

Characterizing Internet Routing Dynamics for Enhanced Network Management

By

Ramakrishna Keralapura

B.E. (Bangalore University) 1998

M.S. (University of Alabama in Huntsville) 2000

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

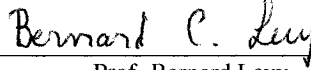
UNIVERSITY OF CALIFORNIA

DAVIS

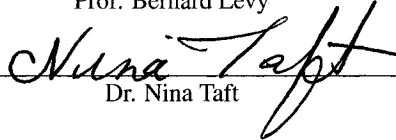
Approved:



Prof. Chen-Nee Chuah



Prof. Bernard Levy



Dr. Nina Taft

Committee in Charge

2007

-i-

UMI Number: 3261167

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 3261167

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Managing large Autonomous Systems (ASes) that constitute the Internet is a herculean task given the scale and complexity of the Internet. Despite enormous efforts in network management, the Internet still falls short of today's expectations. For example, it fails to provide stringent quality-of-service (QoS) guarantees required by delay and/or loss sensitive applications like Voice-over-IP (VoIP) and real-time multimedia streaming. One of the several reasons for this shortcoming in network management is the lack of thorough and comprehensive understanding of the dynamic behavior of routing protocols that control the Internet.

In this dissertation, we begin by analyzing the dynamic behavior of the *Interior Gateway Protocols* (IGP), like *Intermediate System to Intermediate System* (IS-IS) and *Open Shortest Path First* (OSPF) protocols, during network resource failures. We develop a model to characterize the dynamic IGP protocol behavior and show that although these protocols are reliable at larger time scales, the inconsistency in the states of nodes at smaller time scales results in anomalous conditions like routing loops and link overloads. We explain this behavior of IGP not only based on the protocol design, but also on the operational network conditions. We also propose techniques to avoid anomalous network conditions during convergence. Specifically, we propose *Loopless Interface Specific Forwarding* (LISF), a new routing paradigm where packet forwarding is based on both the final destination and the incoming interface, to avoid routing loops and minimize the impact of failures on network traffic.

Traditional *Service Level Agreements* (SLAs), defined by average delay or packet loss, often camouflage the instantaneous performance perceived by end-users. Based on our model for dynamic IGP behavior, we define a set of metrics to capture the *instantaneous service availability* of a net-

work. Using extensive simulations and analysis we show that the traditional approach to characterize networks (using graph-theoretic properties like node degree or network diameter) fails to capture service availability. We also propose a simple yet powerful way to extract the *goodness* of a network and show that this concept has several applications in network management including capacity planning, design, and upgrade. In this dissertation, we address the following question in detail in the context of network upgrade: “how to add new nodes and links into an operational network in a graceful manner so that the perceived network performance from the perspective of existing customers does not deteriorate?”. We propose a two-phase framework to find the optimal upgrade strategy: first, deciding what nodes should be added and how they should be connected to existing topology, and second, deciding the ideal sequence to add these new nodes and links. We formulate the first phase as a non-linear optimization problem and the second phase as a multistage dynamic programming problem.

In the second part of this dissertation, we focus on the interactions between several distributed routing protocols (in layer-3 and layer-7) that coexist in the Internet. In particular, we investigate the unintentional interactions between multiple overlay protocols, as well as between overlay and IP layer protocols. We show that allowing overlay networks to make independent routing decisions at the application level could lead to race conditions that can manifest as oscillations (in both route selection and network load) and cascading reactions (i.e., an event in one overlay network can trigger a series of events in coexisting overlay networks). We identify the causes for synchronization and derive an analytic formulation for the synchronization probability of coexisting overlays. Our model indicates that the probability of synchronization is non-negligible across a wide range of parameter settings, thus implying that the ill-effects of such synchronization should not be ignored. We also use

our model to study the effects of various factors such as path diversity (measured in round trip times) and probing aggressiveness on these race conditions. We extensively analyze the implications of our study on the future design of overlay network protocols and propose several strategies to reduce the impact of race conditions. In addition, we also identify and analyze interactions between overlay and IP networks in the context of traffic engineering and coupling of multiple ASes in the Internet.

To my Grandma

-v-

Contents

List of Figures	ix
List of Tables	xii
1 Introduction	1
1.1 Overview of Internet Routing Infrastructure	2
1.1.1 Intra-Domain Routing	2
1.1.2 Inter-Domain Routing	4
1.2 Motivation and Problem Statement	5
1.3 Our Approach	8
1.4 Dissertation Organization	12
2 Related Work	13
2.1 Network Failures and IGP Convergence	13
2.2 Notion of Service Availability for IP Networks	15
2.3 Overview of Overlay Networks	21
2.3.1 Overlay Network Architecture	21
2.3.2 Interactions between Multiple Overlay and IP Networks	24
3 Intra-Domain Routing Dynamics	26
3.1 Characterizing Interior Gateway Protocol Convergence Behavior	29
3.2 Modeling Service Availability of IP-Networks	35
3.2.1 Service Availability	35
3.2.2 Goodness Factors	36
3.3 Results and Discussion	40
3.3.1 Simulation Setup	40
3.3.2 Service Availability for Known Topologies	43
3.3.3 Goodness Factors vs. Static Graph Properties	44
3.4 Applications of Goodness Factors	50
3.4.1 Goodness Factors from Ingress-node and Link perspectives	50
3.4.2 Network-wide Goodness Factors	51
3.5 Summary and Extensions	54

4	Eliminating Routing Loops during IGP Convergence - LISF	58
4.1	Transient Looping Problem and Existing Approaches	60
4.2	Our Approach	63
4.2.1	Interface-Specific Forwarding	63
4.2.2	Loopless Interface-Specific Forwarding (LISF)	64
4.3	Proof of Loop-free Property of LISF	69
4.4	Performance Evaluation	72
4.4.1	Single Link Failures	72
4.5	Summary	75
5	Applications of Service Availability in Network Management	79
5.1	Graceful Network Upgrade	81
5.1.1	Network Performance from a Customer's Perspective	82
5.1.2	Problem Description	83
5.2	Problem Formulation	85
5.2.1	Phase-1: Finding the Optimal End State	86
5.2.2	Phase-2: Multistage Node Addition Strategy	88
5.3	Numerical Example	93
5.4	Summary	98
6	Cross-Layer Interactions Between Overlay and IP Networks	100
6.1	Hypothesis and Problem Statement	101
6.2	Modeling and Simulating Routing Strategies	102
6.2.1	Overlay Network Dynamics	103
6.2.2	IP-Layer Routing Dynamics	104
6.3	Challenges to Traffic Engineering (TE)	104
6.3.1	Traffic Matrix Estimation	105
6.3.2	Load Balancing	107
6.4	Traffic Oscillations	107
6.5	Coupling of Multiple AS Domains	116
6.6	Discussion	117
7	Race Conditions In Coexisting Overlay Networks	120
7.1	Motivation and Problem Statement	120
7.2	Previous Work in Multiple Overlay Interactions	121
7.3	Simulating Multiple Co-existing Overlays	123
7.3.1	Simulating IP and Overlay Network Dynamics	123
7.3.2	Race Conditions in Multiple Overlays	124
7.4	Why do race conditions occur?	130
7.4.1	Conditions for Traffic Oscillations	131
7.4.2	Conditions for Cascading Reactions	133
7.5	Analyzing Oscillations	135
7.5.1	Probability of Synchronization	135
7.5.2	How long do oscillations last?	141
7.6	Extension of the analytical model	143



7.7	Local and Global Synchronization	146
7.8	Validation of Analytical Model	147
7.9	Sensitivity to Probing Parameters	150
7.9.1	Aggressiveness Factor and Probe Parameter Setting	150
7.9.2	Results and Discussion	151
7.10	Implications of synchronization in overlays	157
7.11	Limiting the Impact of Race Conditions	159
7.11.1	Limiting the Impact of Synchronization	159
7.11.2	Limiting the Impact of Cascading Reactions	163
7.12	Summary	164
8	Conclusions	165
8.1	Summary of Contributions	165
8.2	Future Research Directions	168
8.2.1	Stop Triggers for Oscillations	169
8.2.2	Transparency between ISPs and Application Service Providers	169
8.2.3	Overlay Network Discovery Using <i>DiscOver</i>	171
8.3	Lessons Learned and Concluding Remarks	174
8.3.1	Importance of Realistic Analytical Models	175
8.3.2	Role of Predictability in Network Management	176
8.3.3	Synergistic Coexistence of Distributed Protocols	176
	Bibliography	179

List of Figures

3.1	Example Network Topology for IGP Convergence Illustration	31
3.2	ISP-A topology	41
3.3	ISP-B topology	42
3.4	CDF of Traffic Disruption due to Single Link Failures	44
3.5	CDF of Number of Delay Parameter Violations due to Single Link Failures	45
3.6	Goodness Factor from an Ingress Node Perspective Vs Out-Degree in Various Topologies with Normal Prefix Distribution	46
3.7	Goodness Factor from an Ingress Node Perspective Vs Out-Degree in Various Topologies with Extreme Prefix Distribution	47
3.8	Top Graph shows the Variation of Network Goodness Factor with Network Diameter while the Bottom Graph shows the Variation of Network Goodness with Average Increase in Tree Depth due to Single Link Failures	48
3.9	Top graph shows the Network Goodness Factors while the Bottom Graph shows the Disconnecting Sets for Different Topologies	49
3.10	Top Graph shows the Goodness Factors from the Perspective of Various Ingress Nodes in Topology-6 while the Bottom Graph Shows the Link Badness Factor for Various Links in Topology-6	51
3.11	Network Goodness for Topologies with Different Link Weight Assignment Scheme	52
3.12	Network Goodness for Topologies with Different Prefix Distribution Schemes	53
3.13	Adding a New Node into the Network - Network Goodness for Various Solutions	54
3.14	Network-Wide Time Between Failures on Low Failure Links and the Approximation by Weibull Distribution	55
3.15	TD for Nine Different Associations of Links with Failure Probabilities	57
4.1	Topology-1 used for Illustration	61
4.2	Topology-2 used for Illustration	61
4.3	Scenarios for Illustrating Loop-Freedom under PIPO	69
4.4	Average Service Disruption Time per Link Failure for Various O-D pairs in Sprint Topology with Symmetric Links	73
4.5	Average Service Disruption Time per Link Failure for Various O-D pairs in Sprint Topology with Asymmetric Links	74

4.6	Network Convergence Time due to Various Single Link Failures in Sprint Topology with Symmetric Links	75
4.7	Network Convergence Time due to Various Single Link Failures in Sprint Topology with Asymmetric Links	76
4.8	Average LISF Packet Drop Time per Link Failure for Various O-D pairs in Sprint Topology with Symmetric Links	77
4.9	Average LISF Packet Drop Time per Link Failure for Various O-D Pairs in Sprint Topology with Asymmetric Links	78
5.1	An Example Topology and Potential Node Locations for New Node Placement . . .	93
5.2	Final Solution from Solving the Non-linear Program in Phase-1 with Specific Network and Budget Constraints	94
5.3	Impact on Service Disruption Time and Traffic Disruption when Using Maximum Link Utilization and Node Degree as Metrics to Estimate Network Performance. . .	96
6.1	(a) Illustration Network 1. (b) Illustration Network 2.	105
6.2	Scenario 1 - Two Overlay Networks in One Domain	110
6.3	Link Loads on Various Links in the IP Network for <i>Scenario 1</i>	110
6.4	Scenario 2 - Three Overlay Networks in One Domain	112
6.5	Link Loads on Various Links in the IP Network for <i>Scenario 2</i>	112
6.6	Link Loads on Various Links in the IP Network for <i>Scenario 2</i>	114
6.7	End-to-End Delay for Path A-H in <i>Scenario 2</i>	115
6.8	Scenario 3 - One Overlay Network in Two Domains	116
7.1	Simulation Topology	124
7.2	Link Utilization as a Function of Time	126
7.3	Second Simulation Run: Cascading Reactions.	129
7.4	Third Simulation Run: Cascading Reaction Leading to Oscillations	130
7.5	Two Overlay Networks that Partially Share Primary and Alternate Paths	131
7.6	Three Overlay Networks in Multiple Domains that Partially Share their Primary and Alternate Paths	134
7.7	All Possible Scenarios to Calculate the <i>Region of Conflict</i>	139
7.8	Scenario 1 ($V_1 = R_1/2$ and $V_2 = R_2/2$)	140
7.9	A Comparison of Theoretical and Simulation Results for $P(S)$ Between Two Identical Overlay Networks with Different Values of P	148
7.10	$P(S)$ as a Function of the Ratio P/Q with Different Values of Q for a Given Value of P in Two Identical Overlay Networks	148
7.11	Comparison between Theoretical and Simulation Results for Number of Oscillations.	149
7.12	$P(S)$ Vs $(R_2 - R_1)/R_1$ for Proportional Parameter Overlays with Similar Aggressiveness and Varying RTT	152
7.13	Proportional Parameter Overlays with Mixed Aggressiveness and Varying RTT	153
7.14	Proportional Parameter Overlays with Mixed Aggressiveness and a Chosen Value of RTT	154
7.15	Fixed Parameter Overlays (RON-like) with Same Values of P and Q , and Varying RTT	155



7.16	Fixed Parameter Overlays with Different Values of P and Q , and Varying RTT . . .	155
7.17	$P(S)$ as a Function of P_2 with $P_1 = 5s$	156
7.18	Maximum Number of Oscillations as a Function of Relative RTT	157
7.19	$P(S)$ for Randomized Probing Parameters	160
7.20	Effect of Using Back-off Technique on the Number of Oscillations	162
7.21	Effect of Using Back-off Technique on the Number of Oscillations when the Overlays are Using Different Values for N	162
8.1	Change in Traffic Matrix Entries in a Domain due to Overlay Networks that Span a Single Domain	172
8.2	Change in Traffic Matrix Entries in a Domain due to Overlay Networks that Span Multiple Domains	173



List of Tables

3.1	Summary of Routing Events (with a Route Update Time of 400ms). Network Convergence Time = 1.9s; Service Disruption Time = 1.1s	32
3.2	Convergence Time Depends on Routing Update Time in Nodes	33
3.3	Algorithm to Calculate Service Disruption Time from Node x to Node y due to a Single Link Failure	34
4.1	Summary of Routing Events under OSPF, ORDR, and LISF	62
4.2	Interface-Independent Forwarding Tables at Node B	64
4.3	Interface-Specific Forwarding Tables at Node B	64
4.4	Differences in LISF Methods in Discarding a Packet to d Arriving at i from j	66
4.5	Interface-Specific Forwarding Tables at B under Different LISF Methods	66
4.6	Differences among LISF Methods in Discarding Packets Arriving at Node B	68
5.1	Notations used in Graceful Network Upgrade Problem Formulation	86
5.2	All Possible Upgrade Solutions for Phase-2 (Note: $\omega_{SD} = \omega_{TD} = \omega_{util} = 2$ and $\omega_{degree} = 5$)	97
5.3	Multi-Stage Network Upgrade Solution for Phase-2, With and Without Using SD time and TD as Metrics for Network Performance.	97
6.1	Timer Values for Overlay Networks in <i>Scenario 1 and 2</i>	109
7.1	Timer Values for the Overlays in Simulation	125

Acknowledgements

I consider myself extremely fortunate to have crossed paths with my advisor, Chen-Nee Chuah, during my graduate years. Chen-Nee has always been a major source of inspiration to me, constantly encouraging me to pursue my dreams. She continually gave me the freedom to work on the projects that I found most interesting, while encouraging me to regularly think about the broader impact of my research. Her uncanny ability to identify the scope of new ideas helped me to make the right choices at crucial junctions in graduate school. I can confidently say that in Chen-Nee I found a friend, philosopher, and guide who so easily made my journey through graduate school extremely pleasurable. I sincerely thank Chen-Nee for her constant support, guidance, and mentoring.

Nina Taft has been another great mentor. Her passion for details and her openness to new ideas rubs off on anyone who works with her. Her ability to explain even the most complex concepts in very simple terms makes her stand apart from anyone else who I have worked with. She introduced me to the area of overlay routing and more than half of my dissertation focuses on this concept. She has always been one of my role-models, and continual source of sound advice.

I have also been very fortunate to have had a chance to work with several outstanding researchers including Gianluca Iannaccone, Supratik Bhattacharyya, Srihari Nelakuditi, Jai Ramamirtham, Graham Cormode, Rajeev Rastogi, Yueyue Fan, and Antonio Nucci. I would like to thank all of them for their advice and guidance at various stages of my graduate years. I also like to thank all my committee members - Bernard Levy (both qualifying and dissertation committee), Biswanath Mukherjee (qualifying committee), and Dipak Ghosal (qualifying committee) - for their constructive criticism and helping me to stay on the right track.

Life in Davis would not have been as enjoyable without my friends and lab mates. I had the privilege

to share a lab with some fantastic people like Lihua Yuan, Jianning Mai, Danjue Li, Jason LeBrun, Saqib Raza, Haiping Liu, Chris Dana, and Peng Cheng. I would like to appreciate all their efforts in reviewing my paper drafts, attending all my practice talks, and constantly giving me constructive feedback to improve my work further. We have also had several fruitful discussions about research during our afternoon coffee trips to Silo. I would like to thank all of them from the bottom of my heart for all the things that they have done.

Finally, last but not the least, I would like to thank my family for their constant support during the highs and lows of graduate life. It is beyond words to express my gratitude to my wife, Mallika, without whom it would just be impossible to be where I am today. I would also like to express my most sincere thanks to mom (Kathyaeni), dad (Dr. K.R. Srikantiah), sisters (Dr. Srivani and Soumya), mother-in-law (Geeta Sridhar), and father-in-law (K.R. Sridhar) for their constant encouragement and support through thick and thin. I also want to thank my grandmom for her love and steadfast belief in me. She passed away last year before she could see me graduate and I owe this thesis to her.

Chapter 1

Introduction

Managing large networks that constitute the Internet is a herculean task given the scale and complexity of the Internet. Despite enormous efforts in network management, the Internet still falls short of today's expectations. For example, it fails to provide stringent quality-of-service (QoS) guarantees required by delay and/or loss sensitive applications like Voice-over-IP (VoIP) and real-time multimedia streaming. One of the several reasons for this shortcoming in network management is the lack of thorough and comprehensive understanding of the dynamic behavior of routing protocols that control the Internet. In this dissertation, we focus on characterizing the routing protocol dynamics in the Internet, both at the IP and application layers. Our final goal is to show that better understanding of routing protocol dynamics can significantly improve an Internet service provider's (ISP's) ability to accomplish effective network management.

We begin this chapter by first introducing the Internet routing infrastructure (Section 1.1). We then present our motivation to understand the routing protocol dynamics (Section 1.2) and discuss our approach (Section 1.3). We end this chapter by presenting an outline of this dissertation

(Section 1.4).

1.1 Overview of Internet Routing Infrastructure

Internet can be viewed as a collection of many independent *domains* that can communicate with one another in order to route packets to different parts of the Internet. Each of these domains, also referred to as *Autonomous Systems* (AS) or networks, are composed of many routers and links, that are under the same administrative and technical control. Typically, all the routers that belong to the same domain run the same routing protocol among themselves. The routing protocols used within a single domain are different from the protocols that are used between different domains. The rationale behind different protocols for *intra-domain* and *inter-domain* routing stems from the fact that they have different routing objectives in terms of policies, scale, and performance.

1.1.1 Intra-Domain Routing

Intra-domain routing protocols, also known as *interior gateway protocols* (IGP), are used to determine how routing is performed within a domain. Most commonly used protocols for intra-domain routing are *Routing Information Protocol* (RIP) [64], *Open Shortest Path First* (OSPF) routing protocol [69], and *Intermediate System to Intermediate System* (ISIS) routing protocol [73]. At the heart of these protocols is the routing algorithm that is used to determine the path for a packet from a source router to destination router. The purpose of the routing algorithm is simple: given a set of routers, with links connecting the routers, a routing algorithm finds the “best” path from the source router to the destination router.

In any domain, it is imperative to carefully utilize the network resources in order to max-

imize network performance. This is usually accomplished by the administrators of a domain (or network operators) by assigning “costs” to each link in the network, and routing traffic based on these link costs. The cost may reflect the physical distance traversed by that link, the link speed, or monetary cost associated with the link. The cost of a path is the sum of the cost of the various links along the path. Hence the best path found by routing algorithms usually corresponds to the least-cost path between the source and destination routers.

The routing algorithms used by IGP protocols can be classified into two broad classes[58]:

- *Global Routing Algorithms:* These routing algorithms compute the least-cost path between a source and destination using the global knowledge about the network. In other words, these algorithms take the connectivity between different routers and the link costs as inputs, and compute the least-cost paths. These algorithms are also known as *link state algorithms*. One popular example for this kind of algorithms is *Dijkstra’s shortest path algorithm*, which is used in OSPF and ISIS routing protocols.
- *Decentralized Routing Algorithms:* These routing algorithms compute the least-cost path in an iterative and distributed fashion. That is, the routers do not have a global knowledge, but iteratively exchange information with their neighbors to compute the least-cost paths between source and destination routers. These algorithms are also known as *distance vector algorithms*. An example for this kind of algorithms is *Bellman-Ford algorithm*, which is used in RIP.

1.1.2 Inter-Domain Routing

Inter-domain routing protocols, also known as *exterior gateway protocols* (EGP), are used to determine how routing is performed between different domains. The *Border Gateway Protocol* (BGP) [79][89] is the defacto EGP protocol. A small subset of routers in every domain run BGP and these routers are usually connected to BGP routers in other domains. The relationship between the domains can be that of provider-customer or peer-peer.

BGP uses a *path vector routing algorithm*, which is very similar to the distance vector intra-domain routing algorithm except that: (i) no link costs are assigned to inter-domain links since the link connects two independent domains, and (ii) inter-domain routing is mainly based on policies that are determined by the relationship between different domains. The information exchanged between BGP routers in different domains are mainly determined by their relationship and service level agreements (SLAs). For example, a customer domain will not advertise any routes that it hears from its provider to any of its peers, since it does not want to carry any transit traffic destined to the provider from other domains in the Internet.

Typically, a router can exchange BGP routing information with two types of nodes: (i) routers within the same domain via Internal-BGP (I-BGP), or (ii) external peers outside its domain via External-BGP (E-BGP). The main objective of I-BGP is to distribute routes learned via E-BGP to all the routers within the domain. I-BGP and E-BGP are essentially the same protocol (e.g., same message types and state machine), but they use different rules for re-advertising routes to prevent announcements from looping [89]. The main implication of the rules is the requirement of a fully meshed I-BGP session between each pair of routers in a domain, which poses scalability and management problems. One widely deployed solution to get around this problem is to group routers

into clusters consisting of a leader called a route reflector (RR), and members that are called route clients (RC) [21]. With clustering, the clients only peer with their respective route reflectors, but the route reflectors must be fully meshed.

1.2 Motivation and Problem Statement

Despite its scalability and ease of deployment, the current Internet fails to provide stringent quality-of-service (QoS) guarantees required by some delay and/or loss sensitive applications like Voice-over-IP (VoIP) and real-time multimedia streaming. For example, IGP takes tens of seconds to re-converge after a link/node failure, while path restorations on inter-domain paths can take as long as 15 minutes due to the slow convergence of BGP resulting in poor performance of real-time applications.

The performance (i.e., efficiency, reliability, availability, security, etc.) of the Internet infrastructure critically depends on effectively *managing the Internet routing plane*. Unlike the standard performance metrics (i.e., packet loss, delay, and throughput), reliability, availability, and manageability of the Internet infrastructure are not well understood, often without any theoretical or quantitative frameworks to study them. From this perspective, it is important to understand the behavior of IGP and BGP protocols not only under normal network conditions, but also when unexpected events (like software and hardware malfunctions) occur in the network. There have been numerous works in the past that have extensively studied the behavior of BGP, evaluating its normal and dynamic convergence properties [47, 46, 93, 76, 75, 80, 65, 63, 8, 9]. However, the same is not true when we consider IGP protocols. In other words, there has been a lot of effort in understanding IGP protocol behavior under normal network conditions [69, 73], but little effort in

understanding/characterizing their dynamic convergence behavior after unexpected events occur in the network. Previous studies have shown that these unexpected events like link/node failures occur very often in today's networks and hence it is critical to understand the dynamic behavior of IGP protocols for effective network management [66]. In the first part of this dissertation, we seek to answer some of the following questions (Chapters 3, 4, and 5):

- What events are triggered by IGP protocols when unexpected events like link/node failures occur in the network? What state changes occur in the network before it converges to a new state? How do these network dynamics after a failure affect the packet forwarding capability of a network?
- How can we model/characterize IGP network dynamics (i.e., state changes in the network) after failures? If we can accurately characterize IGP network dynamics, then how can we use this to manage networks more effectively?
- Does the IGP convergence process result in any undesirable or harmful network states? If yes, then how can we fix the IGP protocol behavior to ensure that the network does not enter these undesirable or harmful states?

From the perspective of network management, it is also important to study the interactions between different independent Internet protocols in the wild. In other words, although different protocols are designed and tested to be independent of each other in controlled environments, these protocols could interact with each other when they are deployed in the Internet - perhaps the most complex distributed networked system. Existing studies on routing behavior have focused on individual routing protocols. For instance, the authors in [12] study IS-IS protocol-level behavior

and make recommendations to achieve sub-second IS-IS convergence. In [85], the authors present black-box measurement techniques for estimating key internal delays in OSPF implementations that can affect the speed of update propagation in the network and the time needed to re-converge. Many studies have also been dedicated to examine the growth of BGP routing tables [22], the convergence time after routing changes [46, 47], BGP instability [59], and router misconfigurations [63]. The study in [60] reveals how BGP routing policies impact the selection of Internet paths, in which the best paths are often not the shortest or most efficient routes. Although it is important to focus on individual protocols to study their inefficiencies and problems, it is equally important to study the interactions between different protocols that can potentially affect the overall stability of the routing plane.

Application service providers adopt various strategies to overcome some of the problems in the routing infrastructure of IP networks. One of the most commonly used strategy to avoid non-optimal paths selected by the routing protocols, and to provide more flexible and robust routing for applications, is to construct networks at the application layer (called *Overlay Networks*) on top of the IP layer [16, 1, 98]. Overlay networks intend to provide a platform for delay and loss sensitive applications by giving more control to applications in choosing Internet paths, thus aiming to improve applications' performance. Even though a lot of effort has been devoted in designing new architectures for overlay networks, there is a dearth of studies that explore how these new architectures interact with the existing infrastructure. Such studies will prove to be of immense help in determining the impact of new architectures on the applications that are already deployed. In this context, we seek answers to the following questions (Chapters 6 and 7):

- Do overlay network protocols, that are intended to be independent of other protocols, inad-

vertently interact with the underlying IP layer protocols? Do multiple independent overlay network protocols inadvertently interact with each other?

- If the answer to the above questions is ‘yes’, then what kind of interactions can occur and what are the consequences? Are these interactions beneficial or harmful to the overall system? Can we qualify and quantify these interactions? Can we develop an analytical model to understand and study these interactions?
- If the interactions are harmful, then what steps can be taken to reduce the impact of such interactions? Is it possible to design protocols such that they can synergistically coexist?

1.3 Our Approach

Our aim in the first part of this dissertation is to characterize the dynamic behavior of IGP protocols during network resource failures and use this understanding to address several issues in network management. Using extensive analysis based on the data from a tier-1 Internet service provider (ISP) network, we found that although these algorithms are reliable at larger time scales, the inconsistency in the states of nodes at smaller time scales results in anomalous conditions like *routing loops* and link overloads. These conditions severely affect the packet forwarding capability of the network resulting in poor service availability. We explain this behavior of IGP protocols not only based on their design but also on the operational conditions in the network. We have also developed a comprehensive convergence model that can *predict* the behavior of IGP protocols during network resource failures and quantify the impact of these failures on the performance of the network.

Using the model, one of the problems that we found during IGP convergence was transient

routing loops (also called forwarding loops), mainly caused by routers having inconsistent views of the network. In other words, the network can enter a state where packets go back and forth between a set of routers and may not reach the final destination. To address this problem we propose a novel routing paradigm - *loopless interface-specific forwarding* (LISF) - that avoids transient loops by forwarding a packet based not only on its destination but also its incoming interface. We show that LISF requires no modifications to the existing link-state routing mechanisms and it can be easily deployed in current routers since they already maintain a forwarding table at each interface for lookup efficiency. Using simulations we show the feasibility and evaluate the efficiency of LISF.

Traditional service level agreements (SLAs), defined by average delay or packet loss, often camouflage the instantaneous performance perceived by end-users. Based on our model for dynamic IGP behavior, we have defined a set of metrics that capture the *instantaneous service availability* of a network. Given a network topology and its link weights, we propose a novel technique to compute the associated service availability by taking into account transient routing dynamics and operational conditions, such as BGP table size and traffic distributions. Using extensive simulations and analysis we have shown that the traditional approach to characterizing networks (using graph-theoretic properties) fails to provide insights on the expected performance perceived by end customers. We also show that the amount of service disruption experienced by *similar* networks (i.e., networks with similar intrinsic properties such as average out-degree or diameter) could be significantly different, making it imperative to use our new metrics for characterizing networks.

Based on the set of new metrics, we have also proposed a simple yet powerful way to extract the *goodness* of a network from three perspectives - ingress node (from one node to many destinations), link (traffic traversing a link), and network-wide (across all source-destination pairs).

We show that this concept of *predictable* network goodness can be used to address several network management issues including designing networks from the perspective of both ISPs and customers, determining BGP peering points, and IGP link weight setting among several others.

One of the critical aspects of network management that has not received much attention is network upgrade. We address the question of “how to add new nodes and links into an operational network in a graceful manner so that the perceived network performance from the perspective of existing customers does not deteriorate?”. We propose a two-phase framework to find the optimal upgrade strategy: first, deciding what nodes should be added and how they should be connected to existing topology, and second, deciding the ideal sequence to add these new nodes and links. We formulate the first phase as a non-linear optimization problem and the second phase as a multistage dynamic programming problem. Through a numerical example, we show the feasibility of this framework and demonstrate the advantages of our multistage approach in determining an ideal upgrade sequence. The results clearly highlight the significance of incorporating network performance (for ex, service availability) into the two-phase framework to achieve minimal impact to existing customers.

In the second part of this dissertation, we focus on qualifying, quantifying, and modeling the interactions between different independent protocols that coexist in the Internet. We also focus on how these interactions can hinder an ISP’s effort to perform effective network management. We first explore the interactions between routing protocols in the IP and overlay layers, and then examine the interactions between multiple coexisting overlay routing protocols.

ISPs manage performance of their networks in the presence of failures or congestion by employing common traffic engineering techniques such as link weight settings, load balancing,

and routing policies. Overlay networks attempt to take control over routing in the hope that they might achieve better performance for such failures or high load episodes. We examine some of the interaction dynamics between the two layers of control from an ISP's view. With the help of simple examples, we illustrate how an uncoordinated effort of the two layers to recover from failures may cause performance degradation for both overlay and non-overlay traffic. We also show how current traffic engineering techniques are inadequate to deal with emerging overlay network services.

By allowing end hosts to make independent routing decisions at the application level, different overlay networks may unintentionally interfere with each other. In this dissertation, we describe how multiple similar or dissimilar overlay networks could experience race conditions, resulting in oscillations (in both route selection and network load) and cascading reactions. We pinpoint the causes for synchronization in terms of partially overlapping routes and periodic path probing processes and derive an analytic formulation for the synchronization probability of two overlays. Our model indicates that the probability of synchronization is non-negligible across a wide range of parameter settings, thus implying that the ill-effects of synchronization should not be ignored. Using the analytical model, we find an upper bound on the duration of traffic oscillations. We also show that the model can be easily extended to include a large number of co-existing overlays. We validate our model through simulations that are designed to capture the transient routing behavior of both the IP- and overlay-layers. We use our model to study the effects of factors such as path diversity (measured in round trip times) and probing aggressiveness on these race conditions. Finally, we discuss the implications of our study on the design of path probing process in overlay networks and examine strategies to reduce the impact of race conditions.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. In Chapter 2, we present an overview of intra-domain routing and overlay networks. We also summarize all the prior work in the context of IGP routing dynamics, service availability of IP networks, and interactions between overlay and IP protocols. We also describe how the work in this dissertation is significantly different (in problem definition, approach, results, and conclusions) from all the existing studies.

Chapter 3 presents our effort in characterizing IGP routing dynamics and also demonstrates our attempt in using it to define the service availability (and subsequently the *goodness factors*) of IP networks. We address the problem of transient routing loops during IGP convergence in Chapter 4, and present a new approach to address the problem. In Chapter 5, we explore an application of our IGP convergence and service availability model in the context of graceful network upgrade, and present a two-stage framework to determine the optimal upgrade strategy.

In Chapter 6, we identify several problematic interactions between overlay and IP layer routing protocols, and make a case for future research work in that direction. In Chapter 7, we explore one of the interactions between multiple coexisting overlays in detail and develop an analytical formulation. Using the model, we comprehensively evaluate the interactions and quantify them. Finally, we present some conclusions, contributions of this dissertation, and pointers for future work in Chapter 8.

Chapter 2

Related Work

In this chapter, we provide a brief overview of intra-domain routing as well as overlay networks. Section 2.1 presents a brief overview of IGP protocol operation in real ISP networks and discusses prior work in the context of modeling IGP protocol convergence due to network failures. Section 2.2 summarizes studies in the past related to characterizing service availability of best effort IP networks. In Section 2.3, we introduce overlay networks and provide a background on different types of overlay networks. We also summarize all the earlier efforts in understanding the influence of overlay networks on IP networks.

2.1 Network Failures and IGP Convergence

Although network failures are a well known set of events that occur very frequently in IP networks [66], not much effort has been devoted to understanding the nature and impact of failures on the service offered by the network. It was not until recently the typical patterns of failures in large backbone networks were characterized [66].

Traditionally at the IP layer, link/node failures in a network are detected by the loss of IGP *hello* messages (that are sent every 10 seconds by default) between neighboring nodes. A node that detects the loss of a few consecutive hello messages assumes the occurrence of a failure, generates a failure message, and floods it in the network. All the other nodes that hear this failure message will recompute their shortest path tree and update their forwarding tables. This process typically takes tens of seconds for all the nodes in the network to recompute shortest paths. Packet forwarding along the alternate path (found by shortest path recomputation) resumes after all the nodes update their view to the new network state.

Network failures can also be handled at the optical layer which lies below the IP layer. The advantage of doing this is that failure detection and recovery are very fast compared to the entire process at the IP layer. The convergence time can be reduced from the order of tens of seconds to under a second, but the disadvantage of this technique is that it is very expensive. Given that the optical switches do not have a buffer, they have to be over-provisioned to be able to handle failures.

An alternative approach to handle failures is to detect failures at the optical layer and use the recovery techniques at the IP layer. This approach is cost efficient compared to the all-optical process, and relatively fast compared to the all-IP process. Most of the tier-1 ISPs adopt this technique to detect and restore failures. Even though this approach does not take tens of seconds to converge, in big networks it still takes between 6 to 10 seconds [51]. It is very important to understand the behavior of a network during the period between the occurrence of a failure event and the time when the packet forwarding recovers. One of our contributions is to characterize/model the network dynamics that follow a failure event [55].

Although there has been very little work in understanding the evolution of network states

during IGP convergence, there have been several studies that have attempted to measure and improve the IGP convergence time after network failures. In [13], Alaettinoglu et al claim that the IGP convergence time in real networks was nowhere close to the achievable minimum, and propose changes to ISIS specification and implementations to reduce the convergence time. They show that their changes are easily scalable as the number of nodes and links in a network increases. Similarly, authors in [30] propose a novel routing control methodology based on priority queues, where the “hello” messages are given higher priority than the data messages to help in achieving faster IGP convergence time. Leveraging some of the concepts proposed in this dissertation, the authors in [41] conducted measurement-based experiments to show that the IGP convergence time in large networks can be reduced to under 1 second without compromising the stability of the network.

2.2 Notion of Service Availability for IP Networks

Failure of different links/nodes in a ISP network has different impact both on customers and the ISP. In other words, the service availability of a network as perceived by a customer is different for different failures. Similarly the impact of different failures on the ISP in terms of the resulting SLA violations is different. For example, some link failures can result in high packet losses leading to large number of SLA violations, while some others may not have a big impact on SLAs.

The notion of availability was first introduced in the traditional telephone networks [49]. Telephone networks are deterministic networks when compared to IP networks because of strict admission control policy. The availability of a path in the telephone network can be defined in terms of the availability of the individual components that makeup the path for a call between two

subscribers. For example in [49], the availability of a path is defined in terms of the availability (or unavailability) of the components of a path segment like distribution (the portion of the path segment between the subscriber and the telephone switch in the central office), switch in the central office, facility entrance (the portion of the path segment that includes A/D converters, multiplexers, automated digital terminals, etc.) and inter-office (the portion of the path segment connecting two switches). In [57] and [34], the availability of a telephone network is determined based on the outages reported by the individual telephone companies to FCC. The metrics used for calculating the availability of the network are: *number of outages*, *customer minutes*, and *blocked calls*. The *number of outages* includes all the outages that affected more than 30,000 customers or lasted for more than 30 minutes. This does not capture the smaller-scale outages. The *customer minutes* represents the product of the number of customers affected in each outage and the actual outage time. This metric assumes that every customer tries to use the network for the entire duration of the outage. The *blocked calls* metric represents the actual customer calls that were blocked. This metric tries to compensate for the deficiency of the *customer minutes* metric but ignores the multiple calls that originate from a customer due to a single blocked call.

In the Internet, most of the existing work on availability deals with the optical layer, where *availability* is used as an optimization metric to design path or sub-path protection schemes. Service availability can be defined in numerous different ways. One of the earlier definitions of availability, A , was proposed in [48] as:

$$A = \frac{MTBF}{(MTBF + MTTR)} \quad (2.1)$$

where $MTBF$ is the *mean time between failures* and $MTTR$ is the *mean time to repair*. But this

definition had a fundamental flaw in that $MTTR$ was a part of $MTBF$. In [31], the authors came up with a modified definition for *service availability* that measured the fraction of the time for which the service was available, i.e.,

$$A = \frac{MTTF}{(MTTF + MTTR)} \quad (2.2)$$

where $MTTF$ is the *mean time to failure* and $MTTR$ is the *mean time to repair* [31]. The authors in [31] further analyze mesh networks at the optical layer and state that for dual failures, the availability of such networks is much higher than a 1+1 APS (i.e., Automatic Protection Switching).

Different research works, like [74] and [97], propose solutions to achieve high availability using fast recovery and hence avoid the transient condition during failures that affect the overall delay and loss. In [74], the authors suggest a new protection scheme, called sub-path protection, which incorporates the idea of shared-path protection to achieve high scalability and fast recovery time for a modest sacrifice in resource utilization. The main idea of sub-path protection is to first partition a large network into several smaller domains, and then apply shared-path protection to the network such that an intra-domain path does not use resources of other domains and the primary and backup paths of an inter-domain path exit a domain (and enter another domain) through a common egress (or ingress) domain-border node. This scheme helps in restricting the transient condition to a single domain and hence suppressing its ill effects. In [97], the focus is on providing high availability to the customers by provisioning the connections with different protection schemes using a new framework that the authors propose.

As mentioned before, the restoration mechanisms in the optical layer are very expensive compared to the restoration mechanisms in the higher IP layer, which are both effective and eco-

nomical [50]. Many schemes for fast convergence and improving service availability have been suggested in the IP layer. The authors in [50] show that it is possible to achieve significant improvements in failure restoration time by tuning the timers that govern failure recovery speed as described in the 7-step process [51]. They also conclude that an IP-based restoration approach is viable for an ISP backbone and has several advantages. It provides high network availability at a small fraction of the cost for building protection or restoration mechanisms in the underlying optical layer. It avoids the complexity, high operational costs, and overheads associated with newer technologies such as MPLS. At the same time, it is ideally suited for the IP design paradigm - loosely organized networks with best-effort delivery and no admission control.

To take advantage of the low cost and complexity of the IP layer mechanisms, many tier-1 ISPs use optical layer only for failure detection and the IP layer for protection and restoration [44]. Hence new techniques were investigated and suggested to improve service availability at the network layer. One such approach was suggested in [71] known as *Failure Insensitive Routing* (FIR). Under FIR, when a link fails, its adjacent node suppresses global updating and instead initiates local rerouting of packets that were to be forwarded through the failed link. Though other nodes are not explicitly notified of the failure, they infer it from packets' arrival at the node. When a packet arrives at a node through an unusual interface (through which it would never arrive had there been no failure), corresponding potential failures can be inferred and the next hop can be chosen avoiding those links. Under FIR, the next hop for a packet is determined based on not only the destination address but also the incoming interface. Note that such interface dependent forwarding is feasible with current router architectures as they already maintain a forwarding table at each line card of an interface for lookup efficiency. These interface dependent forwarding tables can be pre-computed

since inferences about link failures can be made in advance. Thus with FIR approach, when a link fails, only nodes adjacent to it locally reroute packets to the affected destinations and all other nodes simply forward packets according to their pre-computed interface specific forwarding tables without being explicitly aware of the failure. Once the failed link comes up again, if the failure event is being suppressed, original forwarding tables are locally restored and forwarding resumes over the recovered link as if nothing ever happened. This approach decouples destination reachability and routing stability by handling transient failures locally and notifying only persistent failures globally. Essentially with FIR, in the presence of link failures, packets get locally rerouted possibly along sub-optimal paths but do not get caught in routing loops or get dropped till the new shortest paths are globally recomputed.

Another mechanism with a similar purpose is proposed in [92] known as *Alternate Routing*, which is based on deflection (i.e. instead of dropping packets in the transient period, routers could forward them to other links so that they can reach the destination through a different path). In this scheme, each node computes a map for each of its links to an alternate link. This alternate link will be used to forward a packet when the original link fails. Since the number of links at a router is typically smaller than the number of destinations in the routing table, the alternative link table is much smaller than a typical routing table. Such an approach leaves the existing routing tables unmodified. A packet is deflected if the node detects a link failure or if the incoming interface and the outgoing interface of the packet are the same. Alternate tables are computed at the same time when the routing tables are computed. The authors in [92] suggest an algorithm to find alternate links based on breadth first search and an algorithm to deflect packets during failures based on sink trees.

The approaches for improving service availability in [71] and [92] involves alternate packet forwarding mechanisms to avoid traffic *black-holes* and routing loops during service convergence. A single (or multiple) link failure in a network increases the traffic on the alternate path resulting in overload in some of the links on the alternate path. If the network is not adequately over-provisioned then the traffic may experience packet losses and service disruption due to the fact that the alternate path is short of capacity to support the traffic that fails-over from the primary path. A *deflection routing* mechanism to counter this problem is suggested in [53]. The authors in [53] propose to improve service availability by distributing the traffic on various alternate paths to avoid the overload on specific links.

Despite the fact that a lot of effort has been devoted to *improving* service availability in the network layer, little work has been carried out in *modeling* service availability to characterize IP networks. Even though many different models for service availability have been suggested in the optical layer there is still a need for a new model in the IP layer due to the fact that the two layers are considerably different. In the optical layer the main focus is on assignment and distribution of traffic on different wavelengths, which makes the optical-layer models inapplicable in the IP layer. One of our contributions in this dissertation is to characterize IP networks using service availability as a metric [55, 56].

In this dissertation, we first characterize the behavior of IGP routing protocols after network failures. We use this model for IGP routing dynamics to estimate (i.e., predict without active or passive measurements) the service availability of a network. We then show that using service availability as a metric to characterize IP networks has several advantages, including better network design and easier management (e.g., network upgrade).

2.3 Overview of Overlay Networks

Overlay networks have emerged as a promising platform to provide customizable and reliable services at the application layer while retaining the best-effort network layer. An overlay network typically consists of pre-selected nodes, located in one or more network domains, that are connected to one another through application-layer routing. One of the underlying paradigms of overlay networks is to give applications more control over routing decisions, that would otherwise be carried out solely at the IP layer. The advent of a wide variety of active measurement techniques has made this possible. An overlay network typically monitors multiple paths between pairs of nodes and selects one based on its own requirements of end-to-end delay, loss rate, or throughput.

2.3.1 Overlay Network Architecture

Typically overlay networks can be classified into two broad categories based on its design: *infrastructure-based* and *peer-to-peer (P2P)* overlay networks. While infrastructure-based overlay networks deploy some pre-selected set of fixed nodes that provide overlay services, P2P overlay networks depend on nodes that frequently enter and leave the network to dynamically change its topology and connectivity, thus offering diverse services. Some of the popular infrastructure-based overlay networks are Detour [84], RON [16], Akamai [1], OPUS [24], etc. Similarly SplitStream [28], Napster [5], Kazaa [4], Gnutella [3], Chord [90], and Bamboo [81] are examples of popular P2P services.

One of the earliest designs for application layer networks was proposed in Detour [84], where the authors identify the problems in Internet routing. They highlight the fact that the paths that applications traverse are typically not ideal in terms of various performance metrics like end-

to-end delay, loss, and throughput. They identify two main reasons for this: (i) Policy-based BGP routing restricts the use of best inter-AS path between different source-destination pairs that span multiple domains, and (ii) Proprietary traffic engineering policies determine the shortest path routing within a domain. The authors go on to suggest that instead of using default Internet routing, overlay networks can be used to provide better paths and hence improve the performance of various applications.

The main focus in RON [16] is to build an overlay network that can provide resiliency to applications. The paper identifies that failures in the Internet could take a long time to recover. For example, a BGP reconvergence after a failure takes more than 10 minutes. The paper proposes an overlay network that recovers from a failures in the order of seconds instead of minutes. With the help of an experimental test bed, the authors show that they achieve significant improvement in performance.

One other popular application of infrastructure-based overlay networks is in the distribution of content in the Internet. Popular websites (like Yahoo! [7], and CNN [2]) that get millions of hits everyday and need to stream video and audio to several people at the same time face the problem of congestion, server overload, and performance degradation. One approach to avoid these problems is to store multiple copies of the content in strategic locations on the Internet, and dynamically redirect requests to different servers to avoid congestions, server overload, and performance degradation. This requires that the servers at various locations on the Internet are always up-to-date. To accomplish, this content delivery networks (like Akamai) build application layer networks that provide services to various websites in replicating their data in different places on the Internet.

While infrastructure-based overlay networks require an organization to own the complete

application-layer network and administer it, P2P networks are built dynamically using different nodes that enter and leave the network. Popular file sharing services like KaZaa form application-layer networks using the nodes of the users who login and use this network to exchange audio, video, and data files among themselves. Notice that the current users can leave the network at anytime and new users can join the network frequently. Hence these overlay networks do not have a fixed topology or connectivity, but depends on the number of users logged in, their locations, and the files that they own.

Another well-known use of overlay networks is in distributed hash tables (DHTs) like Chord [90], Pastry [83], Tapestry [98], Scribe [29], and Bamboo [81]. DHTs are self-organizing, structured peer-to-peer networks that are used to locate objects and route messages in a group of nodes. Typically the nodes and objects in a DHT are assigned identifiers (called *nodeID* and *keys* respectively). The node identifiers and object keys have a certain length (for example, 128 bits long in Pastry) and are derived from a 2^b id space, where b is typically a large value. Given a message and key, DHTs route the messages through the system of nodes such that the number of hops taken by the message to reach the destination object is minimized. The approach is to continually route the message such that the next hop is numerically closer to the destination than the current node.

Implementing multicast in the Internet has been explored for a long time. Many schemes (like [33]) that were proposed required changes to the IP routing plane and hence did not find widespread deployment. Given that IP layer multicast was not feasible in the Internet, the focus shifted to application layer networks to accomplish multicast. SplitStream [28] was one such scheme that proposed the use of overlay networks to achieve multicast. The authors propose to use distributed hash tables (like Pastry [83] or Scribe [29]) to build multicast trees that are used to distribute audio/video

to different hosts in a reward-based approach.

2.3.2 Interactions between Multiple Overlay and IP Networks

The interaction between overlay networks and IP networks was first identified by the work of Qiu et al [77] where the authors investigate the performance of selfish overlay routing in Internet-like environments. The approach in this paper was to model overlay routing and IP routing as a game theoretic problem. In this game, overlay networks and IP network take turns in playing the game before they reach the Nash equilibrium point (when network-level routing is static). The authors evaluate the performance of the network only after the system reaches equilibrium. This approach is based on two assumptions: (i) The system has a Nash equilibrium point and is reachable, and (ii) Overlay networks and the IP network take turns in playing the game. Also, the work ignores a wide variety of dynamics (due to various events like link/node failures, congestions, software bugs, etc.) that occur in the real-world networks. Zhang et al [96] and Liu et al [62] model the interaction between overlay routing and TE as two-player game, where the overlay attempts to minimize its delay and TE tries to minimize network cost. They argue that the lack of common objective for the overlay and IP networks could result in poor performance. In this dissertation, we focus instead on dynamics of the overlay routing layer before the system reaches the equilibrium. Instead of static network-layer routing, we consider events such as link/router failures, flash crowds, and network congestions that lead to dynamic re-computation of routes in overlay applications and/or IGP protocols.

Other works have studied the overlay network probing process, a crucial component of overlay routing. Nakao et al. [70] proposed a shared routing underlay that exposes large-scale, coarse-grained static information (e.g., topology and path characteristics) to overlay services through

a set of queries. They advocate that the underlay must take cost (in terms of network probes) into account and be layered so that specialized routing services can be built from a set of basic primitives. However, sharing network-layer path information may induce synchronized routing decisions in overlay networks and unintentionally lead to route/traffic oscillations, an aspect not addressed in [70].

In this dissertation, we hope to shed some light on this problem through our modeling of overlay and IP-layer dynamics in response to network failures. Finally, based on the work in this dissertation, the authors in [42] explore the problem of oscillations due to intelligent route control (IRC) systems deployed by networks that multihomed to different ISPs. The objective of IRC systems is to probe all the available alternate paths and determine the best path for a network to route traffic. Similar to the findings in this dissertation, the authors show that IRC systems, like overlay networks, do not consider *self-load effects* while moving traffic to alternate paths, resulting in oscillations.

Our aim in the second part of this dissertation is to first identify potential interactions between the overlay and IP-layer routing protocols. We then pinpoint the reasons for these interactions, and qualify and quantify their effect. We also develop a comprehensive analytical model to study the interactions between multiple coexisting overlay networks, and use it to show that these interactions are not pathological, but are very likely to occur in real network conditions. We also study the impact of these interactions on overlay and IP networks to show that it is important to address these issues. Finally, we explore several techniques to reduce the ill-effects of these interactions and provide guidelines for better overlay network design.

Chapter 3

Intra-Domain Routing Dynamics

Service-Level Agreements (SLAs) offered by today's Internet Service Providers (ISPs) are based on four metrics: end-to-end delay, packet loss, data delivery rate, and port availability. The first three metrics are usually computed network-wide and averaged over a relatively long period of time. For the fourth metric, the term "port" refers to the point at which a customer's link attaches to the edge of an ISP's network. Port availability therefore refers to the fraction of time this port is operational and measures a customer's physical connectivity to the ISP's network. None of these SLA metrics capture the ability of the network to carry customer traffic to Internet destinations at any point in time.

The main problem with the existing SLA specifications is that they do not capture the effect of instantaneous network conditions like failures and congestions. A recent study [66] shows that failures occur on a daily basis due to a variety of reasons (e.g., fiber cut, router hardware/software failures, and human errors) and can impact the quality-of-service (QoS) delivered to customers. When a link/node fails, all routers will independently compute a new path around the failure. Dur-

ing that time, routers may lack or have inconsistent forwarding information, resulting in packet drops or transient routing loops [23][87]. However, not all failures impact the network equally. The failure of a critical backbone link carrying heavy traffic may be more detrimental than the failure of an access link connecting a single customer. Yet, these service degradations are camouflaged by the *average* parameters reported in current SLAs. Therefore, to *measure* IP network performance, it is essential to consider the network routing configuration and traffic pattern during link or node failures.

Reports from various tier-1 ISPs suggest that IP backbone networks are usually over-provisioned where the link utilization of backbone links is less than 50% of their total capacity [36][40]. The reports also confirm that congestion due to link or router overload is a very rare event in backbone networks. During a link failure event, traffic on the failed link is rerouted and may congest links along alternate paths. However, such congestions are usually not significant for a single failure event. Heavy congestions may occur when there are multiple failures, but such events are relatively rare. Hence in our current work we ignore the effect of congestions on network performance and only consider failures.

Here we define a set of metrics for *service availability* of IP backbone networks that capture the impact of routing dynamics on packet forwarding. Instead of relying on active or passive measurements, we propose a methodology to *estimate* the service availability of a network in the presence of independent link failures. Specifically, given a topology (nodes, links and connectivity) and routing information (link weights, link delays and BGP peering points), we are able to compute the potential impact on service due to link failures. To achieve this, we carefully model the factors identified in the measurement-based study by Iannaccone et al [51] that contribute to routing

convergence. Convergence refers to the amount of time it takes for traffic forwarding to resume correctly on the backup path after a link failure. We wish to point out here that we focus mainly on single link failures for IP backbone networks since these are the dominant class of failures (i.e., over 70% of all failures in IP backbone networks) as observed by Markopoulou et al [66].

We use the novel concept of service availability to evaluate known topologies, such as full-mesh, ring and Tier-1 ISP backbone topologies. Our simulations show that the performance of a network not only depends on routing dynamics, but also on various other factors such as IGP link weight assignment and BGP prefix distribution. This brings out a necessity to identify new metrics that customers can use to differentiate networks. There have been many attempts to characterize the Internet topologies or to model their graph-theoretic properties [78][35][45]. The resulting models are useful for re-generating topologies that best model real networks, such as GT-ITM [27] and BRITE [67], but they do not provide any insights on the QoS that a particular network can provide.

Using the concept of service availability, we derive *goodness factors* based on three different perspectives: ingress node (from one node to many destinations), link (traffic traversing a link), and network-wide (across all source-destination pairs). The goodness factors reveal how topologies with similar intrinsic graph properties, such as average out-degree or network diameter, do not necessarily offer the same level of service availability.

Finally, we describe several applications for the goodness factors in network planning and provisioning. For example, goodness from an ingress node perspective allows customers to choose the best place to connect to a network (or to choose among different providers), while link-based goodness helps an ISP to identify the set of critical links to be upgraded.

The rest of the chapter is organized as follows. Section 3.1 identifies the importance of

routing dynamics in estimating the end-to-end performance of a network and motivates this work. In Section 3.2, we describe the proposed metrics and introduce the concept of service availability to characterize network topologies. We also define a set of network goodness factors. We discuss our numerical results in Section 3.3. Finally, we examine various applications of goodness factors (Section 3.4) and present some conclusions (Section 3.5).

3.1 Characterizing Interior Gateway Protocol Convergence Behavior

Intra-domain routing protocols, such as IS-IS [73] and OSPF [69], define how each node in the network responds to changes in the topology. Such protocols are also known as *link state protocols*, where each node has complete knowledge of the network topology including all the links present in the network.

Upon detection of a link/node failure or a configuration change in the network, each node is responsible for disseminating the new topology description to all its neighboring nodes and re-computing the forwarding information in its own routing table. From the time of the failure or configuration change to the time all nodes have been informed of the change and have updated their routing tables, traffic disruptions (like packet drops and routing loops) are possible as the nodes may have an inconsistent view of the network.

We define the “convergence time” (CT) of a node due to a failure event in the network as the time taken by the node to update its routing and forwarding information in response to the failure. A node that does not have to update its routing or forwarding information has a convergence time of zero. The convergence time for any node n (that has to update its routing or forwarding information) due to a failure can be summarized as a combination of 3 components:

- *Detection time*: This is the time taken by the adjacent nodes to detect the failure. Today's IP routers provide several mechanisms to perform this function [52], but all of them are based only on local information exchanged between neighboring nodes. For example, *hello* messages at the IP layer or alarms at the optical layer. Hence detection time represents a fixed price that is independent of the network topology or configuration.
- *Notification time*: This represents the time taken by the routing update to propagate through the network to reach n . In link state protocols, messages are flooded throughout the network. Therefore, the notification time strongly depends on the hop distance between n and the adjacent nodes. Each node along the forwarding route needs to process the message update before forwarding it, thus introducing a delay in the propagation of information.
- *Route Computation and Update time*: This is the time spent by node n to compute the new shortest path routing tree (that incorporates the failure information) and then update its forwarding information based on the new routing tree. The update procedure involves applying the changes to all the network prefixes that have been learnt via the BGP inter-domain protocol. The result of this computation is a forwarding table where each prefix is associated with a neighboring node as the *next hop*.

The route computation and update time at any node heavily depends on the number of prefixes for which the next hop information needs to be changed. In turn, the number of prefixes to be updated not only depends on the location of the failure, but also on the distribution of prefixes to the next hops in the forwarding table. Indeed, the closer the failure occurs to a node, the larger will be the number of prefixes affected. Similarly, if a large number of prefixes share the same next hop node, a change in the topology close to that node will result in long route

update time for all nodes in the network.

Based on the per-node convergence time (CT), we define the “network convergence time” as the maximum value of CT among all the nodes in the network. This indicates the time at which all the nodes in the network have learned about the failure, updated their routing tables and have a consistent view of the network.

Service availability of a network depends on the convergence time. Providers attempt to increase the overall availability of their networks by reducing convergence time in order to speed up the recovery after a failure. As we described above, convergence time depends on (i) router technology used for failure detection, (ii) network topology, (iii) routing protocol configuration such as IGP weights and timers, and (iv) location of peering points with other networks that determines the distribution of network prefixes among egress nodes. Clearly, it is not possible to look at a subset of the above mentioned factors to derive the service availability of a network.

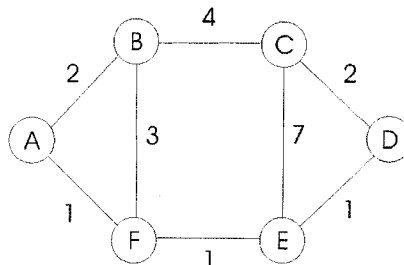


Figure 3.1: Example Network Topology for IGP Convergence Illustration

Consider the network illustrated in Figure 3.1. The number on each link indicates the IGP link weight. Let node A be the traffic source and node D be the traffic sink. Consider the failure of link E – D. We assume that the nodes and the links have similar characteristics with a *detection time* of 500ms, a *notification time* between neighboring nodes of 100ms, and a node *route computation*

and update time of 400ms (except for node C for which we assume this to be 100ms due to the fact that it does not have to change its route to reach D after the failure of link E – D). Consider the disruption observed for the traffic sent from node A to node D following the failure event at time $t_0 = 0$. Table 3.1 shows the routing events, changes in the traffic forwarding path and service availability from node A to node D. In this example, we assume that a node notifies neighboring nodes only after it has completed the update of its own routing table. However, it is easy to verify that if the updates are sent before updating the routing table, the example yields similar results.

Time	Event	Forwarding Path (A – D)	Service from A to D	Notes
0s	Failure of link E – D	A-F-E-D	NO	
0.5s	D,E: failure detection	A-F-E-D	NO	
0.9s	D,E: route update	A-F-E-C-D	YES	Forwarding restored
1.0s	C,F: notified of failure	A-F-E-C-D	YES	
1.1s	C: route update	A-F-E-C-D	YES	Path C-D not affected
1.2s	B: notified of failure	A-F-E-C-D	YES	
1.4s	F: route update	A-F-B-F-...	NO	Routing loop B – F
1.5s	A: notified of failure	A-F-B-F-...	NO	Routing loop B – F
1.6s	B: route update	A-F-B-C-D	YES	
1.9s	A: route update	A-B-C-D	YES	Network Convergence

Table 3.1: Summary of Routing Events (with a Route Update Time of 400ms). Network Convergence Time = 1.9s; Service Disruption Time = 1.1s

Interestingly, as the message update propagates across the network, the forwarding path from node A to node D changes four times. Some of these intermediate paths are valid thus restoring service between A and D, while some are not, causing packet drops (or *traffic black-hole*) and routing loops. For example, packets are dropped until node E has computed a new forwarding path to reach D and routing loops occur when nodes B and F have conflicting forwarding information.

Based on the example we can derive some initial observations:

- The network convergence time provides a rough upper bound on service disruption time (i.e.

the time for which service is not available). The service is not available when the packets cannot reach their destinations due to the lack of forwarding information. Given that traffic forwarding may resume even if all the nodes in the network have not updated their routing table, there may be a significant lack of correlation between network convergence and service availability. For example, in our illustration above, the cumulative time for which the service is not available amounts to 1.1s while the network takes 1.9s to converge. Therefore, network convergence time does not capture the user experience in terms of network service availability.

- The network topology by itself does not help in understanding the service availability of a network. Additional information such as the location of peering points and size of routing tables needs to be considered in characterizing service availability. The time taken to update the routing table depends on the number of prefixes that are affected. Table 3.2 shows the impact of varying the number of prefixes that have D as the next hop node on convergence time and service availability. Even with the same topology and identical failure scenario, increasing route update time may lead to significant differences in service availability.

Routing update time	100ms	200ms	300ms	400ms	500ms	1000ms
Convergence time	1.0s	1.3s	1.6s	1.9s	2.2s	3.7s
Time - service not available	0.8s	0.9s	1.0s	1.1s	1.2s	1.7s

Table 3.2: Convergence Time Depends on Routing Update Time in Nodes

In the following section, we introduce a new topology metric that exploits the knowledge of routing dynamics to define and compare the *goodness* of topologies. We have developed an algorithm to compute the cumulative time for which service is not available (i.e. the service disruption time as presented in Table 3.1). The details of this algorithm are presented in Table 3.3.

STEP 1: Initialize the *Service Disruption Time*, $\phi_l(x, y)$, for the path from x to y due to the failure of the link l to 0 i.e. $\phi_l(x, y) = 0$. If the original path from x to y does not contain link l then QUIT.

STEP 2: Find the convergence time (CT) for each node and list the nodes in the increasing order of convergence time. Let the convergence time of the first node in the list be CT^1 , second node be CT^2 and so on. In general, the convergence time for the n^{th} node in the list is CT^n . Set the *current node*, k , (which is the node number on the sorted list) to 0.

STEP 3: Increment k to 1. Set $\phi_l(x, y) = CT^1$. At the time instant CT^1 after the failure event, find the path that a packet from the source node, x , follows to reach the destination node, y , taking into account that the first node in the list has converged and others have not. If the path has a routing loop or black-hole, then set $previousDisruption = true$ else set $previousDisruption = false$

STEP 4: Increment k by 1. At the time instant CT^k after the failure event, find the path that a packet from the source node, x , follows to reach the destination node, y , taking into account that the intermediate nodes might have converged or not.

STEP 5: If the path that the packet follows does not contain the failed link l and has no routing loop, then set $previousDisruption = false$. Go to Step 7. Else go to Step 6.

STEP 6: If the path that the packet follows contains the failed link l or has a routing loop, then the path from x to y is still disrupted. If $previousDisruption = false$, then do not update the *Service Disruption Time* but set $previousDisruption = true$ else if $previousDisruption = true$, then update the *Service Disruption Time* in the k^{th} iteration as, $\phi_l(x, y) = \phi_l(x, y) + CT^k - CT^{k-1}$

STEP 7: If there are more nodes in the list then go to STEP 4. Else QUIT.

Table 3.3: Algorithm to Calculate Service Disruption Time from Node x to Node y due to a Single Link Failure

3.2 Modeling Service Availability of IP-Networks

3.2.1 Service Availability

We define service availability in three perspectives:

- *Ingress node perspective*: This is a measure of the network performance as seen by a particular ingress node where the traffic enters the network. It provides an insight about the level of service to expect when a customer connects to different ingress nodes of a network. It also helps an ISP to ensure that it can meet SLA specifications for customers connecting to different ingress nodes.
- *Link perspective*: The performance of a network should not heavily depend on the reliability of a few links in the network. In other words, the network should not have critical links whose failure results in serious performance degradation. Service availability from a link perspective evaluates the importance of various links for network performance.
- *Network perspective*: This measures the performance of the entire network from the perspective of an ISP. This is useful in designing a network to achieve high end-to-end performance.

We propose three metrics that capture the service availability of a network due to single link failures:

- *Service Disruption Time (SD time)*: This represents the time for which service between a particular source-destination (O-D) pair is disrupted. From the point of view of an ingress node, it indicates the loss in connectivity with all/some parts of the Internet due to a link failure.

- *Traffic Disruption (TD)*: This metric captures the total traffic disrupted between a particular source and destination node due to single link failures. It is the product of traffic rate between the O-D pair along the failed link and the service disruption time. For an ISP, TD is more important than SD time because of the fact that customers are compensated for the amount of traffic lost, irrespective of the duration for which the service is disrupted.
- *Number of Delay Parameter Violations (DV)*: In a well-designed network, the end-to-end delay along alternate paths is generally higher than the original path. We assume here that the delay parameter is representative of the maximum end-to-end delay that can be tolerated by delay sensitive traffic in the network. If the end-to-end delay along the alternate path exceeds the delay parameter specification, then there is a delay parameter violation. DV measures the number of such delay parameter violations from the perspective of an ingress node due to single link failures.

Note that SD time and TD capture the effect of a link failure during service disruption, while DV is a post-convergence effect. However, given complete network topology specifications (i.e, nodes, links, connectivity, link weights, link delays and delay parameter), BGP prefix distribution and traffic rate in the network, all the metrics can be pre-computed and used to characterize the end-to-end performance of a network.

3.2.2 Goodness Factors

To use the notion of service availability in characterizing network topologies, it is necessary to capture it using a quantitatively measure. Intuitively, this measure should yield a numerical value that can estimate the end-to-end performance of a network and help in differentiating various

topologies. In order to accomplish this, we define a set of *goodness factors* based on different perspectives of service availability. These factors can be defined differently depending on the specific scenario (for example, it could be driven by the cost involved, SLA specifications etc.). The rest of the chapter presents one specific example of such definitions.

We first introduce the notations that we use in the rest of this section. Consider a network with N nodes and M links. Let Γ and Λ represent the set of nodes and links, respectively. For any node, i , there are $(N - 1)$ different destinations in the network and hence $(N - 1)$ different paths with node i as the ingress node. For the failure of link j , all/some/none of $(N - 1)$ paths could be affected. Let Q_{ihj} and T_{ihj} denote the SD time and TD for the source-destination pair $i - h$, due to the failure of link j . Similarly, let S_{ij} denote the number of delay parameter violations from ingress node i along all $(N - 1)$ paths due to the failure of link j .

Goodness from ingress node perspective

Typically, many customers are connected to a network at an ingress node. TD represents the total traffic affected for all the customers connected to the ingress node due to a link failure and does not provide valuable information to individual customers. Instead we use SD time and DV to measure the goodness of a network from an ingress node perspective. We define,

$$GI_i = f(Q_i, S_i) \quad (3.1)$$

where, GI_i is the goodness factor of the network from the perspective of ingress node i . Q_i is the average SD time for node i across all $(N - 1)$ paths and M possible single link failures:

$$Q_i = \frac{1}{M(N-1)} \sum_{\forall h \in \Gamma, h \neq i, \forall j \in \Lambda} Q_{ihj} \quad (3.2)$$

Similarly, S_i is the average of the number of delay parameter violations from node i , to all other nodes in the network due to various single link failures:

$$S_i = \frac{1}{M} \sum_{\forall j \in \Lambda} S_{ij} \quad (3.3)$$

The function f depends on SLA specifications between ISP and customers. For simplicity, we assume that goodness is inversely proportional to various metrics. Hence,

$$GI_i = \frac{C}{(Q_i)^q (S_i)^p} \quad (3.4)$$

where C is a constant. The exponents q and p are SLA dependent. In our simulations, we assume the exponent values to be 1.

Goodness from link perspective

The impact of a link failure on the network performance directly depends on total TD and total DV (i.e. sum of DV for all nodes) due to the failure. High values of these metrics for a link implies that the link is critical to network performance. We define the goodness factor from a link perspective based on similar assumptions on f as in Equation 3.4:

$$GL_j = \frac{C}{(T_j)^t (S_j)^p} \quad (3.5)$$

where, GL_j is the goodness from the perspective of link j . T_j is the total TD due to the failure of

link j :

$$T_j = \sum_{\forall i, h \in \Gamma, i \neq h} T_{ihj} \quad (3.6)$$

Similarly, S_j is the total number of delay parameter violations due to the failure of link j i.e.,

$$S_j = \sum_{\forall i \in \Gamma} S_{ij} \quad (3.7)$$

To find critical links in a network it is more useful to compute the *badness* of a link rather than its goodness. Hence we define the badness of link j in the network as the inverse of its goodness:

$$BL_j = C(T_j)^t(S_j)^p \quad (3.8)$$

Goodness from network perspective

We define the goodness of the entire network as the weighted sum of link goodness factors for various links in the network:

$$GN = \sum_{\forall j \in \Lambda} \frac{C_j}{(T_j)^t(S_j)^p} \quad (3.9)$$

where, C_j represents the weight, i.e., the importance of the link j to the goodness of the entire network. For example, a link that has a low probability of failure can be assigned a higher weight (i.e., C_j) compared to a link that has a high failure probability. We will discuss this further in Section 3.5. However, in the results presented in the next two sections, we assume that the C_j values for all the links are equal to 1.

3.3 Results and Discussion

In this section, we use the metrics proposed in Section 3.2 to quantify the impact of link failures on service availability of a network. First, we illustrate how the algorithm presented in Section 3.1 is used to compute SD time, TD time and DV for different classes of network topologies. Then, we examine how traditional graph properties such as out-degree, network diameter, increase in tree depth, and disconnecting sets correlate to QoS offered by the network. We will show the effectiveness of goodness factors in differentiating various network topologies by capturing the routing dynamics that affect traffic forwarding performance. Lastly, we will discuss the implication of the graphs and how goodness factors can be used in evaluating “quality” of connectivity and network design applications.

3.3.1 Simulation Setup

We built a java-based simulator to emulate intra-domain routing dynamics in the presence of link failures and to implement the algorithm presented in Table 3.3. The inputs to the simulator are complete network topology specifications, BGP prefix distribution, and traffic load along different links in the network. In our simulations, each node in the network topology is mapped to a geographic location. The delays for individual links are then calculated based on the geographical distance between the nodes that the link connects. We categorize the nodes in a network as large, medium, and small, depending on the amount of traffic they generate. We consider 20% of the nodes as large nodes, 30% as medium nodes and the rest as small nodes. The classification of nodes into large, medium, and small nodes is based on the analysis of PoP (Point of Presence) level traffic data from a tier-1 ISP backbone network. The traffic matrix for all the networks are generated using

this model. We distribute BGP prefixes proportional to the traffic between nodes. Large traffic flow from a source node to destination node implies that the source node reaches a large number of prefixes in the Internet through the destination node. Our results with different assumptions for prefix distributions yielded interesting results. We will discuss this further in Section 3.4.

Based on these inputs, the simulator runs Dijkstra's SPF algorithm to find the shortest path from every node to all other nodes in the network. It then simulates single link failures and executes Dijkstra's SPF algorithm again, to find new paths in the network. The SD time for various O-D pairs are calculated based on the algorithm in Section 3.1. TD is then calculated using SD time and network traffic distributions between different source-destination pairs. DV is determined using the link delay values and delay parameter specification. We assume that the maximum tolerable delay to be 100 ms in our simulations.

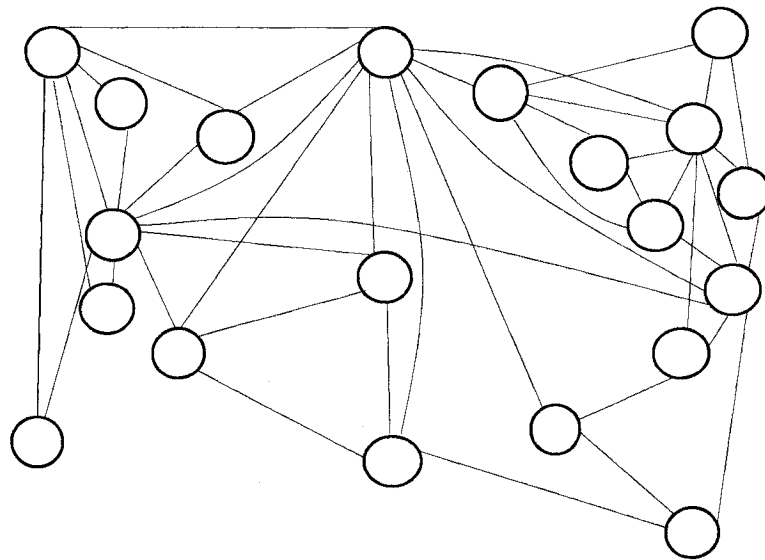


Figure 3.2: ISP-A topology

We consider the following two sets of topologies.

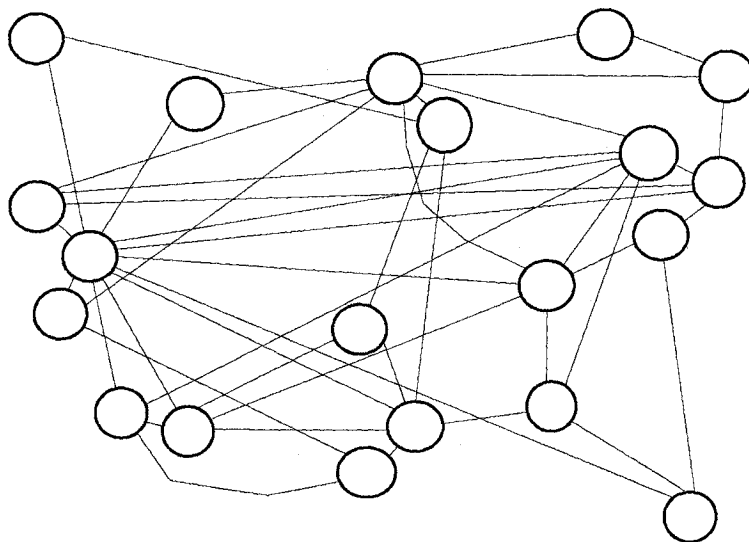


Figure 3.3: ISP-B topology

Set A: The first set of topologies represent standard topologies which includes ring, full-mesh, and two PoP-level tier-1 ISP topologies (ISP-A and ISP-B in Figure 3.2 and Figure 3.3). We use these topologies to show that the metric values calculated from our simulations are intuitively correct. All the topologies considered in the first set have 20 nodes in their networks. ISP-A and ISP-B have 44 links each while the ring and mesh topologies have 20 and 190 links respectively.

Set B: For the second set of topologies, we consider 10 topologies which are “similar” in terms of their intrinsic properties like average out-degree and network diameter. The diameter refers to the maximum depth of all routing trees in the network. One of the topologies considered here is ISP-A (Figure 3.2) from *Set A*. The other 9 topologies were generated by changing link connectivity in ISP-A. Each topology was generated using the following procedure:

- Randomly delete x links from ISP-A topology. We consider $10 \leq x \leq 15$ in our simulations.
- Randomly add x links back into the topology while honoring the following rules: (i) The

minimum out-degree of any node in the network is 2. (ii) The resulting topology is connected, i.e., a spanning tree for the resulting topology has 19 links.

All the topologies considered in *Set B* have 20 nodes and 44 links with an average out-degree of 4.4 per node. We found that the network diameter for the topologies lie in the range of 4-6.

In all cases, we assign equal weights to all the links in a network, thus making it a minimum-hop routing scheme. In reality, different IGP link weight assignment schemes yield different metric values and we will further explore this in Section 3.4. To achieve a fair comparison, we consider the same traffic distributions in all the topologies. Traffic rate to/from large, medium, and small nodes remain the same in all the topologies. Finally, we want to point out that the values of goodness factors are normalized to 1 in all the results presented.

3.3.2 Service Availability for Known Topologies

In this section, we use the metrics proposed for service availability to study the performance of four known topologies (*Set A*). Intuitively, a mesh topology should perform the best among all the networks with the same number of nodes while a ring topology should perform the worst. Figure 3.4 shows the cumulative distribution (CDF) of the total TD in each of the four networks for various single link failure scenarios. Every link in the mesh topology carries traffic between a single O-D pair, thus resulting in small values of total TD for single link failures, while every link in a ring topology carries traffic between multiple O-D pairs, resulting in very high values of total TD. These two topologies are the extreme cases for any topology with the same number of nodes. TD values for other topologies, like ISP-A and ISP-B, lie in-between these extreme values.

Figure 3.5 shows the CDF of total DV for single link failures. There are no delay para-

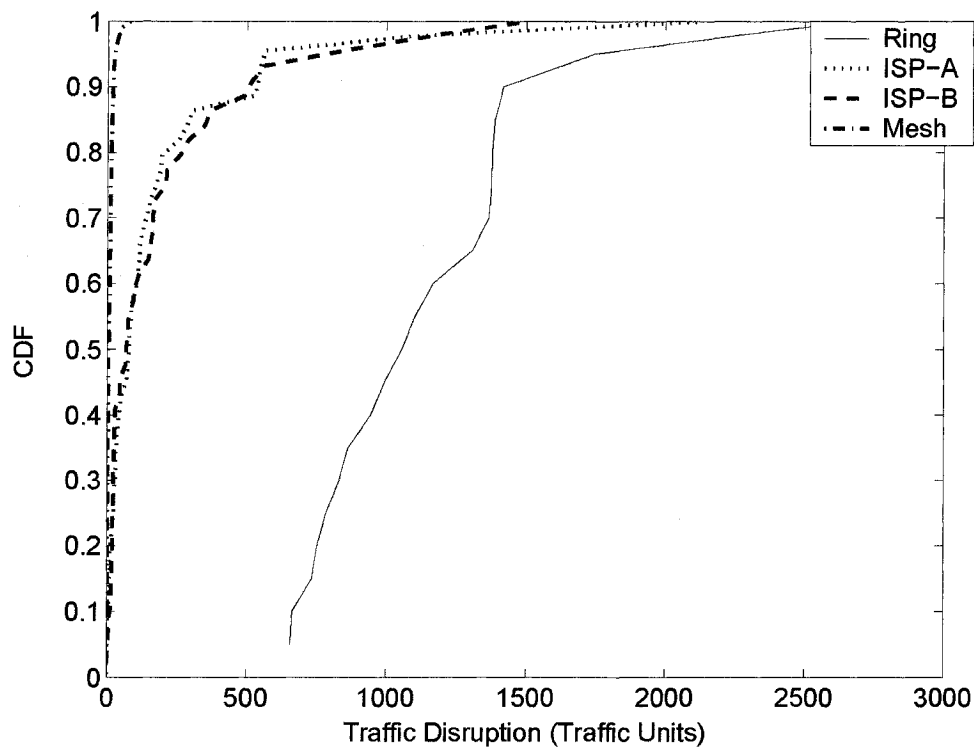


Figure 3.4: CDF of Traffic Disruption due to Single Link Failures

meter violations for the mesh topology after single link failures while ring topology is significantly worse compared to other topologies for obvious reasons.

3.3.3 Goodness Factors vs. Static Graph Properties

The following case studies illustrate the limitations of static graph properties (like out-degree distributions or network diameter) to evaluate the performance of a network. This provides the motivation to characterize network graphs using *goodness factors*, which are directly derived based on estimated performance, i.e., service availability. All the following results are for topologies in *Set B*.

In the past, out-degree of a node has been used as a metric for characterizing a source

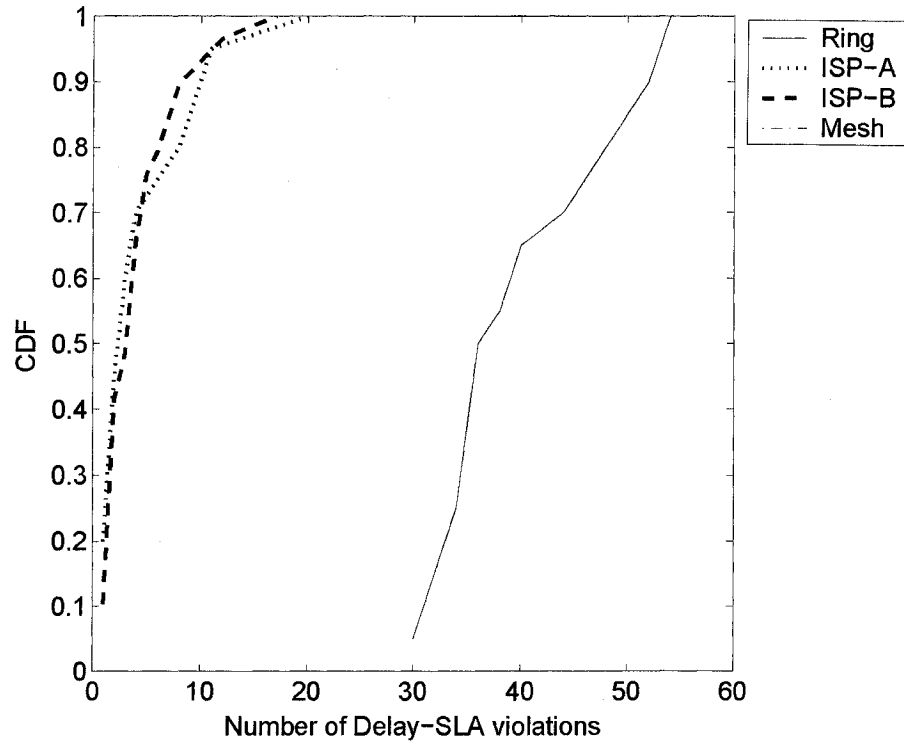


Figure 3.5: CDF of Number of Delay Parameter Violations due to Single Link Failures

node [94]. Figure 3.6 shows the maximum, minimum, and average goodness factor values for various source nodes as a function of their out-degree. The goodness factors for source nodes with the same out-degree are considerably different. Also, higher out-degree does not always result in higher goodness factor values. The main problem in using out-degree to estimate the performance of a source node is that it does not capture the effect of various network characteristics such as BGP prefix distribution and link weight assignment scheme which have a significant impact on source node performance. To show this, we repeat the simulations by distributing BGP prefixes such that all the prefixes have a single exit point in the network (we refer to this as *extreme prefix distribution*). None of the other topology properties were altered. The graph in Figure 3.7 shows that changing BGP prefix distribution in the network changes the source goodness factor for various nodes. This

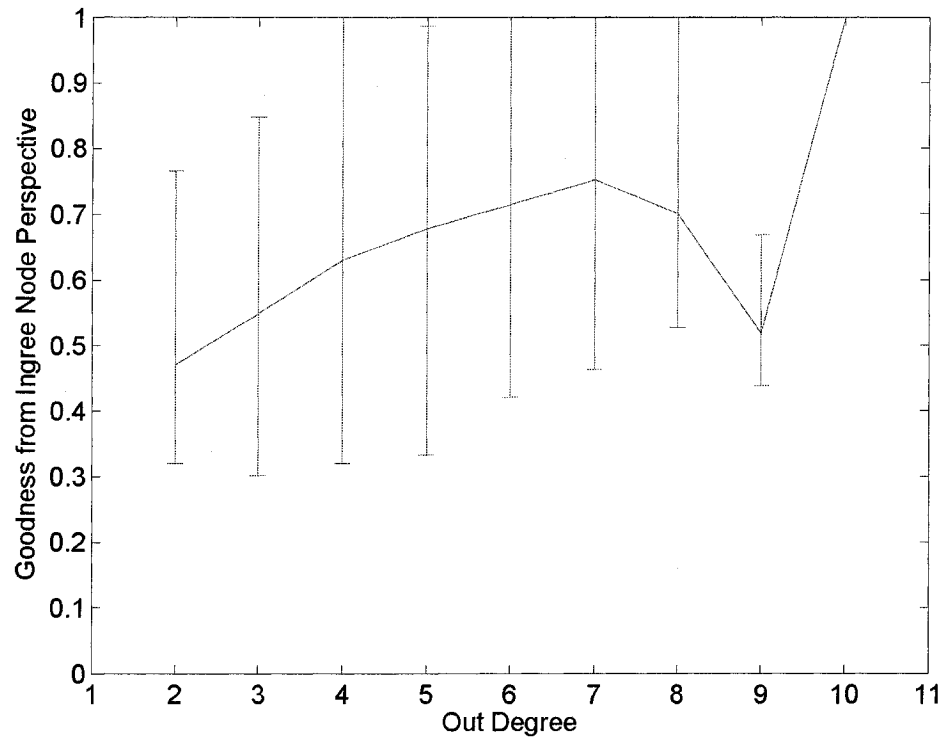


Figure 3.6: Goodness Factor from an Ingress Node Perspective Vs Out-Degree in Various Topologies with Normal Prefix Distribution

implies that out-degree is not an appropriate metric to measure the performance of source nodes.

Network diameter gives an estimate of the maximum convergence time in the network. One would expect that a network with a small diameter would exhibit small convergence time and hence offer better service availability. The top graph of Figure 3.8 shows the network goodness factor for various topologies against network diameter. However, our results show that topologies with smaller diameter do not always result in higher goodness factors. Also, topologies with same diameter exhibit different network goodness factor values. Like out-degree, network diameter does not account for several characteristics of a network and hence is not a good metric for predicting its performance.

Another metric related to network diameter is the increase in tree depth due to single link

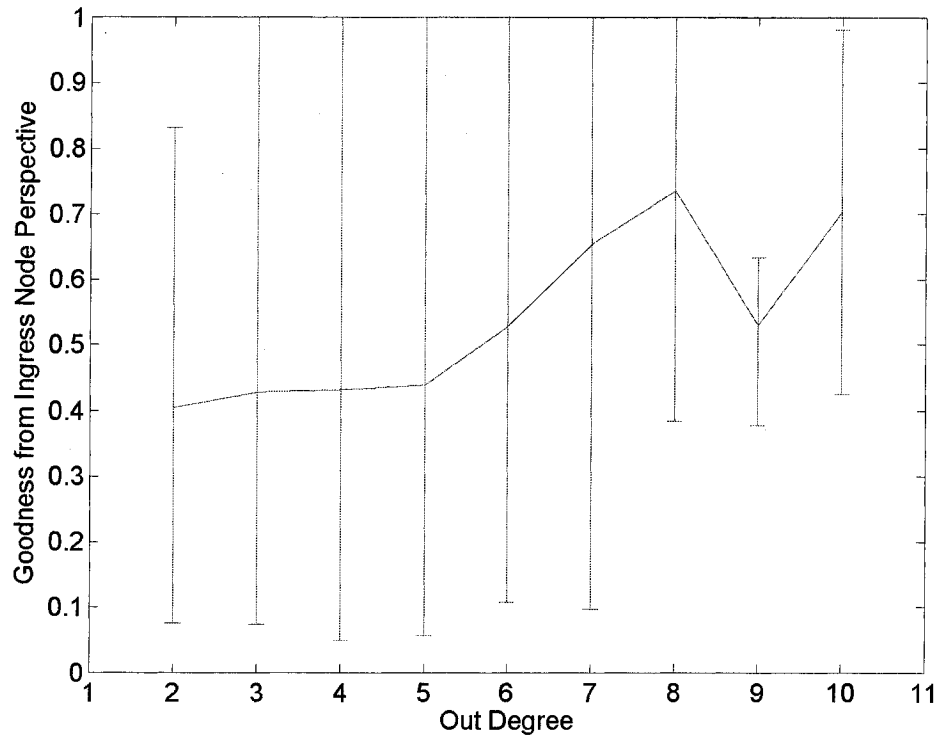


Figure 3.7: Goodness Factor from an Ingress Node Perspective Vs Out-Degree in Various Topologies with Extreme Prefix Distribution

failures. In a well-designed network, the end-to-end delay depends directly on the depth of the routing tree from source to destination. Hence DV depends on the increase in tree depth after a failure. Typically, lower values of increase in tree depth due to single link failures should result in better topologies. From our simulations we find that this is not true. The bottom graph in Figure 3.8 shows the average increase in tree depth for various topologies against the network goodness factors. Even though increase in tree depth can be used to roughly estimate DV, we find that it is not a good metric for predicting network goodness.

Another important traditional metric used to estimate network performance is the number of *disconnecting sets* in a network [78]. Disconnecting set of a topology is defined as a set of links, whose cardinality is less than the minimum node out-degree in the network and whose removal

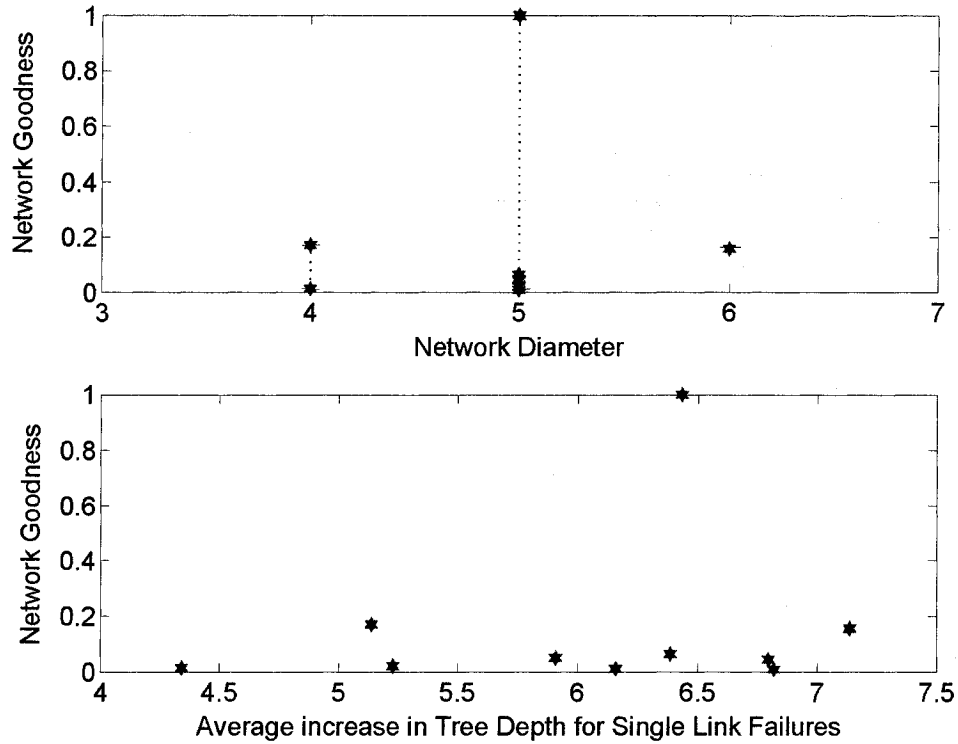


Figure 3.8: Top Graph shows the Variation of Network Goodness Factor with Network Diameter while the Bottom Graph shows the Variation of Network Goodness with Average Increase in Tree Depth due to Single Link Failures

disconnects the topology into two or more smaller topologies [10]. The topologies that we consider here, have nodes with out-degree of two. Hence, by the definition above, the cardinality of the disconnecting set should not be more than 1. Since this definition yields a null set for all the topologies, we define a disconnecting set, D , with the following modifications:

- D partitions a topology into 2 parts with each part having at least 2 nodes.
- Exclude Supersets: If D_1, D_2, \dots, D_n are n disconnecting sets for a network, then $D_i \not\subseteq D_j$ when $i \neq j$.

There are numerous solutions to the above definition of D , but we consider only those sets for which the cardinality of D is less than 3, i.e, sets with at most 3 links removed. Disconnecting

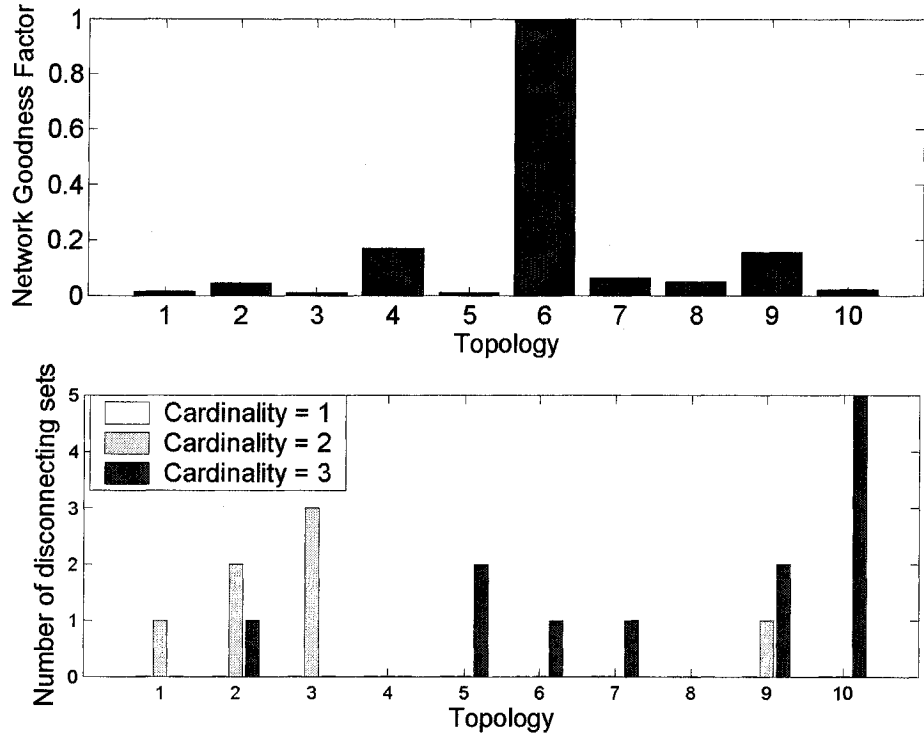


Figure 3.9: Top graph shows the Network Goodness Factors while the Bottom Graph shows the Disconnecting Sets for Different Topologies

sets determine links that have a big impact on network performance when they fail. The failure of links in the disconnecting set could potentially result in high TD and DV. As the cardinality of the disconnecting set increases, the criticality of the links in the set decrease. For example, a single link connecting two sub-topologies is more critical than two links connecting two sub-topologies. The bottom graph of Figure 3.9 shows the number of disconnecting sets with different cardinalities for various topologies. The top graph in the same figure shows the network goodness factor for various topologies. Even though topology-4 and topology-8 have no disconnecting sets, we see that they do not result in the best network goodness factor values. Hence, similar to the other traditional metrics, disconnecting sets are not good metrics to capture service availability of a network.

The top graph in Figure 3.9 shows that even though the topologies are similar, the end-

to-end performance based on service availability is significantly different. ISPs can take advantage of this network differentiation to design their networks to provide high service availability to customers. It is also helpful in estimating the cost of compensating the customers for SLA violations of the network.

3.4 Applications of Goodness Factors

This section explores the various applications of the proposed *goodness factors*, and present some numerical results using topologies in *Set B* in our case study.

3.4.1 Goodness Factors from Ingress-node and Link perspectives

Given an ISP network, the top graph in Figure 3.10 shows the performance that a customer can expect when connected to different ingress nodes. This helps a customer to choose an ideal location to get connected to the network. It also helps the ISP to ensure that it can meet the SLA specifications for customers at a given source node.

The bottom graph in Figure 3.10 shows badness factors of the links (inverse of goodness, Eq. 9) for one specific network topology (Topology-6). The graph shows that there are a handful of critical links (e.g., Link-4 and Link-14) that have a big impact on network performance when they fail. The rest of the link failures only result in minor service degradation. The ability of the badness factors to clearly distinguish the critical links in the network is extremely useful for capacity planning and traffic engineering purposes of an ISP. For example, this information allows the ISP to make an informed decision about bringing down a link for maintenance to minimize the impact on network performance. ISPs can also re-negotiate peering relationships to divert traffic away from

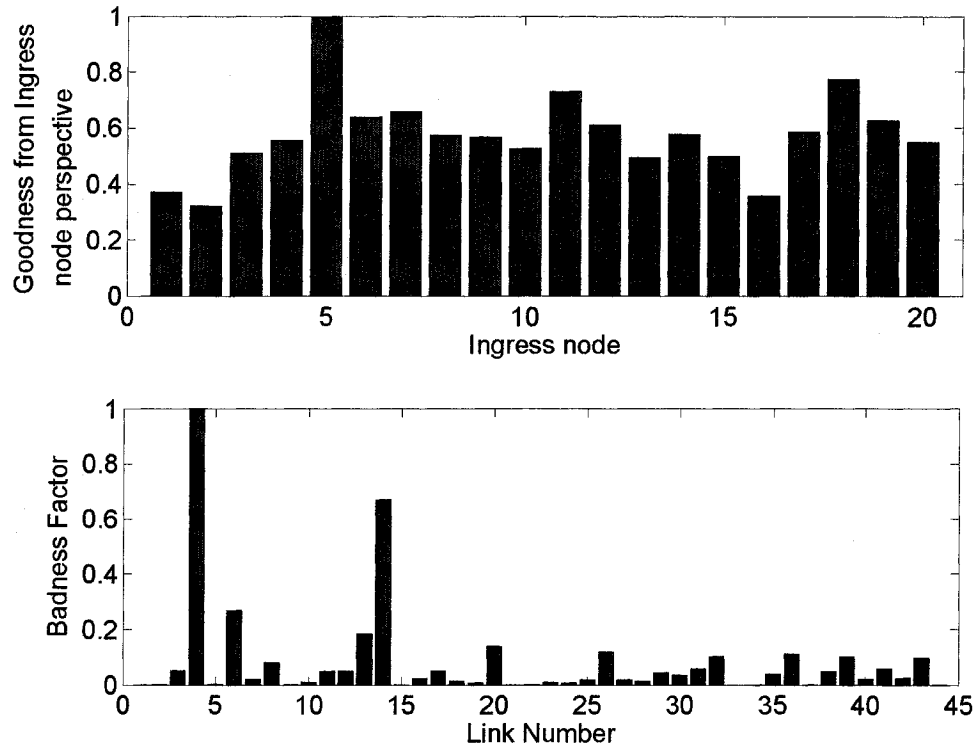


Figure 3.10: Top Graph shows the Goodness Factors from the Perspective of Various Ingress Nodes in Topology-6 while the Bottom Graph Shows the Link Badness Factor for Various Links in Topology-6

critical links. With a better picture of how the failure of individual links can impact performance, the ISP can better estimate whether certain SLAs can be met.

3.4.2 Network-wide Goodness Factors

Our results from the previous sections have established the importance of characterizing topologies based on meaningful performance metrics such as service availability. The proposed *network-wide goodness* aims to capture the impact of routing dynamics on service degradation measured in terms of TD and DV across all source-destination pairs. This goodness metric forms the basis for optimizing network design decisions. We explore three such applications in the following

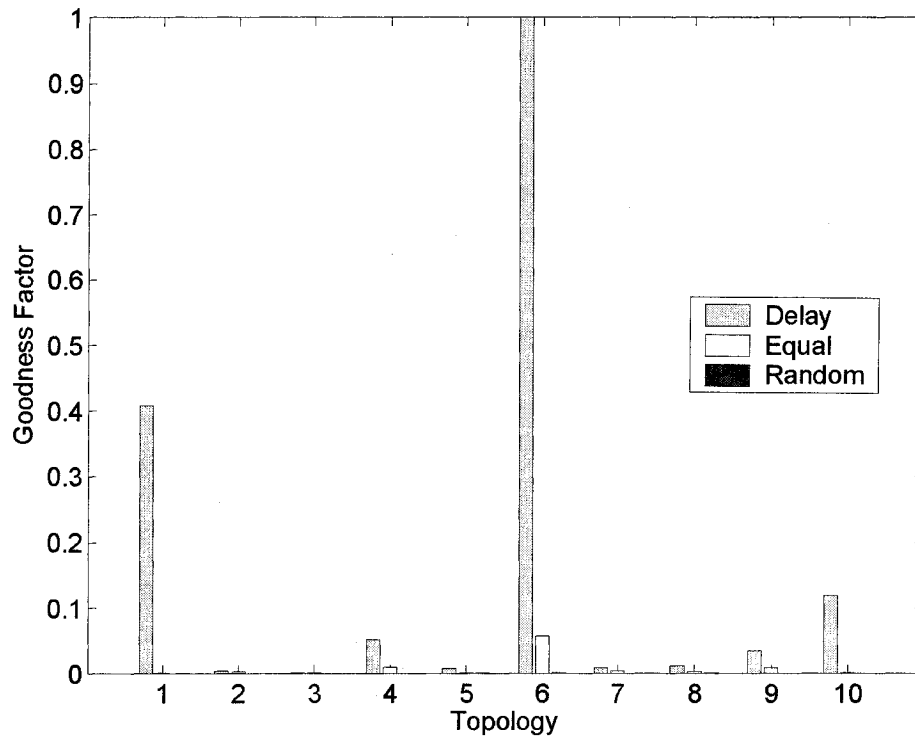


Figure 3.11: Network Goodness for Topologies with Different Link Weight Assignment Scheme

discussion.

IGP Link Weight Assignment: IGP link weights determine the routing trees and hence can significantly influence the service availability of a network. We explore how goodness factors can be used to “evaluate” different link weight assignment schemes in Figure 3.11. We consider the same ten topologies in *Set B* but with three different link weight assignments: (i) weights proportional to link delays, (ii) equal weights, and (iii) random weights. Figure 3.11 shows that the same topology can behave quite differently when different link weights are used. Therefore goodness factors can be used as an optimization metric for selecting link weights to provide the best service availability.

BGP peering points: As discussed in Section 3.3.3, the location of BGP peering points determine the prefix distribution size at different nodes, and hence also influences service availability. In

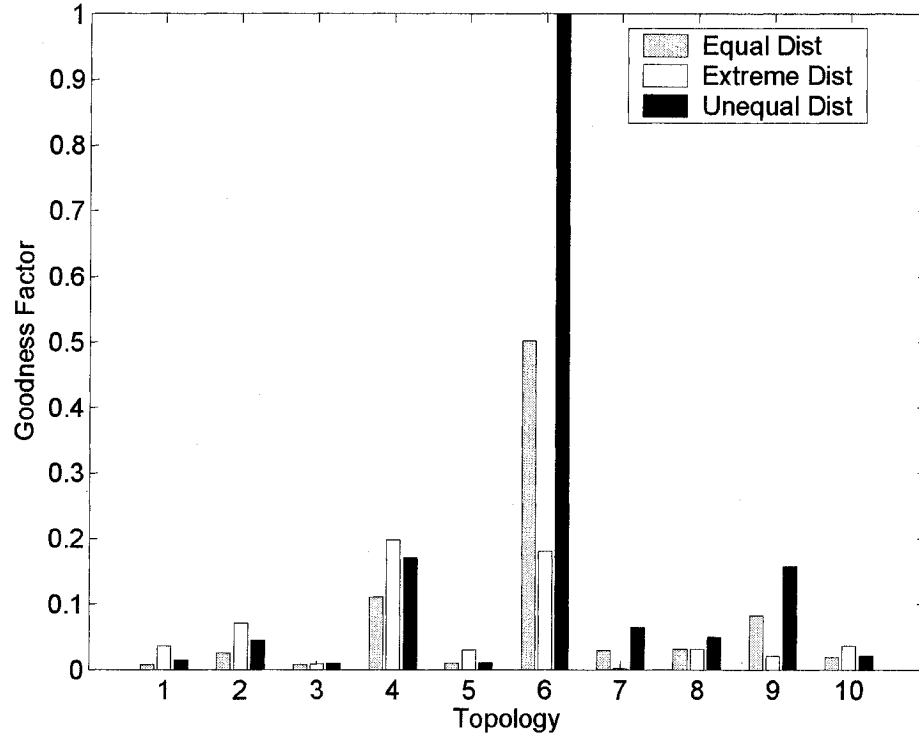


Figure 3.12: Network Goodness for Topologies with Different Prefix Distribution Schemes

Figure 3.12, we compare network goodness for topologies with three ways of distributing BGP prefixes across different nodes: (i) equal distribution, (ii) extreme distribution where all the BGP prefixes are located at one exit point, and (iii) typical prefix distribution observed in a tier-1 ISP. Depending on the network topology, different locations of BGP peering points result in different network goodness in terms of overall service availability. This illustration shows that goodness factors can be a useful metric in determining “ideal” BGP peering points that result in the most desired network performance.

Network upgrade: To cope with customer demands and meet SLAs, an ISP may have to schedule network upgrade to introduce a new node or link into its network. Deciding *where* in the existing network to connect this new node/link to, becomes a design challenge. In this case study, we

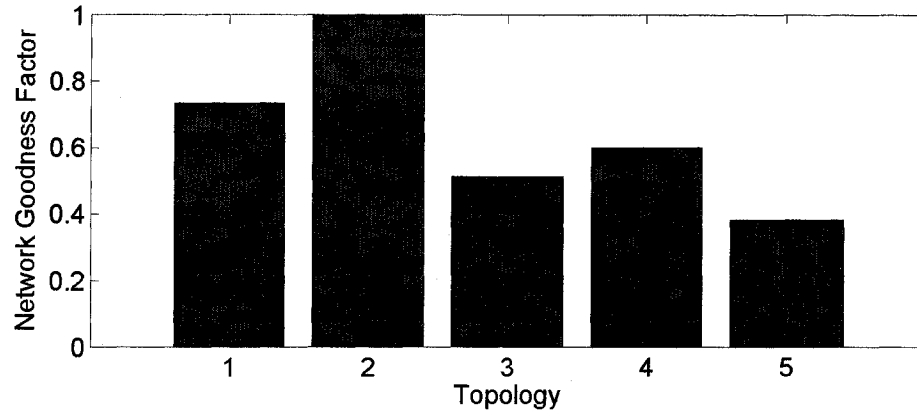


Figure 3.13: Adding a New Node into the Network - Network Goodness for Various Solutions

consider adding a new node with 3 links into ISP-A network (Figure 3.2). Figure 3.13 shows five possible solutions and compares the resulting network-wide goodness of the new network. Solution-2 clearly shows the best network goodness in terms of the offered service availability in the presence of failures.

3.5 Summary and Extensions

In this chapter, we examine the importance of incorporating network dynamics in characterizing topologies. Our simulations show that traditional metrics like out-degree, network diameter, and disconnecting sets that disregard network dynamics do not effectively capture the performance of a network. Hence it calls for a new approach to characterize topologies. To fill this void, we proposed a novel methodology based on the concept of service availability and demonstrated its effectiveness using simulations.

To the best of our knowledge, this is the first work to consider network dynamics in characterizing topologies. The approach is the first step in the right direction and is appealing to

both ISPs and customers alike. We have identified numerous applications for goodness factors in capacity planning, network design and upgrade. We will discuss the network upgrade problem in detail in Chapter 5.

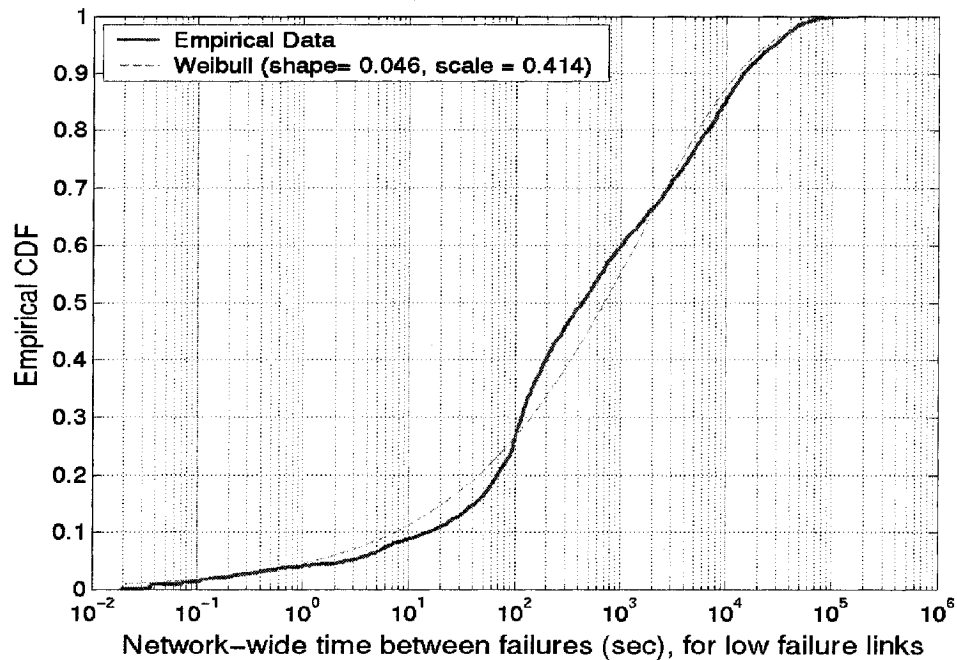


Figure 3.14: Network-Wide Time Between Failures on Low Failure Links and the Approximation by Weibull Distribution

In this chapter, we have characterized various networks based on their service availability assuming that all link failures in backbone networks are equally likely with a uniform probability. In fact, recent studies (like [66]) have shown that different links exhibit different failure characteristics. Some links fail more often while others do not fail as frequently. In addition, some link failures last for a longer duration when compared to some others. In particular, the authors in [66] empirically show that in a tier-1 ISP North American backbone network:

- A majority (70%) of the unplanned failure events are isolated, i.e., only affect a single link at

a time, and hence can be modeled as independent link failures.

- Links are highly heterogeneous, i.e., some links fail significantly more often than others. This motivates the classification of the links into two categories - *high-frequency* and *low-frequency* links, and model them separately. Within each class, the number of failures, $n(l)$, for link l follows roughly a power-law, i.e., $n(l) \propto l^{-k}$, where k is found to be -0.73 for high-frequency links and -1.35 for low-frequency links.
- The empirical cumulative distribution function (CDF) for time between any two failures can be approximated by a Weibull distribution. For example, Figure 3.14 shows the empirical CDF for the network-wide time between failures for low-frequency links. The Weibull parameters can be derived for each set of empirical data based on maximum-likelihood estimation, e.g., in this case, $\alpha = 0.046$ and $\beta = 0.414$. The cumulative distribution of the duration of failures observed over the same period show that most failures are transient (i.e., short-lived): 46% last less than a minute and 85% last less than ten minutes.

It is important to note that link failures in different networks can follow different distributions, and hence their goodness factors not only depend on the topology and operational network conditions, but also on the link failure model of the network.

Figure 3.15 shows the cumulative distribution function of the traffic disruption in the ISP-A network using the failure model observed in [66]. The entire simulation time was eight hours. During each simulation run, different links were associated with different failure probabilities that were generated from a Weibull distribution (as in [66]). Every simulation run resulted in a curve in Figure 3.15. We can see that changing association between failure probabilities and links results in very different distributions of traffic disruption implying that goodness factors depend

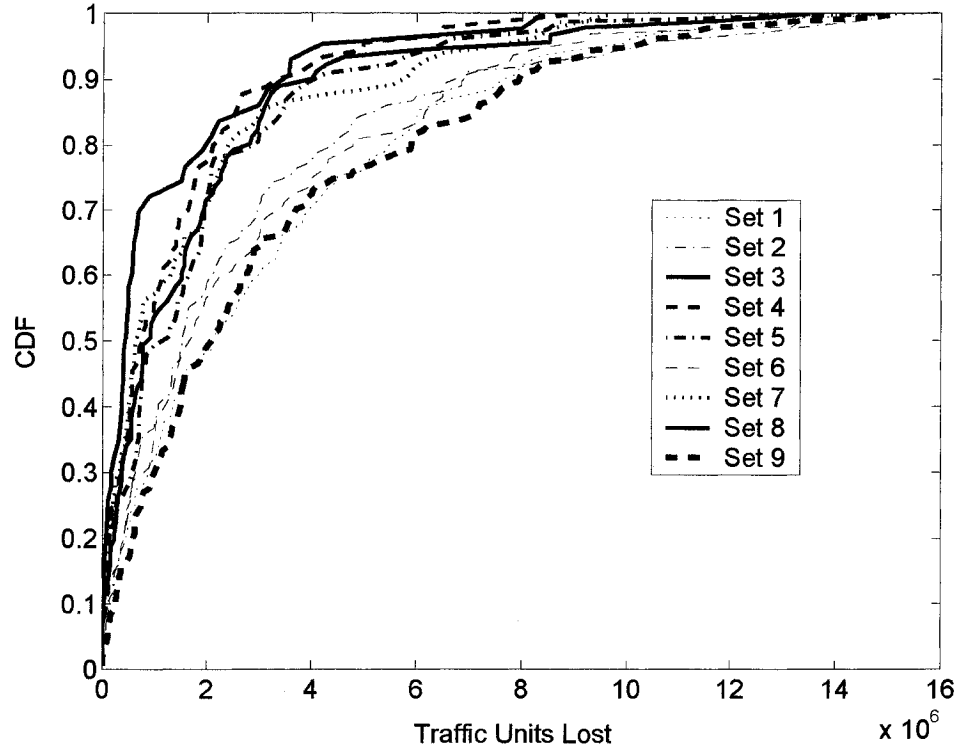


Figure 3.15: TD for Nine Different Associations of Links with Failure Probabilities

heavily on the correct association of failure probabilities with network links. Therefore, to adopt our service availability framework to different operational networks, one should first derive (or accurately estimate using techniques that are beyond the scope of this dissertation) the failure model for the particular network and then use it in our definition of goodness factors (Equation 3.9).

Chapter 4

Eliminating Routing Loops during IGP

Convergence - LISF

As we discussed in the previous chapter, the convergence time after a link failure can be viewed as a sum of three components - detection time, notification time, and route computation and update time. During the convergence period, routers may have inconsistent views of the network and therefore can cause forwarding loops [100]. While these loops last for only a short time and their effect is mitigated by the TTL field in IP datagrams, they can still overwhelm high capacity links and render them unusable. Therefore, it is desirable to avoid any forwarding loops even if they are only transient.

Several remedies for the transient looping problem have been suggested in the literature [100, 43, 41, 25, 26]¹ and an IETF working group has been addressing this issue [82]. Path locking with safe neighbors approach [100] categorizes routes into three types A, B, or C, and

¹Many other schemes have been proposed to deal with failures through fast local rerouting [99, 25, 19]. However, the focus of our work is on schemes specifically designed for loop avoidance during convergence after a network-wide link state update.

installs new routes for B and C types after a fixed configurable delay such that delay for type B is greater than delay for type C routes. While this approach decreases the likelihood of loops, it does not completely eliminate them. Moreover, it introduces additional delays in the installation of new routes, thus compounding the convergence delay. A loop-free path-finding algorithm proposed in [43] blocks a potential loop when it detects that a loop can be formed. To achieve this, a router first reports to all its neighbors that its distance to reach the destination is infinity, and then waits for those neighbors to acknowledge its message with their own distances and predecessor information before updating its successor in the forwarding table. Recently, similar methods have been proposed in [41, 25], where the forwarding table updates in the network are ordered such that a node updates its forwarding table only after all its neighbors that use the node to reach different destinations through the failed link update their forwarding tables. Although these schemes do avoid transient loops, they require additional messages to be exchanged among routers to enforce the ordering of the updates of forwarding tables, resulting in increased convergence delay.

In this chapter, we propose an alternate approach – *loopless interface-specific forwarding* (LISF) – that exploits the existence of one forwarding table per interface to avoid transient loops without requiring any changes to the existing link state routing mechanisms. When all the routers in a network have the same view of the network, there would not be a forwarding loop. Only in the presence of discrepancies in the views of different routers, a packet might get caught in a loop. In such a case, the packet would have arrived through an *unusual* interface of at least one of the routers involved in the loop. Therefore, a forwarding loop can be avoided if the packet were to be discarded in such a scenario rather than forwarded to the usual next hop. LISF does precisely that by selectively discarding packets that arrive through unusual interfaces. The key advantages of LISF

are that it avoids transient loops without increasing the convergence delay and without employing any additional mechanisms to synchronize the forwarding table updates in different nodes.

The rest of this chapter is structured as follows. In Section 4.1, we illustrate the problem of transient loops. Our LISF approach for avoiding forwarding loops and three possible implementations of it are described in Section 4.2. In Section 4.3, we prove that the proposed LISF methods prevent loops in case of symmetric single link failures. The results of our simulations evaluating the LISF methods are presented in Section 4.4. We finally present some conclusions in Section 4.5.

4.1 Transient Looping Problem and Existing Approaches

We now illustrate the occurrence of transient loops, discuss a recently proposed approach for avoiding them, and point out the need for an alternate approach.

We use an example to illustrate the problem of transient loops. Consider the topology shown in Fig. 4.1, where each directed link is labeled with its weight. For the purpose of illustration, let us assume that all the nodes have similar characteristics with a *failure detection time* of 50ms, a *failure notification time* between neighboring nodes of 100ms, and *route computation and update time* of 400ms at a node (100ms for nodes that are not affected by the failure).

Consider a scenario where link E–D fails at time 0s. We examine how this failure impacts the forwarding of packets from source node A to destination node D. Table 4.1 summarizes the routing events under the traditional OSPF and the corresponding changes in the packet's forwarding path from node A to node D. The resulting convergence delay (i.e., the total time for all the nodes in the network to converge after the failure) is 0.65s, and the service disruption time (i.e., the total time for which the service between A and D is disrupted due to the failure) is 0.55s. During the

interval between the forwarding table updates in nodes E and F (i.e., between 0.45s and 0.55s), both the nodes have a different view of the network, resulting in a forwarding loop.

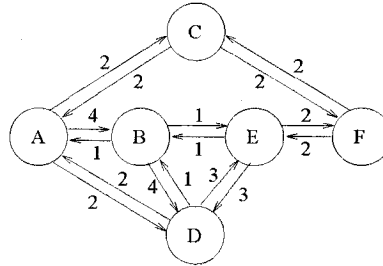


Figure 4.1: Topology-1 used for Illustration

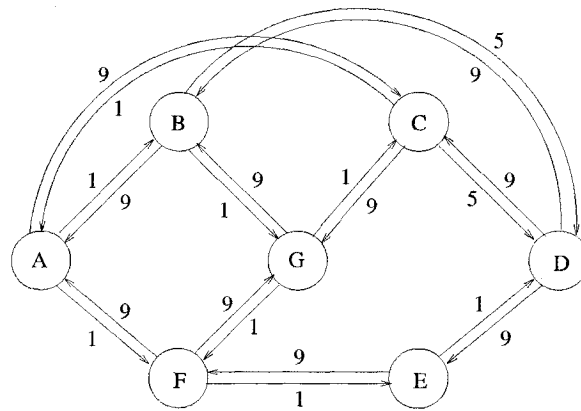


Figure 4.2: Topology-2 used for Illustration

To avoid transient loops during the convergence after a planned link failure or an unplanned failure of a protected link, a method was proposed in [41] that ensures ordered installation of forwarding table entries by exchanging messages between neighboring nodes. Here, we consider a similar approach (which we refer to as ORDR) for avoiding loops in case of unplanned failures (around 80% of all failures according to [66]) of unprotected links. The routing events under ORDR corresponding to Fig. 4.1 are shown in Table 4.1. Note that with this method, F updates its forward-

Time	OSPF		LISF	ORDR
	Events	A to D pkts	A to D pkts	Events
0s	Failure of link E-D	A-F-E-drop	A-F-E-drop	Failure of link E-D
0.05s	D,E: failure detected	A-F-E-drop	A-F-E-drop	D,E: failure detected
0.15s	C,F: failure notified	A-F-E-drop	A-F-E-drop	C,F: failure notified
0.25s	A,B: failure notified	A-F-E-drop	A-F-E-drop	A,B: failure notified
0.35s	B: route update	A-F-E-drop	A-F-E-drop	B: route update
0.45s	D,E: route update	A-F-E-F...(loop)	A-F-E-F-drop	
0.55s	C,F: route update	A-F-B-C-D	A-F-B-C-D	C: route update
0.65s	A: route update	A-B-C-D	A-B-C-D	A: route update
1.05s				D: route update
1.15s				F: route update
1.65s				E: route update

Table 4.1: Summary of Routing Events under OSPF, ORDR, and LISF

ing table 500 ms (assuming update time of 400 ms and the message propagation and processing time of 100 ms) after A updates its table. While this method avoids forwarding loops, its drawback is that it increases the network convergence delay. For the scenario discussed here, this method extends the convergence delay to 1.65s.

Our objective is to develop a scheme that combines the best features of OSPF and ORDR, i.e., low convergence delay and disruption time of OSPF and loop avoidance of ORDR. Such a scheme would ideally respond to the failure of E–D as shown in Table 4.1. Its behavior would be effectively similar to OSPF except that a packet is dropped if it would loop otherwise (as in the case of packets destined to D from F or E during the interval from 0.45s to 0.55s). Consequently, the ideal scheme would have the convergence delay of 0.65s and service disruption time of 0.45s while also avoiding forwarding loops. In the following sections, we present and evaluate a scheme, LISF, that closely approximates this ideal behavior.

4.2 Our Approach

Our approach for avoiding forwarding loops is based on the notion of *interface-specific forwarding*, where a packet's forwarding depends on the incoming interface in addition to the destination address. In this section, we first briefly explain interface-specific forwarding and argue how it can be exploited to avoid loops. We then present three methods of computing interface-specific forwarding table entries and illustrate the differences between these methods in terms of loop avoidance and computational complexity.

4.2.1 Interface-Specific Forwarding

A packet in an IP network is traditionally routed based on its destination address alone regardless of its source address or the incoming interface. Therefore, a single forwarding table that maps a destination address to a next hop and an outgoing interface is sufficient for current routers to perform IP datagram forwarding. Nevertheless, routers nowadays maintain a forwarding table at each line card of an interface for lookup efficiency. However, all these forwarding tables at each interface are identical, i.e., these forwarding tables are interface-independent. For example, interface-independent forwarding tables at node B of Fig. 4.1 are as given in Table 4.2.

Instead of maintaining the same forwarding table at each interface, it is possible to avoid forwarding loops by making the entries of these forwarding tables *interface-specific*. Table 4.3 gives the possible set of interface-specific forwarding table entries at node B of Fig. 4.1. Each entry is marked with '–', **X**, or a nexthop node. The entries marked '–' are obviously never used. The entries marked **X** are not referenced normally, i.e., when there is no failure and all nodes in the network have the same consistent view. For example, in Fig. 4.1, a packet with destination D

		destination				
		A	C	D	E	F
interface	A→B	A	C	C	A	A
	C→B	A	C	C	A	A
	F→B	A	C	C	A	A

Table 4.2: Interface-Independent Forwarding Tables at Node B

		destination				
		A	C	D	E	F
interface	A→B	-	C	X	X	X
	C→B	A	-	X	X	A
	F→B	A	C	X	X	-

Table 4.3: Interface-Specific Forwarding Tables at Node B

should not arrive at B from any of its neighbors since B is not the next hop for them. Similarly, B should not receive from A, a packet destined for F, since A is along the path from B to F. However, in the presence of link failures and inconsistent forwarding tables at different nodes (during the convergence period), a packet may arrive at a node through an unusual interface. Interface-specific forwarding enables special treatment of such packets that arrive through unusual interfaces without introducing any changes to the forwarding plane of the current network infrastructure. Here, we study how interface-specific forwarding can be exploited for the purpose of avoiding loops during the convergence period after a link state change in the network.

4.2.2 Loopless Interface-Specific Forwarding (LISF)

It is clear that under link state routing, when all the routers in a network have the same view of the network, there would not be a forwarding loop. Only in the presence of discrepancies in the views of different routers, a packet might get caught in a loop. However, in such a case, under interface-specific forwarding, the packet would have arrived through an unusual interface of at least one of the routers involved in the loop. So a forwarding loop can be avoided if the packet were to be discarded in such a scenario rather than forwarded to the usual next hop. We refer to this approach of avoiding forwarding loops by selectively discarding packets that arrive through unusual

interfaces as *loopless interface-specific forwarding* (LISF).

Ideally, a packet should be discarded by a router only if its forwarding would definitely result in a loop. However, with only its own local view of the network, a router cannot always determine the actual forwarding path of a packet with certainty. Therefore, the design challenge of LISF is to ensure loop freedom without unnecessarily discarding packets. In this chapter, we study several implementation choices of LISF, ranging from conservative discarding of packets only if there would certainly be a loop otherwise but forwarding even if there could be a loop, to aggressively discarding of packets whenever there could be a loop even if there may not actually be a loop.

Before we proceed to present various LISF methods, we first introduce some notation that would help describe them. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph with vertices \mathcal{V} and edges \mathcal{E} representing the network. We use \mathcal{R}_i^d to denote the next hop² from i to d in \mathcal{G} . Let $\mathcal{F}_{j \rightarrow i}^d$ denote the forwarding table entry, i.e., the next hop to d for packets arriving at i through the interface associated with neighbor j . We use \mathcal{P}_i^d to refer to the shortest path from i to d given the graph \mathcal{G} . Similarly, the cost of the shortest path is denoted by C_i^d .

We now present three different LISF methods. The difference between these methods lies in which of the entries marked **X** in Table 4.3 are set to \ominus , meaning *discard*. These methods are named according to the criterion they use to discard a packet. The operation of these methods, when a packet for destination d arrives at node i from neighbor j , is summarized in Table 4.4 and elaborated in detail below. It should be noted that, under LISF, a node i makes packet forwarding/discarding decisions based solely on its own view of the network.

²Note that LISF works even with Equal Cost Multipath (ECMP) routing. But for ease of explanation, it is assumed that there is only one shortest path per destination.

Method	Discard Condition	Discard Criterion
PIng-POng (PIPO)	$j \in \mathcal{R}_i^d$	in and out interfaces are same
CYCLe (CYCL)	$j \in \mathcal{P}_i^d$	previous node along the path
NO Forward Progress (NOFP)	$C_{\mathcal{R}_i^d}^d \geq C_j^d$	no forward progress

Table 4.4: Differences in LISF Methods in Discarding a Packet to d Arriving at i from j

		destination				
		A	C	D	E	F
interface	A→B	–	C	C	⊖	⊖
	C→B	A	–	⊖	A	A
	F→B	A	C	C	A	–

(PIPO)

		destination				
		A	C	D	E	F
interface	A→B	–	C	C	⊖	⊖
	C→B	A	–	⊖	A	A
	F→B	A	C	C	⊖	–

(CYCL)

		destination				
		A	C	D	E	F
interface	A→B	–	C	⊖	⊖	⊖
	C→B	A	–	⊖	A	A
	F→B	A	C	⊖	⊖	–

(NOFP)

Table 4.5: Interface-Specific Forwarding Tables at B under Different LISF Methods

PIPO: Discard a packet if its incoming and outgoing interfaces are the same, i.e., $\mathcal{F}_{j \rightarrow i}^d = \ominus$ if $j \in \mathcal{R}_i^d$. PIPO discards a packet only when it arrives at a node from its next hop, i.e., along the reverse shortest path to the destination. It is the most conservative of all the methods listed here as it discards a packet only when there is a loop. Otherwise, without PIPO, in such a scenario, packets will ping-pong between two neighboring nodes. For example, in Table 4.5, a packet to destination D arriving at B from C is discarded by PIPO since C is the next hop to D from B. PIPO is also the simplest since it incurs no additional overhead for computing interface-specific forwarding table entries beyond the currently used Dijkstra’s algorithm for computing interface-independent forwarding tables. However, PIPO can ensure loop-freedom only when two nodes are involved in a loop, which is the case when links are *symmetric* (bidirectional with equal weights in both directions) and inconsistency in the views among routers is limited to a single link’s state.

CYCL: Discard a packet if the previous node appears along the path from this node to the destination, i.e., $\mathcal{F}_{j \rightarrow i}^d = \ominus$ if $j \in \mathcal{P}_i^d$. CYCL discards a packet when it arrives from a node which falls

along the shortest path from this node to the destination. When the links are symmetric, CYCL behaves just like PIPO. Only when links are asymmetric and the resulting paths are asymmetric, the operation of CYCL could be different from PIPO. With a less stringent condition than PIPO, CYCL may discard a packet even when there may not actually be a loop, but at the same time, it can avoid some loops that are not avoided by PIPO. For example, in Table 4.5, a packet to destination E arriving at B from F is forwarded by PIPO to A resulting in a loop whereas it will be discarded by CYCL since F is along the shortest path from B to E. The computational complexity of CYCL is similar to that of PIPO as both require only a single shortest path tree computation.

NOFP: Discard a packet if there is no forward progress towards its destination from its previous hop to the next hop of this node, i.e., $\mathcal{F}_{j \rightarrow i}^d = \ominus$ if $C_{\mathcal{R}_i}^d \geq C_j^d$. NOFP discards a packet if its previous hop is not farther from its destination than the next hop of this node. In such a case, there is a potential for a loop and NOFP discards such packets. For example, in Table 4.5, a packet to destination D arriving at B from F is discarded by NOFP since the cost from F to D is 2 whereas the cost from the next hop C is 3. This is in contrast to both PIPO and CYCL which forward the packet to C. While such discarding by NOFP seems unnecessary, NOFP can prevent more loops than PIPO and CYCL even when links are asymmetric and the state of multiple links change simultaneously. For example, in topology shown in Fig. 4.1, suppose link F–E failed. Furthermore, assume that all nodes except nodes B and C are notified of the failure and their forwarding tables reflect the failure. In this scenario, under PIPO and CYCL, a packet from A to D is forwarded along a loop A–B–G–C–A–B–... . On the other hand, under NOFP, it is discarded by B since, according to B's view, the cost of 3 from next hop G to D is not smaller than the cost from A to D which is also 3. The downside, however, is that a straightforward method to implement NOFP requires a

from	to	failed link	PIPO	CYCL	NOFP
C	D	C–D	discard	discard	discard
F	E	F–E	forward to A	discard	discard
F	D	F–E	forward to C	forward to C	discard
C	E	C–D	forward to A	forward to A	forward to A

Table 4.6: Differences among LISF Methods in Discarding Packets Arriving at Node B

computation of $O\left(\frac{|E|}{|V|}\right)$ times Dijkstra on the average (to compute the shortest path trees rooted at each neighbor), whereas PIPO and CYCL have the same complexity as Dijkstra.

The difference between the actions of the above three methods is clearly evident in the presence of failures of links C–D and F–E in Fig. 4.1 as shown in Table 4.6. Here it is assumed that B is not yet aware of the failed links and the forwarding tables of B do not reflect the change. When only F–E fails, packets from F to D arriving at B are discarded by NOFP whereas PIPO and CYCL forward them along a loop-free path via C. Essentially these methods achieve different tradeoffs between loop-avoidance and packet-discarding, which can be summed up as follows:

- Packet looping probability: $\text{PIPO} \geq \text{CYCL} \geq \text{NOFP}$
- Packet discard probability: $\text{PIPO} \leq \text{CYCL} \leq \text{NOFP}$
- Network convergence delay: $\text{PIPO} = \text{CYCL} = \text{NOFP}$

4.3 Proof of Loop-free Property of LISF

We now prove that the LISF methods described in the previous section ensure loop-freedom when at most a single link state change is being propagated in a network with symmetric links. It is clear that if PIPO is loop-free, other two methods are also loop-free since whenever PIPO

discards a packet, they would too. Therefore, it suffices to provide the proof for PIPO, which is given below.

We use the following notation in this section. Let T^D be the shortest path tree with undirected links rooted at a destination D . Note that this tree contains the shortest paths from every node to D since the links are symmetric. We use $S(i, T)$ to denote the subtree of T below the node i . Since the forwarding to a destination is independent of forwarding to other destinations, in the following we prove it for a destination D .

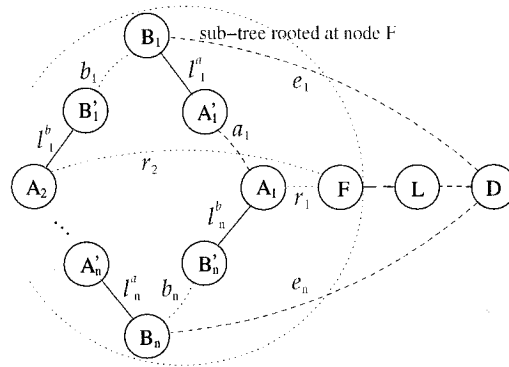


Figure 4.3: Scenarios for Illustrating Loop-Freedom under PIPO

Theorem 1. *A packet destined for D will not loop under PIPO in case of the failure of a link ℓ .*

Proof. If $\ell \notin T^D$, the forwarding path to D is the same with or without the failure of ℓ . Therefore, the forwarding to D is consistent at every node along the path and hence packets destined for D will not be caught in a loop due to the failure of ℓ . In the rest of the proof, it is assumed that $\ell \in T^D$.

Let $\ell = F-L$ and F be the upstream node to L on the path from F to D as in Fig. 4.3. When ℓ is down, only those nodes in $S(F, T^D)$ will be affected (i.e., all nodes outside the subtree will forward along the same path with or without $F-L$ for destination D). Now consider a packet originating from any node in $S(F, T^D)$. That packet may be forwarded to node F and then get

rerouted, or get rerouted somewhere in $S(F, \mathcal{T}^D)$. Once the packet goes outside the subtree, it will be forwarded to D consistently. So the only possible loop is the one consisting of nodes that are all within $S(F, \mathcal{T}^D)$. Thus, to prove loop-freedom under PIPO, we only need to show that there will not be a loop within $S(F, \mathcal{T}^D)$.

Suppose there is a loop in subtree $S(F, \mathcal{T}^D)$. The loop must contain some nodes that are *aware* and some that are *unaware* of the failed link F–L. Pick an arbitrary node, A_1 , in the loop that is aware of the failure. As shown in Fig. 4.3, suppose the packet starts from A_1 and is routed along zero or more “A” nodes (i.e., nodes that are aware of the failure and forward in consistence as A_1), and reaches A'_1 , the last “A” node in this stretch. A'_1 forwards the packet to B_1 , a node unaware of the failure. Note that from A_1 to B_1 , the forwarding is consistent (with $\mathcal{P}_{A_1}^D(\mathcal{E} \setminus \ell)$). We use the dashed line to indicate this subpath of $\mathcal{P}_{A_1}^D(\mathcal{E} \setminus \ell)$. B_1 then reroutes the packet: instead of towards D via the dashed path, it forwards the packet via the dotted path towards B'_1 , i.e., it chooses $B_1 \rightsquigarrow B'_1 \rightarrow A_2 \rightsquigarrow F \rightarrow L \rightsquigarrow D$. Similarly, the packet is rerouted at A_2 , which intends to forward it to D through the next stretch of dashed path. This process continues until the packet is forwarded back to A_1 by B_n ($n \geq 1$).

Now we show that there is a contradiction if such a loop exists. Note that in such a loop, any B_i can not forward a packet back to A'_i after getting the packet from A'_i (e.g., $A'_{i+1} \neq A'_i$), as the packet will get dropped under PIPO when being forwarded back to A'_i . Then consider the reroute decision at node B_i ($1 \leq i \leq n$). Since the node B_i chooses the path $B_i \rightsquigarrow A_{i+1} \rightsquigarrow F$ over $B_i \rightsquigarrow A_{i-1} \rightsquigarrow F$, we have

$$\sum_{i=1}^{n-1} (b_i + l_i^b + r_{i+1}) < \sum_{i=1}^{n-1} (a_i + l_i^a + r_i) \quad (4.1)$$

$$b_n + l_n^b + r_1 < a_n + l_n^a + r_n \quad (4.2)$$

Adding them together, we have

$$\sum_{i=1}^n (b_i + l_i^b) < \sum_{i=1}^n (a_i + l_i^a) \quad (4.3)$$

Similarly, consider the rerouting decision made at node A_i . Since it chooses the path $A_i \rightsquigarrow B_i \rightsquigarrow D$ over the path $A_i \rightsquigarrow B_{i-1} \rightsquigarrow D$,

$$\sum_{i=1}^{n-1} (a_{i+1} + l_{i+1}^a + e_{i+1}) < \sum_{i=1}^{n-1} (b_i + l_i^b + e_i) \quad (4.4)$$

$$a_1 + l_1^a + e_1 < b_n + l_n^b + e_n \quad (4.5)$$

Adding them together, we get

$$\sum_{i=1}^n (a_i + l_i^a) < \sum_{i=1}^n (b_i + l_i^b) \quad (4.6)$$

Obviously, formula (6) above contradicts formula (3). Therefore, a forwarding loop is not possible under PIPO in case of a single link failure. \square

Using similar arguments as above, we can show that PIPO avoids forwarding loops during the convergence period after not only a failure but any change in the state of a link.

4.4 Performance Evaluation

In this section, we evaluate the performance of LISF methods and compare them against OSPF and ORDR. We first simulate single link failures and demonstrate that LISF methods prevent a significant number of loops that are possible under OSPF, and also have lower convergence delay than ORDR. We then experiment with scenarios of multiple failures to further study the tradeoffs of LISF methods between packet-discarding and loop-avoidance.

4.4.1 Single Link Failures

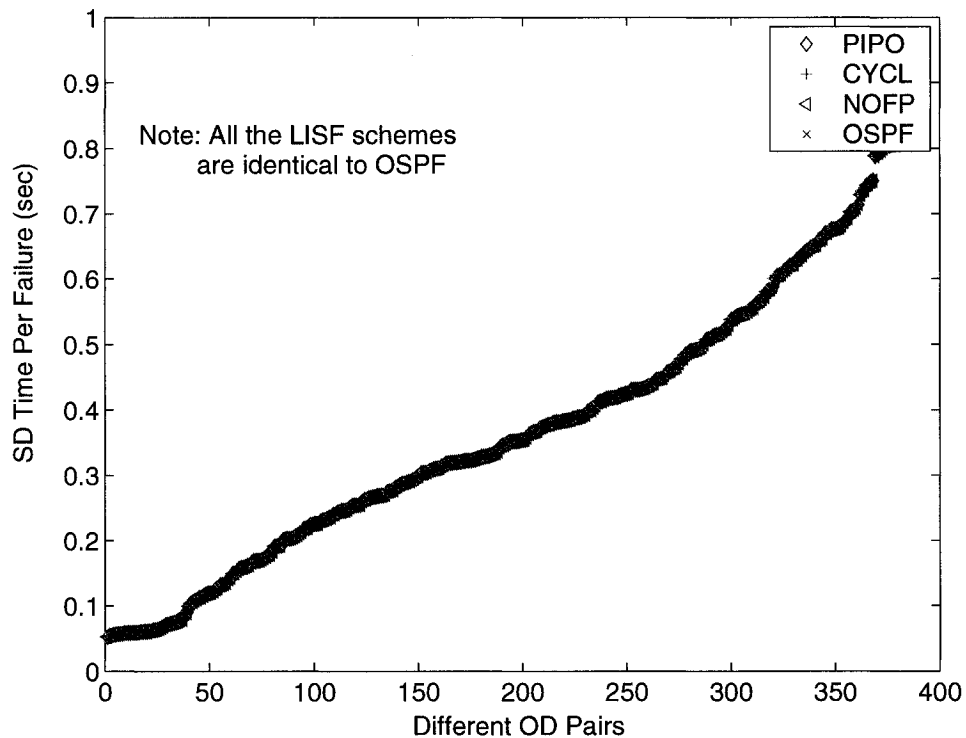


Figure 4.4: Average Service Disruption Time per Link Failure for Various O-D pairs in Sprint Topology with Symmetric Links

To evaluate LISF methods, we use the same control plane simulator presented in Sec-

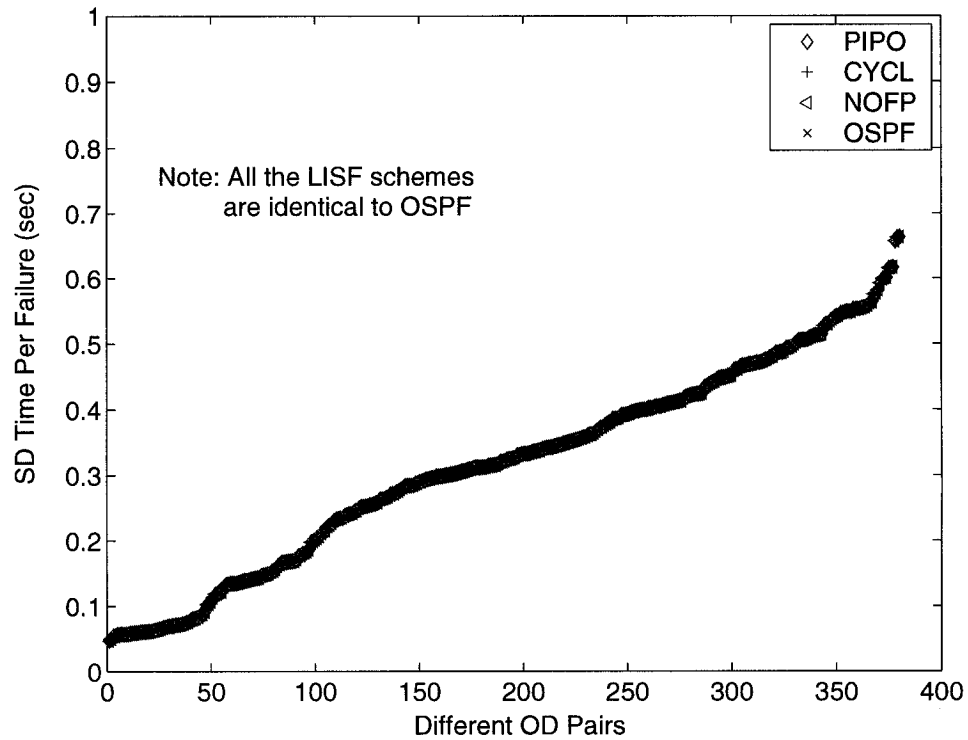


Figure 4.5: Average Service Disruption Time per Link Failure for Various O-D pairs in Sprint Topology with Asymmetric Links

tion 3.3 that emulates intra-domain routing dynamics in the presence of link failures and measures both service disruption (SD) time between different origin-destination (OD) pairs and network convergence (NC) time. We use the tier-1 ISP backbone (PoP-level) topology with 20 nodes and 42 links in our simulations (Figure 3.2). We assign OSPF link weights to different links by randomly picking integer values between 1 and 5. We consider both symmetric links where $X \rightarrow Y$ has the same OSPF weight as $Y \rightarrow X$, and asymmetric links where the OSPF weight for $X \rightarrow Y$ could be different from $Y \rightarrow X$. The forwarding table at each node includes entries for all the prefixes in the Internet. We assume that the rate of FIB update is 20 entries/ms and the number of prefixes is 161352. The other parameters in the simulator are set based on the findings in [52]. In every simulation run, we fail each link in the network exactly once. The results presented below represent

the effect of all the link failures in a simulation run.

The total time during which a forwarding loop exists under OSPF is observed to be 11.851s, whereas LISF methods, as expected, have no loops in case of symmetric link failures. For the case of asymmetric link failures, OSPF has loops for a duration of 6.989s, while it is 0.056s for PIPO. In both cases, there are no loops under CYCL or NOFP. These results demonstrate the effectiveness of LISF methods in avoiding loops. We proceed to show that this is achieved not at the expense of larger convergence delay or longer service disruption.

Figure 4.4 represents the average SD time experienced by all OD pairs due to a single link failure in this network with symmetric links. We can clearly see that the average SD time for a particular OD pair remains the same when LISF is implemented on top of OSPF. This shows that LISF does not add any extra SD time. Figure 4.5 shows a very similar behavior of average SD time for this network with asymmetric links.

Let NC time represent the total time taken for the network to converge after a link failure. Fig. 4.6 and Fig. 4.7 show the NC time due to different link failures for the backbone network with symmetric and asymmetric links, respectively. Given that some nodes in the network have to wait for a longer time to update their FIBs under ORDR, it is easy to see that a network with ORDR exhibits higher NC times when compared to a network with LISF where no synchronization is needed between nodes for updating their FIBs.

Fig. 4.8 shows the packet discard times due to various LISF methods in the ISP network with symmetric links. As expected, with symmetric link failures, the packet discard times of PIPO and CYCL are identical, and NOFP is quite close to PIPO. A similar pattern is observed even with asymmetric links (Fig. 4.9). However, the packet discard times under PIPO and CYCL are not

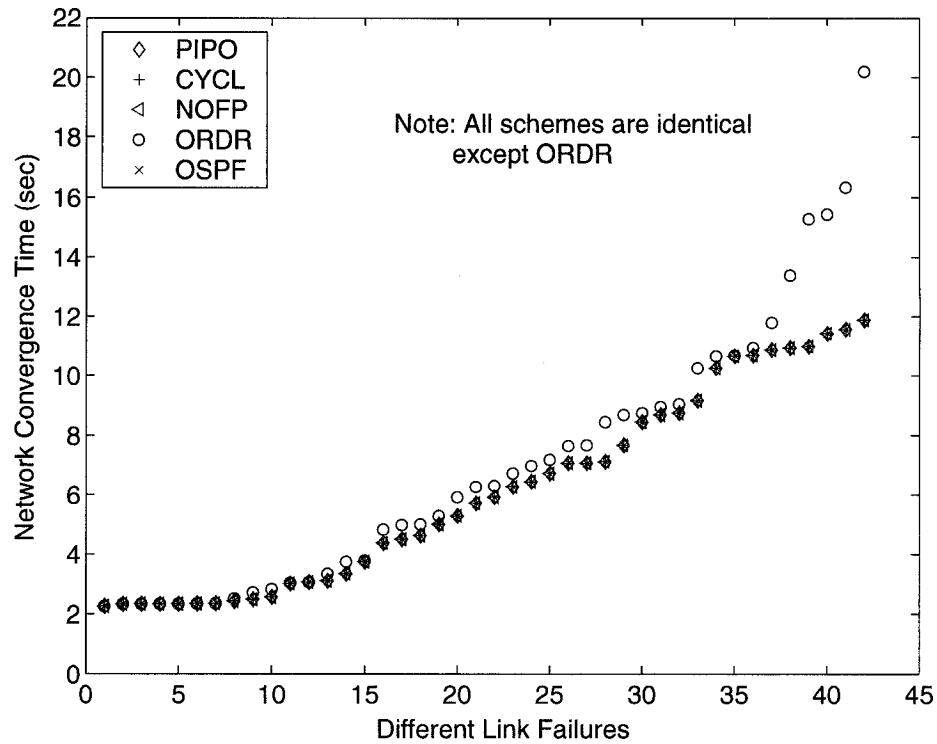


Figure 4.6: Network Convergence Time due to Various Single Link Failures in Sprint Topology with Symmetric Links

identical due to the fact that more loops are avoided by CYCL compared to PIPO.

4.4.2 Multiple Link Failures

To further evaluate LISF methods, we simulated failures of multiple links and nodes. For this study, we did not use the control-plane simulator mentioned above as it can only handle single link failures. Instead, we used a simplified model to approximate link state propagation and route computation and FIB update times. It is assumed that link state propagation takes 1 time unit per hop and route computation and FIB update time is 3 units. We simulate single node failures and also simultaneous failures of 2 nodes, 2 links, and 3 links. In each failure scenario, we forward a packet between every pair of nodes and count the number of node pairs for whom packets are undeliverable

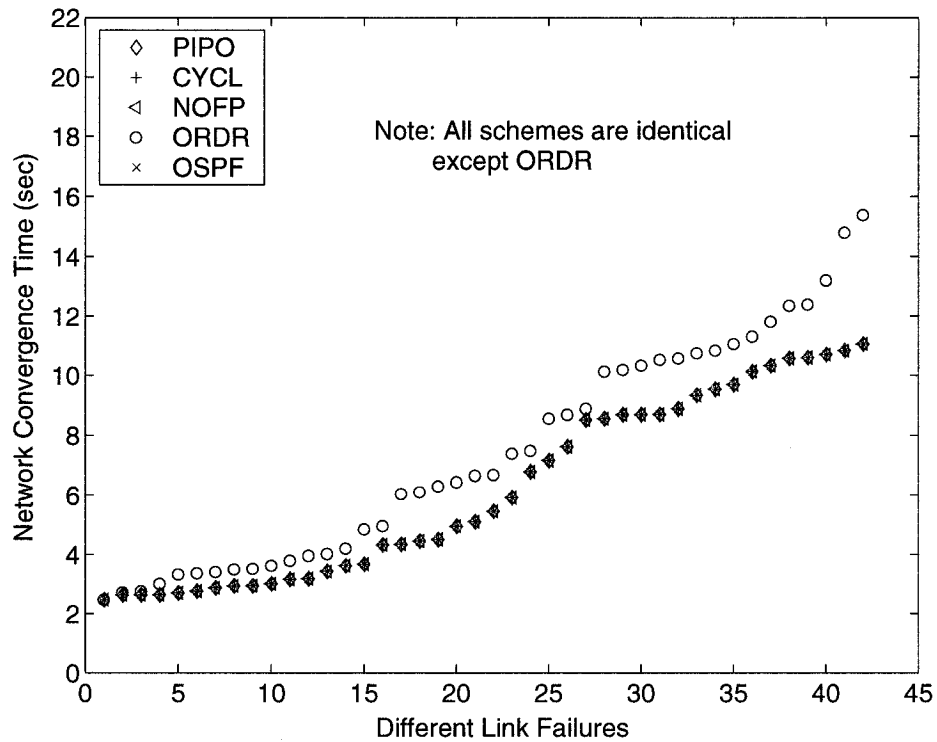


Figure 4.7: Network Convergence Time due to Various Single Link Failures in Sprint Topology with Asymmetric Links

and also those that get caught in a loop.

failures	looping probability				% of undeliverable node pairs			
	OSPF	PIPO	CYCL	NOFP	OSPF	PIPO	CYCL	NOFP
2 links	$10^{-2.3}$	$10^{-5.3}$	$10^{-5.3}$	0	5.9	5.9	5.9	6.2
3 links	$10^{-2.1}$	$10^{-4.9}$	$10^{-4.9}$	0	8.6	8.6	8.6	9.1
1 node	$10^{-3.0}$	0	0	0	4.3	4.3	4.3	4.4
2 nodes	$10^{-2.7}$	$10^{-3.7}$	$10^{-3.7}$	0	8.1	8.1	8.1	8.2

Table 4.7: Comparison of LISF Methods and OSPF in Case of Multiple Failures

Table 4.7 shows the relative performance of different LISF methods and OSPF in terms of their ability to avoid loops and deliver packets. ORDR is not included here as it is not designed to deal with multiple failures. PIPO and CYCL yield identical performance since the links are symmet-

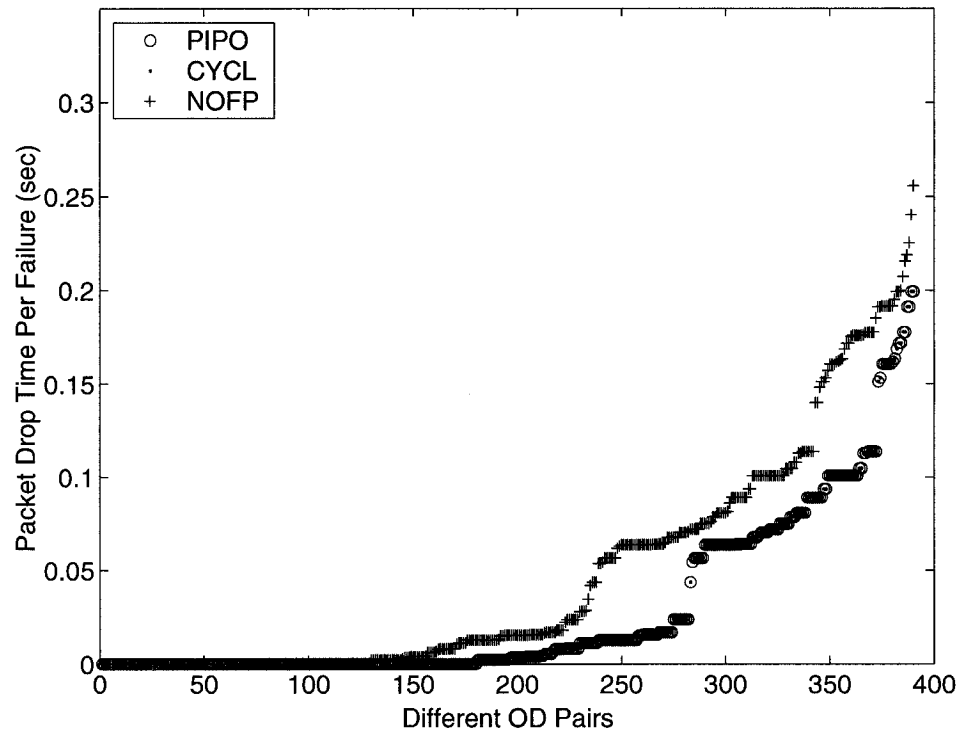


Figure 4.8: Average LISF Packet Drop Time per Link Failure for Various O-D pairs in Sprint Topology with Symmetric Links

ric. Compared to OSPF, loops are close to 1000 times less likely to happen with PIPO and CYCL, whereas no loops occur under NOFP. In terms of packet delivery, both PIPO and CYCL have the same performance as OSPF. The delivery ratio of NOFP is only slightly worse than OSPF. Considering that NOFP prevents loops without excessive discarding of packets, we believe LISF approach with NOFP method is a viable alternative for avoiding transient loops during the convergence of intra-domain routing schemes in IP networks.

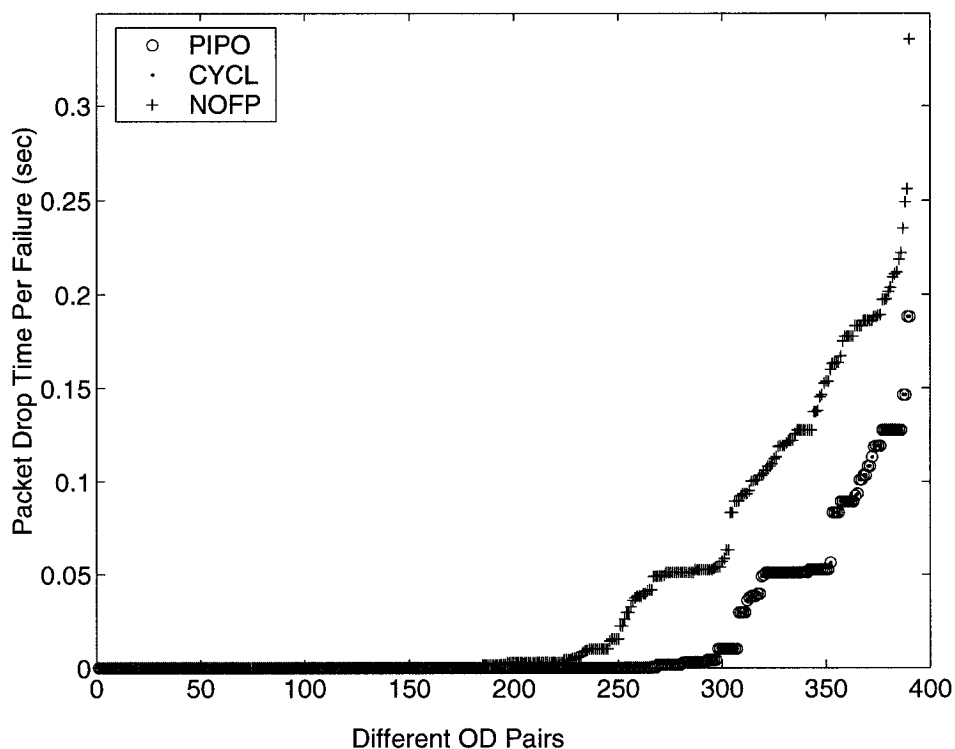


Figure 4.9: Average LISF Packet Drop Time per Link Failure for Various O-D Pairs in Sprint Topology with Asymmetric Links

4.5 Summary

In this chapter, we have proposed a simple interface-specific forwarding based approach called LISF to avoid transient forwarding loops during the network convergence periods. LISF approach selectively discards packets arriving through unusual interfaces when they are likely to be caught in a loop. We have demonstrated that LISF incurs no additional message overhead compared to OSPF and avoids forwarding loops like ORDR without increasing the network convergence time. We have presented several LISF methods and evaluated their performance. We observed that simple PIPO is effective in eliminating most of the loops and NOFP provides the best trade-off between packet-discarding and loop-avoidance.

Chapter 5

Applications of Service Availability in Network Management

Internet service providers (ISPs) face a plethora of critical challenges in the area of network management. One such challenge is the problem of *graceful network upgrade*, where ISPs need to add new nodes and links into their operational network in a graceful manner so that the perceived network performance from the perspective of existing customers does not deteriorate. Although graceful network upgrade is an important aspect of network management, it has not received much attention from the networking community. To the best of our knowledge, as of today, there are no publicly available studies that provide a framework for adding new nodes and links into existing operational networks. Most of the previous works have focused on upgrade in the context of either upgrading individual network components [95] or capacity provisioning [38].

From the perspective of an ISP, planning a successful network upgrade involves three main steps: (i) identifying a set of potential locations where new nodes can be added, (ii) determining

a subset of all the identified locations where nodes should be added (along with the links) such that it results in maximum revenue for the ISP given certain budget and performance constraints, and (iii) identifying an ideal sequence for adding nodes and links into the network such that the upgrade process has minimum impact on the existing customers. There are several techniques like [54, 14, 91, 32, 20], that can be used to address the first step from an economic viewpoint. The focus of our work is on the last two steps assuming that an ISP has already identified a set of potential locations for adding new nodes.

In this chapter, we propose a *two-phase multistage* framework to determine the optimal network upgrade strategy. One of the key features of this framework is the emphasis on considering network performance not only from the perspective of the ISP but also from the perspective of its customers at every stage. We capture the network performance from the perspective of customers using the service availability metrics that we discussed in detail in Chapter 3. In *Phase-1*, we determine the nodes and links that should be added to the network by formulating the problem as a non-linear optimization problem. In *Phase-2*, we determine an ideal sequence for adding the new nodes and the links to ensure minimal impact on network performance through multistage dynamic programming (DP). We believe that multistage node addition in Phase-2 is critical for an ISP for several reasons: (i) It would be extremely hard to debug problems that arise if all the identified nodes and links are added simultaneously. Given that the network is operational during the upgrade process, it is important to ensure that the customers do not suffer from severe performance degradation; (ii) ISP may face delays due to node/link construction schedule and certain nodes/links could be available sooner than others; (iii) The budget may only become available sequentially.

5.1 Graceful Network Upgrade

Our focus in this work is on tier-1 ISP backbone networks where nodes correspond to Points-of-Presence (PoPs) and links correspond to IP-layer virtual connections on top of the underlying optical fiber. Consider a tier-1 ISP backbone network that spans the continental US. All the nodes in such a network are strategically placed in locations such that the ISP can serve a large number of customers who can easily connect to its nodes, thus resulting in the maximum possible revenue. Although such a network could be ideal today, it is important for ISPs to add more nodes into their backbone network to serve customers in regions that have the potential to become business/residential hubs in the future.

In the context of our work, network upgrade *does not* refer to upgrading individual components in the network. An obvious approach for an ISP is to add nodes at all possible locations and connect them in a full mesh to give the best performance, but such a strategy is usually infeasible due to budget constraints. We assume that the entire upgrade procedure has a maximum fixed budget, b , that cannot be exceeded. We also assume that an ISP:

- Can reliably estimate the future traffic demands to identify all potential physical locations where it is beneficial to install new nodes [14, 91, 32, 20].
- Can accurately estimate the revenue/profit associated with the addition of new nodes in the locations identified above [54]. Each node addition results in a revenue/profit that is independent of other node additions in the network.

5.1.1 Network Performance from a Customer's Perspective

Recent studies in tier-1 carriers have found that performance deterioration inside tier-1 ISP networks are mainly caused by *link failures* that occur frequently due to several reasons like fiber cuts, software bugs, and hardware errors [51]. The authors in [66] developed a general model that captures the duration and frequency of occurrence of such link failures. Based on this link failure model and the observed transient behavior during routing convergence, we proposed two metrics that capture the network service availability from a customer's perspective in Chapter 3, Section 3.2.1:

- **Service Disruption Time (SD Time):** This metric captures the average time for which a customer connected to an ISP through ingress node gets disconnected from various prefixes in the Internet due to link failures. It is important for an ISP to ensure that the SD time for all the nodes in their network is below the acceptable values specified in their service level agreements (SLA).
- **Traffic Disruption (TD):** This metric captures the total traffic disrupted between different nodes in the network due to link failures. TD should be included in the 'cost' function since an ISP typically has to compensate customers when the total TD for the customer exceeds the threshold value specified in the SLA.

Using extensive simulations, we previously showed that SD time and TD capture several important characteristics of an operational network that cannot be described by traditional graph-theoretic metrics (like node degree, network diameter, etc.). In the rest of our discussions in this chapter, we use SD time and TD (either as constraints or as high cost penalties when violated) to capture the network performance from a customer's perspective.

Note that SD time and TD metrics depend on operational network conditions like traffic

demand, exit points for various Internet prefixes, and shortest paths between different nodes in the network. Adding new nodes and links could change these operational network conditions, thus affecting the network performance as seen by customers. Our goal in this work is to minimize the performance deterioration due to changing operational network conditions. It is also important to note that SD time and TD metrics can be easily replaced by other context-specific metrics while keeping the same overall framework proposed in this chapter.

5.1.2 Problem Description

Consider a network with n nodes and l links. We use \mathcal{N} to represent the set of all nodes and \mathcal{L} to represent the set of all links in the original network. We use several attributes to capture different characteristics of the network:

- **Adjacency Matrix, M:** An $n \times n$ matrix where each element is either a 1 or 0. $M_{i,j} = 1$ indicates that there is a link that directly connects node i and node j , and $M_{i,j} = 0$ indicates that there is no link between i and j .
- **Distance Matrix, D:** An $n \times n$ matrix indicating the physical distance between different nodes in the network.
- **Service Disruption Time Matrix, S:** An $l \times n \times n$ matrix where $S_{i,j,k}$ represents the total service disruption time for the source-destination pair $j - k$ due to failure of link i .
- **Traffic Disruption Matrix, T:** Similar to S , this is a $l \times n \times n$ matrix where $T_{i,j,k}$ represents the total traffic disrupted between the source-destination pair $j - k$ due to failure of link i .

Note that the adjacency matrix, M , and the distance matrix, D are symmetric matrices since we assume that all the links are bidirectional, while service disruption time matrix, S , and traffic disruption matrix, T , may not be symmetric.

Let g represent the total number of locations where nodes can be placed to upgrade the network. We represent the revenue/profit from an ISP's perspective by \mathbf{r} ($g \times 1$ vector) where each element r_k indicates the revenue/profit when the new node at position k is added to the network. Similarly we use \mathbf{nc} ($g \times 1$ vector) to represent the estimated building and installation cost for new nodes at different potential locations. Note that the cost of building and installing nodes at different locations could be different. We use lc to indicate the cost per unit length of installing new links. The cost here refers to the cost of installing optical fiber between different nodes. Finally, we assume that all the links (in the current network and upgraded network) have enough capacity to carry the traffic demands and hence we ignore using link capacities as variables in our problem formulation. However it is easy to include link capacities as variables in our two-phase network upgrade framework.

The graceful network upgrade problem can be defined as follows: Given a total budget constraint, find the locations where the nodes should be added and how these nodes should be connected to the other nodes (original and newly added nodes) in the network so as to maximize the revenue from the network while maintaining certain performance constraints. We should also determine the optimal sequence to perform this upgrade. During the network upgrade process we should ensure that the network performance (in terms of service disruption time and traffic disruption) from the perspective of customers connected to the original network does not deteriorate.

Let \mathbf{k}' ($(n + g) \times 1$ vector) represent a 0/1 vector where, $k'_i = 1$ indicates that a node exists at location i , and $k'_i = 0$ indicates that no node exists at location i . Since we do not remove any of the original nodes in the network during the upgrade process, we have:

$$\mathbf{k}' = \begin{pmatrix} \mathbf{u}_n \\ \mathbf{k}'' \end{pmatrix}$$

where, \mathbf{u}_n ($n \times 1$ vector) is a unit vector with all its elements equal to 1, and \mathbf{k}'' is a $g \times 1$ vector whose elements can be 0/1 depending on the locations where the nodes are added to the network.

Let \mathbf{M}' represent the adjacency matrix of the upgraded network:

$$\mathbf{M}' = \begin{pmatrix} \mathbf{M} & (\mathbf{M}'')^T \\ \mathbf{M}'' & \mathbf{M}''' \end{pmatrix}$$

where, \mathbf{M} is the adjacency matrix of the original network; \mathbf{M}'' is a $g \times n$ matrix that indicates the links between the nodes added during the upgrade process and the original nodes in the network; \mathbf{M}''' is a $g \times g$ matrix that indicates the connectivity between the new nodes added into the network during the upgrade process.

In order to determine optimal strategy for network upgrade, we have to determine the values of \mathbf{k}'' , \mathbf{M}'' , and \mathbf{M}''' such that all the constraints are met. We will discuss more about the constraints in Section 5.2. A summary of all the notations used in this chapter is presented in Table 5.1.

5.2 Problem Formulation

We decouple the graceful network upgrade problem into the following two phases:

Symbol	Description
n	Number of nodes in original network
l	Number of links in original network
\mathcal{N}	Set of all nodes in original network
\mathcal{L}	Set of all links in original network
\mathcal{N}'	Set of all nodes in upgraded network
\mathcal{L}'	Set of all links in upgraded network
$\mathbf{M} [n \times n]$	Adjacency matrix of original network
$\mathbf{D} [n \times n]$	Distance matrix of original network
$\mathbf{S} [l \times n \times n]$	SD time matrix of original network
$\mathbf{T} [l \times n \times n]$	TD matrix of original network
α	Link utilization threshold
b	Total budget for network upgrade
g	Number of potential locations to add nodes
$\mathbf{r} [g \times 1]$	Revenue vector for potential node locations
$\mathbf{nc} [g \times 1]$	Node installation cost
lc	Link installation cost per unit length
$\mathbf{k}' [(n + g) \times 1]$	Node placement vector for upgraded network
$\mathbf{M}' [(n + g) \times (n + g)]$	Adjacency matrix of upgraded network
$\mathbf{D}' [(n + g) \times (n + g)]$	Distance matrix of upgraded network
$\mathbf{u}_n [n \times 1]$	Unit matrix of size $n \times 1$

Table 5.1: Notations used in Graceful Network Upgrade Problem Formulation

5.2.1 Phase-1: Finding the Optimal End State

The main objective of this phase is to determine an end state that maximizes the total revenue/profit for the ISP. This can be formulated as follows:

Objective function: Maximize $((\mathbf{k}'')^T \cdot \mathbf{r})$ subject to the following constraints:

Budget constraint: The total cost of building and installing the nodes and links should be less than the total budget:

$$\frac{lc}{2} \{ \mathbf{u}_{n+g}^T \cdot (\mathbf{D}' \otimes \mathbf{M}') \cdot \mathbf{u}_{n+g} - \mathbf{u}_n^T \cdot (\mathbf{D} \otimes \mathbf{M}) \cdot \mathbf{u}_n \} + ((\mathbf{k}'')^T \cdot \mathbf{nc}) \leq b$$

where, \otimes operator represents element-wise multiplication of the two matrices and \mathbf{D}' is the distance matrix of the upgraded network. Since all the links are bidirectional and are considered twice in the adjacency and distance matrices we need the factor 2 in the denominator of the first term in the above equation.

Node degree constraint: One of the basic requirements for resiliency that all ISPs require is that every node in the network has at least two independent paths to route traffic originating at the node.

In other words, the degree of all the nodes added into the network should be at least 2:

$$\sum_{j=1}^{n+g} \mathbf{M}'_{i,j} \geq 2 \cdot \mathbf{k}_i \quad \forall i = n+1, n+2, \dots, n+g$$

SD time constraint: For an ISP it is important to ensure that the network performance experienced by customers connected to the nodes in the original network does not deteriorate. Let \mathbf{S}' , \mathcal{N}' , and \mathcal{L}' represent the SD time matrix, set of nodes, and set of links respectively in the final upgraded network. Hence we can write the SD time constraint as:

$$\sum_{i \in \mathcal{L}', q \in \mathcal{N}'} \mathbf{S}'_{i,j,q} \leq \sum_{i \in \mathcal{L}, q \in \mathcal{N}} \mathbf{S}_{i,j,q} \quad \forall j \in \mathcal{N}$$

TD constraint: Similar to SD time, the traffic disruption experienced by customers should not become worse after an upgrade. We use \mathbf{T}' to represent the TD matrix of the upgraded network.

Hence,

$$\sum_{i \in \mathcal{L}', q \in \mathcal{N}'} \mathbf{T}'_{i,j,q} \leq \sum_{i \in \mathcal{L}, q \in \mathcal{N}} \mathbf{T}_{i,j,q} \quad \forall j \in \mathcal{N}$$

Link utilization constraint: Maximum link utilization has frequently been used by ISPs as a measure of network quality and failure resiliency [6]. Network operators always want to ensure that the

maximum link utilization in their network is below a certain threshold value (say α). We represent the utilization of a link i by LU_i . Hence,

$$\max_{\forall i \in \mathcal{L}'} LU_i \leq \alpha$$

The linearity or non-linearity of Phase-1 depends mainly on the performance metrics that are chosen as constraints. The above formulation is non-linear since \mathbf{S} and \mathbf{T} are non-linear terms (as presented in Chapter 3 and [56]). This problem can be solved by well-established non-linear programming techniques such as simulated annealing, genetic algorithms, and tabu search.

Note that, in the above formulation: (i) Absolute values for the SD time and TD constraints can be used instead based on the SLA between ISP and customers. (ii) Constraints can be imposed on SD time and TD experienced by *new nodes* added to the network depending on the SLA that the ISP plans for the customers connecting to the new nodes. However, we currently ignore the above conditions in both Phase-1 and Phase-2, since our focus in this chapter is to establish the usefulness of the basic framework without adding secondary details.

5.2.2 Phase-2: Multistage Node Addition Strategy

The main objective of this phase is to determine the optimal sequence in which nodes and links should be added into the network. In this phase, we make two assumptions:

- We assume that we can obtain a solution from the first phase, i.e., the values of \mathbf{k}' and \mathbf{M}' . In other words, we have complete knowledge of the initial network state that we start with and the final state that we intend to reach.
- We also assume that the ISP wants to add no more than one node at a time. Hence every stage

in our multistage node addition strategy adds one node and all the links associated with that node into the network.

We formulate Phase-2 as a multistage DP problem, and use network performance at every stage as the cost involved in upgrading the network from one stage to the next. We characterize the network performance by four metrics: service disruption time, traffic disruption, max link utilization, and the degree of different nodes in the network. Note that these metrics can be used either as constraints or as costs in the utility function. We choose to use these metrics as costs since using them as constraints may not result in any feasible solution. On the other hand, assigning a high cost when the performance criteria is not met will lead to feasible solutions albeit higher cost. In this work, we treat all the four performance criteria with the same importance. However, a weighting scheme can be introduced to distinguish their importance.

We use the following notations in the dynamic program problem formulation. Let \mathcal{N}^e and \mathcal{L}^e represent the set of nodes and links respectively in the network after e node addition stages. Hence, $|\mathcal{N}^e| = |\mathcal{N}| + e = n + e$. We use \mathbf{v}_n^e to represent a 0/1 vector of nodes that are remaining to be added after e stages. Let h be the node added to the network in stage e . We can represent the optimal return function after stage e by $\Phi^e(\mathbf{v}_n^e, h)$. Note that the return function after any stage depends on the location of the node added and the locations of the nodes remaining to be added. Finally, we represent the SD time and TD values after stage e by \mathbf{S}^e and \mathbf{T}^e respectively.

We model the cost associated with network performance at stage e due to the addition of node h as follows:

- **Cost of SD time, $c_{SD}^{e,h}$:** ISPs require the SD time experienced by customers (i.e. nodes) in the original network should not increase after any stage of network upgrade. Hence we define a cost function such that the cost is very large only when the current SD time for the original nodes is larger than the original SD time.

For every node, i , in the original network we define the cost of SD time as:

$$c_{SD}^{e,h}(i) = \left\{ \frac{2\delta^{e,h}(i) - \delta(i)}{\delta(i)} \right\}^{2\omega_{SD}(i)} \quad \forall i \in \mathcal{N} \quad (5.1)$$

where, $\omega_{SD}(i)$ is a high order index value and,

$$\begin{aligned} \delta^{e,h}(i) &= \sum_{j \in \mathcal{L}^e, q \in \mathcal{N}^e} \mathbf{S}_{j,i,q}^e \\ \delta(i) &= \sum_{j \in \mathcal{L}, q \in \mathcal{N}} \mathbf{S}_{j,i,q} \end{aligned}$$

Hence the overall cost of SD time in the network is:

$$c_{SD}^{e,h} = \sum_{i \in \mathcal{N}} a_{SD}(i) \cdot c_{SD}^{e,h}(i) \quad (5.2)$$

where, $a_{SD}(i)$ is a weighting factor that indicates the importance of the performance of node i to the ISP.

- **Cost of TD, $c_{TD}^{e,h}$:** We can express the cost of TD in a very similar fashion as cost of SD time.

Hence,

$$c_{TD}^{e,h}(i) = \left\{ \frac{2\tau^{e,h}(i) - \tau(i)}{\tau(i)} \right\}^{2\omega_{TD}(i)} \quad \forall i \in \mathcal{N} \quad (5.3)$$

where, $\omega_{TD}(i)$ is a high order index value and,

$$\begin{aligned}\tau^{e,h}(i) &= \sum_{j \in \mathcal{L}^e, q \in \mathcal{N}^e} \mathbf{T}_{j,i,q}^e \\ \tau(i) &= \sum_{j \in \mathcal{L}, q \in \mathcal{N}} \mathbf{T}_{j,i,q}\end{aligned}$$

Hence the overall cost of TD in the network is:

$$c_{TD}^{e,h} = \sum_{i \in \mathcal{N}} a_{TD}(i) \cdot c_{TD}^{e,h}(i) \quad (5.4)$$

where $a_{TD}(i)$ is a weighting factor that indicates the importance of the performance of node i to the ISP.

• **Cost of node degree, $c_{degree}^{e,h}$** : Since the most common failures in networks are single link failures, it is important to ensure that all the nodes have a node degree of at least 2. Note that during the multistage upgrade process it may not be feasible to ensure that the newly added node will have a degree of at least 2. However, after the completion of the upgrade process all the nodes will have a degree of at least 2 since our solution in Phase-1 guarantees this. Hence a high cost at any stage resulting in degree of 0 or 1 will help in finding a strategy with a node degree of at least 2 in all the stages:

$$c_{degree}^{e,h}(i) = \left\{ \frac{1}{(\kappa^{e,h}(i))(\kappa^{e,h}(i) - 0.5)} \right\}^{2\omega_{degree}(i)} \quad \forall i \in \mathcal{N}^e \quad (5.5)$$

where $\omega_{degree}(i)$ is a high order index value to ensure that the cost for a node with small degree is high, and $\kappa^{e,h}(i)$ is the degree of node i after the addition of node in location h in stage e . Hence the total cost for node degree for the network at stage e is:

$$c_{degree}^{e,h} = \sum_{i \in \mathcal{N}^e} a_{degree}(i) c_{degree}^{e,h}(i) \quad (5.6)$$

where $a_{degree}(i)$ is a weighting factor that indicates the importance of the performance of node i to the ISP.

- **Cost of link utilization, $c_{util}^{e,h}$:** A high utilization on a link implies that it will not be able to accommodate more traffic when there is a failure elsewhere in the network. Hence the operators set a threshold value (α) for the maximum link utilization.

We can express the cost of link utilization as:

$$c_{util}^{e,h} = \left\{ \frac{2\sigma^{e,h} - \alpha}{\alpha} \right\}^{2\omega_{util}} \quad (5.7)$$

where $\sigma^{e,h} = \max_{\forall i \in \mathcal{L}^e} LU_i$.

The total cost of upgrade at stage e is $c^{e,h} = c_{SD}^{e,h} + c_{TD}^{e,h} + c_{degree}^{e,h} + c_{util}^{e,h}$. The functional equation of the multistage dynamic programming formulation now becomes:

$$\Phi^e(\mathbf{v}_n^e, h) = \max_{\forall h \in \mathbf{v}_n^{e-1}} \left\{ -c^{e,h} + \gamma \Phi^{e+1}(\mathbf{v}_n^{e+1}, h_1) \right\} \quad (5.8)$$

with the boundary condition,

$$\Phi^{n'-n+1}(\mathbf{v}_n^{n'-n+1}, h_2) = 0 \quad (5.9)$$

where $h_1 \in \mathbf{v}_n^e$, $h_2 \in \emptyset$, and $n' = |\mathcal{N}'|$. Also, $\gamma \leq 1$ represents the discounting factor for future benefits. In the rest of this chapter we use $\gamma = 1$.

5.3 Numerical Example

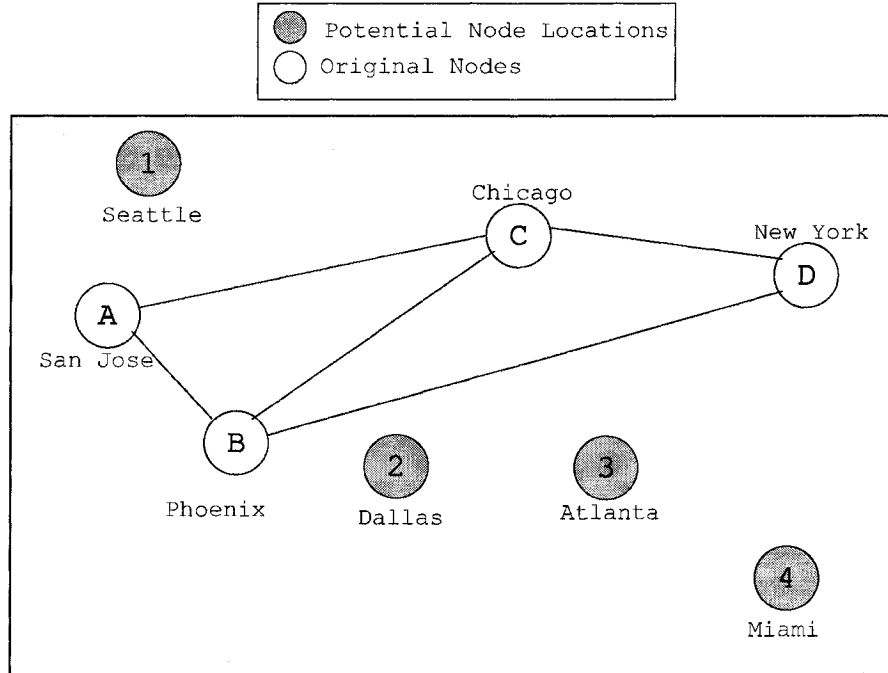


Figure 5.1: An Example Topology and Potential Node Locations for New Node Placement

In this section, we illustrate the feasibility and advantages of using our two-phase multistage upgrade process described earlier. For the ease of exposition, we use a simple example, although the framework can apply to real-world topologies of $O(100)$ nodes.

We consider the topology in Figure 5.1 where nodes are situated in geographically distributed locations in the US. Our aim is to create a scenario similar to that of tier-1 IP backbone (PoP-level) topologies that span the continental US. The original topology that we consider has four nodes (A, B, C, and D) and five links between them. Although IP-links are virtual links on top of the optical infrastructure, in this work we assume that the propagation delay for different links is dependent on the geographical distance between the nodes that they connect. Based on the findings in Chapter 3, we categorize the nodes in the network as large, medium and, small depending on

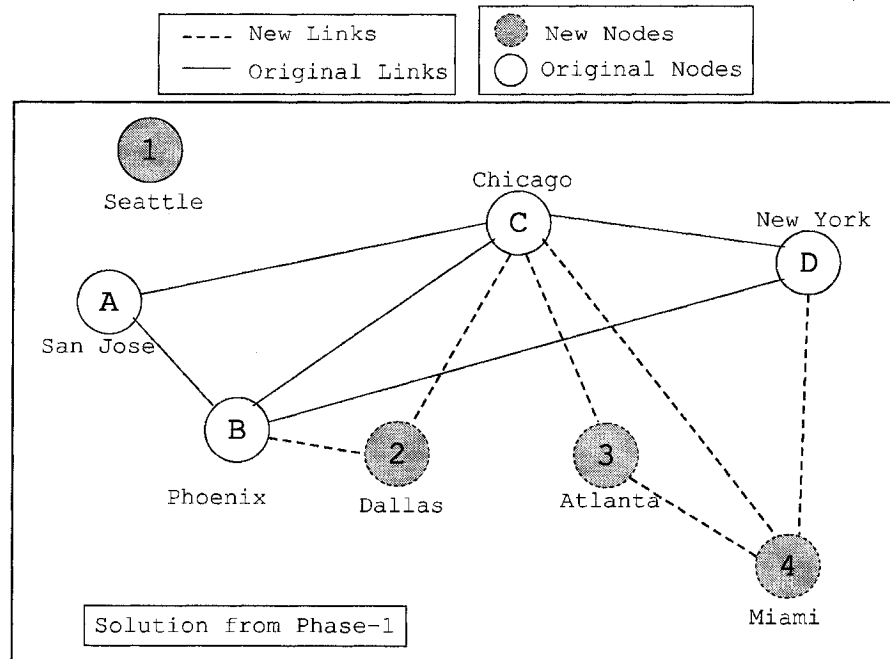


Figure 5.2: Final Solution from Solving the Non-linear Program in Phase-1 with Specific Network and Budget Constraints

the amount of traffic they generate (typically in the ratio 4:2:1) [56]. In our example we consider nodes C and D as large nodes, B as a medium node, and A as a small node. We generate our traffic matrix based on this assumption. Another important consideration in the network is the BGP prefix distribution among different nodes, i.e., BGP exit points to reach different prefixes in the Internet. SD time and TD for different nodes depend on the prefix distribution in the network [56]. We model the prefix distribution to incorporate the elephant and mice phenomenon where 80% of the traffic is destined to 20% of the prefixes. Finally we assume equal link weights for all the links in the network reducing the routing to minimum hop routing.

As a starting point, we identify four distinct locations (1, 2, 3, and 4) where new nodes can be added to the original network (see Figure 5.1). We assume that nodes 1 and 4 are large nodes, 3 is a medium node, and 2 is a small node. We also assume that an ISP can identify the

potential locations for adding nodes and estimate the traffic and revenue for each of the locations. Note that these estimations could come with a number of uncertainties and dependencies among each other. However, as a first step, we assume that the traffic and the associated revenue generated by a node is independent of other nodes in the network.

In our simple example, we assume: $\mathbf{r} = \{45000, 25000, 38000, 50000\}$, $\mathbf{nc} = \{14000, 13000, 13800, 14000\}$, $lc = 1$, and $b = 50000$. Our first objective (i.e. Phase-1) is to determine which of the four potential nodes should be added to the network to maximize revenue and how should they be linked to the other nodes while ensuring that the budget, SD time, TD, node degree, and link utilization constraints are satisfied. We use the Tabu Search technique to explore the final solution. Since the problem is small, we also use a brute force approach to confirm that the above solution is in fact the global optimal solution. The final solution that we obtained for Phase-1 is shown in Figure 5.2. We can see that only three of the four potential nodes were included.

Typically network operators use maximum link utilization in the network as a measure of network performance. However, link utilization does not consider the network performance from users' perspective and hence optimization decision based solely on link utilization could result in performance degradation for some/all customers. To reinforce this concept, we find the solution to the problem in Phase-1 by ignoring the SD time and TD constraints, and use only link utilization and node degree as constraints for network performance. This approach will try to find a solution (referred to as *utilization solution*) that minimizes the maximum link utilization in the network among all the feasible solutions that maximize the revenue.

In Figure 5.3, we compare the SD time and TD values for nodes A, B, C, and D before and after the upgrade (using the utilization solution). We compute the utilization solution by varying

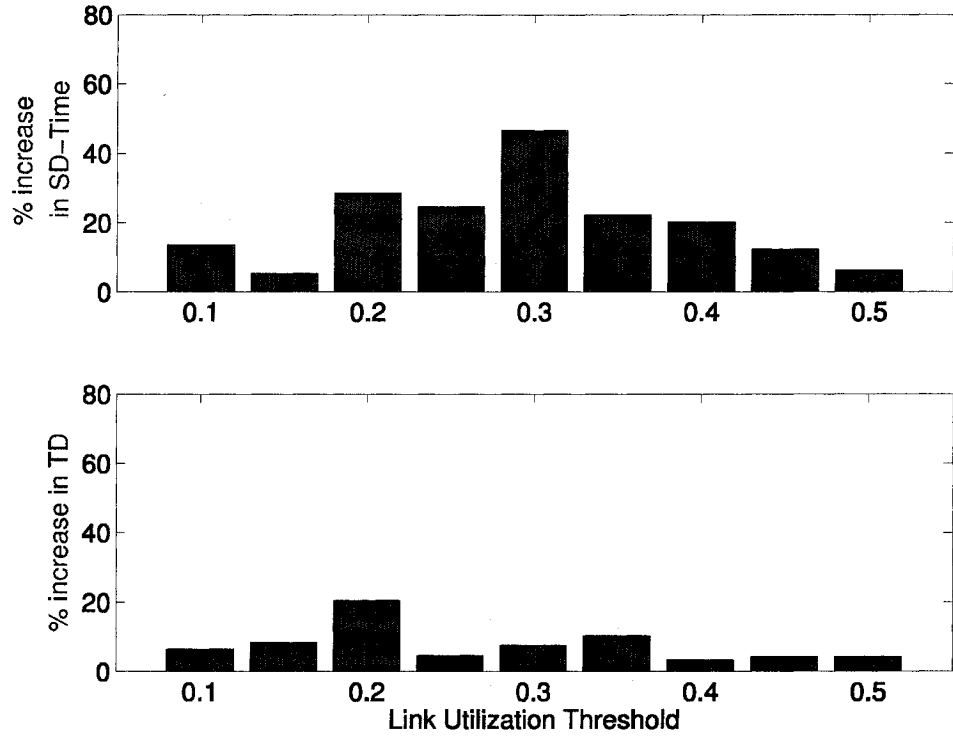


Figure 5.3: Impact on Service Disruption Time and Traffic Disruption when Using Maximum Link Utilization and Node Degree as Metrics to Estimate Network Performance.

α (the maximum link utilization threshold value) between 0.1 and 0.5, a range used in real world networks. We can see that, in the case of our simple network, SD time could increase up to 46% and TD could increase up to 20%. These values could vary significantly depending on the operating network conditions. The main conclusion from Figure 5.3 is that it is important to consider SD time and TD constraints while performing network upgrade.

Now that we have determined the optimal end state, our next objective is to determine an optimal strategy for multistage node addition. Here we assume that the ISP adds one node at a time, along with all the links associated with that node. We implemented the dynamic programming solution for Phase-2 using Matlab that takes the initial and final network conditions as input, and outputs the optimal sequence to reach the final stage from the initial stage.

All Possible Solutions	Stage- $\{1,2,3\}$	Total Cost
<i>Solution-1</i>	2, 3, 4	1056.76
<i>Solution-2</i>	2, 4, 3	16.74
<i>Solution-3</i>	3, 2, 4	1090.33
<i>Solution-4</i>	3, 4, 2	1065.42
<i>Solution-5</i>	4, 2, 3	1.654
<i>Solution-6</i>	4, 3, 2	5.503

Table 5.2: All Possible Upgrade Solutions for Phase-2 (Note: $\omega_{SD} = \omega_{TD} = \omega_{util} = 2$ and $\omega_{degree} = 5$)

Multistage Solution	With SD time and TD		Without SD time and TD	
	Node Added	Links Added	Node Added	Links Added
<i>Stage-1</i>	4	4 - C, 4 - D	4	4 - C, 4 - D
<i>Stage-2</i>	2	2 - B, 2 - C	3	3 - 4, 3 - C
<i>Stage-3</i>	3	3 - 4, 3 - C	2	2 - B, 2 - C

Table 5.3: Multi-Stage Network Upgrade Solution for Phase-2, With and Without Using SD time and TD as Metrics for Network Performance.

In our current example, we need to add three nodes and six links into the network. Hence we need two intermediate stages to reach the final stage. Note that there are 6 different ways in which this can be accomplished. All possible solutions for multistage network upgrade are shown in Table 5.2.

In the final solution from Phase-1, node 3 is connected to nodes 4 and C (see Figure 5.2). If 3 is added into the network before 4 then it results in very high network cost (Equation 5.5) since the degree of node 3 will be less than 2. Hence the optimal solution should avoid adding node 3 before node 4. This eliminates 3 out of the 6 possible solutions (see Table 5.2). Close observation of the remaining 3 solutions from the dynamic programming output indicates the following: adding node 2 before node 4 results in very small degree and link utilization costs, but results in high SD time and TD costs due to high network convergence time after link failures. This could be

attributed to the distribution of exit points in the network for various prefixes. Note that given different operational conditions (like traffic matrix and BGP prefix distribution), this condition may not be true. Finally adding node 3 before node 2 also resulted in high SD time and TD costs. Hence, for this example, the optimal solution is as shown in Table 5.3. However, ignoring SD time and TD costs in the DP formulation resulted in a different sequence for node addition. The cost for this sequence is much higher than the original sequence (see Table 5.2, Solution-5 and Solution-6). Note that although the sequence was determined by ignoring SD time and TD cost, the final cost includes the SD time and TD costs to emphasize the importance of considering user-centric performance metrics during the upgrade process.

It is important to note that the outcome discussed above depends on the operational network conditions and will vary depending on the state of the network. Our main focus in this chapter is to show that the two-phase approach provides a framework that helps in finding an optimal strategy for network upgrade.

5.4 Summary

In this chapter, we proposed a two-phase framework to determine the optimal network upgrade strategy. To the best of our knowledge this is the first piece of work to address this problem. We showed that it is important to consider the performance perceived by customers during the upgrade process. We also used an example to show the feasibility and advantages of our proposed framework.

Although we have made some simplifying assumptions, we believe that this is the first step towards developing a more comprehensive framework for the network upgrade process. However,

it is straightforward to relax these assumptions. For example, the estimated traffic and revenue could dynamically change over time and could have several interdependencies. We also ignored the situation where the budget becomes available sequentially. Such changes can be easily incorporated into our framework by using adaptive and stochastic dynamic programming techniques that can adjust the optimal path after every stage based on the current state and the estimated values of other inputs.

Another important point to note is that our framework does not require closed form expressions for network performance metrics (i.e. different constraints and costs). The same framework will still apply when the performance metrics need to be computed using complex algorithms. In practice, simulation techniques can be combined with our multistage dynamic programming framework to efficiently compute optimal solutions. We believe that the two-phase framework presented in this chapter is powerful yet flexible. Finally, we also believe that, with minor modifications, this framework can be directly applied in different network scenarios such as wireless cellular networks.

Chapter 6

Cross-Layer Interactions Between Overlay and IP Networks

Overlay networks have emerged as a promising platform to provide customizable and reliable services at the application layer to support multicast (e.g., SplitStream [28]), content delivery (e.g., Akamai [1]), resilient connectivity (e.g., RON [16]), and distributed hash table services [81, 98], among others.

Overlay networks typically consists of pre-selected nodes, located in one or multiple network domains, that are connected to one another through application-layer routing. One of the underlying paradigms of overlay networks is to give applications more control over routing decisions, that would otherwise be carried out solely at the IP layer. The advent of a wide variety of active measurement techniques has made this possible. An overlay network typically monitors multiple paths between pairs of nodes and select one based on its own requirements of end-to-end delay, loss rate, or throughput.

6.1 Hypothesis and Problem Statement

Allowing routing control at both the application and the IP layers could have profound implications on how Internet Service Providers (ISPs) design, maintain, and run their networks. The network architecture, along with the set of algorithms and *traffic engineering* (TE) policies that ISPs use, are based on certain assumptions about how their customers and traffic behave. Overlay networks could call into question some of these assumptions, thus rendering it more difficult for ISPs to achieve their goals. For example, it is very important for an ISP to perform load balancing across *all* of its links to ensure good performance and also to limit the impact of failures. We hypothesize that the co-existence of multiple overlays could severely hamper an ISP's ability to achieve this goal.

Another TE issue stems from an ISP's need to estimate its *traffic matrix* (TM). ISPs need to understand the traffic demands for many TE tasks such as capacity planning, reliability analysis, and link weight assignment. We believe that overlay networks could make an ISP's job of estimating its TM more difficult, resulting in erroneous decisions by the ISP.

In addition, routing decisions at two independent layers could lead to short-term or long-term traffic oscillations. If overlay networks react to events in the IP network (e.g., failures or congestion) independently of an ISP, race conditions could occur and lead to traffic oscillations. Such traffic oscillations not only affect the overlay traffic but also impact the background (non-overlay) traffic in the network. We hypothesize that such reactions by overlay networks that span multiple domains could threaten the effectiveness of BGP in isolating different domains in the Internet.

These critical issues raise an important question: *Can overlay networks and underlying IP networks form a synergistic co-existence?* Given the increasing popularity of overlay networks,

it is critical to address issues that arise from the interaction between the two layers. We hypothesize that it could be problematic to have routing control in two layers, when each layer is unaware of things happening in the other layer. ISPs may neither be aware of which nodes are participating in an overlay, nor their routing strategies. Overlay networks are not aware of the underlying ISP's topology, load balancing schemes, or timer values for failure reaction.

Qiu et al. [77] describe the interactions between overlay networks and ISPs after the routing control mechanisms reach the Nash equilibrium point. Here, we explore some of the issues that arise due to dynamic interactions in the presence of unexpected or unplanned events such as network failures. We believe that the dynamic network behavior in the presence of failures (that are common, everyday events for ISPs [52]) is very important for ISPs and needs to be addressed.

In the rest of the chapter, using simple illustrations, we show that these dynamic interactions could be problematic and make the case for subsequent research in this direction. It is important to note this work is not about network performance issues measured by the classic metrics such as loss, delay, or throughput. Instead, we are interested in understanding how the network management techniques currently deployed by ISPs are impacted by overlay networks. We believe that ISPs need to clearly understand the implications of overlay network behavior, and if needed, develop mechanisms or new services in order to handle traffic from overlay networks.

6.2 Modeling and Simulating Routing Strategies

To quantify the interactions between the overlay layer and the underlying IP layer, we built a Java-based control plane simulator to analyze (a) the conflicts in decisions made by two different layers and (b) the impact of such decisions on the data traffic.

6.2.1 Overlay Network Dynamics

While different overlay networks designed for a wide range of applications may differ in their implementation details (e.g., choice of topologies or performance goals), most of them provide the following common set of functionalities: path/performance monitoring, failure detection, and restoration. In our simulation model, we attempt to capture the most generic properties of an overlay network:

- Most overlay routing strategies select a path between a source-destination pair with the best performance based on end-to-end delay, throughput, and/or packet loss. Our model assumes the overlay path with the shortest end-to-end delays will be selected, but can be extended to include other metrics.
- Most overlay networks monitor the actively used paths by sending frequent probes to check if the paths adhere to acceptable performance bounds. If the probing event detects a problematic path (due to failures, congestion, etc. at the IP-layer), then the overlay network sends probes at a higher rate to confirm the problem before selecting an alternate path. Our model assumes regular probes are sent out every P seconds. If a probe does not receive a response within a given *timeout* (or T) value, then the path is probed at a higher rate (every Q seconds). If a path remains bad after N such high frequency probes, the overlay will find an alternate path (or the next best path) between the source and destination nodes. For instance, RON [16] can be modeled with $P=12s$, $Q=3s$, and $N=3$ while Akamai network can be modeled with much smaller values of P , Q , and N [61]. As soon as an alternate path is found, the traffic is moved to the alternate path, which is now probed every P seconds to ensure that it is healthy¹.

¹As long as the current path adheres to the performance bounds, an overlay does not shift traffic to an alternate path even if the alternate path starts to exhibit better performance.

Note that all the parameters, P , Q , N , and *timeout* (or T), are configurable and can be set to different values to simulate the routing strategies of different overlay networks. In our simulations, multiple overlay networks can be simulated, each with its own topology and routing strategy.

6.2.2 IP-Layer Routing Dynamics

Within each domain, we emulate an IP-layer *interior gateway protocol* (IGP) that implements Dijkstra's shortest path algorithm. We generate link failures and model IGP dynamics in response to failures as outlined in [52] and Chapter 3.

In any IP network, the link utilization level determines the delay, throughput, and losses experienced by traffic flows traversing the link. To simulate realistic link delays, we use a monotonically increasing piecewise linear convex function similar to [39] and [77]. Note that in all our test scenarios in this chapter, we assume that a load of 50 units on a link is the threshold value beyond which it exhibits very high delay values.

In the following sections, we identify numerous issues that result in potentially harmful interactions between the IP layer networks and the overlay networks, and explore each of them in detail to make the case for further research in this area.

6.3 Challenges to Traffic Engineering (TE)

ISPs deploy TE techniques to control how the traffic flows in the network and balance load across the network. In general, overlay networks attempt to bypass ISP-dictated path and find alternate paths that maximize their own performance. Qiu et al. [77] have shown how *selfish* routing

in overlay networks may have a negative impact on intra-domain TE, i.e., overlay networks tend to overload links on the shortest paths.

ISPs apply TE mainly in reaction to changes in the topology (due to link failures [39, 72]) or traffic demands (due to flash crowd events or BGP failures). In these scenarios, a common way for ISPs to manage the traffic is by changing the IGP link weights. ISPs make two assumptions while using this technique: (i) traffic demands do not vary significantly over short timescales, and (ii) changes in the path within a domain do not impact traffic demands. Overlay network routing defeats these assumptions as illustrated below.

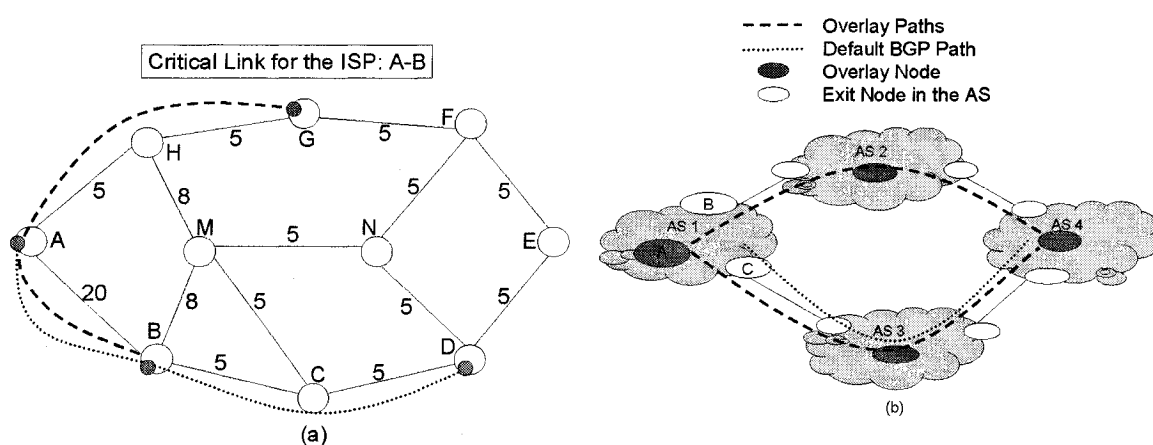


Figure 6.1: (a) Illustration Network 1. (b) Illustration Network 2.

6.3.1 Traffic Matrix Estimation

ISPs employ techniques such as [86] to compute their TM i.e., a matrix that specifies the traffic demand from origin nodes to destination nodes in a network. A TM is a critical input for many traffic engineering tasks, such as capacity planning, failure analysis, link weight setting, etc.

Conventionally, overlay nodes typically encrypt the information about the final destination

in the data packets and hence the IP layer is unaware of the true destination. Traffic between two overlay nodes could traverse multiple overlay hops. At each hop, traffic exits the IP network and reaches the overlay node, which deciphers the next overlay hop information and inserts the traffic back into the IP network. Consider the network in Figure 6.1(a), which has multiple overlay nodes (at A , B , D and G) in a ISP domain. Suppose the traffic between nodes A and D is 10 units. If IP routing is used then the TM entry for the source-destination pair $A - D$ is 10. But if overlay routing intervenes and decides to route the traffic through an overlay path (say, $A - B - D$) which offers better latency, then the TM entry for the pair $A - D$ is *duplicated* as two entries of 10 units each, one for $A - B$ and another for $B - D$, while the value for the entry $A - D$ is 0. This implies that overlay networks could introduce dynamism in TM values, thus requiring the ISP to perform frequent estimation of TM to maintain its accuracy.

There are many flows whose ultimate destination lies outside the ISPs' domain; the traffic from these flows traverses the ISP and thus appears inside the TM. The traffic matrix will specify an exit router within the ISP's domain for such flows. If this traffic belongs to an overlay network that spans multiple domains, and uses its own path selection mechanism, then the exit point within a single ISP domain could change, resulting in a *shift* in TM entry. For example, consider the network in Figure 6.1(b). Suppose that the layer-3 path from $AS1$ to $AS4$ is through $AS3$. If the overlay network discovers a better path through $AS2$, then the overlay network could switch the routing path, thus changing the associated exit point. The TM entry in $AS1$ for $A - C$ now *shifts* to $A - B$. If this were to happen for large flows, it could affect a significant portion of the TM. If this were to happen often, it would increase the dynamic nature of the TM. TM entry *duplication* and *shift* are two examples that warrant frequent TM updates.

6.3.2 Load Balancing

Another important traffic engineering task is to define intra-domain routing policies. For example, considering again the network in Figure 6.1(a). Suppose that the ISP has two important customers connected to nodes A and B , and hence assigns a high link metric to link $A - B$ (as shown in Figure 6.1(a)) to discourage the use of this link by other node pairs. Suppose that node D wants to send traffic to node A . Using IP-layer routing, the path traversed by the traffic would be $D - N - M - H - A$. However, the overlay node D can choose to reach A by forwarding through another overlay node, such as B . In this case, the path $D - C - B - A$ is used. Similar choices can be made for the traffic from A to D , B to G and G to B . This undermines the ISP's intent, and an ISP could thus erroneously assume that the majority of resources on link $A - B$ are used by its two important customers. The impact of bypassing routing or load balancing policies could be magnified when multiple overlay networks co-exist and make independent decisions.

6.4 Traffic Oscillations

Overlay networks attempt to provide “enhanced” services to applications by routing their traffic through paths that adhere to strict performance constraints. A degradation in path performance will trigger overlay networks to find an alternate path that satisfies the performance constraints and re-route the traffic accordingly. If multiple overlays co-exist, then a performance degradation event will trigger a reaction in all overlays traversing the same problematic spot in the network. Two or more overlays reacting at moments that are close in time can result in race conditions. Having routing control in two layers in the network is equivalent to having two closed loop systems reacting simultaneously yet independently to the same set of events. This is a classic situation for

race conditions that lead to traffic oscillations.

We will see that there are a number of events that trigger oscillations, such as link or node failures, IGP convergence, and congestion (high loads). There are also a number of ways and/or events that cause oscillations to stop, such as self-disentangling (explained below) and failure restoration. The different combinations of such start and stop triggers means that there exists a variety of scenarios in which oscillations occur and that oscillations can last for varying amounts of time, some short and some long.

The co-existence of multiple overlays has not been well explored. We believe that it increases the likelihood of harmful traffic oscillations that could affect a significant portion of traffic due to race conditions. We consider three simple examples to explore these ideas on race conditions and multiple overlays. Although our examples consider small test topologies to identify and illustrate different possible interactions, our observations would apply to large networks when a sub-graph of the network resembles our test topologies.

We first consider a scenario with two overlay networks on top of a single ISP domain (*Scenario 1*, Figure 6.2(a)). The IP network consists of 7 nodes and 8 links while the two overlay networks on top of the IP network have 4 nodes each with mesh connectivity. The numbers on the links in Figure 6.2 represent the link loads. While the numbers enclosed in the boxes and circles represent the overlay traffic load, the non-enclosed numbers represent the background traffic in the IP network. We assume that the only traffic in both the overlay networks is from node *A* to node *D* and is equal to 20 units each. The timer values (as in Section 6.2) used for the overlay networks are shown in Table 6.1. Note that these values are similar, but not the same, for both overlay networks. For example, two overlays that both cater to video streaming could end up with similar application

Scenario-1				
Timer	P	Q	N	T
Overlay-1	310	150	3	100
Overlay-2	300	150	3	120
Scenario-2				
Timer	P	Q	N	T
Overlay-1	500	150	3	100
Overlay-2	500	150	3	110
Overlay-3	200	150	3	100

Table 6.1: Timer Values for Overlay Networks in *Scenario 1 and 2*

level timer values.

Initially the overlay traffic between nodes A and D traverses the IP path $A - C - D$ (Figure 6.2(a)). Notice that the link loads on all the IP links are below the congestion threshold value of 50 units. Now consider the event that link $A - C$ experiences a failure. If both overlay networks react faster than the IP layer, they might independently decide to move their traffic to the top path. This can happen if previous probing indicated that the path $A - B - H - D$ offers better performance than the bottom path $A - E - F - D$. In other words, the first overlay network decides to reroute its traffic through B and the second overlay network decides to reroute its traffic through H without being aware of each other's traffic shift.

This in turn results in very heavy load on links $A - B$ and $H - D$. Both the overlay networks react to this congestion and find a new path to reach the destination node D from A . Again both the overlay networks decide to move the traffic to the bottom path at nearly the same time, as shown in Figure 6.2(c). These traffic shifts create overload on links $A - E$ and $F - D$. Once again both overlay networks react to this by re-routing the traffic back to the top path. This results in traffic oscillations between the top and bottom paths until one of the overlay networks reacts and

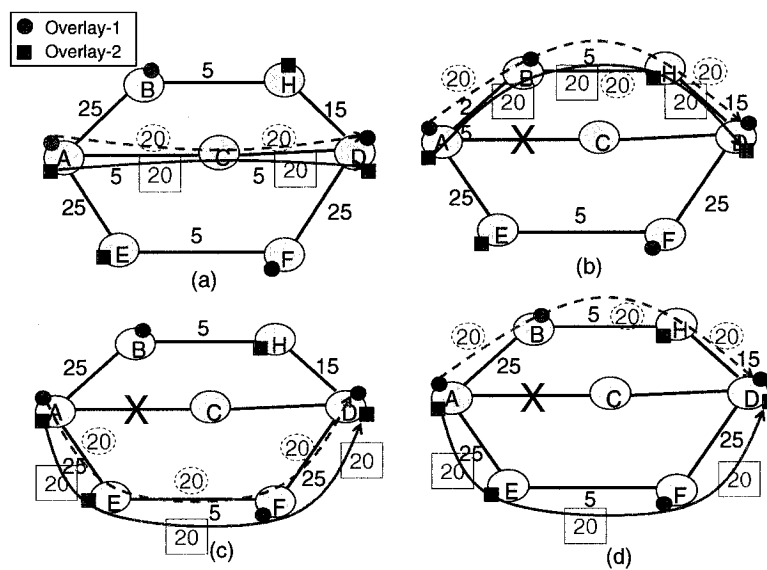


Figure 6.2: Scenario 1 - Two Overlay Networks in One Domain

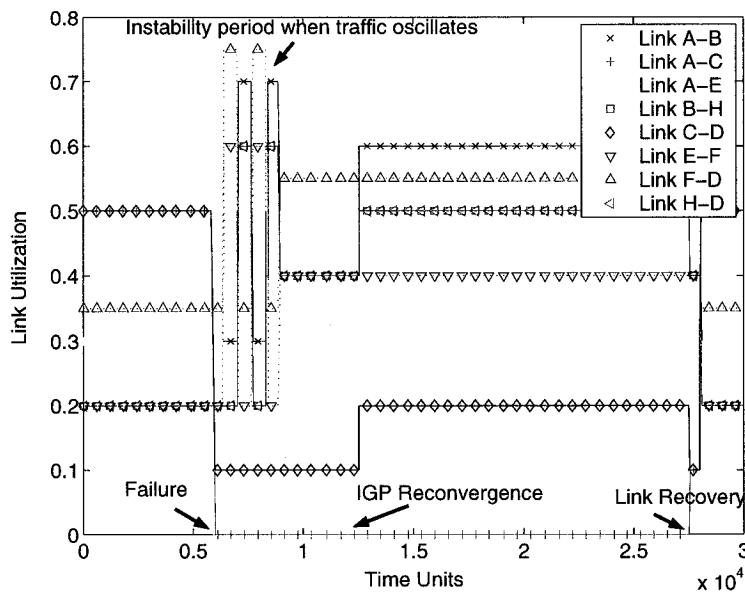


Figure 6.3: Link Loads on Various Links in the IP Network for Scenario 1

reroutes traffic faster than the other (Figure 6.2(d)), thus breaking the deadlock.

The resulting oscillations for *Scenario 1* are depicted in Figure 6.3 that shows the utilization of various links in the IP network across time. We see that soon after the link failure event, loads on links $A - B$, $A - E$, $B - H$, $E - F$, $H - D$ and $F - D$ start oscillating. In this case, the oscillations stop before the IGP protocol converges. Note that the same reaction could have been triggered by a flash crowd event rather than a failure. A traffic surge targeting a server at node D can create sufficient congestion to seriously degrade the path quality between $A - D$ and thus engender the same reaction.

This type of scenario can happen when the path probes from the two overlays end up being spaced close in time; in essence the two overlay networks can get *synchronized* in their detection of “better alternate paths” and in their traffic shifting. In this case the traffic shift in one overlay network is not visible to the high frequency probes of the other. Even though probes in different overlays are initiated at different points in time, it is possible for them to get synchronized. If the inter-probe times are similar, but not exactly the same, then over time the alignment between two such probes will grow, separating them out enough to break the synchronization. At this point, one overlay will detect congestion and move its traffic before the second one does. If the second overlay detects the drop in load (since the first one moved) via its high frequency probes, then it no longer moves. When this happens, we say that the two overlays *disentangle* themselves. Our observation that overlays can become synchronized reveals a behavior similar to that presented in [37] on general synchronization of periodic routing messages. From that work we surmise that the cause of synchronization may lie in the periodic nature of the overlay probing processes.

We now consider *Scenario 2*, with the same IP network, but add one more overlay network

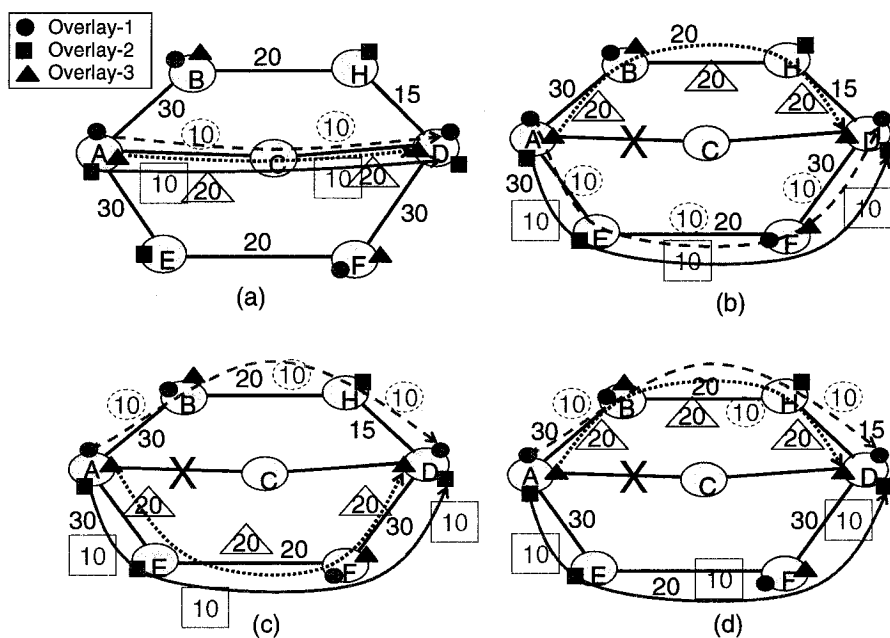


Figure 6.4: Scenario 2 - Three Overlay Networks in One Domain

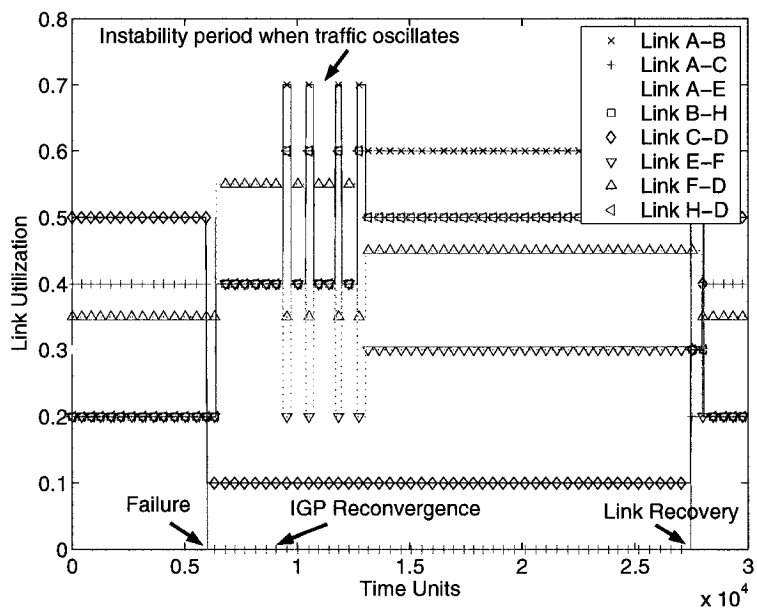


Figure 6.5: Link Loads on Various Links in the IP Network for Scenario 2

as shown in Figure 6.4(a). Similar to *Scenario 1*, the only overlay traffic in all the three networks is between nodes A and D . Table 6.1 shows the timer values for the three overlay networks. The third overlay network probes the network more often than the other two, thus reacting faster to performance degradation events.

Figure 6.4(b) shows the network state after the link $A - C$ fails. Note that the third overlay network reacts first and chooses the top path while the other two networks choose the bottom path. At this point in time, none of the links are overloaded and all the overlay networks have found a stable path. However, after the IGP protocol converges, the new layer-3 path selected from node A to node D is $A - B - H - D$. The first two overlay networks are oblivious to the fact that the underlying path between the source and destination nodes has changed and hence assume that the original overlay link $A - D$ has recovered. The networks start probing the overlay link $A - D$ (i.e. the IP path $A - B - H - D$) and find that it offers better performance than their current path (i.e. $A - E - F - D$) due to the fact that the links along the path $A - B - H - D$ are less loaded than $A - E - F - D$. The first overlay network reacts faster than the second overlay network and moves to the new path. This results in a situation as shown in Figure 6.4(d), where the first and third overlay networks use the top path and the second overlay network uses the bottom path. This overloads the link $A - B$ and the third overlay network reacts to this link overload faster than the first overlay network, resulting in the situation depicted in Figure 6.4(c). This traffic shift overloads link $A - E$, leading to oscillations. The oscillations stop when the overlay networks land in the situation where the first two overlay networks use the same path and the third one uses the other path (Figure 6.4(b)).

Our simulation tracks the dynamic evolution of such a scenario. Figure 6.5 shows the

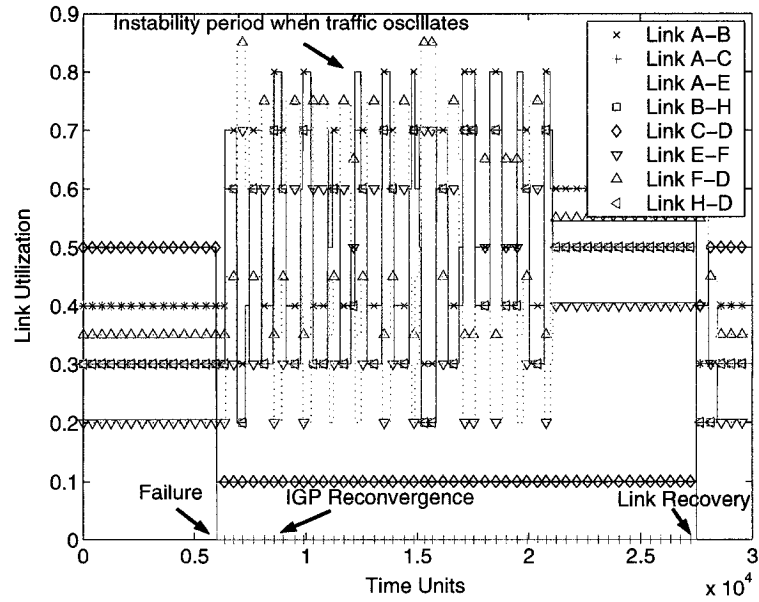


Figure 6.6: Link Loads on Various Links in the IP Network for *Scenario 2*

utilization of various links during the oscillations. We see that the network is stable until IGP re-converges, after which loads on several links start oscillating and continue until the overlay networks disentangle themselves. Note that in this scenario the trigger for oscillations was the IGP convergence event, whereas in the previous scenario the trigger was the failure itself.

Whether or not oscillations happen, and how long they last, depend very much upon the arrival times and temporal inter-spacing of the probes at the failure event. We considered a variation of *Scenario 2* in which we altered the random start times of each of the overlay probes, and changed the link loads slightly. Due to the way the probes ended up being aligned, oscillations occurred as depicted in Figure 6.4(c). Soon after the link failure, loads on many links start oscillating, as shown in Figure 6.6. These oscillations continue even after the IGP re-convergence event, and stop only when the overlay networks disentangle themselves and reach a state similar to the one shown in Figure 6.4(b).

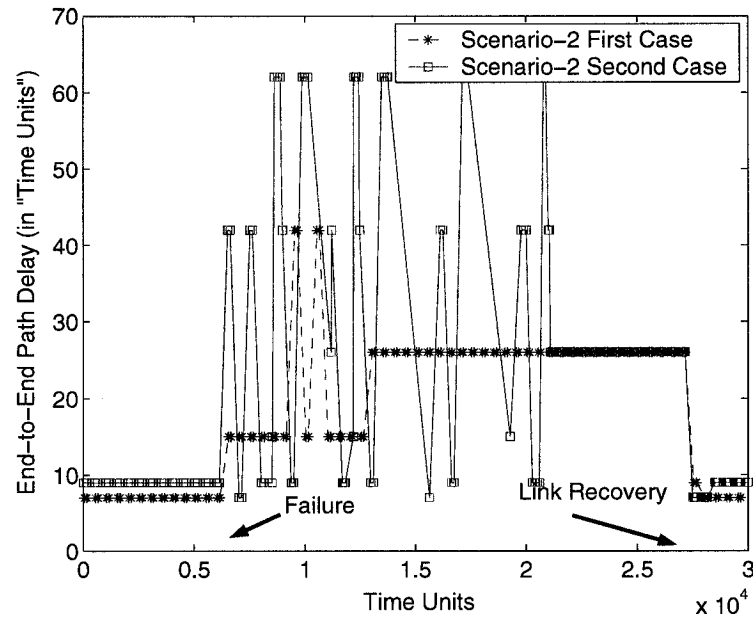


Figure 6.7: End-to-End Delay for Path A-H in *Scenario 2*

From these simple examples, we conclude: (1) Traffic oscillations involving multiple overlay networks can be short-term or long-term depending on the network state, overlay timer values and their synchronization at the time of the event that triggers oscillations. (2) A variety of events occurring at the IP layer like IGP re-convergence, failure recovery, self-disentangling, etc., can influence the start or stop of oscillations at the overlay layer. We explore overlay network synchronization and traffic oscillations in detail in the next chapter.

Overlay networks route traffic independently of the underlying IP network, ignoring the potential impact on the background traffic. Figure 6.7 shows the end-to-end delay experienced by the traffic from *A* to *H* in *Scenario 2*. The background traffic experiences highly variable end-to-end delay that could result in jitter, service disruption, and packet losses. This is a serious problem for ISPs since they are held responsible for the performance of *all* their customer traffic, including

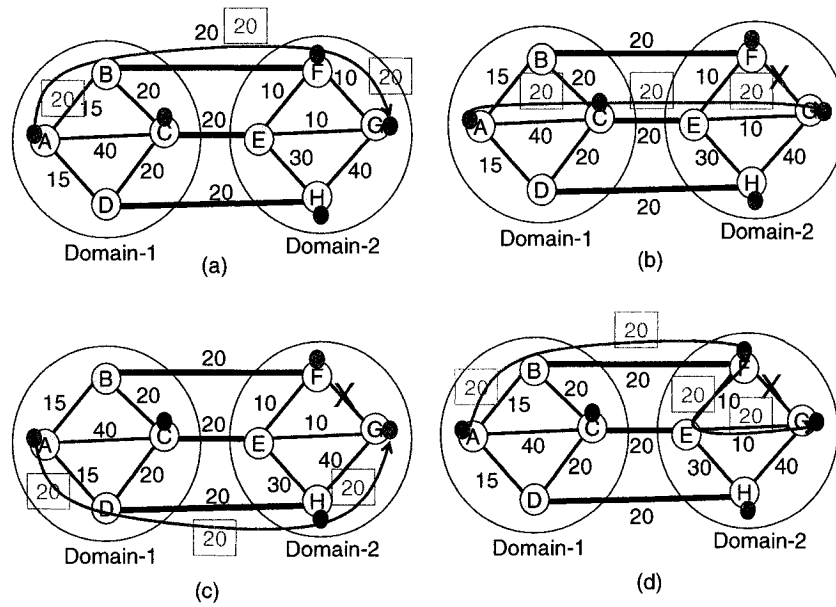


Figure 6.8: Scenario 3 - One Overlay Network in Two Domains

non-overlay traffic.

6.5 Coupling of Multiple AS Domains

We next consider a scenario with a single overlay network that spans multiple domains (*Scenario 3*, Figure 6.8(a)). The overlay path between the nodes A and G is $A - F - G$ and the underlying path is $A - B - F - G$. Now consider the event that link $F - G$ fails in 'Domain-2'. Figure 6.8(b) shows the state of the network soon after this failure. The new overlay path between A and G is $A - C - G$. This results in overloading the link $A - C$. The overlay network reacts to this overload and finds another alternate overlay path through H (Figure 6.8(c)). This traffic shift overloads the link $H - G$ and hence the overlay network moves the traffic back to the overlay path $A - C - D$. This results in traffic oscillations in 'Domain-1' (and 'Domain-2') until the IGP

converges in ‘Domain-2’. At this point, the overlay network finds the stable overlay path $A - F - G$ and the oscillations stop (Figure 6.8(d)).

From this scenario we can infer: (1) When an overlay network spans multiple domains, the network state in one domain could influence the network behavior in another domain. (2) One of the aims of BGP is to decouple different domains so that events in one domain do not affect another domain. Overlay networks can defeat this objective by inducing a coupling of the two adjacent ASes (e.g., two ISPs). An event in one domain may result in traffic oscillations in the other.

6.6 Discussion

We have identified several problematic interactions that can occur between IP networks and overlay networks: (i) traffic matrices become more dynamic and more ambiguous, making them harder to estimate; (ii) some types of load balancing policies can be bypassed and (iii) different ASes can get coupled due to per-domain events; (iv) multiple overlays can get synchronized, which in turn leads to traffic oscillations that can last for varying amounts of time (discussed in detail in the next chapter); and (v) oscillations can impact non-overlay traffic.

We believe that it is imperative for ISPs to have better knowledge about overlay networks to cope with the interactions and provide good service to *all* its customers. The scenarios raised here imply that traditional traffic engineering techniques may not be sufficient for this purpose when overlay networks become widespread. In order for ISPs and overlays to form a more synergistic co-existence, ISPs may want to think about how to design incentives for overlay applications to avoid behaviors that are problematic for carriers. Similarly it is important for overlay networks to understand the implications of their routing strategies and adopt mechanisms to curb harmful

interactions.

When the original Internet was developed, it was designed as an overlay on top of X.25 networks. Although we may try to draw some lessons learned by revisiting that history, we suspect the analogy may be thin for two reasons. First, the Internet was designed as a single overlay, whereas now we are looking at a situation with potentially many overlays co-existing. Secondly, the goal of the Arpanet was to provide an additional service (i.e., connectionless best-effort communications) that the underlying networks were not providing. The routing control and failure restoration in overlays are *competing* with the same kind of services offered at the IP layer. We could also draw upon lessons learned from the design of IP over SONET or DWDM networks [15, 52] in which the division of labor is carefully thought out such that each layer is responsible for different kinds of failures.

We thus believe that the types of issues raised herein should be addressed within the research community sooner rather than later. Some interesting topics for further examination in the area of integration between ISPs and overlay networks include:

- Evaluate the benefit of using an overlay service compared to augmenting IP-layers with other route control techniques, such as multi-homing as in [11].
- Develop a formal framework for understanding overlay network behavior that would help distinguish general problems versus pathological cases.
- Define mechanisms to allow different overlay networks to share routing and performance information among themselves. Such coordination among networks (similarly to how BGP allows coordination between ISPs) could help avoid synchronization events.

- Define mechanisms to share information between the overlay network and the IP layer so that the two networks can be aware of the events and state information affecting each other.

In the next chapter, we address some of the above issues. We examine very realistic network scenarios and show that the interactions between overlay and IP networks are not pathological. We also develop a formal framework to understand and analyze the behavior of multiple co-existing overlay networks. Using this framework we show that it is necessary to explicitly address the problem of race conditions between different overlays while designing the overlay protocols.

Chapter 7

Race Conditions In Coexisting Overlay Networks

As discussed in the previous chapter, numerous application-layer overlay networks are being deployed today in the Internet and the volume of traffic that they carry is increasing [88]. Since most overlay networks are designed independently with different target applications in mind, our suspicion is that as overlay traffic load increases, different overlays may unintentionally interfere with each other. It is therefore very important to examine the impacts of the co-existence of multiple overlay networks when their traffic represents a significant, if not dominant, portion of the total traffic.

7.1 Motivation and Problem Statement

Based on our results in Chapter 6, we arrive at a hypothesis that two (or more) co-existing overlays can experience race conditions and become synchronized leading to route and traffic os-

cillations, and cascading reactions (i.e., an event in one overlay can trigger a series of events in coexisting overlays). This hypothesis is formulated based on two key observations. First, in the work by Floyd et al [37], the authors discuss how there are many examples of seemingly independent periodic processes in the Internet that can inadvertently become synchronized. They warned that the phenomenon of inadvertent synchronization of period processes would most likely become an increasing problem in computer networks. Typically overlay networks use a periodic probing process to detect events that deteriorate path performance, and to identify alternate paths between source and destination pairs. IP layer events such as failures can trigger an overlay network to move its traffic to an alternate path. The work in [37] would suggest that a situation in which two different overlays, both using *periodic* probing, that can react to the same IP layer triggering event, is a candidate scenario for the synchronization problem. A second motivation for suspecting that synchronization might arise comes from control theory. Different overlays are simultaneously and independently conducting routing control at the application layer. This corresponds to a situation in which multiple independent control loops coexist, yet react to the same events (e.g, failures). This is a classic situation for race conditions. This dissertation seeks to explore this hypothesis and to quantify the likelihood of oscillations and cascading reactions, how long they can last, and the conditions under which they occur.

7.2 Previous Work in Multiple Overlay Interactions

Interactions between multiple co-existing overlay networks were first addressed by Qiu et al. [77], where the authors investigate the performance of selfish routing after the system reaches the Nash equilibrium point (when network-level routing is static). They also show that selfish routing

can achieve optimal average latency at the cost of overloading certain links. Liu et al. [62] model the interaction between overlay routing and IP traffic engineering as a two-player game, where the overlay attempts to minimize its delay and IP traffic engineering tries to minimize network cost. In our current work, we focus instead on dynamics of the overlay routing layer before the system reaches the equilibrium. Instead of static network-layer routing, we consider events such as link/router failures, flash crowds, and network congestions that lead to dynamic re-computation of routes in overlay applications and/or IGP protocols.

Other works have studied the overlay network probing process, a crucial component of overlay routing. Nakao et al. [70] proposed a shared routing underlay that exposes large-scale, coarse-grained static information (e.g., topology and path characteristics) to overlay services through a set of queries. They advocate that the underlay must take cost (in terms of network probes) into account and be layered so that specialized routing services can be built from a set of basic primitives. However, sharing network-layer path information may induce synchronized routing decisions in overlay networks and unintentionally lead to route/traffic oscillations, an aspect not addressed in [70]. We hope to shed some light on this problem through our modeling of overlay and IP-layer dynamics in response to failures.

While our work was in part inspired by the work in [37], the particular periodic process we focus on is different from the one considered in [37]. They focused on routing protocols such as EGP, IGRP, and RIP that send a periodic update message to ensure routing tables are kept up to date. The process we explore uses two types of periodic probes and only reacts to external triggers. Also, they study the scenario of many routers participating in the same protocol, whereas as we consider two different instances of the protocol, with non-identical parameters that only partially overlap in

the underlying physical network.

7.3 Simulating Multiple Co-existing Overlays

To validate our hypothesis (and the analytical model presented later in Section 7.5), we used the simulator that we described in Chapter 6. The simulator implements the control planes at both the overlay and IP layers. It allows us to analyze the conflicting decisions made by multiple overlays and their impact on both overlay and other IP traffic. We model the overlay networks, for both simulations and analysis, using configurable parameters P , Q , N , and T , as described in Chapter 6.

7.3.1 Simulating IP and Overlay Network Dynamics

Within each ISP domain, we emulate an IP-layer interior gateway protocol (IGP) that implements Dijkstra's shortest path algorithm. To simulate real-world network scenarios, we introduce link failures and carefully model the IGP dynamics in response to failures as outlined in [52] and Chapter 3. In any IP network, the link utilization determines the delay, throughput, and losses experienced by traffic flows that traverse the link. To simulate realistic link delays, we model delay on any link as a monotonically increasing piecewise linear convex function of its utilization (as in [39] and [77]).

In our simulations, a single IP network could have multiple overlay networks with nodes resident in its domain. Each overlay can have a different topology and routing strategy. Note that we do not require that all of the nodes participating in an overlay be resident in the same domain. All overlay networks adopt the same routing strategies and path probing mechanisms described in

Chapter 6.

We assigned the background traffic between various IP nodes in the network based on the findings in [68]. The traffic between overlay nodes in various overlay networks was assigned such that *the overlay traffic accounts for a significant portion of the IP traffic* (an implicit assumption made in all of our discussions). In our simulations, the combined overlay traffic from all the overlay networks was typically 30-50% of the overall traffic. We generated numerous events at the IP layer and observed the reactions of overlay and IP networks, both at the control plane and the data forwarding plane. In the following section, we present some of our key observations.

7.3.2 Race Conditions in Multiple Overlays

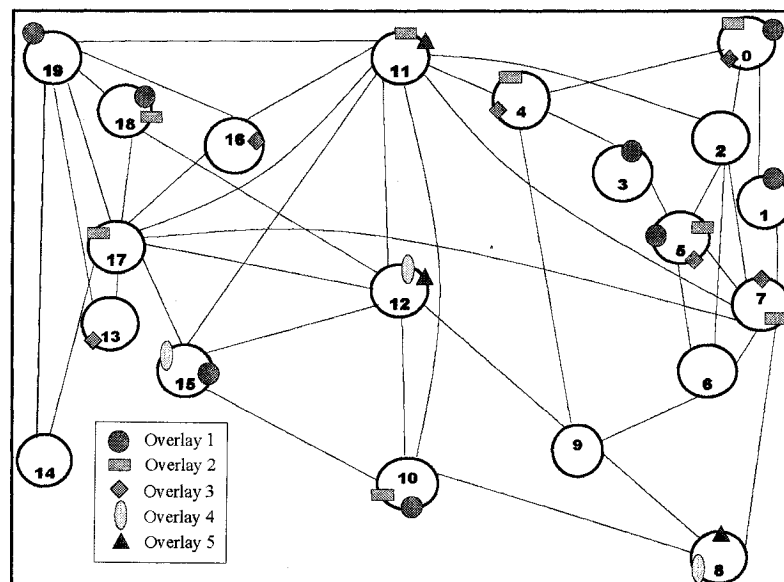


Figure 7.1: Simulation Topology

As mentioned before, our hypothesis is that the co-existence of multiple independent over-

lays can exhibit race conditions that lead to unexpected network instability. To test our hypothesis, we consider a scenario with five overlay networks deployed on top of a tier-1 ISP backbone topology (similar to [55]), as shown in Figure 7.1. To simulate a realistic scenario with heterogeneous overlay networks, we chose different timers values (i.e., P , Q , T , and N) for each of the overlay networks (Table 7.1). Notice that the timers of the first two overlays have significantly larger values compared to the other three. Even though our simulation topology has a single domain, it is straightforward to see that the observations and results that we present in the rest of the chapter can occur even when overlay networks span multiple domains.

Timer	P (ms)	Q (ms)	T (ms)	N
Overlay-1	2000	600	300	3
Overlay-2	2000	1000	350	3
Overlay-3	1000	500	200	3
Overlay-4	800	400	120	3
Overlay-5	700	300	100	3

Table 7.1: Timer Values for the Overlays in Simulation

We ran numerous simulations by generating different IP-level events and various traffic loads in the overlay networks. Results reveal many different possible interactions between overlay networks triggered by different events. For ease of presentation, we only show the dynamics of multiple overlays in response to a very common event, link failures.

Figure 7.2 shows the utilization of a subset of links in one of the simulation runs that lasted for 70s. On the x-axis, we mark the timeline of various IP-layer events such as link failures, routing *re-convergence* (IGP routing protocol has converged and identified alternate path), and link recovery (failed link becomes operational again). We consider two link failure events: (i) *link 10-12* fails at $t = 10$ s, and (ii) *link 2-5* fails at $t = 50$ s. The failures are far apart such that the first link

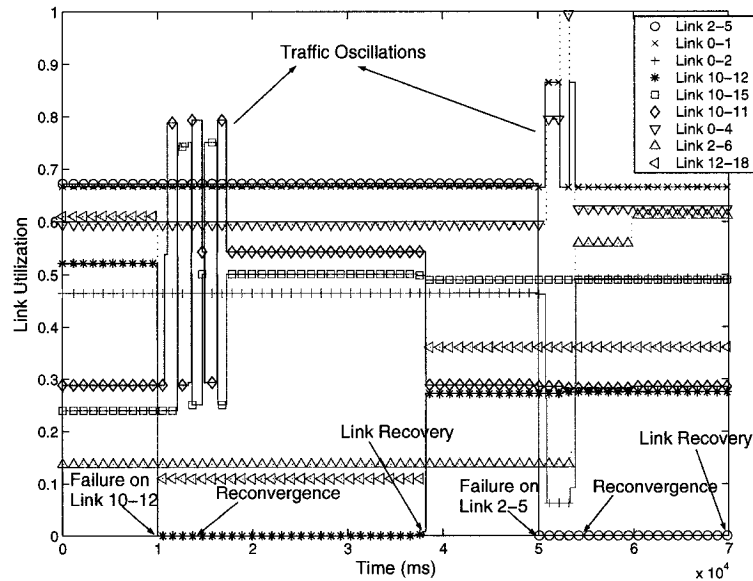


Figure 7.2: Link Utilization as a Function of Time

is operational (i.e. recovered) before the second failure event. Note that we do not show multiple simultaneous link failures for ease of illustration. However, multiple failures will exacerbate the race conditions described here.

Traffic Oscillations: In Figure 7.2, we can clearly see that soon after the link failure events, loads on some of links start oscillating. There are two sets of oscillations: one corresponding to the failure of *link 10-12* and the other for the failure of *link 2-5*. During these two sets of oscillations, the paths between some source-destination pairs in the overlay layer change constantly. The first set of oscillations involves two overlay networks (*Overlay-1* and *Overlay-2*) while the second set involves three (*Overlay-1*, *Overlay-2*, and *Overlay-3*). We observe that the overlay paths involved in these oscillations do share at least a few common links in the IP layer. Note that the two sets of oscillations involve different number of overlays, implying that different number of overlay networks can interact with each other resulting in oscillations.

Stop Triggers for Oscillations: Even though both sets of oscillations in Figure 7.2 are triggered by failure events, the stop trigger for the oscillations are different. In the first set, the oscillations stop when the overlay networks de-synchronize, or disentangle themselves, with no external trigger. In the second set, oscillations stop when IGP re-converges. Different runs of our simulations clearly indicate that there is a variety of events that act as stop triggers for these oscillations (e.g., IGP re-convergence, link recovery, self-disentanglement, etc.). An important observation here is that certain IP layer events that act as stop triggers for oscillations at some point in time might not affect the oscillations at another point in time. Also, most of the IP layer events that act as stop triggers are heavily dependent on the network conditions at the IP layer. For example, IGP convergence depends not only on timer values set by the ISPs, but also on the location of BGP peering points [55]. The order of occurrence of these stop triggers is not deterministic, thus introducing unpredictability in the duration of the oscillations. In essence, the end of oscillations depends on numerous factors, thus making it non-trivial to accurately estimate the impact of oscillations on overlay or non-overlay traffic.

Cascading Reactions: We also observed that a reaction from one overlay network can trigger a series of reactions in other overlay networks. We refer to such a domino effect as *cascading reactions* or *cascading route changes*.

We ran the same simulation as before but with a slightly different traffic load in the overlay networks. The top graph in Figure 7.3 shows the load on IP links that were common to the overlay paths between nodes 10 and 18 in *Overlay-1* and *Overlay-2*. The bottom graph shows the load on some of the links used by *Overlay-4* and *Overlay-5* between nodes 8 and 15, and, 8 and 12 respectively.

By observing the link loads on these IP links (Figure 7.3) and correlating it with the path changes at the overlay control layer, we found the following. Soon after the failure of *link 10-12*, *Overlay-1*, which uses the failed link as a part of the primary path between nodes 10 and 18, moves its traffic on to an alternate path through *link 10-15*. We can see this as an increase in the load on *link 10-15* in the top graph of Figure 7.3. A little later, *Overlay-2* also moves its traffic onto the same alternate path. We can see this as a further increase in the load on *link 10-15* in the top graph. This increase in load on *link 10-15* deteriorates the performance of the path from node 8 to node 15 in *Overlay-4*. The primary path for this source-destination pair is $8 - 10 - 15$. *Overlay-4* reacts to this performance deterioration and moves its traffic to an alternate path (i.e., $8 - 9 - 12 - 15$). This results in a decrease in load on *link 10-15* as in the top-graph and a simultaneous increase in load on *link 8-9*, as shown in the bottom graph. This traffic shift overloads *link 8-9* and *link 9-12*, prompting the traffic between nodes 8 and 12 in *Overlay-5* to find an alternate path. Note that the primary path from node 8 to node 12 in *Overlay-5* was $8 - 9 - 12$. *Overlay-5* finds an alternate path (i.e., $8 - 7 - 11 - 12$) and moves its traffic. This leads to a decrease in load on *link 8-9* and an increase on *link 11-12* (bottom graph). This shows that a reaction in one or more overlay networks could lead to a series of reactions from other overlays. A critical observation here is that such cascading reactions could become common as the number of overlays that co-exist increases.

Figure 7.4 shows the results from our third simulation run which differs from the second run only in the overlay traffic load. The cascading reactions observed are similar to the second simulation run except that: (i) the alternate path found by *Overlay-5* between nodes 8 and 12 is $8 - 10 - 15 - 12$ instead of $8 - 7 - 11 - 12$. (ii) The reaction of *Overlay-4* to the routing decisions made by *Overlay-1* and *Overlay-2* results in oscillations in both *Overlay-4* and *Overlay-5*. The

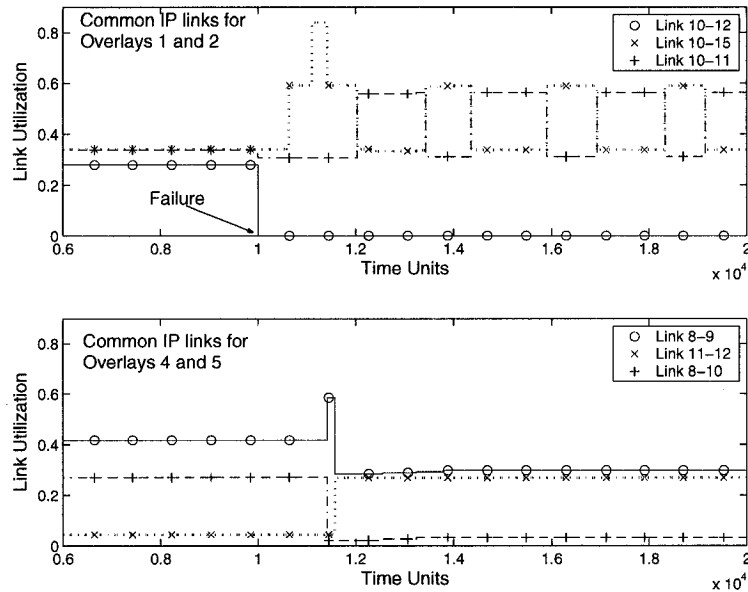


Figure 7.3: Second Simulation Run: Cascading Reactions.

resulting load fluctuations on *link 10-15*, *link 10-11*, *link 8-10*, and *link 8-9* can be observed in the top and bottom graphs of Figure 7.4. Note that all four overlays (*Overlay-1*, *Overlay-2*, *Overlay-4*, and *Overlay-5*) start oscillating after the failure of *link 10-12* resulting in highly unstable network condition. The only difference between the simulation runs in Figures 7.3 and 7.4 is the overlay load conditions. In other words, an increase in load in a overlay network could completely change the complexity of the problem. It is worth noting that a gradual increase in overlay traffic load over time could be a major factor that could lead to such cascading reactions.

Our observation of cascading reactions in multiple overlay networks is similar to the *bistable* behavior detected in traditional non-hierarchical telephone networks [18]. Typically, these networks are built as a full-mesh of 4ESS switches with logical links between them. By default, under low-load conditions, all the calls use the direct (one-hop) path between switches. Under high loads, when the direct path is full, the network picks a two-hop alternate path to carry a call. These

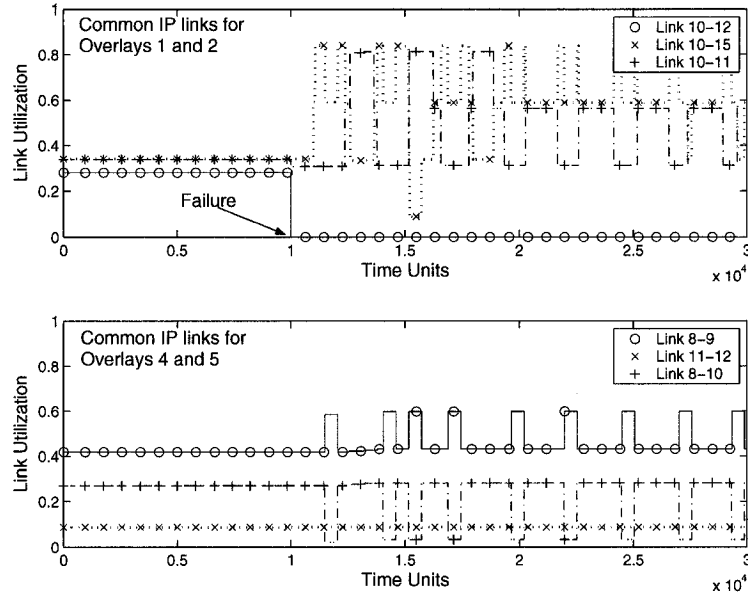


Figure 7.4: Third Simulation Run: Cascading Reaction Leading to Oscillations

two-hop calls consume more resources in the network, and could force other potential direct calls to use two-hop paths. This might result in most of the calls using two-hop paths thus reducing the network capacity by half. This is very similar to cascading reactions observed in overlay networks. The solution for this problem in telephone networks was to use *trunk reservation* schemes that prevent a link to be used for two-hop calls when its utilization exceeds a certain threshold value. This scheme is inapplicable in the context of overlay networks since (i) there is no single administrative control for different overlays, and (ii) IP networks lack admission control and hence reservation of bandwidth on different links is not feasible.

7.4 Why do race conditions occur?

In this section, we present the arguments for why and when race-conditions (hence traffic oscillations and cascading reactions) between multiple co-existing overlays occur.

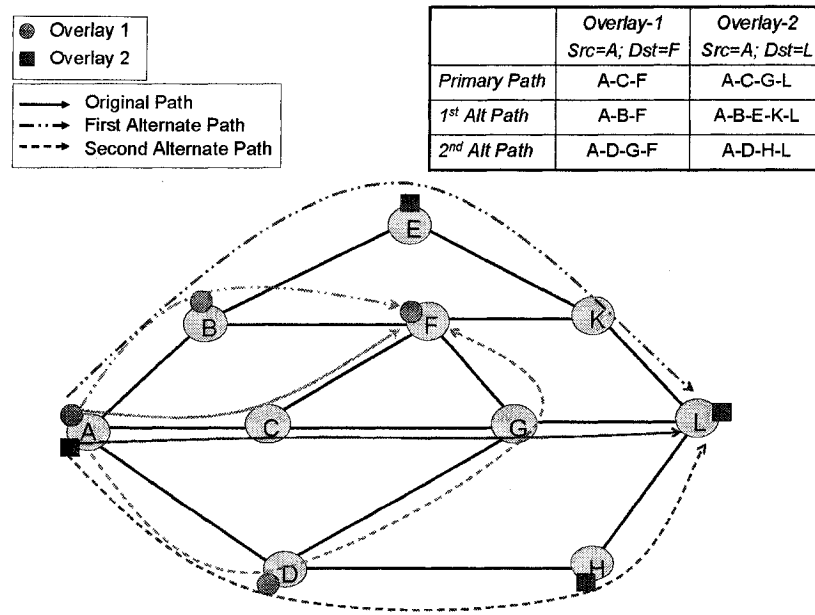


Figure 7.5: Two Overlay Networks that Partially Share Primary and Alternate Paths

7.4.1 Conditions for Traffic Oscillations

Oscillations are initiated when coexisting overlays satisfy the following conditions:

Path Performance Degradation. An event must trigger a perturbation of the network state that leads overlay networks to revisit their routing decisions and look for alternate paths. This event can be an increase in the traffic demand of the IP network or one of the overlay networks, or a link failure event that results in a reduction of capacity. Since different overlays are controlled by autonomous timers and routing algorithms, a path performance degradation event could provoke independent reactions from different overlays.

Topology (i.e., Primary and Backup Paths). The node locations determine how the paths of coexisting overlay networks overlap. Oscillations may occur when the primary and alternate paths share at least one common link. Figure 7.5 illustrates this case with overlays on top on an IP

network. The node pair $A - F$ in *Overlay-I*, and pair $A - L$ in *Overlay-II* share the link $A - C$ on their primary paths. Assume, for simplicity of discussion, that the “top” path is their first alternate choice. If link $A - C$ fails, then the first alternate path for $A - F$ and that for $A - L$ would share link $A - B$. If this link becomes a bottleneck, forcing the overlay networks to move their traffic again, then the overlay source-destination pairs $A - F$ and $A - L$ would now move to the “bottom” path. However, they would still share link $A - D$ that could itself become a bottleneck. Hence the topology criteria for synchronization to occur is: there is a pair of overlay nodes in each of two different overlay networks, such that the two primary paths share at least one common bottleneck link. This condition is intuitive. Two overlays will not get synchronized if they do not share portions of physical paths. For oscillations to sustain, the two overlays must share bottleneck link/s in both their first and second alternate path choices.

Periodic Probing Process. As mentioned in Section 6.2, overlays periodically probe their paths. Consider the two overlays in Figure 7.5 and a failure on *link A-C* that is common to their primary paths ($A - F$ in *Overlay-I* and $A - L$ in *Overlay-II*). Suppose the timing of the probing processes for two overlays is such that the last high frequency probes, for each of the two overlays, expire within a short time window of the other. Then both overlays will shift their traffic to their first choice alternate path roughly at the same time. When this occurs we say that two overlays get *synchronized*. This happens when the window of time is so short that the overlay that moves second does not have time to re-probe its path to realize that some traffic load from the other overlay has already moved. Now, if the traffic load on the first choice alternate path becomes high, then the overlays could react again moving their traffic to the second choice alternate path. Such reactions can continue and overlays move their traffic in a lock-step fashion between the two alternate paths

until the distance between the probes grows large enough to end the synchronization. When this happens we say that the two overlays *disentangle* themselves.

Since overlay networks have no control over performance degradation events inside an ISP (first condition), and since they may not have much control over the placement of overlay nodes that eventually determines the overlap of paths (second condition), we focus our calculation of the probability of synchronization in terms of just the probing process parameters.

7.4.2 Conditions for Cascading Reactions

Cascading reactions tend to occur when there are a large number of overlays co-exist and they satisfy the following conditions:

Path Performance Degradation and Periodic Probing Process. Similar to case of oscillations (in Section 7.4.1), cascading reactions are triggered by a perturbation of the network state that overlay networks detect and react to through their periodic probing process.

Topology (i.e., Primary and Backup Paths). Although oscillations also require the topology condition to be satisfied, the requirement for cascading reactions are quite different from that of oscillations. Cascading reactions could occur when the alternate path of first overlay overlaps with the primary path of the second overlay, the alternate path of the second overlay overlaps with the primary path of the third overlay, and so on. Figure 7.6 illustrates a scenario with three overlay networks spanning multiple domains that could lead to cascading reactions. Consider the primary and alternate paths between the node pairs in the three overlays as indicated in the figure. The alternate path between the node pair $A - G$ in *Overlay-1* partially overlaps with the primary path between the node pair $C - K$ in *Overlay-2*. Same is the case for the node pair $C - K$ in *Overlay-2* and

node pair $B - H$ in *Overlay-3*. For simplicity of discussion assume that the link $D - G$ fails. Soon after the failure, *Overlay-1* will move the traffic between the node pair $A - G$ from its primary path ($A - D - G$) to the alternate path ($A - C - F - G$). This alternate path shares the link $C - F$ with the primary path between node pair $C - K$ in *Overlay-2*. If this link gets overloaded degrading the performance, then *Overlay-2* could move the traffic between node pair $C - K$ to the alternate path, $C - B - E - H - K$. Notice that *Overlay-2* and *Overlay-3* now share links $B - E$ and $E - H$. If anyone of these links gets overloaded, then *Overlay-3* could move its traffic to the alternate path, $B - A - D - F - H$. If there are more overlays in the system that satisfy the topology condition then these reactions could percolate to several overlays affecting their performance.

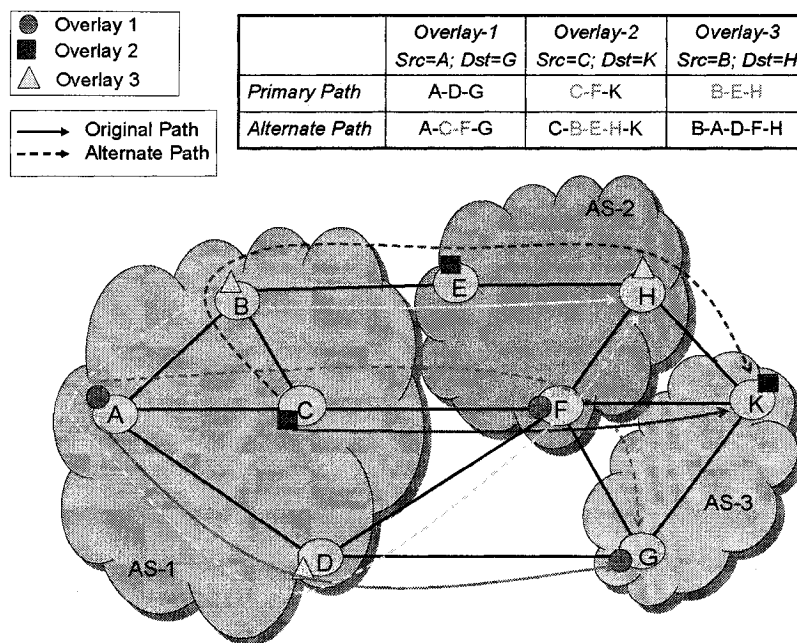


Figure 7.6: Three Overlay Networks in Multiple Domains that Partially Share their Primary and Alternate Paths

Since a particular overlay network cannot know whether its paths overlap with those of another overlay, and since it cannot predict performance failures, avoiding these conditions is beyond

the control of an overlay network. However, an overlay can control its own probing process, thus in the rest of this work we focus on modeling the impact of the path probing process on oscillations.

7.5 Analyzing Oscillations

In this section, we focus on how the path probing parameters affect synchronization. First, we develop an analytic formulation of the probability of synchronization for two overlays as a function of the parameters in the path probing procedure. Second, we derive an upper bound on how long two overlays can remain synchronized, for a given set of parameters.

7.5.1 Probability of Synchronization

For our analysis, we assume the first two conditions for synchronization hold, i.e., the two overlays share at least one link on their primary paths and one event on the shared links occurs (e.g., failure or congestion) that causes probe packets to be lost or excessively delayed.

As described in Section 6.2, overlay networks probe their paths at regular intervals of P seconds. If the path is healthy, the probe should return in one round trip time, with a measure of the path delay (or an assessment of another chosen metric). If the probe does not return before the timeout T expires, then the overlay starts sending its high-frequency probes (N will be sent) every Q seconds. Thus, the probing procedure for each overlay i on path j is specified by five parameters: the probe interval P_i , the high frequency probe interval Q_i , the timeout T_i , the number of high frequency probes N_i , and the round trip time R_{ij} over path j . Note that T_i is the same for low- and high-frequency probes. By definition $P_i \geq Q_i \geq T_i \geq R_{ij}$.

The probing procedure implies that (under normal circumstances) on a given path, there

will be exactly one probe in every time period of length P . Now suppose that an event (e.g., a link failure) occurs at time t_l . We assume that a probe sent on path j in overlay i at time t_0 “senses” the state of the path at $t_0 + R_{ij}/2$, i.e., the probe is dropped if the path is not operational at that time¹. Hence, the overlay network will detect the failure event with the probe sent at t_0 if $t_0 \in [t_l - R_{ij}/2, t_l - R_{ij}/2 + P]$. We call this period the *detection period*. The overlay will then react at time $t_0 + T_i$ sending the high frequency probes as discussed above.

Consider two overlay networks, O_1 and O_2 . Let t_1 and t_2 be the actual times at which the initial probes are sent during the detection period. We assume that t_1 and t_2 are equally likely to occur anywhere in their detection period and hence are uniformly distributed in their detection period. Once an overlay network detects the failure, it begins sending the high frequency probes every Q_i time units. The final high frequency probe will be sent out at $f_i = t_i + N_i Q_i$ for $i = 1, 2$.

An overlay network actually moves its traffic to an alternate path immediately after the final high frequency probe has timed out ($f_i + T_i$ for O_i). Two overlay networks will synchronize if they both move their traffic to the same bottleneck link (shared by their chosen alternate paths) in a “short” window of time. By “short” we mean here that the window is small enough that when one overlay moves, the second overlay does not see or detect that move through its probing process and thus moves itself onto the same link(s).

There are two cases for synchronization - in one case O_1 moves its traffic first and O_2 moves shortly thereafter, or vice versa. Consider the case of O_1 moving first. Suppose that O_2 sends out the final high frequency probe after O_1 sends its final high frequency probe, but before O_1 moves its traffic. We assume that O_2 decides at time f_2 what its alternate path will be if this last

¹To simplify our analysis during failures we ignore the exact values of propagation delays between the source, the failed spot, and destination. Thus we approximate the instant at which a probe is dropped by $R_{ij}/2$.

probe does not return, and hence it doesn't have time to detect the traffic move by O_1 before it moves its own traffic. Hence if we have the timing $f_1 < f_2$ and $f_2 - f_1 < T_1$, then both overlays move their traffic without being aware of the other's reaction. This is the condition for synchronization when O_1 moves first. Similarly, if O_2 moves first, the networks will synchronize if $f_2 < f_1$ and $f_1 - f_2 < T_2$. Hence the two overlays get synchronized if *any one* of the following two conditions are satisfied:

$$\text{Sync Condition-1: } 0 < f_2 - f_1 < T_1 \quad (7.1)$$

$$\text{Sync Condition-2: } 0 < f_1 - f_2 < T_2 \quad (7.2)$$

Since the above two conditions are independent of each other, we can combine them as follows:

$$\begin{aligned} -T_1 &< f_1 - f_2 < T_2 \\ -T_1 &< (t_1 + N_1Q_1) - (t_2 + N_2Q_2) < T_2 \\ b &< t_1 - t_2 < a \end{aligned} \quad (7.3)$$

where $a = N_2Q_2 - N_1Q_1 + T_2$; $b = N_2Q_2 - N_1Q_1 - T_1$

We assume that t_1 and t_2 can occur anywhere in their detection period with a uniform probability. It is important to notice that the actual value of t_l is irrelevant and hence for the ease of understanding we consider $t_l = 0$. Thus the range of t_1 is $[-R_1/2, P_1 - R_1/2]$ and the range of t_2 is $[-R_2/2, P_2 - R_2/2]$, where R_1 is the RTT for the primary path in overlay O_1 and R_2 is the RTT for the overlapping primary path in the other overlay², O_2 .

²Since we focus only on the primary path in both the overlays, we drop the second subscript in R_{ij} .

For a specific set of parameters $(P_i, Q_i, T_i, N_i, R_i)$ for $i = 1, 2$, we can represent the system as a two dimensional graph with the x-axis representing probe t_1 and the y-axis representing probe t_2 . All the allowable values for the tuple (t_1, t_2) lie inside the rectangle with the vertices: $(-R_1/2, -R_2/2)$, $(P_1 - R_1/2, -R_2/2)$, $(P_1 - R_1/2, P_2 - R_2/2)$ and $(-R_1/2, P_2 - R_2/2)$ (see Figure 7.8). This geometric representation allows us to compute the probability of synchronization, $P(S)$, of two overlays in an intuitively simple way. We define *region of conflict* to be the portion of this rectangle in which synchronization will occur, i.e., the region that satisfies the two constraints specified in Equation 7.3. The boundaries of the *region of conflict* are thus determined by the boundaries of the rectangle and their intersection with the two parallel lines of slope 1:

$$\text{Line 1: } t_1 - t_2 = a \quad (7.4)$$

$$\text{Line 2: } t_1 - t_2 = b \quad (7.5)$$

Since t_1 and t_2 can occur anywhere in their detection period with uniform probability, synchronization will occur if the point (t_1, t_2) lies inside the region of conflict. Now the probability of synchronization, $P(S)$, can be defined to be the ratio of the area of the region of conflict to the total area of the rectangle. This two-dimensional representation captures the influence of all the parameters $(P_i, Q_i, N_i, T_i, R_i)$ since these quantities ultimately define all the corners and line intersection points needed to compute the relevant areas.

There are a number of ways in which *Line-1* and *Line-2* intersect the boundaries of the rectangle. All possible scenarios that determine this area are shown in Figure 7.7. Consider Figure 7.8 that represents *Scenario 1* in detail. *Line-1* intersects the bottom and right edges, while *Line-2* intersects the left and top edges. As evident in Figure 7.8, we can clearly see that the area A of the rectangle is composed of three distinct regions: A_1 (area of the region below *Line-1* and the

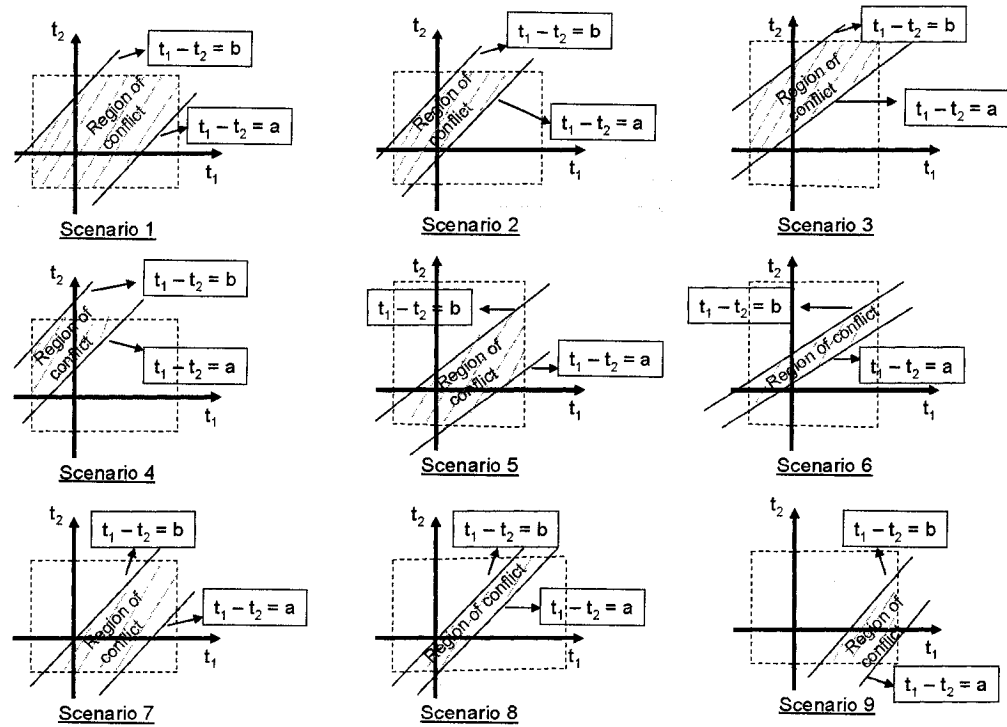


Figure 7.7: All Possible Scenarios to Calculate the *Region of Conflict*

rectangle boundaries), A_2 (area of the region above *Line-2* and the rectangle boundaries) and the *region of conflict*. Hence the *region of conflict*, A_C , can be expressed as,

$$A_C = A - A_1 - A_2 \quad (7.6)$$

Thus we can express the probability of synchronization as

$$P(S) = \text{Probability}(b < t_1 - t_2 < a) = \frac{A_C}{A} \quad (7.7)$$

In *Scenario 1*, A_1 and A_2 are triangular regions with two equal edges (due to the fact that both *Line-1* and *Line-2* have a slope of 1). From Figure 7.8, we see that A_C can be computed using

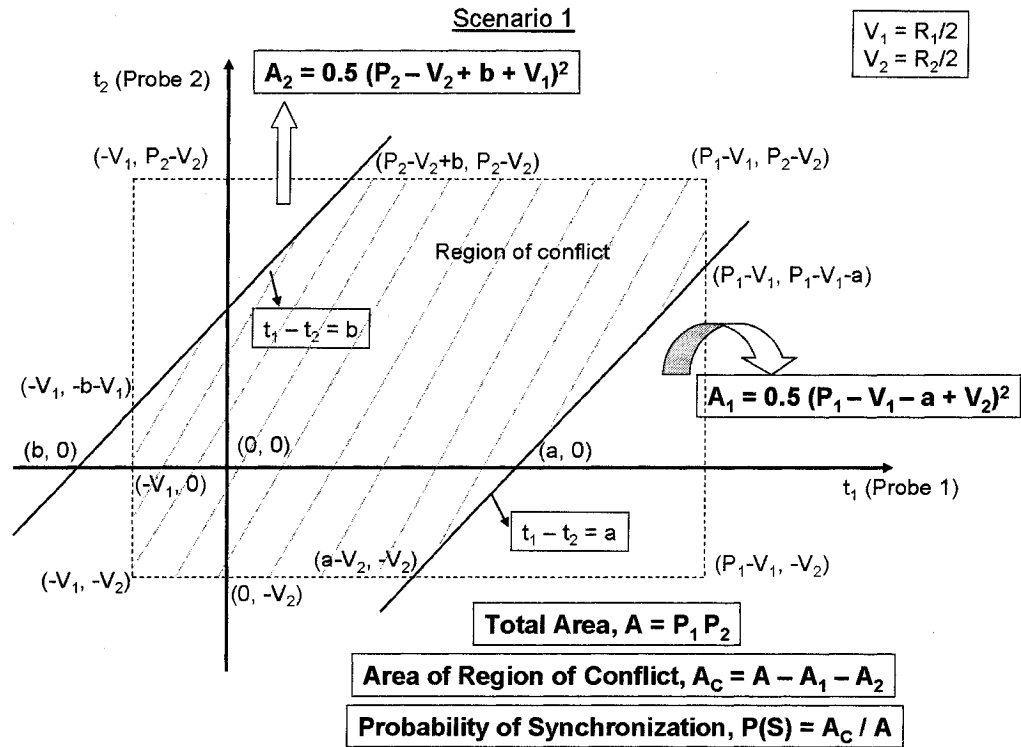


Figure 7.8: Scenario 1 ($V_1 = R_1/2$ and $V_2 = R_2/2$)

$$A = P_1 P_2 \quad (7.8)$$

$$A_1 = 0.5(P_1 - R_1/2 - a + R_2/2)^2 \quad (7.9)$$

$$A_2 = 0.5(P_2 - R_2/2 + b + R_1/2)^2 \quad (7.10)$$

Even though the above equations for A_1 and A_2 are valid for *Scenario 1*, they do not hold for scenarios where *Line-1* intersects boundaries other than the right and bottom edges of the rectangle, and/or *Line-2* intersects boundaries other than the left and top edges of the rectangle. For example, if we consider *Scenario 2* in Figure 7.7, *Line-1* intersects the top and bottom edges of the rectangle. If we use Equation 7.9 to calculate A_1 , then A_1 is larger than it should actually be since it

includes space outside the rectangle. We thus need to add another term, E , to A_C in Equation 7.6 to compensate for the *excess* included in A_1 . In the case of *Scenario 2* (when *Line-1* intersects the top edge of the rectangle), the excess term to be removed is $E = 0.5(P_1 - R_1/2 - a - P_2 + R_2/2)^2$. We do not include the compensation areas for each of the nine scenarios since it is a straight forward computation. Although this model results in 9 different scenarios, each of them with a different equation for $P(S)$, it is still attractive since it is conceptually very simple. This model can be extended to include an arbitrary number of overlays and we describe this in the next section.

7.5.2 How long do oscillations last?

Suppose that two overlays react to an event within a short window of time, and land up on alternate paths that share at least one common link. As depicted in Figure 7.5, such a reaction by both overlays could overload the common link, prompting them to find another alternate path. These reactions lead to oscillations that last until the overlay networks disentangle themselves or are influenced by an external event (e.g., a shift in the traffic loads). If no external events stop the oscillations, then it is important to ask how long these oscillations will last. In this section, we derive an upper bound on the number of oscillations.

To break synchronization, what matters is the temporal spacing between the two probing processes that govern their reaction times (moving traffic). Let $s_0 = t_1 - t_2$ denote the difference in time between the initial detection of the path problem. Since we are concerned with the difference between the reaction times, we can map the \mathbb{R}^2 *region of conflict* space onto \mathbb{R}^1 space on a real line for all possible scenarios. In other words, we can represent all the points in the *region of conflict* by the value of $t_1 - t_2$ at that point. Here all the points in the *region of conflict* are mapped to the region between the points b and a on the real line.

Every time an overlay network shifts traffic to an alternate path it starts probing the new overlay path every P seconds. If both the overlays shift their traffic almost simultaneously resulting in performance degradation on the first choice alternate path, then this will trigger another response from both overlays. They will shift their traffic to their second choice alternate path. If these second choice paths become overloaded, each overlay may move back to its first choice path, thus entering into oscillations. The time for an overlay to detect a problem on a new path and then move its traffic is given by $P_i + N_i Q_i + T_i$. Thus each time the synchronized overlays move together from one set of alternate paths to another, the spacing between the probes change by $c = P_1 + N_1 Q_1 + T_1 - P_2 - N_2 Q_2 - T_2$. After k ($k = 1, 2, \dots$) such reactions, the spacing between the probes can be expressed as:

$$s_k = s_0 + k.c \quad (7.11)$$

Note that we have implicitly assumed here that the parameter values for the primary and alternate paths remain the same for both the overlays; hence the value of c is the same regardless of whether we are on the first or second set of alternate paths. We make this assumption for two reasons. First, it allows our model to remain tractable; without this the size of the box (feasible region for probe values) in our model for $P(S)$ would change with each traffic shift. Second, this is not unreasonable for scenarios in which two overlay networks select locations for their nodes that are similar either because they are strategic or resident where the traffic demands are high. Also, if the values of c are different for each of the alternate paths then the *rate* at which the spacing moves towards the boundary condition could either increase or decrease, thus introducing the possibility that the number of oscillations could be better or worse than the case that we consider here.

From Equation 7.3, the stop condition for the oscillations is given by $|s_k| > a - b$ (note

that $a > b$). The worst case in the number of oscillations happens when s_0 is equal to a (or b) and moves towards b (or a) by c seconds at each step. It is then straightforward to derive \bar{k} , the upper bound on the number of oscillations, $\bar{k} = \left\lceil \frac{a-b}{|c|} \right\rceil$. Hence,

$$\bar{k} = \left\lceil \frac{T_1 + T_2}{|P_1 - P_2 + N_1 Q_1 - N_2 Q_2 + T_1 - T_2|} \right\rceil \quad (7.12)$$

Notice that when the overlays have identical parameters, they remain synchronized forever. The model thus follows our intuition that once two overlay gets synchronized, if the spacing between the probes never changes, they remain synchronized always.

7.6 Extension of the analytical model

In the previous section, we provided a framework to evaluate the likelihood of synchronization between two overlay networks that satisfy the conditions outlined in Section 7.4.1. A natural question to ask would be ‘Does the same framework apply when there are m overlays that coexist and satisfy the conditions in Section 7.4.1?’. In this section we address this question and show that our analytical framework can be extended to m coexisting overlays when all the overlays satisfy the conditions outlined in Section 7.4.1.

Consider m coexisting overlay networks, O_1, O_2, \dots, O_m . Let f_1, f_2, \dots, f_m be the times at which the final high frequency probes are sent by overlays O_1, O_2, \dots, O_m after a path performance degradation event is detected. Let $S = \{s_1, s_2, \dots, s_m\}$ be an ordering of $\{1, 2, \dots, m\}$ such that $f_{s_i} < f_{s_j}$ if $i < j$.

Similar to the conditions for synchronization of two overlays outlined in Equations 7.1 and 7.2, the condition for the synchronization of m overlays is given by a set of $m - 1$ inequalities:

$$0 < f_{s_m} - f_{s_i} < T_{s_i} \quad \text{for } i = 1, 2, \dots, m - 1 \quad (7.13)$$

The above equations represent the required conditions for m coexisting overlays to synchronize. The ordering in S depends on the actual times at which the final high frequency probes are sent out by different overlays and hence there could be several different combinations. However, from Equation 7.13, we can clearly see that the order in which the final high frequency probes are sent out by the first $m - 1$ overlays does not affect the synchronization condition. The only requirement is that the final high frequency probe from O_{s_m} should be sent out after all the other overlays have sent their final high frequency probes. In other words, $S = \{1, 2, 3, 4 \dots m\}$ results in the same synchronization condition as $S = \{2, 1, 3, 4 \dots m\}$ or $S = \{4, 2, 1, 3 \dots m\}$ as long as $s_m = m$. Hence, the different possible combinations of S depends on the value of s_m . Note that s_m can take any one of the m values ($\{1, 2, 3 \dots m\}$) and hence there can be m different combinations of S . This implies that in a system with m overlays there are m independent synchronization conditions for all the overlays to synchronize. In each of these synchronization conditions, there will be $m - 1$ sets of inequalities (as we can see in Equation 7.13) that are required to be satisfied in order for overlays to synchronize.

To illustrate the above reasoning let us first consider the case where there are only two overlays, i.e., $m = 2$. In this case, there should be m , i.e., 2 independent synchronization conditions and $m - 1$, i.e., 1 set of inequality in each of the synchronization conditions. We can clearly see that this is in fact true based on Equations 7.1 and 7.2. Similarly in the case of three overlays, there are 3 independent synchronization conditions with 2 sets of inequalities in each of these conditions.

Equation 7.13 can be rewritten as:

$$\begin{aligned}
0 < t_{s_m} + N_{s_m} Q_{s_m} - t_{s_i} - N_{s_i} Q_{s_i} < T_{s_i} \quad \text{for } i = 1, \dots, m - 1 \\
d_{im} < t_{s_m} - t_{s_i} < T_{s_i} + d_{im} \quad \text{for } i = 1, \dots, m - 1
\end{aligned} \tag{7.14}$$

where $d_{im} = N_{s_i} Q_{s_i} - N_{s_m} Q_{s_m}$. The set of inequalities in Equation 7.14 involves m overlays and contains m uniform random variables ($t_{s_1}, t_{s_2}, t_{s_3}, \dots, t_{s_m}$). The range of these variables are similar to the definitions in Section 7.5.1, i.e., the range of t_{s_i} is $[-R_{s_i}/2, P_{s_i} - R_{s_i}/2]$ for $i = 1, 2, \dots, m$. Hence this system of inequalities represents a m -dimensional space and the range of values for each random variable ensures that the allowed values for the variables are bounded inside a finite volume (i.e., $V = P_{s_1} P_{s_2} \dots P_{s_m}$) enclosed by the hyperplanes $t_{s_i} = -R_{s_i}/2$ and $t_{s_i} = P_{s_i} - R_{s_i}/2$ for $i = 1, 2, \dots, m$.

A key observation in the inequalities defined in Equation 7.14 is that every inequality depends on only two variables. In other words, the two hyperplanes defined by $t_{s_m} - t_{s_i} = d_{im}$ and $t_{s_m} - t_{s_i} = T_{s_i} + d_{im}$ are parallel to all other axes except the axes in the direction of t_{s_m} and t_{s_i} . Note that with only two varying dimensions for each inequality, the situation is similar to the one described in Section 7.5.1. There are 9 possible scenarios (similar to Figure 7.7) that could occur for each inequality and there are $m - 1$ such inequalities in a synchronization condition. Hence, for each of the m synchronization conditions there are 9^{m-1} possible scenarios, each one with its own *compensation volumes*. The probability of synchronization for a particular synchronization condition is the ratio of the volume enclosed by the hyper-planes defined by the inequalities in Equation 7.14, and the total volume V . Since the synchronization conditions are independent of each other, the *total* probability of synchronization of m overlays is the sum of the probabilities of

synchronization of all the m synchronization conditions.

7.7 Local and Global Synchronization

In the previous section, we discussed how likely it is for all m coexisting overlay networks to synchronize. A natural question that comes up is the following: given m coexisting overlay networks, how likely is it that a particular overlay network gets involved in local or global synchronization. In other words, is an overlay network more likely to synchronize locally with a few other overlay networks or is it more likely to exhibit global synchronization where the overlay synchronizes with several other coexisting overlays. In this section we try to address this question.

Let us assume a system with m overlays where all the overlays satisfy the conditions outlined in Section 7.4.1. Let us now consider the i^{th} overlay network, O_i . The probability with which this overlay network (i.e., O_i) synchronizes with just one other overlay network can be expressed as:

$$P^{i,1} = \sum_{\substack{j=1 \\ j \neq i}}^m \left[P_{ij} \prod_{k \neq i,j} (1 - P_{ik})(1 - P_{jk}) \right] \quad (7.15)$$

where, P_{ij} is the probability of synchronization between overlays O_i and O_j . The term $(1 - P_{ik})$ represents the probability that the overlay O_i does not synchronize with O_k . The second part of the above equation represents the probability that O_i and O_j do not synchronize with any other overlay networks in the system.

Generalizing the above equation, the probability that O_i synchronizes with exactly r other overlay networks can be written as:

$$P^{i,r} = \sum_{\substack{j_1=1 \\ j \neq i}}^m \sum_{\substack{j_2=j_1+1 \\ j \neq i}}^m \dots \sum_{\substack{j_r=j_{r-1}+1 \\ j \neq i}}^m [P_{i j_1 \dots j_r} \Phi] \quad (7.16)$$

$$\Phi = \prod_{\substack{k \neq i \\ k \neq j_1, \dots, j_r}} [(1 - P_{ik})(1 - P_{j_1 k}) \dots (1 - P_{j_r k})] \quad (7.17)$$

where $P_{i j_1 \dots j_r}$ represents the probability of synchronization of O_i with r other overlays (i.e., the result from previous section where the total number of overlays in the system is $r + 1$).

We define that an overlay network, O_i , in a system of m coexisting overlay networks is synchronized locally if it synchronizes with less than g other overlays and is synchronized globally if it synchronizes with more than h overlays. Note that g and h can be set to different values depending on the scenario as long as $1 \leq g < h \leq m - 1$. Using Equations 7.16 and 7.17, we can now express the probability of local synchronization of O_i , P_{LS}^i , as:

$$P_{LS}^i = \sum_{k=1}^g P^{i,k} \quad (7.18)$$

Similarly the global synchronization probability, P_{GS}^i , is:

$$P_{GS}^i = \sum_{k=h}^{m-1} P^{i,k} \quad (7.19)$$

7.8 Validation of Analytical Model

To validate both our analytic formulation and our implementation in the simulator, we compare $P(S)$ computed using the model (Section 7.5) versus that seen in simulation. We consider two similar networks (i.e., all the parameters are identical), but vary the value of the probe interval.

The results are given in Figure 7.9. The simulation results are based on running the simulation 1000 times and calculating the number of times the overlay networks got synchronized. We can see that the analytical results closely match the simulation results.

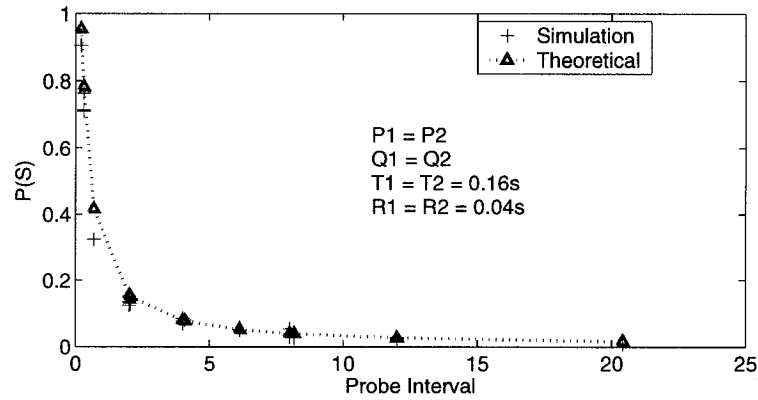


Figure 7.9: A Comparison of Theoretical and Simulation Results for $P(S)$ Between Two Identical Overlay Networks with Different Values of P

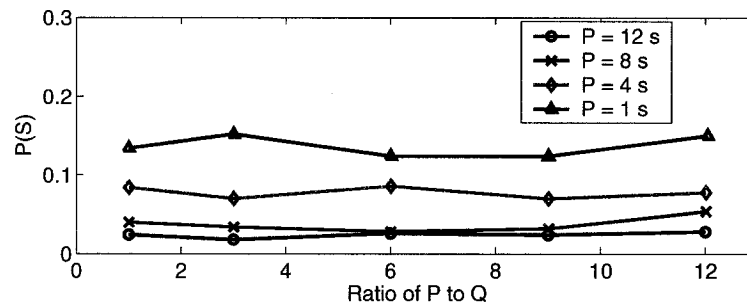


Figure 7.10: $P(S)$ as a Function of the Ratio P/Q with Different Values of Q for a Given Value of P in Two Identical Overlay Networks

When the two overlay networks are identical (i.e., $P_1 = P_2 = P$, $Q_1 = Q_2 = Q$, $T_1 = T_2 = T$, $N_1 = N_2 = N$, and $R_1 = R_2 = R$), it is easy to see that we have $a = T$ and $b = -T$. Hence the probability of synchronization (from Equations 7.7, 7.8, 7.9, and 7.10) collapses to the simple equation $P(S) = T(2P - T)/P^2$. If our model is correct, this implies that $P(S)$ is independent of Q , N , and RTT . Figure 7.10 shows the variation of $P(S)$ (generated

using the simulator) as a function of Q for constant values of P . We can clearly see that for a given probe interval, varying Q does not impact the probability of synchronization between two identical networks thus confirming the accuracy of our model.

To verify the correctness of the theoretical upper bound on the number of oscillations (Equation 7.12), using our simulator we simulate oscillations in two synchronized overlays that are dissimilar (Figure 7.11). We run the simulations 50 times for each set, but the figure represents only those cases where the overlays synchronize and oscillate. We can clearly see that the theoretical upper bound on the number of oscillations before the overlays disentangle is larger than the actual number of oscillations in our simulations, yet lies near the values observed in simulation. Note that we are exploring the number of oscillations in a small parameter space, but the main purpose of this figure is to validate our model. We look at a wider parameter space in Section 7.9.2.

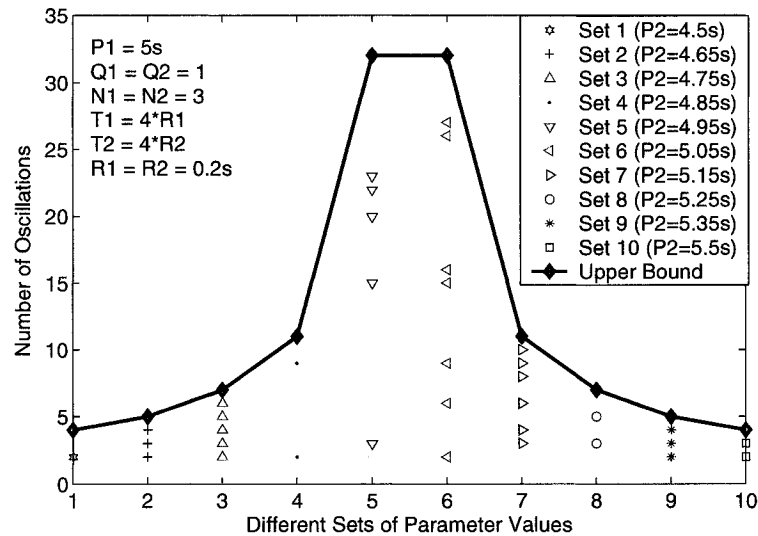


Figure 7.11: Comparison between Theoretical and Simulation Results for Number of Oscillations.

7.9 Sensitivity to Probing Parameters

To assess whether or not these race conditions pose a problem in the design of overlay networks, we need to understand whether such situations are pathological (and thus very unlikely to occur) or if there is a reasonable chance that this can happen over some non-narrow range of the parameter space. For the case of two overlays, we have ten parameters, and thus $P(S)$ describes a probability in 10-dimensional space. We now study how $P(S)$ varies with respect to some of these parameters, or combinations of them. Due to the complexity of the parameter space, we direct our attention to address the following questions: (i) Is $P(S)$ non-negligible in operating regions that can occur in the Internet? (ii) Can we count on naturally occurring variations in RTT (due to path length diversity) to reduce $P(S)$ to negligible values? (iii) If not, which parameter settings can drive $P(S)$ to low values? In other words, how should an overlay network designer choose the probing parameters to reduce the likelihood of synchronization, especially when the behavior of other overlays are not known?

7.9.1 Aggressiveness Factor and Probe Parameter Setting

We start with the simplest case of two overlays with identical parameter setting because this case provides some insight about overlays in general. Recall from Section 7.8 that for two identical overlays, we have $P(S) = T(2P - T)/P^2$. We see that $P(S)$ depends only on the probe interval and the timeout values of the overlays. The maximum value of $P(S) = 1$ occurs when $T = P$, i.e., for these parameter settings, the overlay networks will definitely synchronize. If $P = 2T$ then $P(S) = 0.75$. In order to decrease the probability of synchronization to less than 0.05 (i.e. 5% chance of synchronization) we need to use $P \approx 40T$.

We are thus motivated to characterize overlay networks by their probing frequencies. We consider overlays that probe frequently and move their traffic quickly as aggressive. We define an *aggressiveness factor*, α , of an overlay network as the ratio of the timeout and probe interval, $\alpha_i = T_i/P_i$. Note that $RTT \leq T \leq P$, hence $0 < \alpha \leq 1$. For two identical overlay networks we have $P(S) = 2\alpha - \alpha^2$, which shows that as the networks increase their aggressiveness (i.e. as $\alpha \rightarrow 1$), $P(S)$ increases.

In many of our sample scenarios in the next section, we varied RTT values between 20ms and 300ms to capture a variety of realistic Internet overlay paths that span either a small or large geographic distance (i.e. nearby cities to international routes). We choose the timeout value to be four times the RTT. This is motivated by the type of approach usually followed in TCP in which timeout values are set to be the mean RTT plus 3 or 4 times the standard deviation. Assuming the standard deviation is similar to the mean, we use $T = 4 * RTT$ in our calculations.

The other probing parameters P and Q can be set in two ways: (i) *Proportional* values, where P and Q are set to be multiples of T (and hence RTT) and are different for each path in the overlay; (ii) *Fixed* values, where P and Q are constants independent of T and RTT and therefore the same for all the paths.

7.9.2 Results and Discussion

In Figure 7.12, we explore the impact of variations in RTT values on $P(S)$ for two proportional parameter overlays. Although both overlays have the same aggressiveness in this example, the actual values of P , Q , and T will differ for each overlay because the parameters ultimately depend on the particular RTT. We observe in this figure that when both overlays are aggressive (e.g., $T/P = 1/3$), $P(S)$ can be as high as 55%. When both are non-aggressive (e.g., $T/P = 1/20$),

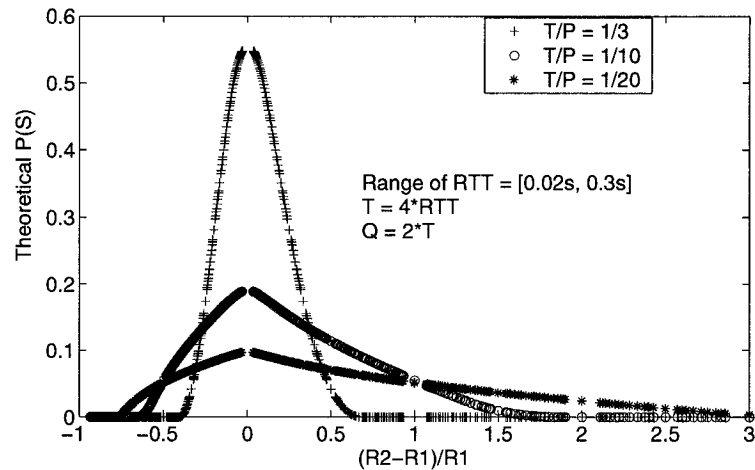


Figure 7.12: $P(S)$ Vs $(R_2 - R_1)/R_1$ for Proportional Parameter Overlays with Similar Aggressiveness and Varying RTT

$P(S)$ never gets above 10%. In this figure, we plot $P(S)$ versus the relative difference in RTT values between two overlays. The figure indicates that when one RTT is more than twice the value of the other, then this synchronization issue is not a concern as $P(S)$ is near or at zero. However, when the RTTs are less than 50% different from one another, then we can have non-negligible probability of synchronization. This could happen for two overlay networks that both span a similar geographic region. Since the dependence here is on the relative RTT's, the actual size of this geographic region does not matter.

As overlays are not widely deployed and their performance requirements are not yet well understood, it is not clear how to decide for which values should $P(S)$ be considered “significant”, or “non-negligible”. Here we consider $P(S)$ to be non-negligible if it exceeds 10%. Admittedly, this number is subjective, however we will see plenty of scenarios in which $P(S)$ is considerably far away from zero to indicate that synchronization problems should not be neglected.

In Figure 7.13, we examine some scenarios in which the two overlays have different aggressiveness factors. In these scenarios, the first overlay is set to be aggressive, while the second

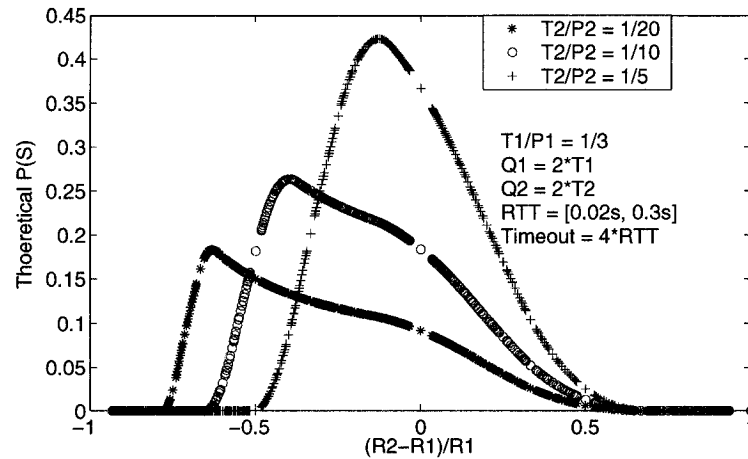


Figure 7.13: Proportional Parameter Overlays with Mixed Aggressiveness and Varying RTT

overlay varies from aggressive ($\alpha_2 = 1/5$) to non-aggressive ($\alpha_2 = 1/20$). We see that even in the case of one aggressive and one non-aggressive overlay network, $P(S)$ can still be non-negligible for a wide range of relative RTT values. However, this figure indicates that an overlay might benefit from using non-aggressive parameters even if another overlay behaves aggressively. To further explore this hypothesis, we consider a wider variety of cases in Figure 7.14.

Figure 7.14 shows the value of $P(S)$ as a function of the aggressiveness factors of the two overlays. Each curve in the graph represents the value of $P(S)$ for a fixed value of T_1/P_1 but different values of T_2/P_2 . We can clearly see that as the aggressiveness of both the overlays increase, there is a higher chance of synchronization. This probability significantly decreases when the overlays are non-aggressive. This confirms that as long as one of the overlays is non-aggressive, the probability of synchronization is low. In other words, setting a high value of P is critical to reducing $P(S)$. We wish to point out that there could be fairness issues when one overlay is very aggressive, and exploits the non-aggressive parameter settings of the other overlay.

We now look at the impact of using a fixed parameter approach to choosing P and Q .

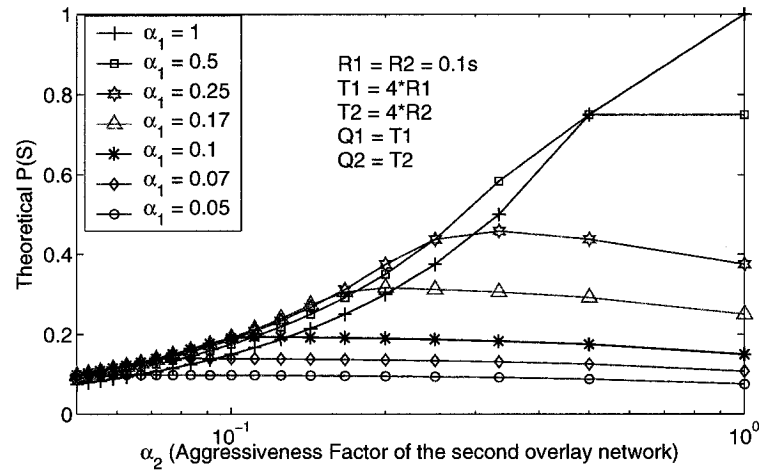


Figure 7.14: Proportional Parameter Overlays with Mixed Aggressiveness and a Chosen Value of RTT

In Figure 7.15, we consider the case of two RON networks. The pattern in this figure is explained as follows. Each straight line of points belongs to the cases of a fixed R_1 as R_2 is varied through its entire range. An interesting observation from this plot is that even when the relative difference between the RTT's is zero, $P(S)$ does *not* take on a single value, but instead can take on any of a number of values. $P(S)$ is at its minimum when both RTT values are small ($R_1 = 20\text{ms}$, $R_2 = 20\text{ms}$), and achieves its maximum when both RTT values are large ($R_1 = 300\text{ms}$, $R_2 = 300\text{ms}$). This suggests that in fixed parameter overlays, unlike proportional parameter overlays, the absolute value of RTT is an important factor in determining $P(S)$. We observe that RON networks are designed to be non-aggressive (T/P varies between 0.007 and 0.1) and this results in low synchronization probabilities. However, there still do remain a number of cases in which $P(S)$ exceeds 10%.

In Figure 7.16, we consider a case of two fixed parameter overlays with different values of P and Q . We see a similar behavior as in the previous case of fixed parameter overlays. We wish to point out that in these two cases $P(S)$ is significant for a wider range of values on the x-axis when

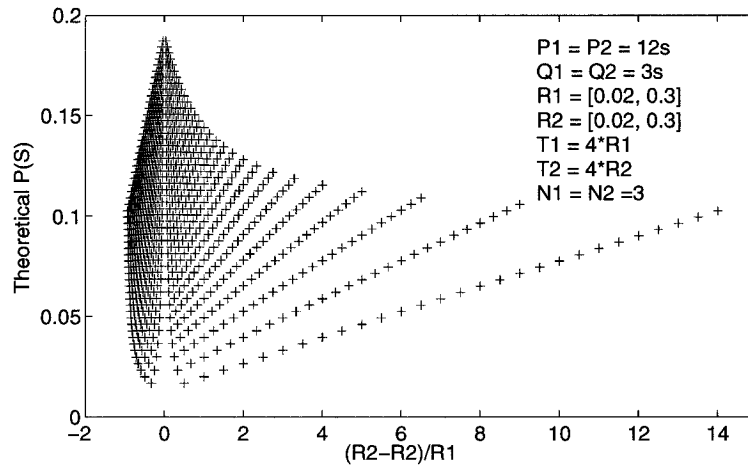


Figure 7.15: Fixed Parameter Overlays (RON-like) with Same Values of P and Q , and Varying RTT

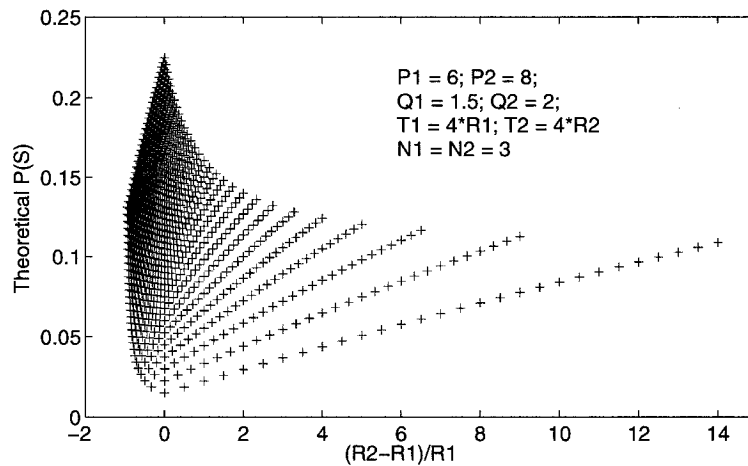


Figure 7.16: Fixed Parameter Overlays with Different Values of P and Q , and Varying RTT

compared to Figure 7.12. In other words, $P(S)$ does not reach zero once the relative difference exceeds 60% or 70%. Using a fixed approach to parameter selection means P is constant, however, since the aggressiveness is determined by $\alpha = T/P$, the aggressiveness is varying per path (since T is proportional to RTT). The overlay is more aggressive on long paths and less so on shorter paths. The increased aggressiveness on longer paths could explain why $P(S)$ does not disappear when the relative difference of RTTs is high (e.g., 300%).

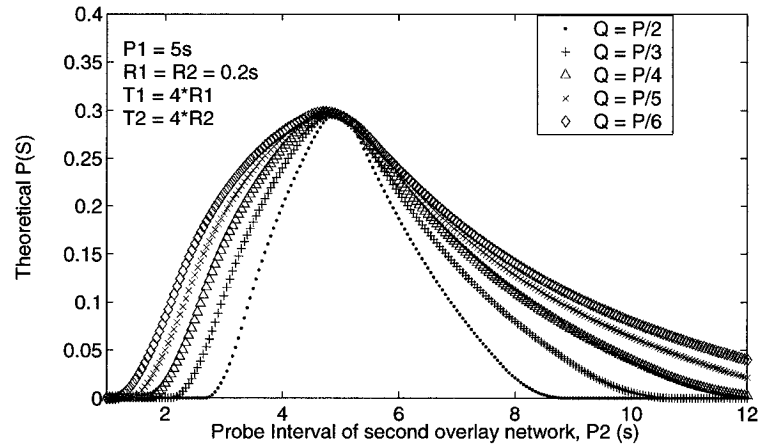


Figure 7.17: $P(S)$ as a Function of P_2 with $P_1 = 5s$

The above results show the influence of P and T on the probability of synchronization. Figure 7.17 explores the influence of Q on $P(S)$ for fixed parameter overlays. For a given value of P_2 , as Q_2 grows, $P(S)$ decreases. For a given value of Q_2 , the range of cases (i.e., values of P_2) in which synchronization can happen are more numerous when Q_2 is small than when it is large. Thus, in general, it seems beneficial to use a higher value of Q to reduce the chances of synchronization. The tradeoff here is a delayed reaction by the overlay. However, most importantly, based on Figure 7.17 we can conclude that the influence of the high frequency probes (Q) on $P(S)$ is far less significant compared to the influence of the probe interval P or the timeout T .

Finally, Figure 7.18 shows the upper bound on the number of oscillations between two overlays after they are synchronized for both proportional and fixed parameter overlays. Even though we are exploring a small subset of the parameter space, there are a considerable number of cases where the number of oscillations is more than 5.

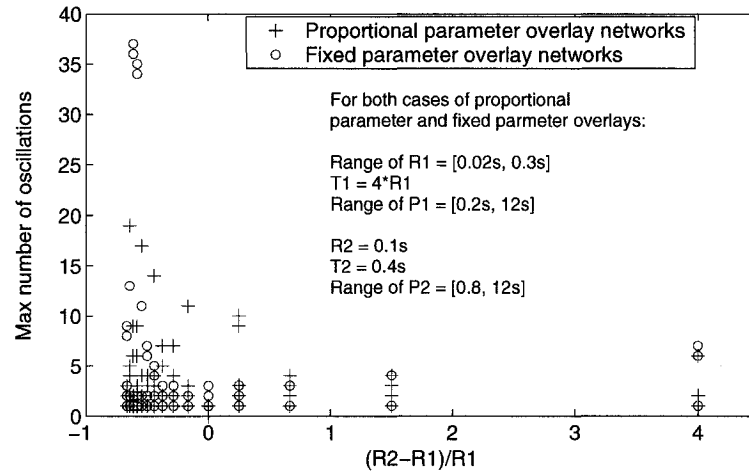


Figure 7.18: Maximum Number of Oscillations as a Function of Relative RTT

7.10 Implications of synchronization in overlays

Today's Internet does not have multiple overlay networks deployed, that is, not the type that use continuous probing to do path selection. We have explored the possibility of race conditions occurring should multiple overlays get deployed so that we may accumulate some wisdom about how to design such networks before they become widely used. We have seen a variety of scenarios in which the probability of synchronization exceeds 10%. These scenarios included cases in which RTTs were varied, probe rates were varied, and the relationships among the parameters were varied. We thus believe that there does exist a non-narrow range of the parameter space in which synchronization is non-negligible. We also illustrated that once synchronization occurs, the resulting oscillations can sometimes last for a long time. We thus believe that these issues should be taken into consideration when overlay networks are designed and configured.

One of the questions we set out to explore was whether we can count on variations in RTT alone to avoid synchronization. When using a proportional approach to parameter selection,

$P(S)$ can be significant (e.g., 40%) when the relative difference in RTTs is small (roughly less than 70%) and the overlays are aggressive. When using a fixed approach to parameter setting, $P(S)$ can exceed 10% even when the relative difference is as high as 300%. Moreover, when using the fixed approach, the absolute value of RTTs matter and large RTTs can bring about larger $P(S)$. This implies, for example, that designing a cross-continental overlay network is more challenging than designing one in a single country. Since synchronization can occur, even when overlay paths have dissimilar RTTs (whether large or small), overlay network designers should *not* rely upon differences in RTTs to avoid synchronization.

We believe that overlay networks should be designed with care so as to mitigate race conditions as much as possible. This is non-trivial as we have shown there is no “ideal” set of parameters that ensures avoidance of synchronization. Our results indicate that using a proportional approach to parameter selection might be better than using a fixed one. The proportional approach narrows down the range of relative RTTs in which synchronization can occur. Also, proportional parameter overlays depend only on the *relative* RTTs and can exploit the path diversity in the Internet better than fixed parameter overlays. But in reality, different overlay networks could easily end up choosing the same strategic locations to place their nodes, resulting in similar RTT values for various paths in different overlays, thus making it harder to achieve and exploit path diversity. In other words, using either a proportional or fixed parameter approach could result in a fair chance of experiencing oscillations.

We saw that $P(S)$ is more sensitive to the low-frequency probe P than the higher frequency probe Q . It appears that the best approach for averting race conditions, is for overlays to be nonaggressive in their probing, i.e., by using large values of P . The tradeoff here is a slower

reaction time. We showed that being nonaggressive can result in smaller values of $P(S)$ even if other overlays are aggressive. There may be implications here in terms of fairness. We also show that it is beneficial to be nonaggressive, as we suspect that the trend is towards building more aggressive overlays because of the popular belief that overlays can outperform layer-3 networks in terms of their reaction time to performance degradation events. We wish to point out that when many overlays start to co-exist, aggressive probing can have negative consequences and overlays can inadvertently step on each other.

7.11 Limiting the Impact of Race Conditions

As we showed earlier, race conditions among multiple overlay networks could affect not only the overlay traffic, but also non-overlay traffic. In this section, we discuss methods that could potentially limit the impact of race conditions between overlays.

7.11.1 Limiting the Impact of Synchronization

To limit the impact of synchronization among multiple overlay networks we can take two approaches: (i) reduce the probability of synchronization among overlays, and/or (ii) reduce the number of oscillations once the overlays get synchronized.

Reducing the Probability of Synchronization

Intuitively, one way to make it less likely that two overlays actually get synchronized would be to add randomness into the probing procedure. The idea of adding randomness was illustrated to help in the case of periodic routing protocols in [37]. Here we study the resulting

behavior of overlays when we add randomness to their probing parameters. The hope is that these random values will drive the reaction times of the overlay networks far apart, thus reducing the possibility of synchronization.

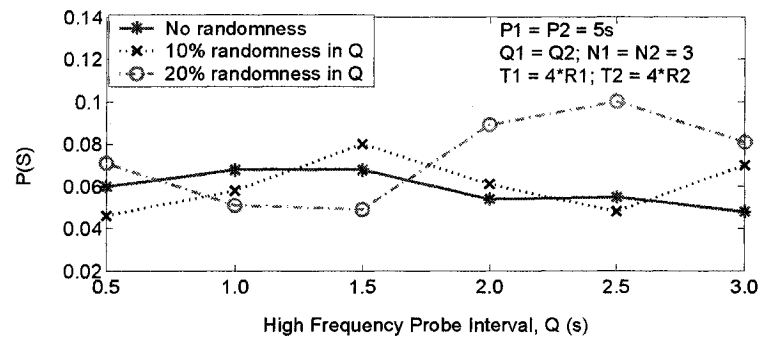


Figure 7.19: $P(S)$ for Randomized Probing Parameters

Figure 7.19 shows the effect of adding randomness into overlay probing parameters on the likelihood of overlay network synchronization in our simulator. The two overlay networks have identical high frequency and low frequency probing intervals (i.e., the same values of P and Q). In the results in Figure 7.19, the value of P is kept constant, while the value of Q is randomized by a certain percentage of its own value (for example, 10%, 20%, etc.). Let us consider the case of adding 10% randomness. In this case, we add a random value chosen from a uniform distribution between 0 and $0.1Q$ to the original value of Q . Although we use a similar strategy to add randomness to both the overlays, the random values chosen for each of the two overlays are independent of each other.

In Figure 7.19 we can see that adding randomness to the high frequency probe interval does not help in decreasing the probability of synchronization. Based on the fact that $P(S)$ depends on the difference terms (like $N_1Q_1 - N_2Q_2$ in Equation 7.3), the randomness added to the the same parameters in two overlays could either increase or decrease the value of $P(S)$. We have

repeated these experiments by adding randomness to other probing parameters (i.e., P and T), but found very similar results (not shown due to space constraints). Hence adding randomness does not always ensure that two overlays are less likely to synchronize.

Reducing the Number of Oscillations

In order to reduce the number of oscillations after a synchronization event, we propose an approach based on the well-known behavior of TCP. Whenever a flow using TCP experiences a packet loss due to congestion the protocol backs off from using an aggressive packet transfer rate. Typically this back-off occurs at an exponential rate to reduce the impact of congestion. In our case of multiple overlays, we propose to use a similar back-off technique where an overlay network successively increases the reaction time each time it decides to switch routes between the same source and destination nodes (if the reactions occur in a small time interval). In other words, this is similar in spirit to dampening, i.e., to slow down the reaction time of a protocol so as to avoid responding too quickly. Note that the back-off technique is also similar to the idea of non-aggressive probing. The main difference is that while using non-aggressive probing, the parameter (or timer) values are always large, but while using back-off strategy the parameter values are increased only when oscillations are detected.

Figure 7.20 shows the effect of using back-off techniques on the number of oscillations between two overlays that are synchronized. In these simulations, we use two different approaches to accomplish back-off: (i) Exponential back-off where the amount of time that an overlay waits to change its path increases exponentially when the overlay has changed its path in the recent past. Note that in Figure 7.20 both overlays double the reaction time when the overlay has changed its path in the previous 20 seconds. (ii) Random back-off where the overlays wait for a random amount

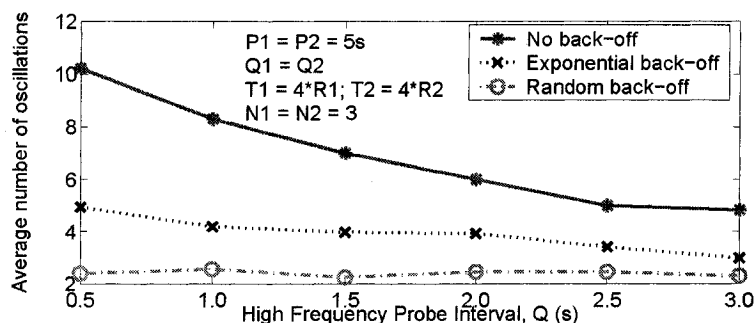


Figure 7.20: Effect of Using Back-off Technique on the Number of Oscillations

of time before deciding to move to an alternate path.

We can clearly see in Figure 7.20 that, on an average, using random back-off reduces the impact of synchronization more than exponential back-off. Since both overlays use the same back-off parameters there is a higher chance that the overlays will remain synchronized for a longer period when they use exponential back-off strategy. However, while using random back-off strategy, both overlays are more likely to wait for different amounts of time before reacting, and hence it results in reducing the impact of synchronization significantly.

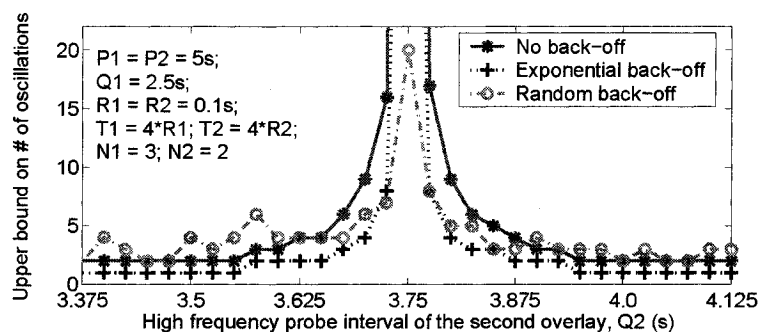


Figure 7.21: Effect of Using Back-off Technique on the Number of Oscillations when the Overlays are Using Different Values for N

Figure 7.21 explores the impact of using back-off technique on the *upper bound* of the number of oscillations. First we observe that when the number of oscillations is low, the two back-

off techniques perform similarly. Second, we see a huge spike at 3.75 on the x-axis, illustrating extreme sensitivity to a particular parameter setting. From Equation 7.12, we can see that there are plenty of combinations of parameter settings that will make the denominator zero, or close to zero, causing such spikes. Since overlays today do not coordinate their parameter choices, uncovering such situations is beyond the ability of an overlay network. This implies there are plenty of isolated cases where the race conditions we describe could be severe. For these cases random backup-off appears more effective in that it would quickly curb the spike by moving the parameters away from the sensitive setting.

7.11.2 Limiting the Impact of Cascading Reactions

Given the rise in popularity of overlay networks, we believe that the likelihood of occurrence of cascading reactions cannot be ignored and hence it is important to limit the impact of cascading reactions.

In addition to the presence of performance degradation and periodic process, the cascading reactions depend heavily on the topology overlapping of multiple overlays. The degradation of path performance depends on different events in the ISP network like failures, flash crowds, worms, and Distributed Denial-of-Service (DDoS) attacks, and is beyond the control of overlay networks.

The placement of overlay nodes in an overlay network combined with the routing strategy used by the underlying ISP network determines the virtual paths used by the overlay network. Different overlay networks typically do not share any information about their node placements, thus making it harder to avoid the overlap of primary and alternate paths. Another factor that could impact the topology condition for cascading reactions is the routing strategy of the ISP. A change in routing strategy in the ISP network could completely change the path overlap in the overlay

networks. A given overlay network has no control over the factors outlined above. Unless all the overlay networks and the ISP share information with each other (similar to the suggestion in [70]) it is hard to come up with solutions to mitigate the impact of cascading reactions.

7.12 Summary

In this chapter, we have shown that co-existing overlay networks can experience race conditions that could affect both the overlay and non-overlay traffic in the ISP network. We analytically modeled the probability of synchronization between two or more overlay networks and showed that overlay networks could get synchronized even when they use dissimilar set of probing parameters. We also explored strategies to reduce the impact of race conditions that can be used as a guideline for overlay network design in the future.

Chapter 8

Conclusions

The efficiency and performance of the Internet infrastructure critically depends on how well Internet service providers (ISPs) manage and run their individual networks. An important concern for ISPs in the context of efficient network management is the impact of routing protocol dynamics on the stability, robustness, and service availability of their network. This dissertation extends the current understanding in two main aspects: (i) the behavior of IGP protocols during failures, and (ii) the impact of interactions between protocols that run at layer-3 and layer-7. In this chapter, we summarize the overall contributions of this dissertation and point out the potential directions for future research.

8.1 Summary of Contributions

The main contributions of this dissertation can be summarized as follows:

- We characterize the dynamic behavior of IGP protocols in the face of network failures and develop a model to predict this behavior. We define a set of metrics for *service availability*

of IP backbone networks that capture the impact of routing dynamics on packet forwarding. Using extensive analysis and experimentation, we show that traditional graph-theoretic properties like node-degree, network diameter, tree-depth, etc. do not provide any insights on the service offered by a network, making it imperative to use our new metrics. With the help of realistic simulations, we show the feasibility and advantages of using our metrics.

- Instead of relying on active or passive measurements, we propose a methodology to *estimate* the service availability of a network in the presence of link failures. The main advantage of this methodology is that it can be used in the *network design* phase to design networks with higher service availability. We derive *goodness factors* based on three different perspectives: ingress node (from one node to many destinations), link (traffic traversing a link), and network-wide (across all source-destination pairs). We describe several applications for goodness factors in network planning and provisioning like IGP link weight setting, BGP peering points for a network, etc.
- We propose a new paradigm for packet forwarding called *loopless interface specific forwarding* (LISF) to address the problem of transient forwarding loops during IGP convergence. We compare our approach to the existing approaches to show that LISF does not incur any additional message overhead or increased convergence delay. We propose several algorithms for LISF (from being very conservative to being very aggressive) that are applicable in a variety of different IGP routing scenarios. We show the feasibility of deploying our algorithms in today's routers without any changes to the current link-state routing mechanisms.
- We also show that the novel concept of service availability can be used to address several other critical aspects of network management like graceful network upgrade, i.e., adding new

nodes and links into an operational network in a graceful manner so that the perceived network performance from the perspective of existing customers does not deteriorate. We propose a two-phase framework that can be used to determine the optimal network upgrade strategy in ISP backbone networks. We believe that with only minor modifications, this framework can be applied in the context of different network service providers like cellular service providers, WiMax service providers, etc. In both the phases of our framework, we highlight the importance of considering network performance from the perspective of both customers and ISPs. Using experiments, we show that such an approach performs significantly better than the traditional approach where network performance is considered only from the perspective of an ISP.

- We identify several interactions between overlay network routing protocols at layer-7 and the IP routing protocols at layer-3. We show that allowing routing control at both the application and the IP layers could have profound implications on how ISPs design, maintain, and run their networks. With help of simulations, we show that overlay networks could hamper ISP's ability to perform effective traffic engineering. In particular, we argue that overlay networks could make it difficult for ISPs to estimate Traffic Matrices (TM) and perform load balancing. We also show that overlay networks that span multiple domains could threaten the effectiveness of BGP in isolating different domains in the Internet.
- We also examine the interactions between several co-existing overlay networks and show that two (or more) co-existing overlays can experience race conditions and become synchronized leading to route and traffic oscillations, and cascading reactions (i.e., an event in one overlay can trigger a series of events in coexisting overlays). We pinpoint the reasons for these race

conditions in terms of partially overlapping paths and periodic probing process.

- We develop an analytical method to compute the probability of synchronization between two overlay networks as a function of the path probing parameters. We also provide an upper bound on the number of oscillations that the two overlays will experience after they are synchronized, in the absence of external events acting as stop triggers. We also show that our analytical model is extensible when an arbitrary number of overlay networks coexist.
- We show that the synchronization probability is not negligible for a wide range of parameters, suggesting that particular care must be given by network designers to the configuration of overlay routing protocols. We illustrate that oscillations can occur even for two overlays that deploy considerably dissimilar path probing parameters. We study the impact of path diversity and probing aggressiveness on the probability of synchronization. We also explore techniques to reduce the impact of race conditions. We show that adding randomness to the probing parameters does *not always* help in decreasing the probability of synchronization. However a back-off mechanism, if deployed in overlays, could help in significantly reducing the ill-effects of oscillations. We also show that it is harder to avoid cascading reactions and hence application and network providers should revisit deployment strategies of overlays in the future.

8.2 Future Research Directions

There are several interesting directions for future research that one could pursue based on the work presented in this dissertation. In this section, we will briefly discuss some of these open future research directions.

8.2.1 Stop Triggers for Oscillations

Although we have extensively examined the causes, impact, and remedies for oscillations, a detailed study of triggers that stop these oscillations was beyond the scope of this dissertation. One possible direction of future research is to gain a comprehensive understanding of stop triggers for oscillations. As we observed in Chapter 7, certain IP layer events that act as stop triggers for oscillations at some point in time might not affect oscillations at another point in time. Also, most of the IP layer events that act as stop triggers are heavily dependent on the network conditions at the IP layer. For example, IGP convergence depends not only on timer values set by the ISPs, but also on the location of BGP peering points [55]. The order of occurrence of these stop triggers is not deterministic, thus introducing unpredictability in the duration of the oscillations. In essence, the end of oscillations depends on numerous factors, thus making it non-trivial to accurately estimate the impact of oscillations on overlay or non-overlay traffic.

Further research is needed to investigate the following: (i) What are the various possible stop triggers? (ii) What is the stop trigger for a given set of oscillations? (iii) What is the frequency of occurrence of different stop triggers? (iv) How does the network topology and conditions influence the occurrence of stop triggers?, (v) Is the occurrence of stop triggers related to the number of overlay networks that are synchronized? (vi) How different are the characteristics of triggers that stop local and global oscillations?

8.2.2 Transparency between ISPs and Application Service Providers

As we saw in the first part of this dissertation, one of the main reasons for poor performance of real-time delay and/or loss sensitive applications, such as VoIP and multimedia streaming,

is the lack of transparency between ISPs and application service providers. In other words, the applications lack information regarding the behavior of the underlying network layer protocol. Based on the framework developed in this dissertation for predicting the duration and impact of service disruption, one possible solution for this problem is to design a mechanism where ISPs can share information regarding the network behavior with applications. Using such information, applications can dynamically adapt their strategies to significantly improve their throughput. In fact, we have started a basic study to explore this idea in the context of real-time multimedia streaming and found that the performance from the perspective of end users dramatically increased by giving applications more information about the underlying network [17]. Such information about the networks can be provided by ISPs as a service to the applications, and hence could prove to be beneficial to ISPs, application service providers, as well as end users.

Another motivation for creating transparency between ISPs and application service providers arises from our findings in Chapters 6 and 7. We have shown how overlay networks at the application-layer (or layer-7) could interact with ISPs at the IP-layer (or layer-3) creating severe problems for ISPs. The main reason for application service providers to rely on overlay networks is due to the fact that ISPs do not give them the flexibility in choosing the best paths for their applications. Given the potential problems for ISPs in traffic engineering, load balancing, and coupling of different domains, it would be beneficial for ISPs to share information with application service providers. This will help application service providers to choose the paths that best meet the requirements of the applications. Although this solution appears simple at the first glance, the complexity stems from the economic and game theoretic aspects of designing an effective and fair solution.

8.2.3 Overlay Network Discovery Using *DiscOver*

As pointed out in Chapters 6 and 7, ISPs could potentially face numerous challenges due to the existence of overlay networks. For example, it could be very hard for ISPs to engineer the traffic in their network to achieve the kind of load balancing that they need to satisfy the SLAs with their customers. ISPs also face the threat of overlay networks affecting the background traffic due to traffic oscillations. Furthermore, overlays that span multiple domains could couple different domains with each other, thus making the state of one domain dependent on the state of another domain. The main reason for these problems is the fact that overlay networks are opaque to ISPs in terms of their topology and routing strategies. In this future research direction, we envision a measurement based technique, called *DiscOver* (i.e., *Discover Overlay Topology and Routing Strategy*), that helps ISPs to discover overlay node locations and routing strategies.

Typically, overlay networks use an active probing mechanism to monitor the health of various alternate paths. When the quality of the currently used path deteriorates, overlay networks find an alternate path (if one exists) that satisfies their quality criteria and reroute traffic along those paths. Even though these alternate paths meet the quality criteria of overlay networks, they may violate the TE policies of the ISP. Such rerouting techniques modify the network traffic matrix (TM) either by *duplicating* a single TM entry in multiple places if the overlay path spans a single domain, or by *shifting* a TM entry to another place in the TM if the overlay path spans multiple domains (as presented in Chapter 6). The important point to observe here is that these changes in the traffic matrix entries are *not random*, but follow a particular *pattern* that is determined by the overlay network topology and routing policies. We believe that observing changes in the subsequent estimations of TM over a long period of time will help ISPs discover not only the location of overlay

nodes but also the routing strategies that they use. As an example let us look at the two types of changes in the TM entries (i.e., *duplication* and *shift*), and how these changes can help in identifying the overlay nodes in a network.

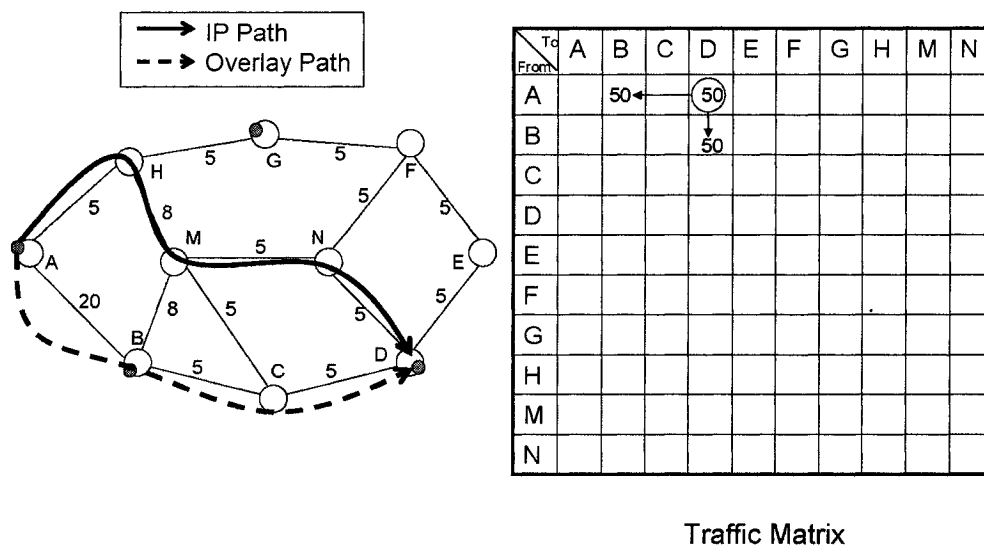


Figure 8.1: Change in Traffic Matrix Entries in a Domain due to Overlay Networks that Span a Single Domain

Overlays Spanning a Single Domain. Consider the network in Figure 8.1, which has multiple overlay nodes (at A, B, D and G) in an ISP domain. Suppose the traffic between nodes A and D is 50 units. If IP routing is used then the TM entry for the source-destination pair A – D is 50 as shown in the traffic matrix in Figure 8.1. However, if overlay routing intervenes and decides to route the traffic through an overlay path (say, A – B – D) that offers better latency, then the TM entry for the pair A – D is *uplicated* as two entries of 50 units each, one for A – B and another for B – D, while the value for the entry A – D is 0. This means that the changes in TM entries that an ISP observes during TM estimations combined with the logs of the events in the network can help

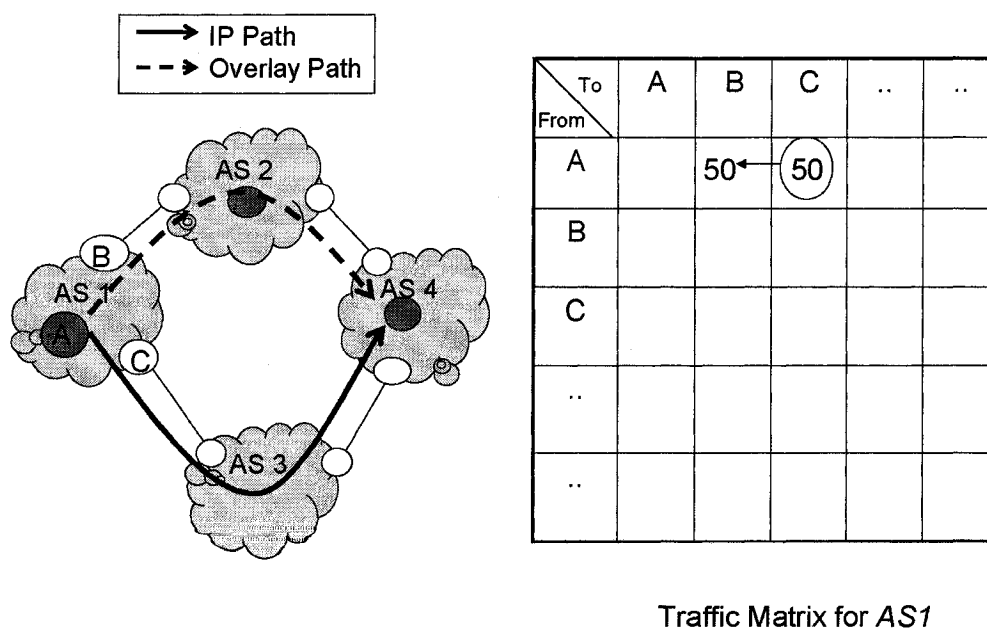


Figure 8.2: Change in Traffic Matrix Entries in a Domain due to Overlay Networks that Span Multiple Domains

in determining the locations of overlay nodes and their routing strategies.

Overlays Spanning Multiple Domains. There are many flows whose ultimate destination lies outside the ISPs' domain; the traffic from these flows traverses the ISP and thus appears inside the TM. The traffic matrix will specify an exit router within the ISP's domain for such flows. If this traffic belongs to an overlay network that spans multiple domains, and uses its own path selection mechanism, then the exit point within a single ISP domain could change, resulting in a *shift* in TM entry. For example, consider the network in Figure 8.2. Suppose that the layer-3 path from AS1 to AS4 is through AS3. If the overlay network discovers a better path through AS2, then the overlay network could switch the routing path, thus changing the associated exit point. The TM entry in AS1 for A – C now *shifts* to A – B as shown in the traffic matrix in Figure 8.2. This implies that

an ISP could correlate the event inside the network (i.e. IGP events) and events outside the network (i.e. BGP events) with the changes in TM entries to assist in locating overlay nodes and their routing strategies.

There are still numerous questions that need to be answered in terms of feasibility of this approach. For example: (i) how can we differentiate the TM change patterns due to overlay networks from other events such as worm attacks, DDoS attacks, flash crowds, etc.? (ii) how does the inaccuracies in TM estimation impact the effectiveness of this approach?

8.3 Lessons Learned and Concluding Remarks

Due to its extremely complex nature, network management is often considered as an *art* more than *science* by network operators. A *perfect* network management mechanism has been a distant and elusive goal despite enormous efforts over the last 10 to 15 years. Managing a large network is extremely difficult for several reasons: (i) Need to accurately control and predict the behavior of the network to ensure correctness, efficiency, fault tolerance, high availability, and performance, (ii) Need to configure the network to reflect both economic and routing policies of the ISP or domain controller, and (iii) Need to ensure that the network is safe, secure, and resilient to internal and external attacks. In this dissertation, we have systematically addressed some of the issues in controlling and predicting the behavior of a network to help in enhancing the current capabilities in network management.

8.3.1 Importance of Realistic Analytical Models

One of the basic requirements for effectively controlling a network is to gain a comprehensive understanding of the behavior of the routing protocols that form the backbone for communication between any two nodes/hosts in the Internet. In this dissertation, we first analyzed, and subsequently developed a model to characterize the dynamic behavior of intra-domain routing protocols (like ISIS and OSPF) during network resource failures. Using this model, we identified the occurrence of *routing loops*, an anomalous network condition at very small timescales, during IGP convergence. Although these routing loops last for a small duration, they can easily overwhelm even the high bandwidth backbone links, significantly deteriorating the network performance. We have proposed several algorithms to prevent routing loops based on a new paradigm for routing called *Loopless Interface Specific Forwarding (LISF)*. An important point to note here is that, had there been no model for IGP convergence, we would not have been able to discover the occurrence routing loops. One could argue that measurement-based studies could have accomplished the same task. However, we believe that, in several situations like the one here, measurement-based techniques could be severely constrained by the location of the measurement equipment. For example, our efforts in finding routing loops using the measurement data from a tier-1 ISP network was not successful mainly due to the fact that the measurement equipments were not deployed to collect the data at the required locations. Even though measurement-based studies are extremely important to understand the functioning of networks, we strongly believe that it is equally important to develop realistic analytical models to understand protocol behavior.

8.3.2 Role of Predictability in Network Management

Another important advantage of developing realistic analytical models for IGP protocols is in accurately *predicting* and *estimating* the future network behavior, a critical requirement for effective network management. Based on our model for dynamic IGP behavior, in this dissertation, we have defined a set of metrics to capture the *instantaneous service availability* of a network. We have used this concept of service availability to characterize networks and extract their *goodness* from the perspective of both ISPs and customers. The important point to observe here is that all of the metrics for service availability (and hence the goodness of a network) are predictable and do not require active or passive measurements. This implies that we can use this concept in the design and planning stages of network management to uncover better network configurations that can offer high availability and performance. In fact, we have demonstrated the importance and advantages of accurately predicting network behavior in determining graceful network upgrade strategies.

8.3.3 Synergistic Coexistence of Distributed Protocols

The Internet is comprised of several ASes, and each of these ASes run their own intra-domain routing protocol inside their AS, and a common inter-domain routing protocol (i.e., BGP) between ASes. Traditionally, these protocols have been designed to be independent of each other. With the exception of known and deliberate channels of communication between the two sets of protocols, network management efforts have dealt with them independently. Over the last 6-7 years, large-scale (i.e., Internet-scale) overlay networks have become really popular for a variety of applications including p2p file sharing, resilient routing, and content distribution. These overlay networks try to accomplish routing at the application layer using their own distributed rout-

ing protocols. These routing protocols are designed to work independently on top of the layer-3 routing protocols and provide applications with more flexibility than the traditional layer-3 routing protocols in choosing end-to-end paths in the Internet.

In this dissertation, we have shown that different distributed routing protocols (in layer-3 and layer-7) that coexist in the Internet could unintentionally interact with each other. We showed that interactions between overlay and IP routing protocols could have profound implications on how ISPs design, maintain, and manage their networks. For example, we have described in detail how overlay networks can hamper an ISP's ability to perform effective traffic engineering and load balancing. We also examined the interactions between several coexisting overlay networks and showed that two (or more) coexisting overlays can experience race conditions and become synchronized leading to route and traffic oscillations, and cascading reactions. These race conditions affect both overlay and non-overlay traffic, creating new problems for effective network management. The main reason for these problems is the fact that these distributed protocols were designed independently without taking into account potential interactions with one another. Different distributed protocols should form a synergistic coexistence in the Internet. One of the key lessons learned here is that it is extremely important, from the perspective of both network and application designers to perform detailed study on how different protocols interact with one another to ensure stability of the system.

In this dissertation, we have addressed several issues that are critical to efficient network management. We have shown the importance and usefulness of realistic analytical models in identifying and solving existing problems in the routing infrastructure. We have also discussed in detail the role of "predictability" in the context of network management, and made a strong case for future

research in this direction. Finally, we have explored the unintentional interactions between different routing protocols in the Internet to show that it is extremely important to model interactions between different network control protocols/mechanisms to detect and prevent race conditions. Although, the “perfect” network management mechanism is still a distant and elusive dream, we strongly believe that the work in this dissertation brings us a few steps closer to realizing that dream.

Bibliography

- [1] Akamai. <http://www.akamai.com>.
- [2] CNN. <http://www.cnn.com>.
- [3] Gnutella. <http://www.gnutella.com>.
- [4] Kazaa. <http://www.kazaa.com>.
- [5] Napster. <http://www.napster.com>.
- [6] Sprint ATL Tutorial. http://ipmon.sprint.com/pubs_trs/tutorials/Backbone-Architecture-And-Performance.pdf.
- [7] Yahoo Inc. <http://www.yahoo.com>.
- [8] AGARWAL, S., CHUAH, C.-N., BHATTACHARYYA, S., AND DIOT, C. The Impact of BGP Dynamics on Intra-domain Traffic. In *Proceedings of ACM Sigmetrics* (June 2004).
- [9] AGARWAL, S., NUCCI, A., AND BHATTACHARYYA, S. Controlling Hot Potatoes in Intradomain Traffic Engineering. Sprint ATL Research Report RR04-ATL-070677.pdf, Sprint ATL, July 2004.

- [10] AHUJA, R., MAGNANTI, T., AND ORLIN, J. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, 1993.
- [11] AKELLA, A., PANG, J., MAGGS, B., SESHAN, S., AND SHAIKH, A. A Comparison of Overlay Routing and Multihoming Route Control. In *Proceedings of ACM Sigcomm* (Sept. 2004).
- [12] ALAETTINOGLU, C., AND CASNER, S. ISIS Routing on the Qwest Backbone: A Recipe for Subsecond ISIS Convergence. In *North American Network Operators Group (NANOG) 24* (Feb. 2002).
- [13] ALATTINOGLU, C., JACOBSON, V., AND H.YU. Towards Milli-Second IGP Convergence. IETF Internet Draft, Nov. 2000. draft-alaettinoglu-ISIS-convergence-00.txt.
- [14] ALTMANN, J., AND RHODES, L. Dynamic Netvalue Analyzer: A Price Plan Modeling Tool for ISPs Using Actual Network Usage Data. In *IEEE International Workshop on Advance Issues in E-Commerce and Web-Based Information Systems* (2002).
- [15] AMERICAN NATIONAL STANDARD FOR TELECOMMUNICATIONS. Technical Report on Enhanced Network Survivability Performance. ANSI T1.TR.68, Feb. 2001.
- [16] ANDERSON, D., BALAKRISHNA, H., KAASHOEK, M., AND MORRIS, R. Resilient Overlay Networks. In *Proceedings of ACM Symposium on Operating Systems Principles* (Oct. 2001).
- [17] ANDREOPOULOS, Y., KERALAPURA, R., VAN DER SCHAAR, M., AND CHUAH, C.-N. Failure-Aware, Open-Loop, Adaptive Video Streaming With Packet-Level Optimized Redundancy. *IEEE Transactions on Multimedia* (Dec. 2006).

- [18] ASH, G. *Dynamic Routing in Telecommunication Networks*. McGraw-Hill, 1997.
- [19] ATLAS, A. U-turn Alternates for IP/LDP Fast-Reroute. IETF Internet Draft, Feb. 2005. draft-atlas-ip-local-protect-uturn-02.
- [20] BAILEY, H., YELTON, R., AJIBULU, A., HOPKINS, M., LOUTH, G., AND NIVA, M. A Multifaceted Approach to Forecasting Broadband Demand and Traffic. *IEEE Communications Magazine* 33 (Feb. 1995).
- [21] BATES, T., CHANDRA, R., AND CHEN, E. BGP Route Reflection - An Alternative to Full Mesh IBGP. RFC 2796, Apr. 2000.
- [22] BELLOVIN, S., BUSH, R., GRIFFIN, T. G., AND REXFORD, J. Slowing Routing Table Growth by Filtering Based on Address Allocation Policies. In *North American Network Operators Group (NANOG) 22* (May 2001).
- [23] BOUTERMANS, C., IANNACCONE, G., AND DIOT, C. Impact of Link Failures on VoIP Performance. In *Proceedings of NOSSDAV* (May 2002).
- [24] BRAYNARD, R., KOSTIC, D., RODRIGUEZ, A., CHASE, J., AND VAHDAT, A. Opus: An Overlay Peer Utility Service. In *Proceedings of IEEE Open Architectures and Network Programming (OpenArch)* (June 2002).
- [25] BRYANT, S., FILSFILS, C., PREVIDI, S., AND SHAND, M. IP Fast Reroute using Tunnels. IETF Internet Draft, May 2004. draft-bryant-ipfrr-tunnels-00.txt.
- [26] BRYANT, S., AND ZHANG, M. A Framework for Loop-Free Convergence. IETF Internet Draft, Oct. 2004. draft-bryant-shand-lf-conv-frmwk-00.txt.

- [27] CALVERT, K., DOAR, M., AND ZEGURA, E. W. Modeling Internet Topology. In *IEEE Communications Magazine* (June 1997).
- [28] CASTRO, M., DRUSCHEL, P., KERMARREC, A., NANDI, A., ROWSTRON, A., AND SINGH, A. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proceedings of ACM Symposium on Operating Systems Principles* (Oct. 2003).
- [29] CASTRO, M., DRUSCHEL, P., KERMARREC, A., AND ROWSTRON, A. SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communication* (Oct. 2002).
- [30] CHOUDHURY, G., MAUNDER, A., AND SAPOZHNIKOVA, V. Faster Link-State IGP Convergence and Improved Network Scalability and Stability. In *Proceedings of IEEE International Conference on Local Computer Networks* (2001).
- [31] CLOUQUEUR, M., AND GROVER, W. Availability Analysis of Span-Restorable Mesh Networks. *IEEE Journal on Selected Areas in Communication* (May 2002).
- [32] CURRAN, K., FLYNN, P., AND LUNNEY, T. A Decision Support System for Telecommunications. *International Journal of Network Management*, 12 (2002).
- [33] DEERING, S. Host Extensions for IP Multicasting. RFC 1112, Aug. 1989.
- [34] ENRIQUEZ, P., BROWN, A., AND PATTERSON, D. Lessons from the PSTN for Dependable Computing. In *Proceedings of the 2002 Workshop on Self-Healing, Adaptive and Self-Managed Systems* (June 2002).

- [35] FALOUTSOS, C., FALOUTSOS, P., AND FALOUTSOS, M. On Power-Law Relationships of the Internet Topology. In *Proceedings of ACM Sigcomm* (Sept. 1999).
- [36] FILSFILS, C. Deploying Tight-SLA Services on an IP Backbone, June 2002. <http://www.nanog.org/mtg-0206/ppt/filsfils>.
- [37] FLOYD, S., AND JACOBSON, V. The Synchronization of Periodic Routing Messages. In *Proceedings of ACM Sigcomm* (Sept. 1993).
- [38] FORSYTH, P., DHALLUIN, Y., AND VETZAL, K. R. Managing Capacity for Telecommunications Networks under Uncertainty. *IEEE/ACM Transactions on Networking* 10, 4 (Aug. 2002).
- [39] FORTZ, B., AND THORUP, M. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of IEEE Infocom* (Mar. 2000).
- [40] FRALEIGH, C., MOON, S., LYLES, B., COTTON, C., KHAN, M., MOLL, D., ROCKELL, R., SEELY, T., AND DIOT, C. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network* (Nov. 2003).
- [41] FRANCOIS, P., AND BONAVENTURE, O. Avoiding Transient Loops during IGP Convergence in IP Networks. In *Proceedings of IEEE Infocom* (Mar. 2005).
- [42] GAO, R., DOVROLIS, C., AND ZEGURA, E. Avoiding Oscillations due to Intelligent Route Control Systems. In *Proceedings of IEEE Infocom* (Apr. 2006).
- [43] GARCIA-LUNA-ACEVES, J., AND MURTHY, S. A path-finding algorithm for loop-free routing. *IEEE/ACM Transactions on Networking* 5, 26 (Feb. 1997).

- [44] GIROIRE, F., NUCCI, A., TAFT, N., AND DIOT, C. Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection. In *Proceedings of IEEE Infocom* (Mar. 2003).
- [45] GOVINDAN, R., AND TANGMUNARUNKIT, H. Heuristics for Internet Map Discovery. In *Proceedings of IEEE Infocom* (Mar. 2000).
- [46] GRIFFIN, T., AND PREMORE, B. An Experimental Analysis of BGP Convergence Time. In *Proceedings of International Conference on Network Protocols* (Nov. 2001).
- [47] GRIFFIN, T., AND WILFONG, G. An Analysis of BGP Convergence Properties. In *Proceedings of ACM Sigcomm* (Sept. 1997).
- [48] GROVER, W. High Availability Path Design in Ring-based Optical Networks. *IEEE/ACM Transactions on Networking* (Aug. 1999).
- [49] HAMILTON, C. Telecommunication-Network Dependability: A Baseline on Local-Exchange Network Availability. In *Proceedings of Reliability and Maintainability Symposium* (1991).
- [50] IANNACCONE, G., CHUAH, C., BHATTACHARYYA, S., AND DIOT, C. Feasibility of IP Restoration in a Tier-1 Backbone. Tech. Rep. RR03-ATL-030666, Sprint ATL, Mar. 2003.
- [51] IANNACCONE, G., CHUAH, C., MORTIER, R., BHATTACHARYYA, S., AND DIOT, C. Analysis of Link Failures in an IP Backbone. In *Proceedings of ACM Sigcomm Internet Measurement Workshop* (Nov. 2002).
- [52] IANNACCONE, G., CHUAH, C.-N., BHATTACHARYYA, S., AND DIOT, C. Feasibility of

- IP Restoration in a Tier-1 Backbone. *IEEE Network Magazine, Special Issue on Protection, Restoration and Disaster Recovery* (2004).
- [53] IYER, S., BHATTACHARYYA, S., TAFT, N., AND DIOT, C. An Approach to Alleviate Link Overload as Observed on an IP Backbone. In *Proceedings of IEEE Infocom* (Mar. 2003).
- [54] JAGANNATHAN, S., ALTMANN, J., AND RHODES, L. A Revenue-based Model for Making Resource Investment Decisions in IP Networks. In *IEEE/IFIP Symposium on Integrated Network and System Management* (2003).
- [55] KERALAPURA, R., CHUAH, C., IANNACCONE, G., AND BHATTACHARYYA, S. Service Availability: A New Approach to Characterize IP Backbone Topologies. In *Proceedings of IEEE International WorkShop on Quality of Service* (June 2004).
- [56] KERALAPURA, R., MOERSCHELL, A., CHUAH, C., IANNACCONE, G., AND BHATTACHARYYA, S. A Case for Using Service Availability to Characterize IP Backbone Topologies. *Journal of Communications and Networks* (June 2006).
- [57] KUHN, R. Sources of Failure in the Public Switched Telephone Network. *IEEE Computer* 30, 4 (1997).
- [58] KUROSE, J., AND ROSS, K. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Computer Science, 2001.
- [59] LABOVITZ, C., MALAN, G. R., AND JAHANIAN, F. Internet Routing Instability. *IEEE/ACM Transactions on Networking* 6, 5 (Oct. 1998).
- [60] LABOVITZ, C., WATTENHOFER, R., VENKATACHARY, S., AND AHUJA, A. The Impact

- of Internet Policy and Topology on Delayed Routing Convergence. In *Proceedings of IEEE Infocom* (Apr. 2001).
- [61] LEIGHTON, T. The Challenges of Delivering Content and Applications on the Internet. In *NSDI Keynote* (May 2005).
- [62] LIU, Y., ZHANG, H., GONG, W., AND TOWSLEY, D. On the Interaction Between Overlay Routing and Traffic Engineering. In *Proceedings of IEEE Infocom* (Mar. 2005).
- [63] MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Understanding BGP Misconfiguration. In *Proceedings of ACM Sigcomm* (Aug. 2002).
- [64] MALKIN, G. RIP Version 2. RFC 2453, Nov. 1998.
- [65] MAO, Z. M., GOVINDAN, R., VARGHESE, G., AND KATZ, R. H. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proceedings of ACM Sigcomm* (Aug. 2002), ACM Press, pp. 221–233.
- [66] MARKOPOULOU, A., IANNACCONE, G., BHATTACHARYYA, S., CHUAH, C. N., AND DIOT, C. Characterization of Failures in an IP Backbone Network. In *Proceedings of IEEE Infocom* (Mar. 2004).
- [67] MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. BRITE: An Approach to Universal Topology Generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)* (Aug. 2001).
- [68] MEDINA, A., TAFT, N., SALAMATIAN, K., BHATTACHARYYA, S., AND DIOT, C. Traffic

- Matrix Estimation: Existing Techniques and Future Directions. In *Proceedings of ACM Sigcomm* (Aug. 2002).
- [69] MOY, J. OSPF Version 2. RFC 2328, Apr. 1998.
- [70] NAKAO, A., PETERSON, L., AND BAVIER, A. A Routing Underlay for Overlay Networks. In *Proceedings of ACM Sigcomm* (Aug. 2003).
- [71] NELAKUDITI, S., LEE, S., YU, Y., AND ZHANG, Z. Failure Insensitive Routing for Ensuring Service Availability. In *Proceedings of IEEE International WorkShop on Quality of Service* (June 2003).
- [72] NUCCI, A., SCHROEDER, B., BHATTACHARYYA, S., TAFT, N., AND DIOT, C. IGP Link Weight Assignment for Transient Link Failures. In *ITC* (2003).
- [73] ORAN, D. OSI IS-IS Intra-domain Routing Protocol. RFC 1142, Feb. 1990.
- [74] OU, C., AND ZANG, H. Sub-Path Protection for Scalability and Fast Recovery in Optical WDM Mesh Networks. *IEEE/ACM Transactions on Networking* (2002).
- [75] PEI, D., WANG, L., MASSEY, D., WU, S., AND ZHANG, L. A Study of Packet Delivery Performance during Routing Convergence. Technical Report UCLA-CSD-TR-030004, UCLA CSD. citeseer.ist.psu.edu/pei03study.html.
- [76] PREMERE, B. An Experimental Analysis of BGP Convergence Time. In *Proceedings of International Conference on Network Protocols* (Nov. 2001), IEEE Computer Society, p. 53.
- [77] QIU, L., YANG, Y., ZHANG, Y., AND SHENKER, S. On Selfish Routing in Internet-Like Environments. In *Proceedings of ACM Sigcomm* (Aug. 2003).

- [78] RADOSLAVOV, P., TANGMUNARUNKIT, H., YU, H., GOVINDAN, R., SHENKER, S., AND ESTRIN, D. On Characterizing Network Topologies and Analyzing Their Impact on Protocol Design. Tech. Rep. USC-CS-TR-00-731, Mar. 2000.
- [79] REKHTER, T. A Border Gateway Protocol 4 (BGP-4). RFC 1771, Mar. 1995.
- [80] REXFORD, J., WANG, J., XIAO, Z., AND ZHANG, Y. BGP Routing Stability of Popular Destinations. In *Proceedings of ACM Sigcomm Internet Measurement Workshop* (2002), ACM Press, pp. 197–202.
- [81] RHEA, S., GEELS, D., ROSCOE, T., AND KUBIATOWICZ, J. Handling Churn in a DHT. In *USENIX ATC* (June 2004).
- [82] ROUTING AREA WORKING GROUP. <http://psg.com/zinin/ietf/rtgwg>, 2004.
- [83] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware* (Nov. 2001).
- [84] SAVAGE, S., COLLINS, A., HOFFMAN, E., SNELL, J., AND ANDERSON, T. The End-to-End Effects of Internet Path Selection. In *Proceedings of ACM Sigcomm* (Aug. 1999).
- [85] SHAIKH, A., AND GREENBERG, A. Experience in Black-box OSPF Measurements. In *Proceedings of ACM Sigcomm Internet Measurement Workshop* (Nov. 2001).
- [86] SOULE, A., NUCCI, A., LEONARDI, E., CRUZ, R., AND TAFT, N. How to Identify and Estimate the Largest Traffic Matrix Elements in a Dynamic Environment. In *Proceedings of ACM Sigmetrics* (June 2004).

- [87] SRIDHARAN, A., MOON, S., AND DIOT, C. On the Causes of Routing Loops. In *Proceedings of ACM Sigcomm Internet Measurement Conference* (Oct. 2003).
- [88] SRIPANIDKULCHAI, K., MAGGS, B., AND ZHANG, H. An Analysis of Live Streaming Workloads on the Internet. In *Proceedings of ACM Sigcomm Internet Measurement Conference* (Oct. 2004).
- [89] STEWART, J. *BGP-4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1998.
- [90] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM Sigcomm* (August 2001).
- [91] STORDAHL, K., AND MURPHY, E. Forecasting Long-term Demand for Services in the Residential Market. *IEEE Communications Magazine* 33 (Feb. 1995).
- [92] VELLANKI, S., AND REDDY, A. N. Improving Service Availability during Link Failure Transients. Tech. Rep. TAMU-ECE-2003-02, Texas A&M University, ECE Dept, Feb. 2003.
- [93] WANG, L., ZHAO, X., PEI, D., BUSH, R., MASSEY, D., MANKIN, A., WU, F., AND ZHANG, L. Observation and Analysis of BGP Behavior Under Stress. In *Proceedings of ACM Sigcomm Internet Measurement Workshop* (2002).
- [94] WAXMAN, B. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communication* 6, 9 (Dec. 1988), 1617–1622.
- [95] XIAO, J., AND BOUTABA, R. Customer-centric Network Upgrade Strategy: Maximizing

- Investment Benefits for Enhanced Service Quality. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)* (2004).
- [96] ZHANG, H., LIU, Y., TOWSLEY, D., AND GONG, W. Understanding the Interaction between Overlay Routing and Traffic Engineering. In *ACM Sigcomm Poster Session* (Aug. 2004).
- [97] ZHANG, J., ZHU, K., ZANG, H., AND MUKHERJEE, B. A New Provisioning Framework to Provide Availability-Guaranteed Service in WDM Mesh Networks. In *IEEE ICC* (May 2003).
- [98] ZHAO, B., HUANG, L., STRIBLING, J., AND KUBIATOWICZ, J. Exploiting Routing Redundancy via Structured Peer-to-Peer Overlays. In *ICNP* (Nov. 2003).
- [99] ZHONG, Z., NELAKUDITI, S., YU, Y., LEE, S., WANG, J., AND CHUAH, C.-N. Failure Inferencing based Fast Rerouting for Handling Transient Link and Node Failures. In *Global Internet Symposium* (Miami, 2005).
- [100] ZININ, A. Analysis and Minimization of Microloops in Link-State Routing Protocols. IETF Internet Draft, Oct. 2004. draft-zinin-microloop-analysis-00.txt.