# Improving Dependability for Internet-scale Services

by

Phillipa Gill

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

# Abstract

Improving Dependability for Internet-scale Services

Phillipa Gill

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2012

The past 20 years have seen the Internet evolve from a network connecting academics, to a critical part of our daily lives. The Internet now supports extremely popular services, such as online social networks and user generated content, in addition to critical services such as electronic medical records and power grid control. With so many users depending on the Internet, ensuring that data is delivered dependably is paramount. However, dependability of the Internet is threatened by the dual challenges of ensuring (1) data in transit cannot be intercepted or dropped by a malicious entity and (2) services are not impacted by unreliability of network components. This thesis takes an end-to-end approach and addresses these challenges at both the core and edge of the network. We make the following two contributions:

1. **A strategy for securing the Internet's routing system.** First, we consider the challenge of improving security of interdomain routing. In the core of the network, a key challenge is enticing multiple competing organizations to agree on, and adopt, new protocols. To address this challenge we present our three-step strategy that creates economic incentives for deploying a secure routing protocol (S*BGP). The cornerstone of our strategy is S*BGP's impact on network traffic, which we harness to drive revenue-generating traffic toward ISPs that deploy S*BGP, thus creating incentives for deployment.

2. **Empirical study of data center network reliability.** Second, we consider the dual challenge of improving network reliability in data centers hosting popular content. The scale at which

these networks are deployed presents challenges to building reliable networks, however, since they are administered by a single organization, they also provide opportunity to innovate. We take a first step towards designing a more reliable network infrastructure by characterizing failures in a data center network comprised of tens of data centers and thousands of devices.

Through dialogue with relevant stakeholders on the Internet (*e.g.,* standardization bodies and large content providers), these contributions have resulted in real world impact. This impact has included the creation of an FCC working group, and improved root cause analysis in a large content provider network.

# Acknowledgements

First, I would like to thank my advisors, Yashar Ganjali and David Lie for their support during my PhD and for giving me the freedom to chart my own course during my degree. They have been excellent sounding boards for a wide variety of topics. This dissertation also owes a lot to my collaboration with Sharon Goldberg. Her tenacity in approaching research problems, and tireless editing of drafts and talks has helped to improve my research skills.

I would also like to thank my committee members, Angela Demke Brown and Nick Feamster for their feedback on the work presented in this thesis. This thesis has also benefited from collaboration with Michael Schapira, Navendu Jain, and Nachi Nagappan. I also thank Bianca Schroeder for advising me in the first year of my PhD and teaching me the finer points of data acquisition and analysis.

I have been fortunate to collaborate with many researchers over the years on a variety of projects: Martin Arlitt, Niklas Carlsson, Lee Humphreys, Balachander Krishnamurthy, Anirban Mahanti, Carey Williamson, and Walter Willinger. I especially thank Anirban for starting me on my path in networking research, and Martin for getting me interested in network measurement. I thank Bala for mentoring on a variety of topics in online social networks and privacy over the past few years.

Although we have yet to collaborate, I thank Jennifer Rexford for the time she spent giving advice on various aspects of job hunting and research in the final year of my degree. The time she spent giving feedback on my job talk especially helped me to think more critically about how I choose research problems.

I thank my lab-mates in the Systems and Networking group for providing much-needed distraction from my work: Peter Feiner, Dan Fryer, Monia Ghobadi, Andy Hwang, Eric Logan, Ioan Stefanovici, and Amin Tootoonchian.

Finally, I thank my parents for teaching me the value of hard work that has served me well during my research career.

# Bibliographical Notes

Chapter 2 presents work done in collaboration with Michael Schapira and Sharon Goldberg. This work was published in SIGCOMM'11 [41] and was further expanded upon in Computer Communications Review [43]. This thesis focuses on the simulation and empirical methodology for evaluating our strategy for deploying S*BGP. Additional theorems and proofs can be found in the full version of [41], which appears as a Boston University Technical Report [42].

Chapter 3 presents work done in collaboration with Navendu Jain and Nachiappan Nagappan. This work was published in SIGCOMM'11 [40] and was funded by a Microsoft Research internship.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Since its beginnings as an internetwork connecting academics, the Internet has evolved into a critical resource supporting many aspects of our daily lives. A key driving force of this shift was the development of the Web, which allowed users to access content via a user-friendly browser interface [72]. By making the Internet easy-to-use, the Web drove many existing services such as news, banking and commerce to develop and maintain presence online. Widespread broadband deployment in the early 2000s [92] further increased the popularity of the Internet by enabling new classes of services, such as online social networks and sites hosting user generated content such as YouTube. We refer to this class of extremely popular services as "Internet-scale services."

Reliance on the Internet has grown to extreme levels, with half of Americans going online to access their daily news [112], and services like Facebook visited by 175 millions users per day [7]. Indeed, the Web's importance as a communication medium has been highlighted by recent uprisings in the Middle East and the corresponding government attempts to restrict Internet access [27]. In addition to relying on the network for communication, the Internet is increasingly being used for critical applications such as electronic medical records and power grid control. With the Internet playing a key role in many aspects of daily life it is critical that applications have an infrastructure they can depend on for security and reliability.

## 1.1 Our goal: A network services can depend on

This thesis centers on the topic of improving the dependability of network infrastructure to better support today's popular services and meet the demands of critical applications as they move online. We consider two key challenges to providing a dependable network by answering the following questions:

**Q1: Security.** How do we ensure network traffic will not be intercepted or dropped as a result of malicious agents in the network?

**Q2: Reliability.** How do we design systems that are resilient to unexpected failures?

This section overviews relevant background and challenges for each of these questions.

### 1.1.1 Protecting network traffic from malicious agents in the network

Recent attacks on heavily fortified critical systems highlight the potential for security vulnerabilities to lead to severe real world consequences [32]. However, attackers do not need to spend effort attacking heavily fortified systems in order to wreak havoc. As a result of its beginnings connecting cooperative organizations, much of the core Internet infrastructure was designed assuming entities could be trusted. As a result, key components of the Internet's infrastructure were designed without security in mind and are surprisingly vulnerable to attack.

**The interdomain routing system.** The Internet's interdomain routing system illustrates how assumptions of trust on the Internet have impacted security of the network. The interdomain routing system is comprised of two parts: the data-plane and the control-plane. The data-plane is responsible for routing traffic along paths discovered using Border Gateway Protocol (BGP), a control-plane protocol. BGP allows autonomous systems (ASes) to discover paths in a distributed way by exchanging messages with each other. In a BGP message an AS sends its chosen path to a given destination, denoted by an IP prefix, to its neighbor. BGP messages are an agreement to transit traffic along the advertised path. As a result, depending on business relationships ASes may only advertise a path to a subset of their neighbors, referred to as an

*export policy*. When an AS receives multiple advertisements for the same destination prefix they will choose between them based on their relationship with the neighboring AS as well as the path length, referred to as a *routing policy*. Since ASes are managed by independent organizations they may configure these policies differently (*e.g.,* to prefer a neighbor that provides them with cheaper transit). Since we do not know the routing policies of every AS on the Internet, we use a model of routing policies proposed in prior work [36] and further described in Appendix A.

**Insecurity of BGP.** When we interact with services online, BGP plays a key role in making sure our traffic reaches its intended destination. Unfortunately, BGP lacks mechanisms for validating paths advertised in BGP messages; thus leaving it extremely vulnerable to attack. This vulnerability has been illustrated by a variety of incidents over the past 15 years. In 2008, Pakistan Telecom accidentally hijacked traffic destined to YouTube [102] causing the service to be unavailable until the problem was rectified. More recently, an incident where China Telecom potentially intercepted traffic to tens of thousands of prefixes highlights the potential for surveillance of sensitive data over the network [26].

**S*BGP: problem solved?** BGP's vulnerabilities have been well known for many years and multiple protocols have been proposed to improve its security by validating network paths *e.g.,* S-BGP [67] and soBGP [120][1]. However, deployment of S*BGP has been at an impasse for more than a decade. Recent standardization efforts by the IETF [55] and regional Internet registries [6, 103] show potential to clear this logjam by standardizing S*BGP and deploying the cryptographic root of trust required by S*BGP. However, even if these efforts succeed there is still a question about whether ASes will have any incentive to deploy S*BGP in practice. Indeed, it is easy to be pessimistic here given previous experiences with IPv6. Worse yet, the benefits of S*BGP do not materialize until multiple networks have deployed the protocol so there is no incentive for early adopters!

Ensuring that services are not impacted by the security vulnerabilities of BGP requires that

[1] Collectively we refer to these protocols as S*BGP.

we tackle this problem of deployment and work to understand how we can create incentives for ASes to deploy S*BGP in practice.

### 1.1.2 Designing systems to be resilient to failure

While security has been a long-standing challenge to the dependability of the Internet, popular services offered by content providers like Google and Facebook present new challenges to how we build reliable networks. Content providers are increasingly deploying large-scale data centers to provide dynamic scaling and harness economies of scale. While data centers offer many desirable scaling properties, their scale is also their downfall. Large-scale data center networks present many network management challenges and are also more failure-prone, as even a small component failure rate will manifest as numerous failures. Thus, it is critical that these networks are designed to prevent configuration errors and to be resilient to component failures when they occur. Indeed, numerous recent incidents [16, 25] illustrate how severe the impact of data center outages can be. This impact is especially high with data centers hosting multiple applications, all of which are at risk when failures occur. The high user-perceived impact of data center failures also translates into high economic impact, with a recent article estimating an average cost of half a million dollars per data center outage [51].

**Data center networks.** Data centers commonly have a hierarchical structure with rack-mounted servers connected to Top of Rack (ToR) switches, which connect to an aggregation layer and network core above that (more detail given in Section 3.2). To deliver services to geographically distributed users, data center networks span facilities in multiple physical locations which presents challenges to traffic engineering and network management. These networks also need to support flows with conflicting demands: long-lived, throughput-sensitive "elephant" flows and short-lived, latency-sensitive "mice" flows.

**Designing for scale.** Deploying large-scale data centers presents many challenges that are actively being tackled in the research community. First, the hierarchical structure of data center networks creates bottlenecks as traffic traverses higher layers of the topology. As a result, many

applications are designed to use resources local to their rack when possible [63]. However, the desire for agile cloud applications (*e.g.,* virtual machine migrations) demand a less constrained network topology. This has led to a variety of proposals considering how to increase the bisection bandwidth of data center networks by redesigning the topology [46, 50, 88, 111]. In addition to increasing bisection bandwidth, these proposals also address the challenge of addressing and routing to large numbers of servers in their topologies. Traffic engineering to meet the requirements of flows with varied requirements has also led to a rich body of work in network management [70, 84, 110, 127].

**Designing for reliability?** While there has been a large body of work on improving the scale and performance of data center networks, the issue of reliability has remained largely unaddressed. Data center networks must be highly reliable to allow applications to depend on the network for the high performance and scalability promised by new network designs. Studying network reliability enables us to prevent failures by designing network architectures that are robust to component failure and easier to manage. Further, we can mitigate failures in today's data center networks by understanding and preventing their root causes, as well as studying and improving the effectiveness of redundancy built into these networks.

Designing more reliable data center networks requires we first understand the failure properties of today's data centers. To date, there has been little study of failures in data center networks. This thesis takes an important first step towards improving reliability by performing an empirical analysis of data center network reliability.

## 1.2 The challenges of effecting change on the Internet

The goal of this thesis is to improve the dependability of Internet-scale services. However, achieving our goal is complicated by the nature of the Internet, with many organizations placing constraints on the design of new systems. On one hand, you have commercial organizations (*e.g.,* Internet service providers (ISPs) and content providers) who are reluctant to deploy new

systems that increase management overheads or require the purchase of additional hardware. On the other hand, you have standardization bodies (*e.g.,* Internet Engineering Task Force (IETF)) that facilitate the specification of new protocols and government agencies (*e.g.,* Federal Communications Commission (FCC)) that must balance public and commercial interests, while facilitating the deployment of new systems and protocols.

Improving the dependability of systems and protocols on the Internet requires taking a practical approach. Thus, this thesis uses network measurement as a foundation to (1) study the magnitude of economic incentives for deploying S*BGP and (2) understand reliability problems in existing networks. This understanding of existing systems is critical to inform the design of new systems and protocols. Additionally, this thesis benefits from dialogue with various stakeholders to understand how the research addresses problems faced in practice.

We tackle the problem of network dependability by taking an end-to-end approach, addressing issues at both the core and edge of the network. Each setting presents a unique set of challenges which we summarize in the following section.

### 1.2.1 The network core: A problem of deployment

Effecting change in the core of the Internet is a notoriously challenging problem. Many years have been spent addressing challenges faced in the core of the network such as increasing the number of available IP addresses using IPv6 [28], addressing consistency issues with BGP convergence [59] and mitigating the security flaws of BGP [67, 120] but deployment for these solutions has been extremely slow, if there has been any deployment at all! Clearly, deployment of new protocols and innovations in the network core is an extremely challenging problem. First, the challenge of standardization must be overcome and agreement must be reached between the competing interests of organizations on the Internet. However, once a protocol is standardized, gaining widespread deployment is another challenge that must be overcome.

The challenge of deployment has plagued innovations in many diverse settings (*e.g.,* farming techniques, technology such as laptops [104]) and has been well studied using models

of innovation diffusion [56, 66, 87, 104, 116, 126]. It is tempting to apply these models to the problem of deploying protocols on the Internet, however, deployment of Internet protocols differs from other innovations. Unlike traditional models of innovation diffusion, where individuals deploy based on their immediate neighbors, an AS's decision to deploy a protocol may be impacted by ASes multiple hops away *e.g.,* S*BGP requires all ASes on a path deploy before security benefits are observed. This additional complexity renders existing innovation diffusion work inapplicable to the Internet domain and raises many interesting challenges which we address in Chapter 2.

### 1.2.2 The network edge: A problem of scale

Internet-scale services present many challenges to edge networks. Addressing these challenges requires taking an empirical approach to characterize the resource requirements of new services and how large scale deployments behave in practice (*e.g.,* [12, 37, 38, 63]). By first understanding requirements and properties of the systems we aim to improve, we can ensure our solutions meet these requirements and address problems that manifest in practice. In previous work, we consider challenges faced by a campus edge network when supporting user generated content on YouTube [37, 38]. This thesis considers scaling challenges at the other edge of the network: data centers used to deliver content. Chapter 3 takes an empirical approach to tackle the problem of designing more reliable data center networks by first understanding the reliability of existing data center networks.

Even though they face their own set of challenges, edge networks have a distinct advantage in terms of deploying innovations. Since these networks are controlled by a single organization, there is potential to innovate in ways that were previously inconceivable on the Internet as a whole. Indeed, this advantage has been recognized by the research community with many recent proposals to modify end hosts to improve congestion control [5], design new network architectures [46, 50, 88, 111] and even completely reimagine network management [70, 84, 110, 127].

## 1.3 Contribution and impact

This thesis tackles the problem of improving network dependability in terms of security and reliability. Since we address these issues at both the core and edge of the network, we must contend with different sets of challenges to arrive at actionable solutions. This section summarizes the contributions and impact of this thesis.

### 1.3.1 A strategy for transitioning to BGP security

Chapter 2 considers the challenge of deploying secure routing on the Internet. Any new protocol on the Internet will necessarily undergo a transition phase where only a fraction of ASes have deployed the protocol. We consider this phase in the deployment of S*BGP and study incentives for ASes to deploy S*BGP. Indeed, since S*BGP requires all ASes on the path to deploy the protocol before security benefits are realized, it appears that there will be little incentive for ASes to deploy the protocol. We challenge this pessimistic view by presenting a three-step strategy for deploying S*BGP. The cornerstone of our strategy is S*BGP's impact on routing decisions which we harness to drive revenue-generating traffic toward ISPs deploying S*BGP, thus creating incentives for deployment. Additionally, our strategy leverages (1) *early adopter* ASes that deploy the protocol initially (*e.g.,* because of government incentives or concern for security) and (2) a simplex (unidirectional) version of the protocol to facilitate deployment by ASes that do not generate revenue by transiting traffic. These "stub" networks will never have incentive to deploy S*BGP in our model, thus simplex S*BGP was crucial to secure the vast majority of destinations on the Internet.

We evaluate our strategy by developing a model of the deployment process inspired by related work on innovation diffusion [66]. We then simulate this model on an empirically derived AS-graph. We consider two key questions about deployment using a combination of analysis and simulations on an empirically derived graph of the Internet's interdomain topology:

1. **How many networks will deploy S*BGP?** For each set of early adopters we run simulations

on the Internet graph to observe how much traffic shifts towards networks that deploy S*BGP, thus creating incentives for deployment. We characterize the effectiveness of our strategy in terms of how many ASes eventually transition to S*BGP. Indeed, we observe that when deployment costs of S*BGP are low (*e.g.,* 5% of transit traffic revenue) a set of 5-10 early adopters is enough to drive deployment at 85% of the ASes on the Internet!

2. **Which networks should be early adopters?** Since early adopters may be brought on board via government incentives it is important to understand which ASes should be targeted. However, it is NP-hard to choose the set of early adopters that will maximize spread of the protocol [41]. Since we cannot choose the optimal set of early adopters, we run simulations on the Internet graph to understand which types of networks are the most effective early adopters. Our simulations indicate that high-degree Tier 1 networks are most effective.

**Impact.** Chapter 2 benefited from discussion with the standards bodies working on BGP security. We further engaged the network operator community by presenting our work at the meeting of the North American Network Operators Group (NANOG). By engaging these groups, we positioned our study of BGP security within the realities of today's standardization process. Our candidate deployment strategy and economic incentives it presents, have spurred the creation of a working group within the FCC to develop processes for incremental deployment of BGP security solutions [53].

### 1.3.2 Empirical study of failures in a data center network

Chapter 3 considers the challenge of improving reliability of data center networks. We study data center network reliability by analyzing a year's worth of network error logs collected from thousands of network devices across tens of geographically distributed data centers. We characterize network failure patterns and overall reliability of a data center network. Second, we leverages lessons learned through characterization to guide the design of future data center networks with improved reliability.

We take an operator-centric approach in this work and consider the following three questions motivated by issues encountered by our network operators:

1. **What are the properties of failures in the network?** Given limited resources for debugging and trouble shooting data center networks, it is important to identify the most failure prone components so that steps can be taken to reduce their impact. Indeed, we observe that load balancer middle boxes are a particularly failure prone component in the data center network with 1 in 5 load balancers experiencing at least one failure over the year. Worse yet, these failures are highly variable, with some load balancers experiencing up to 400 failures per year!

2. **How do failures impact network traffic?** After characterizing the properties of network failures, we study how these failures impact network traffic. We accomplish this by correlating link failures with network traffic on failed links to understand how link failures impact network traffic in terms of byte and packet volumes. We find that most failures lose a large number of packets relative to the number of lost bytes, likely due to loss of protocol-specific keep alive messages or ACKs.

3. **How effective is redundancy in the network at masking failure impact?** Ideally, failures should be completely masked from applications, via redundant links and devices in the network. Indeed, our network employs many redundant devices, incurring cost to purchase and maintain these additional devices. We find that this redundancy is partially effective and is able to reduce the median impact of failure (in terms of lost bytes or packets) by up to 40%.

**Impact.** The work in Chapter 3 was done in collaboration with a large content provider. As a result, we had a unique perspective on the challenges of managing an extremely large-scale network. By interacting closely with the organization's network operators, we ensured that our results were relevant to their concerns. Namely, we focused on helping them improve efficiency by identifying the most failure-prone components and failures that have the most

impact. Subsequent to our study, our results have informed the development of a tool that helps operators understand failures in real time.

### 1.3.3 Methodological contributions

Our study of S*BGP deployment and data center network reliability required developing new methodologies for simulating and analyzing empirical network data.

**Pushing the boundaries of BGP simulation.** Evaluating our model of S*BGP deployment required us to develop a simulation framework of the deployment process that pushed the boundaries of existing BGP simulations. We developed custom algorithms and data structures, and employ distributed computing [128] to achieve a $1,000x$ constant speedup that made our simulations feasible (Section 2.5).

**Correlating data sets to find failures with impact.** The size of the network we study in Chapter 3 presented a unique opportunity to study network reliability at scale, but posed challenges to characterization. Specifically, the logs we employ are used by the operators to troubleshoot the network on a day-to-day basis and are not meant for statistical analysis of failures. As a consequence, we develop a methodology to deal with inconsistencies in the data (*e.g.,* multiple failures on the same link simultaneously). Additionally, we take the extra step of correlating network failures with logs of network traffic to understand the impact of these failures on network traffic. Section 3.3 summarizes these techniques.

### 1.3.4 Thesis outline

The contributions of this thesis are presented in Chapter 2 and Chapter 3, respectively. Each chapter in this thesis presents a stand-alone overview of the contribution with motivation, background, results and conclusion. Chapter 2 presents our strategy for deploying S*BGP. Our empirical analysis of data center network failures are presented in Chapter 3. Finally, Chapter 4 presents conclusions and directions for future work.

# Chapter 2

# A Strategy for Transitioning to BGP Security

## 2.1 Introduction

The Border Gateway Protocol (BGP), which sets up routes from autonomous systems (ASes) to destinations on the Internet, is amazingly vulnerable to attack [17]. Every few years, a new failure makes the news; ranging from misconfigurations that cause an AS to become unreachable [86, 102], to possible attempts at traffic interception [26]. To remedy this, a number of stop-gap measures have been developed to *detect* attacks [64, 76]. The next step is to harden the system to a point where attacks can be *prevented*. After many years of effort, we are finally seeing the initial deployment of the Resource Public Key Infrastructure (RPKI) [6, 9, 83, 103], a cryptographic root-of-trust for Internet routing that authoritatively maps ASes to their IP prefixes and public keys. With RPKI on the horizon, we can now realistically consider deploying the S*BGP protocols, proposed a decade ago, to prevent routing failures by validating AS-level paths: Secure BGP (S-BGP) [67] and Secure Origin BGP (soBGP) [120].

### 2.1.1 Economic benefits of S*BGP adoption

While governments and industry groups may have an interest in S*BGP deployment, ultimately, the Internet lacks a centralized authority that can mandate the deployment of a new secure routing protocol. Thus, a key hurdle for the transition to S*BGP stems from the fact that each AS will make deployment decisions according to its own local business objectives.

**Lessons from IPv6?** Indeed, we have seen this problem before. While IPv6 has been ready for deployment since around 1998, the lack of tangible local incentive for IPv6 deployment means that we are only now starting to see the seeds of large-scale adoption. Conventional wisdom suggests that S*BGP will suffer from a similar lack of local incentives for deployment. The problem is exacerbated by the fact that an AS cannot validate the correctness of an AS-level path unless all the ASes on the path deployed S*BGP. Thus, the security benefits of S*BGP only apply after a large fraction of ASes have already deployed the protocol.

**Economic incentives for adoption.** We observe that, unlike IPv6, S*BGP can impact routing of Internet traffic, and that this may be used to drive S*BGP deployment. These crucial observations enable us to avoid the above issues and show that global S*BGP deployment is possible even if local ASes' deployment decisions are *not* motivated by security concerns! To this end, we present a prescriptive strategy for S*BGP deployment that relies solely on Internet Service Providers' (ISPs) local economic incentives to drive global deployment; namely, ISP's interest in attracting revenue-generating traffic to their networks.

Our strategy is prescriptive (Section 2.3). We propose guidelines for how (a) ASes should deploy S*BGP in their networks, and (b) governments, industry groups, and other interested parties should invest their resources in order to drive S*BGP deployment forward.

**1. Break ties in favor of secure paths.** First, we require ASes that deploy S*BGP to actually use it to inform route selection. However, rather than requiring security be the first criterion ASes use to select routes, we only require secure ASes to *break ties* between equally-good routes in favor of secure routes. This way, we create incentives for ISPs to deploy S*BGP so

they can transit more revenue-generating customer traffic than their insecure competitors.

**2. Make it easy for stubs to adopt S*BGP.** 85% of ASes in the Internet are *stubs* (*i.e.,* ASes with no customers) [21]. Because stubs earn no revenue from providing Internet service, we argue for driving down their deployment costs by having ISPs sign BGP announcements on their behalf or deploy a simplex (unidirectional) S*BGP [78] on their stub customers. In practice, such a simplex S*BGP must either be extremely lightweight or heavily subsidized.

**3. Create market pressure via early adopters.** We propose that governments and industry groups concentrate their regulatory efforts, or financial incentives, on convincing a small set of *early adopters* to deploy S*BGP. We show that this set of early adopters can create sufficient market pressure to convince a large fraction of ASes to follow suit.

We note that while our focus here is on S*BGP, our strategy may be employed to drive deployment of other network protocols. Specifically, any protocol that requires full path deployment that can be incorporated into the routing decision process may benefit from our strategy. This may include protocols such as IPv6 or proposals such as consensus routing [59].

### 2.1.2 Evaluation: Model and simulations

To evaluate our proposal, we needed a model of the S*BGP deployment process.

**Inspiration from social networks?** At first glance, it seems that the literature on technology adoption in social networks would be applicable here [56, 66, 87, 104, 116, 126]. However, in social networks models, an entity's decision to adopt a technology depends only on its immediate *neighbors* in the graph; in our setting, this depends on the number of secure *paths*. This complication means that many elegant results from this literature have no analogues in our setting (Section 2.10).

**Our model.** In contrast to earlier work that assumes that ASes deploy S*BGP because they are concerned about security [10, 19], our model assumes that ISPs' local deployment decisions are based solely on their interest in increasing customer traffic (Section 2.4).

We carefully designed our model to capture a few crucial issues, including the fact that (a) traffic transited by an ISP can include flows from any pair of source and destination ASes, (b) a large fraction of Internet traffic originates in a few large content provider ASes [73, 75], and (c) the cost of S*BGP deployment can depend on the size of the ISP's network. The vast array of parameters and empirical data relevant to such a model (Section 2.9) mean that our analysis is *not* meant to *predict* exactly how the S*BGP deployment process will proceed in practice; instead, our goal was to evaluate the efficacy of our S*BGP deployment strategy.

**Theorems, simulations and examples.** We explore S*BGP deployment in our model using a combination of theoretical analysis and simulations on empirical AS-level graphs [8, 21] (Sections 2.6-2.8). Every example we present comes directly from these simulations. Instead of artificially reducing algorithmic complexity by subsampling [71], we ran our simulations over the full AS graph (Section 2.5). Thus, our simulations ran in time $O(N^3)$ with $N = 36K$, and we devoted significant effort to developing parallel algorithms that we ran on a 200-node DryadLINQ cluster [128].

### 2.1.3 Key insights and recommendations

Our evaluation indicates that our strategy for S*BGP deployment can drive a transition to S*BGP (Section 2.6). While we cannot predict exactly how S*BGP deployment will progress, a number of important themes emerge:

**1. Market pressure can drive deployment.** We found that when S*BGP deployment costs are low, the vast majority of ISPs have incentives to deploy S*BGP in order to differentiate themselves from, or keep up with, their competitors (Section 2.6). Moreover, our results show this holds even if 96% of routing decisions (across all source-destination AS pairs) are *not* influenced by security concerns (Section 2.7.6).

**2. Simplex S*BGP is crucial.** When deployment costs are high, deployment is primarily driven by simplex S*BGP (Section 2.7).

**3. Choose a few well-connected early adopters.** The set of early adopters cannot be random; it should include well-connected ASes like the Tier 1's and content providers (Section 2.7). While it is NP-hard to even *approximate* the *optimal* set of early adopters [41], our results show that even 5-10 early adopters suffice when deployment costs are low.

**4. Prepare for incentives to disable S*BGP.** We demonstrate that ISPs can have incentives to *disable* S*BGP (Section 2.8). Moreover, it has been shown that there could be deployment oscillations (where ASes endlessly turn S*BGP on and off), and it is computationally hard to even *determine* whether such oscillations exist [41].

**5. Minimize attacks during partial deployment.** Even when S*BGP deployment progressed, there were always some ASes that did not deploy (Section 2.6, 2.7). As such, we expect that S*BGP and BGP will coexist in the long term, suggesting that careful engineering is required to ensure that this does not introduce new vulnerabilities into the interdomain routing system.

**Chapter organization.** Background on BGP security solutions is presented in Section 2.2. Section 2.3 presents our proposed strategy for S*BGP deployment. To evaluate the proposal, we present a model of the deployment process in Section 2.4 and our simulation framework in Section 2.5. In Section 2.6-2.8 we explore the model using theoretical analysis and simulations, and present an in-depth discussion of our modeling assumptions in Section 2.9. Section 2.10 presents related work.

## 2.2 Background

BGP's vulnerabilities have been well known and much effort has gone into designing protocols to detect [45, 64, 76, 99, 113, 123, 130] and prevent attacks[1] [54, 67, 118, 120] via path validation. Path validation incurs overheads in terms of cryptographic operations, increased BGP message sizes and management challenges (*e.g.,* public key infrastructure). To mitigate these overheads, there have been many proposals to reduce the number of cryptographic oper-

[1] "Attacks" may also be the result of misconfigurations.

ations [18, 54, 91, 118, 131], reduce message sizes [131, 132] and mitigate management overheads [118]. In this work, we focus on deployment of path validating protocols, but note that there has also been recent interest in ensuring that routing policies and traffic flows comply with stated routing policies [89, 133].

**Which part of the system to secure?** Existing proposals tackle the problem of routing security from two main angles. The first approach is to validate at the control plane *e.g.,* routes advertised in BGP [54, 67, 113, 118, 120] or paths visible by observing BGP data from multiple public vantage points [76, 99]. Securing the control plane ensures that chosen paths are indeed valid, however, it does not guarantee that traffic will actually flow along these paths. Thus, some proposals also employ data plane probing to observe routing anomalies [113, 130]. By monitoring the data plane these proposals can gain visibility into the path taken by traffic. This thesis focuses on the problem of deploying control plane protocols that validate BGP routes. Indeed, deploying data plane security mechanisms is even more challenging, since these protocols need to run on traffic in real time and the overheads/upgrade cost can be significant.

### 2.2.1 Detecting routing misconfigurations and attacks

There has been a large body of work considering how to detect when attacks on routing occur [45, 64, 76, 99, 113, 123, 130]. Attacks may be detected upon receipt of a routing message using heuristics to detect invalid routes [45, 64, 113] or by monitoring routing data to detect when an attack has occured [76, 123, 130], and locate the hijacking AS [99].

**Detecting anomalous routes.** There have been many proposals to help ASes identify when routing messages contain incorrect information [45, 64, 113]. One approach is to simply maintain a database of routing information and have ASes check this database in order to validate routes they receive. This approach has been attempted in the past using the Internet Routing Registry (IRR) [90], however, a lack of incentives for ASes to update their information, and privacy concerns about revealing connectivity led to the registry becoming out of date. Interdomain Route Validation (IRV) aims to address these issues by allowing ASes to maintain a

database of their connectivity and routing policies which may be queried by ASes that want to validate their routing messages [45]. This creates incentives for ASes to participate in IRV *e.g.,* to provide it as a value added service, and mitigates privacy concerns of storing potentially sensitive routing information in a public repository.

Instead of relying on an external data source, Listen and Whisper [113] and Pretty Good BGP (pgBGP) [64] provide mechanisms for ASes to independently validate the routes they receive. Listen and Whisper has the origin of a routing message insert a secret into its routing message. Thus, an AS can detect routing inconsistencies by verifying that the secret matches on multiple routes to the same origin. Pretty Good BGP takes a more pragmatic approach and monitors routes over time [64]. New routes for the same origin are treated as "suspicious" and not used until they have been announced for a sufficient period of time (*e.g.,* 1 day).

**Detecting and attributing attacks after they occur.** Alternate approaches monitor the state of the routing system, either using control plane [76, 99, 123] or data plane [99, 130] monitoring. Prefix hijack alert system (PHAS) takes a similar approach to pgBGP by monitoring the set of ASes that originate a prefix [76]. However, PHAS builds a centralized system for monitoring origin ASes and allows organizations to observe changes in origin for their prefixes. Presumably, the organization knows which ASes *should* be announcing the prefixes and can identify misconfigurations/attacks when they occur.

iSPY [130] also gives organizations the ability to identify attacks, but does so using data plane probing. The intuition behind iSPY is that when an organization's prefix is hijacked, traceroute probes will fail once they reach ASes that have accepted the incorrect routing information. iSPY allows organizations to detect attacks by continuously monitoring data plane connectivity (using traceroute). By studying the extent of connectivity changes, iSPY is able to distinguish attacks from transient connectivity failures. In contrast to iSPY, which focuses on detecting attacks, LOCK focuses on locating the attacker [99]. This is accomplished by probing AS paths to the victim prefix and observing ASes on, or adjacent to, these paths to find the most likely attacker AS.

### 2.2.2 Preventing attacks by validating paths

With RPKI providing an authoritative mapping from ASes to their cryptographic public keys, there are two main protocols that have been proposed that prevent the propagation of bogus AS path information:

**Secure BGP (S-BGP) [67].** S-BGP provides *path validation*, allowing an AS $a_1$ that receives a BGP announcement $a_1a_2...a_kd$ to validate that every AS $a_j$ actually sent the announcement in the path. With S-BGP, a router must cryptographically sign each routing message it sends, and cryptographically verify each routing message it receives. Further, S-BGP requires a secure mapping between ASes and the prefixes they originate to authenticate the origin of routing announcements. This functionality will most likely be implemented via route origin attestations (ROAs) stored in the RPKI [77].

**Secure Origin BGP (soBGP) [120].** soBGP provides a slightly weaker security guarantee called *topology validation*, that allows an AS to validate that a path it learns physically exists in the network. To do this, soBGP requires neighboring ASes to mutually authenticate a certificate for the existence of a link between them, and validate every path it learns from a BGP announcement against these cryptographic certificates.

Because our study is indifferent to attacks and adversaries, it applies equally to each of these protocols. We refer to them collectively as S*BGP, and an AS that deploys them as *secure*.

### 2.2.3 Reducing the complexity of path validation

The security guarantees provided by S-BGP [67] make it an appealing proposition for securing the Internet's routing system. However, S-BGP incurs overheads in terms of expensive cryptographic operations, increased message sizes and management overheads. We now overview proposals to mitigate these challenges in turn.

**Reducing cryptographic overheads.** Many proposals aim to reduce the number of cryptographic operations that must be completed by BGP speakers [18, 91, 118, 131]. Signature

amortization enables ASes to sign each route once, rather than for each neighboring AS. This is accomplished by also signing a bit vector specifying the intended recipients of the message. This reduction in complexity requires a trade off in terms of routing policy privacy which may be undesirable for some organizations. Additionally, Merkle hash trees are used to allow the AS to compute even fewer signatures with a trade off in terms of message size. Signature amortization is applied by Zhao *et al.* [131] in combination with aggregated signatures to reduce message size overhead. Pretty Secure BGP [118] combines signature amortization with a distributed trust model for origin authentication.

Butler *et al.* [18] also work to reduce the number of signatures ASes must compute. Their approach requires the AS to send the set of all paths that may be offered to the neighbor in a single signed message. The AS then uses a chained hash function to signal to their neighbor which path is currently in use. This has the advantage of reducing the complexity of failover when an AS needs to switch to a back up path, and does not require revealing policy information to multiple neighbors. However, ASes still need to reveal more of their routing information to neighbors than they currently do today *e.g.,* the availability of back up links that may only be used as a last resort.

Finally, Hu *et al.* [54] take a different approach altogether, proposing Secure Path Vector routing, which leverages symmetric key cryptography. The authors develop a chained signature scheme using hash chains, hash trees, and one time signatures, to prevent an attacker from removing or changing ASes in the network path.

**Reducing message sizes.** S-BGP requires that each AS on the network path sign the BGP message before propagating it to its neighbors. Naively including all signatures results in message size increasing linearly with path length. This problem has been recognized in the cryptography community where aggregated signatures [15, 79] have been proposed. Aggregate signatures allow $n$ signers to sign $n$ messages and combine the $n$ signatures into a single shorter signature. Zhao *et al.* [131, 132] apply aggregate signatures to reduce the size of routing messages in S-BGP.

**Reducing management overheads.** S-BGP requires a certified mapping between ASes and their IP address allocations to authenticate the origin of routing messages. However, building such a mapping is non-trivial and indeed has been attempted before by the Internet Routing Registry [90] which is widely known to be out of date. Wan *et al.* circumvent this challenge by using a decentralized mechanism to certify the prefixes allocated to ASes [118]. This is accomplished by having networks attest to the prefixes they own as well as the prefixes owned by their peers. This scheme has the advantage of lower complexity but is subject to attack by colluding ASes. Attacks can be mitigated by requiring $k$ neighbors attest to prefixes owned by an AS. Increasing $k$ improves security but increases the size of the list of prefixes that must be checked when validating that an AS is authorized to announce a prefix.

### 2.2.4 Verifying and enforcing routing policy

More recently, there has been interest in not only preventing routing attacks, but validating routing policy. ICING accomplishes this by performing highly optimized cryptographic operations on network packets in order to verify that: (1) each node on the path has consented to transit the traffic and (2) traffic has been handled by the prior hops on the path [89]. Path validating protocols are incredibly difficult to deploy, because they require algorithms that are efficient enough to execute on network traffic in real time. Naous *et al.* demonstrate the feasibility of ICING through implementation and evaluation. They find that ICING would cost 93% more than an IP router on the same platform, incurring a significant, but affordable cost.

Similar to S*BGP, SPIDeR focuses on securing the control plane [133]. However, instead of validating network paths, SPIDeR allows ASes to verify that their neighboring ASes select and propagate routes according to agreed upon policy (*e.g.,* to prefer certain routes over others). This is accomplished by having ASes exchange signed commitments with their neighbors. Neighboring ASes then collaborate, and compare commitments to verify that routes are selected according to agreed upon policy. Commitments are structured so that ASes do not learn more about routing policy than they would with BGP. Unsurprisingly, the guarantees of

SPIDeR do not come without overhead, with control plane bandwidth increasing by 176%. It also does not give the end-to-end path validation of proposals like S*BGP. However, the locality of SPIDeR's guarantees mean that even a small group of ASes deploying can derive benefits from the system. Thus, while a strategy similar to the one we propose can benefit SPIDeR, there may be other methods for driving small local deployments.

## 2.3 S*BGP deployment strategy

### 2.3.1 How to standardize S*BGP deployment

To create local economic incentives for ISPs to deploy S*BGP, we propose that Internet standards should require ASes to deploy S*BGP as follows:

#### Simplex S*BGP for stubs

For stubs, Internet access is a cost, rather than a revenue source, and it seems unlikely that security concerns alone will suffice to motivate stubs to undertake a costly S*BGP deployment. However, because stubs propagate only *outgoing* BGP announcements for *their own IP prefixes* we suggest two possible solutions to this problem: (1) allow ISPs to sign on behalf of their stub customers or (2) allow stubs to deploy simplex (unidirectional) S*BGP. Indeed, the latter approach has been proposed by the Internet standards community [78].

**Simplex S-BGP.** For S-BGP, this means that stubs need only sign outgoing BGP announcements for their own IP prefixes, but not validate incoming BGP announcements for other IP prefixes[2]. Thus, a stub need only store its own public key (rather than obtaining the public keys of each AS on the Internet from the RPKI) and cryptographically sign only a tiny fraction of the BGP announcements it sees. Simplex S-BGP can significantly decrease the computational

[2] A stub may even choose to delegate its cryptographic keys to its ISPs, and have them sign for him; while this might be a good first step on the path to deployment, ceding control of cryptographic keys comes at the cost of reduced security.

load on the stub, and can potentially be deployed as a software, rather than hardware, upgrade to its routers.

**Simplex soBGP.** For soBGP, this means that a stub need only create certificates for its links, but need not need validate the routing announcements it sees. Simplex soBGP is done offline; once a stub certifies his information in the soBGP database, its task is complete and no router upgrade is required.

The objective of simplex S*BGP is to make it easy for stubs to become secure by lowering deployment costs and computational overhead. While we certainly allows for stubs (*e.g.,* banks, universities) with an interest in security to move from simplex S*BGP to full S*BGP, our proposal does not require them to do so.

**Impact on security.** With simplex S*BGP, a stub lacks the ability to validate paths for prefixes other than its own. Since stubs constitute about 85% of ASes [21], a first glance suggests that simplex S*BGP leads to significantly worse security in the global Internet.

We argue that this is not so. Observe that if a stub $s$ has an immediate provider $p$ that has deployed S*BGP and is correctly validating paths, then no false announcements of fully secure paths can reach $s$ from that provider, unless $p$ *himself* maliciously (or mistakenly) announces false secure paths to $s$. Thus, in the event that stubs upgrade to simplex S*BGP and all other ASes upgrade to full S*BGP, the only open attack vector is for ISPs to announce false paths to their *own* stub customers. However, we observe the impact of a single misbehaving ISP is small, since 80% of ISPs have less than 7 stub customers, and only about 1% of ISPs have more than 100 stub customers [21]. Compare this to the insecure status quo, where an arbitrary misbehaving AS can impact about half of the ASes in the Internet (around $15K$ ASes) on average [44].

### Break ties in favor of fully secure paths

In BGP, an AS chooses the path to a given destination AS $d$ based on a *ranking* on the outgoing paths it learns from its neighbors (*e.g.,* Appendix A). Paths are first ranked according to inter-

domain considerations (local preference, AS path length) and then according to intradomain considerations (*e.g.,* MEDs, hot-potato routing)[3].

**Secure paths.** We say that a path is secure iff *every* AS on that path is secure. We do this because an AS cannot validate a path unless every AS on the path signed the routing announcement (S-BGP) or issued certificates for the links on the path (soBGP).

**Security as part of route selection.** The next part of our proposal suggests that once an AS has the ability to validate paths, it should actually use this information to inform its routing decisions. In principle, an AS might even modify its ranking on outgoing paths so that security is its highest priority. Fortunately, we need not go to such lengths. Instead, we only require secure ASes to *break ties* between equally good interdomain paths in favor of secure paths. This empowers secure ISPs to attract customer traffic away from their insecure competitors. To ensure that a newly-secure AS can *regain* lost customer traffic, we require that original tie-break criteria (*e.g.,* intradomain considerations) be employed in the case of equally good, *secure* interdomain paths. Thus, the size of the set of equally-good interdomain paths for a given source-destination pair (which we call the *tiebreak set*) gives a measure of competition in the AS graph.

**Route selection at stubs.** For stubs running simplex S*BGP, we consider both the case where they break ties in favor of secure paths (*i.e.,* because they trust their providers to verify paths for them) and the case where they ignore security altogether (*i.e.,* because they do not verify paths) (Section 2.7.7).

**Partially secure paths.** We do not allow ASes to prefer partially-secure paths over insecure paths, to avoid introducing *new* attack vectors that do exist even without S*BGP (*e.g.,* attack in Appendix B).

We shall show that S*BGP deployment progresses quite effectively even if stubs ignore security and tiebreak sets are very small (Section 2.7.7-2.7.6).

[3]For simplicity, we do not model intradomain routing considerations. However, it should be explored in future work.

### 2.3.2 How third parties should drive deployment

**Early adopters.** To kick off the process, we suggest that interested third parties (*e.g.,* governments, regulators, industry groups) focus regulation, subsidies, or external financial incentives on convincing a set of *early adopter* ASes to deploy S*BGP. One regulatory mechanism may be for the government to require their network providers to deploy S*BGP first. In the AS graph ( [8, 21]), providers to the government include many Tier 1 ISPs who may be difficult or expensive to persuade via other means.

**ISPs upgrade their stubs.** Next, we suggest that a secure ISP should be responsible for upgrading all its insecure stub customers to simplex S*BGP. To achieve this, interested third parties should ensure that simplex S*BGP is engineered to be as lightweight as possible, and potentially provide additional subsidies for ISPs that secure their stubs. (ISPs also have a local incentives to secure stubs, *i.e.,* to transit more revenue-generating traffic for multi-homed stubs (Section 2.6.1).)

## 2.4 Modeling S*BGP deployment

We evaluate our proposal using a model of the S*BGP deployment process. For brevity, we now present only the details of our model. Justification for our modeling decisions and possible extensions are in Section 2.9.

### 2.4.1 The Internetwork and entities

**The AS graph.** The interdomain-routing system is modeled with a labeled AS graph $G(V,E)$. Each node $n \in V$ represents an AS, and each edge represents a physical link between ASes. Per Figure 2.1, edges are annotated with the standard model for business relationships in the Internet [36]: *customer-provider* (where the customer pays the provider), and *peer-to-peer* (where two ASes agree to transit each other's traffic at no cost). Each AS $n$ is also assigned

weight $w_n$, to model the volume of traffic that *originates* at each AS. For simplicity, we assume ASes divide their traffic evenly across all destination ASes. However, our results are robust even when this assumption is relaxed (Section 2.7.8).

We distinguish three types of ASes:

**Content providers.** Content providers (CPs) are ASes whose revenue (*e.g.,* advertising) depends on reliably delivering their content to as many users as possible, rather than on providing Internet transit. While a disproportionately large volume of Internet traffic is known to originate at a few CPs, empirical data about Internet traffic volumes remains notoriously elusive. Thus, based on recent research [73, 75, 105] we picked five content providers: Google (AS 15169), Facebook (AS 32934), Microsoft (AS 8075), Akamai (AS 20940), and Limelight (AS 22822). Then, we assigned each CP weight $w_{CP}$, so that the five CPs originate an $x$ fraction of Internet traffic (equally split between them), with the remaining $1-x$ split between the remaining ASes.

**Stubs.** Stubs are ASes that have no customers of their own and are not CPs. Every stub $s$ has unit weight $w_s = 1$. In Figure 2.1, ASes 34376 and 31420 are stubs.

**ISPs.** The remaining ASes in the graph (that are not stubs or CPs) are ISPs. ISPs earn revenue by providing Internet service; because ISPs typically provide transit service, rather that originating traffic (content), we assume they have unit weight $w_n = 1$. In Figure 2.1, ASes 25076, 8866 and 8928 are ISPs.

### 2.4.2 The deployment process

We model S*BGP deployment as an infinite round process. Each round is represented with a state $S$, capturing the set of ASes that have deployed S*BGP.

**Initial state.** Initially, the only ASes that are secure are (1) the ASes in the set of early adopters and (2) the direct customers of the early adopter ISPs that are stubs. (The stubs run simplex S*BGP.) All other ASes are insecure. For example, in Figure 2.1, early adopters ISP

8866 and CP 22822 are secure, and stub 31420 runs simplex S*BGP because its provider is secure.

**Each round.** In each round, *every* ISP chooses an action (deploy S*BGP or not) that improves its utility relative to the current state. We discuss the myopic best-response strategy that ISPs use to choose their actions in Section 2.4.3. Once an ISP becomes secure, it deploys simplex S*BGP at *all* its stub customers (Section 2.3.2). Because CPs do not earn revenues by providing Internet service, some external incentive (*e.g.,* concern for security, subsidies) must motivate them to deploy S*BGP. Thus, in our model, a CP may only deploy S*BGP if it is in the set of early adopters.

Once ASes choose their actions, paths are established from every source AS $i$ to every destination AS $d$, based on the local BGP *routing policies* of each AS and the state $S$ of the AS graph. We use a standard model of BGP routing policies, based on business relationships and path length (see Appendix A). Per Section 2.3.2, we also assume that routing policies of secure ASes require them to break ties by preferring fully secure paths over insecure ones, so that the path to a given destination $d$ depends on the state $S$. Paths to a destination $d$ form a tree rooted at $d$, and we use the notation $T_n(d,S)$ to represent the subtree of ASes routing through AS $n$ to a destination $d$ when the deployment process is in state $S$. Figure 2.1 (right) shows part of the routing tree for destination 22822; notice that $T_{8866}(22822,S)$ contains ASes $31420, 25076, 34376$.

**Termination.** We proceed until we reach a stable state, where no ISP wants to deploy (or disable) S*BGP.

### 2.4.3 ISP utility and best response

We model an ISP's utility as related to the volume of traffic it transits for its customers; this captures the fact that many ISPs either bill their customers directly by volume, or indirectly through flat rates for fixed traffic capacities. Utility is a function of the paths chosen by each AS. Because path selection is a function of routing policies (Appendix A) and the state $S$,

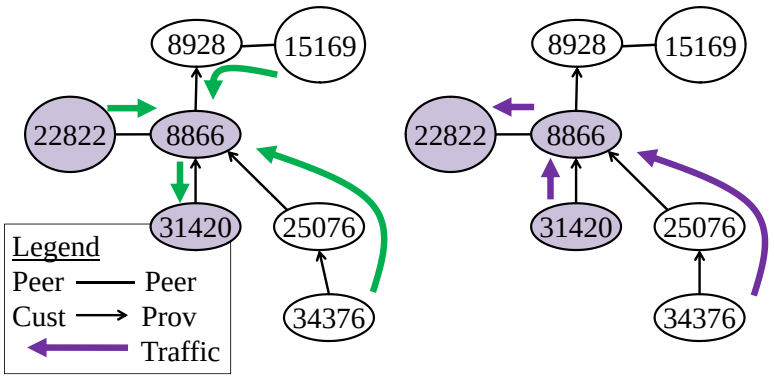



Figure 2.1: Destinations (left) 31420, (right) 22822

it follows that *the utility of each ISP is completely determined by the AS weights, AS graph topology, and the state S.*

We have two models of ISP utility that capture the ways in which an ISP can transit customer traffic:

**Outgoing utility.** ISP $n$ can increase its utility by forwarding traffic *to* its customers. Thus, we define outgoing utility as the amount of traffic that ISP $n$ routes to each destination $d$ via a customer edge. Letting $\hat{D}(n)$ be the set of such destinations, we have:

$$u_n(S) = \sum_{\substack{\text{Destns} \\ d \in \hat{D}(n)}} \sum_{\substack{\text{Sources} \\ i \in T_n(d,S)}} w_i \tag{2.1}$$

Let's use Figure 2.1 to find the outgoing utility of ISP $n = 8866$ due to destinations 31420 and 22822. Destination 31420 is in $\hat{D}(n)$ but destination 22822 is not. Thus, two CPs (Google AS 15169 and Limelight 22822), and 3 other ASes (*i.e.,* AS 8928, 25076, 34376) transit traffic through $n = 8866$ to destination $d = 31420$, contributing a $2w_{CP} + 3$ outgoing utility to $n = 8866$.

**Incoming utility.** An ISP $n$ can increase its utility by forwarding traffic *from* its customers. Thus, we define incoming utility as the amount of traffic that ISP $n$ receives via customer edges for each destination $d$. We restrict the subtree $T_n(d,S)$ to branches that are incident on $n$ via

customer edges to obtain the *customer subtree* $\hat{T}_n(d,S) \subset T_n(d,S)$, we have:

$$u_n(S) = \sum_{\substack{\text{Destns} \\ d}} \sum_{\substack{\text{Sources} \\ i \in \hat{T}_n(d,S)}} w_i \tag{2.2}$$

Let's compute outgoing utility of $n = 8866$ due to destinations 31420 and 22822 in Figure 2.1. For destination 31420, ASes 25076 and 34376 are part of the customer subtree $\hat{T}_n(d,S)$, but $15169, 8928$ and 22822 are not. For destination $d = 22822$, ASes $31420, 25076, 34376$ are part of the customer subtree. Thus, these ASes contribute $2+3$ incoming utility to ISP $n = 8866$.

Realistically, ISP utility is some function of both of these models; to avoid introducing extra parameters into our model, we consider each separately.

**Myopic best response.** We use a standard game-theoretic update rule known as *myopic best response*, that produces the most favorable outcome for a node in the next round, taking other nodes' strategies as given [52]. Let $(\neg S_n, S_{-n})$ denote the state when $n$ 'flips' to the opposite action (either deploying or undeploying S*BGP) that it used in state $S$, while other ASes maintain the same action they use in state $S$. ISP $n$ changes its action in state $S$ iff its *projected utility* $u_n(\neg S_n, S_{-n})$ is sufficiently high, *i.e.,*

$$u_n(\neg S_n, S_{-n}) > (1+\theta) \cdot u_n(S) \tag{2.3}$$

where $\theta$ is a threshold denoting the increase in utility an ISP needs to see before it is willing to change its actions. Threshold $\theta$ captures the cost of deploying BGP security; *e.g.,* an ISP might deploy S*BGP in a given round if S*BGP deployment costs do not exceed $\theta = 5\%$ of the profit it earns from transiting customer traffic. Since $\theta$ is multiplicative, it captures the idea that deployment costs are likely to be higher at ISPs that transit more traffic. The update rule is myopic, because it focuses on increasing ISP $n$'s utility in the next round only. It is best-response because it does *not* require ISP $n$ to speculate on other ASes' actions in future rounds; instead, $n$ takes these actions as given by the current state $S$.

**Discussion.** Our update rule requires ASes to predict their future utility. In our model, ASes have full information of $S$ and $G$, a common approach in game theory, which enables them to project their utility accurately. We discuss the consequences of our update rule, and the impact of partial information in Sections 2.9.1-2.9.2.

## 2.5 Simulation framework

We run simulations of our deployment model over an empirically derived graph of the Internet's topology: the AS graph. In each step, an ISP $n$ will decide whether or not to deploy S*BGP by comparing utility $u_n(S)$ and projected utility $u_n(\neg S_n, S_{-n})$. Computing these values requires us to determine the path from *every* source AS to *every* destination AS in the current state ($S$) and for *every* ISP $n$'s unique projected state $(\neg S_n, S_{-n})$. In this section, we highlight the challenges encountered in developing these simulations and how we address them. Specifically, we present our basic algorithm for computing paths to each destination (Section 2.5.2 and then describe the data structures and amortization methods we use to make this computation tractable.

### 2.5.1 Challenges when simulating on the AS graph

A number of studies have relied on modeling and simulation of the interdomain routing system, ranging from work on network reliability [59, 122], to BGP convergence [97], to geopolitical control of network traffic [65], to secure routing [10, 19, 41]. Researchers commonly model the interdomain routing system as a graph $G = (V, E)$ where the vertices are autonomous systems (ASes), and edges indicate physical connections between ASes [47]. Simulations are used to determine how ASes select routes for traffic using BGP (the Border Gateway Protocol, the de facto routing protocol in the global Internet). Because the AS graph is quite large (currently, it is thought to contain 36K vertices, and 130K edges [21, 30]), algorithms that run over the full AS graph quickly become computationally intensive. Thus, in designing simulations,

researchers must contend with (a) the fact the interdomain routing system does *not* use shortest-path routing, and (b) scalability issues.

**Routing policies differ from shortest path.** Because ASes are controlled by autonomous, economically-motivated entities, their routing decisions often incorporate considerations (*e.g.,* economics, geography ) beyond path length. Thus, standard shortest-path graph algorithms do not work 'out of the box'; instead, researchers have developed custom simulators that compute paths based on routing policy decisions of individual ASes (*e.g.,* using the routing policy model of Appendix A).

Some BGP simulators are designed to allow network operators to answer "what-if" questions about routing configurations within a single AS [34, 100], or simulate BGP at the packet level [31, 97]. The faithful modeling provided by these simulators comes at a high computational cost, making them ill-suited to simulations over a large number of ASes. In contrast, BSIM [64] and BGPSIM [121] abstract away intradomain and packet-level considerations in favor of modeling BGP on an AS graph. While considerably more scalable than previous work, the computational overhead of these simulators remains high because they use discrete event simulation. As such, BGPSIM [121] must be run on a supercomputer, while BSIM [64] can only run a limited number of simulations over the entire AS graph.

**Scale of the AS-level topology.** Alternatively, scalability issues may be mitigated by scaling down the AS graph itself. For example, one can use AS-graph generation tools like GT-ITM [129], BRITE [85], or Inet [57] to create a smaller synthetic AS graph, or use the techniques in [71] to scale down an empirically-derived AS graph (*e.g.,* as provided by CAIDA [30] or UCLA [21]) to a smaller set of ASes.

However, these approaches come with the risk that scaling has modified or even destroyed important graph properties. The authors of [71] show that their scaling techniques preserve several important graph-theoretic properties (*e.g.,* degree distribution, spectral properties). However, the AS graph contains structures which may not be captured by matching standard graph-theoretic properties. For example, we find that our results rely heavily on the prevalence of a

subgraph (the "Diamond" in Figure 2.3) that is difficult to describe using standard properties. Moreover, the risk is compounded because *a priori* it not always clear which graph properties will be important for a particular study; this often becomes clear only *after* the study is complete.

**Our simulator.** To avoid potential inaccuracies that might result from scaling down the AS-level topology, we advocate running simulations on full empirical AS graphs (*e.g.,* [21, 30]). To support running repeated simulations over the full AS graph, we developed lightweight simulation algorithms [41, 44] and parallelized them across a compute cluster running DryadLINQ [128]. Our approach drastically reduces computational overhead by *algorithmically* computing paths chosen by BGP. Moreover, to provide the speedup required to run simulations over the entire AS graph, our simulator (a) abstracts away intradomain details, (b) assumes that all ASes use the routing policies proposed in [36].

### 2.5.2 Routing tree algorithm

At the core of our simulation methodology is a *routing tree algorithm* that computes the best available paths for each source AS to a given destination AS $d$, *i.e.,* $T(d)$ (once BGP has converged so that no AS wishes to change its route). The algorithm assumes that ASes use the routing policies of Appendix A, and takes time $O(|V|^2)$ to compute paths between all source-destination pairs, which is significantly faster than the $O(|V|^3)$ algorithm proposed in [122].

The algorithm is a specialized three-stage breadth-first search (BFS) on the AS graph; each stage of the BFS computes shortest paths of a particular type (*i.e.,* customer / peer/provider). We explain the algorithm using Figure 2.2:

$1^{st}$ *stage: Customer paths.* We construct a partial routing tree by performing a BFS 'upwards' from the 'root' node $d$, using only customer-to-provider edges. (In Figure 2.2, we add edges $(d,1)$, $(d,2)$, $(d,5)$, $(2,6)$, and $(5,6)$).

$2^{nd}$ *stage: Peer paths.* Next, we use only single peer-to-peer edges to connect *new* ASes to the ASes *already added* to the partial routing tree in the $1^{st}$ stage of the algorithm. (In Figure 2.2,

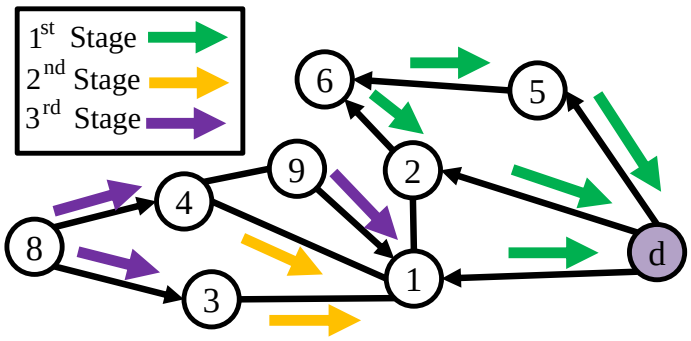



Figure 2.2: Routing tree algorithm

this amounts to adding edges $(1,4)$ and $(1,3)$ but not $(1,2)$ or $(4,9)$).

$3^{rd}$ *stage: Provider paths and Tiebreak.* Next, we traverse the existing partial routing tree with a BFS, and add *new* ASes to the tree using provider-to-customer edges. (In Figure 2.2, this amounts to adding edges $(1,9)$, $(4,8)$ and finally $(3,8)$). Each source AS that is traversed now has a set of equally-good paths (in terms of **LP** and **SP**) to the destination. Thus, for each source AS, we sort its set of paths according to **TB** and determine its chosen path to $d$.

**Running time.** Consider a AS graph $G(V,E)$ where $E_{cp} \subseteq E$ is the subset of customer-provider edges. The $1^{st}$ and $3^{rd}$ stages visit customer-provider edges only (giving a worst-case running time of $|V|+|E_{cp}|$ each), while the $2^{nd}$ stage visits peer-peer edges only ($|V|+|E \backslash E_{cp}|$). Thus, the full algorithm has worst-case running time of $3|V|+2|E|$ to compute paths to a *single destination*. Since the AS graph is sparse, with $|E| \approx 4|V|$, we can compute paths for *all source-destination pairs* paths in time $11|V|^2$.

### 2.5.3 Overview of our algorithmic approach

**Scaling challenges.** The combination of the dynamic state $S$ with the addition of **SecP** to the routing policy means that paths can change in *each iteration*, and moreover we need to compute paths in state $(\neg S_n, S_{-n})$ for *each ISP* in each iteration. Since there are $15\% \cdot |V|$ ISPs in the AS graph, the bottom line is that *in each iteration* we need compute all-pairs paths $15\%|V|$ times. While the routing tree algorithm can be used for the all-pairs path computation,

this gives a running time of $\sim 0.15|V| \cdot 11|V|^2 = 1.65|V|^3$, which quickly becomes intractable; if the routing tree algorithm for a single destination takes 10 *ms*, running it $15\%|V|^2 = 194M$ times takes 22 *days* for a *single iteration*!

Thus, we now show how we achieved the $1000x$ constant speedup that made our simulations feasible:

**Amortization.** Instead of repeating the routing tree algorithm $15\%|V|^2$ in each iteration, our approach was to run the (relatively-slow) routing tree algorithm *once*, and save the intermediate results; we then use these intermediate results as input to a faster algorithm that we run repeatedly in each iteration, thus *amortizing* our computation. In Section 2.5.4 we show how our faster algorithm exploits the sparsity of the AS graph to run in average time $1.18|V|$ (*cf.,* the routing tree algorithm's $11|V|$ worst-case running time). More concretely, our C# routing tree algorithm implementation ran in $\approx 10ms$, while our faster algorithm ran in $\approx 2ms$.

**Parallelization.** Even with the amortized algorithm, we still needed parallelization to run our simulations in a reasonable amount of time; notice that with a 2 *ms* amortized algorithm, a single iteration would take $0.15|V|^2 \cdot 2ms = 4$ *days*! Thus, we parallelize our algorithm to run on 200 machines running DryaLINQ [128] to further speed up the simulations.

**Total running time.** The combination of the $5x$ speedup from amortization and the $\sim 200x$ speedup from parallelization gives running time of about $0.15 \cdot 1.18/200 \cdot |V|^3 = \frac{1}{1100}|V|^3$, giving us a $\sim 1000x$ constant speedup. Indeed, after a few more optimizations (discussed in our report [42]) a single iteration could run in 10-20 *minutes*.

### 2.5.4 Speedups via amortization

The following observation allowed us to use amortization:

**Observation 2.5.1.** *If ASes use the routing policies of Section A, the length and type (*i.e., *customer, peer, provider) of any node i's path to a destination d is independent of the state S of the AS graph.*

The proof of this observation can be found in our report [42]. For each destination AS $d$, we do the following. First, we run the full routing tree algorithm *only once* to precompute the set of equally-good (in terms of type and length) paths from each source AS $n$ to destination AS $d$. Then, for each of the $15\%|V|$ states $(\neg S_n, S_{-n})$ in each iteration, we repeat the following: given the tiebreak sets, we compute $n$'s chosen path to $d$ according the state, **SecP**, and **TB** (Appendix A).

The following values are precomputed from the routing tree algorithm for each destination $d$:

**BucketTable:** Table 2.1 is a BUCKETTABLE for the AS graph in Figure 2.2 with $d$ as the destination. For each AS $n$, a BUCKETTABLE stores the (1) the path type, *i.e.,* the relationship between $n$ and the next hop on $n$'s best path to $d$, (*i.e.,* the decision in the **LP** step) (2) the length of $n$'s best path to $d$ (*i.e.,* the decision in the **SP** step). Each AS $n$ is placed in a cell of the BUCKETTABLE according to its best path length (row) and best path type (column), *e.g.,* AS 8 with a three hop path to $d$ through a provider is in cell $(3, provider)$.

**TieBreakSet:** We have a TIEBREAKSET for each source AS $n$ that stores the set of neighbors that offer $n$ equally good paths to $d$ according to **LP** and **SP**. The ASes in TIEBREAKSET are sorted by their ranking according to **TB**. For example, $\text{TIEBREAKSET}(6,d) = \{2,5\}$ because according to **LP** and **SP**, both AS 2 and AS 5 offer AS 6 equally good paths to $d$ (both are 2-hop paths where the next hop is a customer); AS 2 is stored first because it wins the **TB** step.

**Amortized routing tree algorithm.** This algorithm computes the routing tree for a given destination $d$ and state $S$ using the precomputed BUCKETTABLE and TIEBREAKSETS associated with $d$. Observe that every node in AS $n$'s tiebreak set must have a path to $d$ that is exactly one-hop shorter than the path that AS $n$ has to $d$. Thus, we process each node $n$'s routing decision in ascending order of path length, starting with the destination $d$. To do this, we start at the $0^{th}$ row of the BUCKETTABLE (which can contain only the destination $d$) and walk down the rows BUCKETTABLE, processing each node $n$ in the row as follows:

- If $n$ is secure in state $S$ and there are nodes in $i$'s tiebreak set with a secure path to $d$, then $n$

Table 2.1: BucketTable for routing tree in Figure 2.2

| Row | Cust. | Peer | Prov. |
|---|---|---|---|
| 0 | d | - | - |
| 1 | 1, 2, 5 | - | - |
| 2 | 6 | 3, 4 | 9 |
| 3 | - | - | 8 |

chooses a path through through first node in the TIEBREAKSET that offers $n$ a secure path. AS $n$ is marked as using a secure path to $d$.

- Otherwise, $n$ chooses a path through the first node in its the TIEBREAKSET and $n$ is marked as using an insecure path to $d$.

The average running time of the amortized routing tree algorithm is $t|V|$ for a single destination, where $t$ is the average size of the TIEBREAKSET in the AS graph. Because the AS graph is quite sparse, it turns out that $t = 1.18$, which gives us up to a $6x$ speedup over the $11|V|$ routing tree algorithm.

### 2.5.5 Speedups via parallelization

We leverage the fact that all our algorithms are independent across destinations, making them particularly amenable to Map-Reduce style parallelization. Thus, we parallelized (*i.e.,* mapped) our computation of the subtrees $T(d,S)$ for each state $(\neg S_n, S_{-n})$ across destinations $d$. We then aggregated (*i.e.,* reduced) them across the states $(\neg S_n, S_{-n})$ to obtain utility, *i.e.,* $\sum_d T_n(d)$ in state $(\neg S_n, S_{-n})$. We ran our code on a shared cluster of about 200 machines running DryadLINQ [128]; by parallelizing across destinations, each machine was roughly assigned computation for about $|V|/200 = 150$ destinations (but see [128] for details on how to implement this in DryadLINQ).

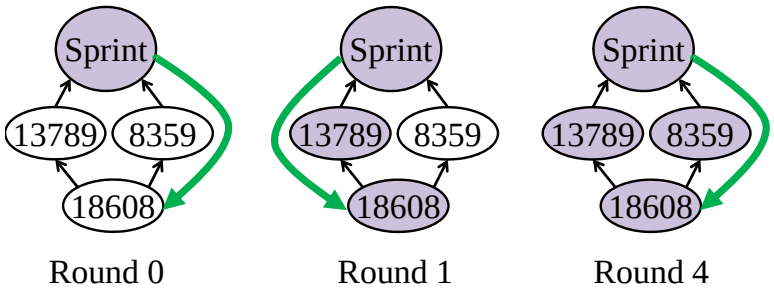



Figure 2.3: A DIAMOND: ISPs 13789 and 8359 compete for traffic from Sprint (AS 1239)

## 2.6 Case Study: S*BGP Deployment

We start by showing that even a small set of early adopters can create enough market pressure to transition the vast majority of ASes to S*BGP.

**Case study overview.** We focus on a single simulation where the early adopters are the five CPs (Google, Facebook, Microsoft, Limelight, Akamai, see Section 2.4.1), and the top five Tier 1 ASes in terms of degree ( Sprint (1239), Verizon (701), AT&T (7018), Level 3 (3356), Cogent (174)). Every ISP uses an update rule with a relatively low threshold $\theta = 5\%$, that the five CPs originate $x = 10\%$ of the traffic in the Internet, and that stubs *do* break ties in favor of secure routes. We now show how even a small set of ten early adopters (accounting for less that 0.03% of the AS graph) can convince 85% of ASes to deploy S*BGP, and secure 65% of all paths in the AS graph.

### 2.6.1 Competition drives deployment

We start by zooming in on S*BGP deployment at two competing ISPs, in a scenario we call a DIAMOND.

**Figure 2.3:** Two ISPs, AS 8359 and AS 13789, compete for traffic from Sprint (AS 1239) to their stub customer, AS 18608. Sprint is an early adopter of S*BGP, and initially the three other ASes are insecure. Both ISPs offer Sprint equally good two-hop customer paths to the stub, and AS 8359 is chosen to carry traffic by winning the tie break. In the first round, AS 13789 computes its projected utility, and realizes it can gain Sprint's traffic by adopting S*BGP

Table 2.2: Occurrences of the DIAMOND scenario for early adopter ASes (sorted by degree)

| | | | | | |
|---|---|---|---|---|---|
| **Tier 1s** | AS 174 | 878 | **CPs** | AS 22822 | 175 |
| | AS 3356 | 1,400 | | AS 15169 | 892 |
| | AS 7018 | 340 | | AS 20940 | 178 |
| | AS 701 | 706 | | AS 8075 | 1,149 |
| | AS 1239 | 728 | | AS 32934 | 82 |

and upgrading its stub to simplex S*BGP. (See Section 2.9.2 for more discussion on how ISPs compute projected utility.) By the fourth round, AS 8359 has lost so much utility (due to traffic lost to ASes like 13789) that he decides to deploy S*BGP.

Of course, Figure 2.3 is only a very small snapshot of the competition for traffic destined to a single stub AS 18608; utility for each ISPs is based on customer traffic transited to *all* destinations in the AS graph. Indeed, this DIAMOND scenario is quite common.

In Table 2.2, we summarize the number of diamonds we counted, each involving at two ISPs, a stub and one of the early adopters.

### 2.6.2 Global deployment dynamics

**Figure 2.4:** We show the number of ASes (*i.e.,* stubs, ISPs and CPs) and the number of ISPs that deploy S*BGP at each round. In the first round, 548 ISPs become secure; because each of these ISPs deploy simplex S*BGP in their stubs, we see that over 5K ASes become secure by the end of the first round. In subsequent rounds, hundreds of ISPs deploy S*BGP in each round; however, the number of newly secure stubs drops dramatically, suggesting that many ISPs deploy S*BGP to regain traffic lost when their stubs were secured by competitors. After the 17th iteration, the process tapers off, with fewer than 50 ASes becoming secure in each round. The final surge in deployment occurs in round 25, when a large AS, 6939, suddenly became secure, causing a total of 436 ASes to deploy S*BGP in the remaining six rounds. When the process terminates, 85% of ASes are secure, including 80% of the 6K ISPs in the AS

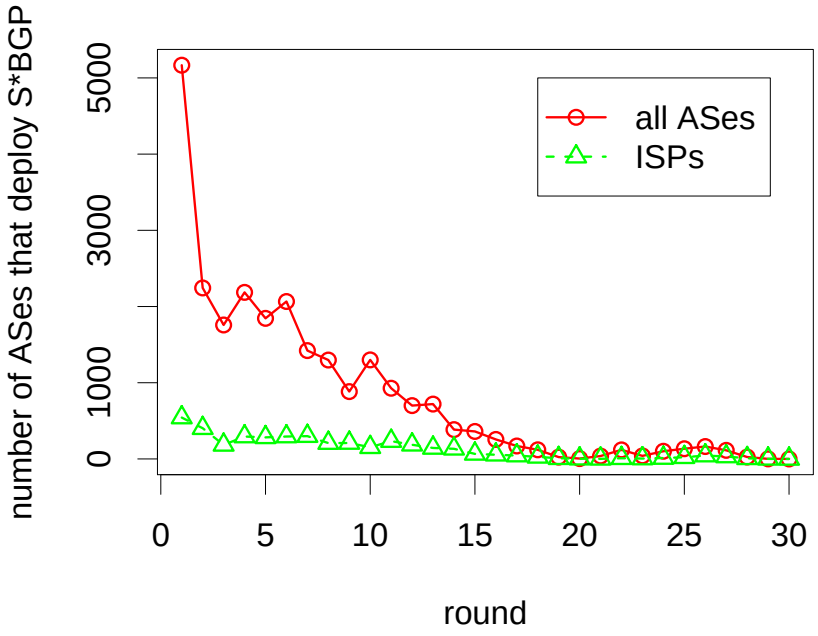



Figure 2.4: The number of ASes that deploy S*BGP each round

graph.

### 2.6.3 Impact of ISP degree on deployment

The reader might be surprised to find that ISPs with high degree are more likely to deploy S*BGP:

**Figure 2.5:** We consider the cumulative fraction of ISPs adopting S*BGP in each round, separated by degree. Interestingly, ISPs with low degree ($\leq$ 10) are less likely to become secure. Indeed, we found a consistent set of about 1000 ISPs that *never* deploy S*BGP in *any* of our simulations (not shown). These ISPs had average degree 6, and remained insecure because they never had to compete for customer traffic; indeed, they were usually providers to only single-homed stub customers.

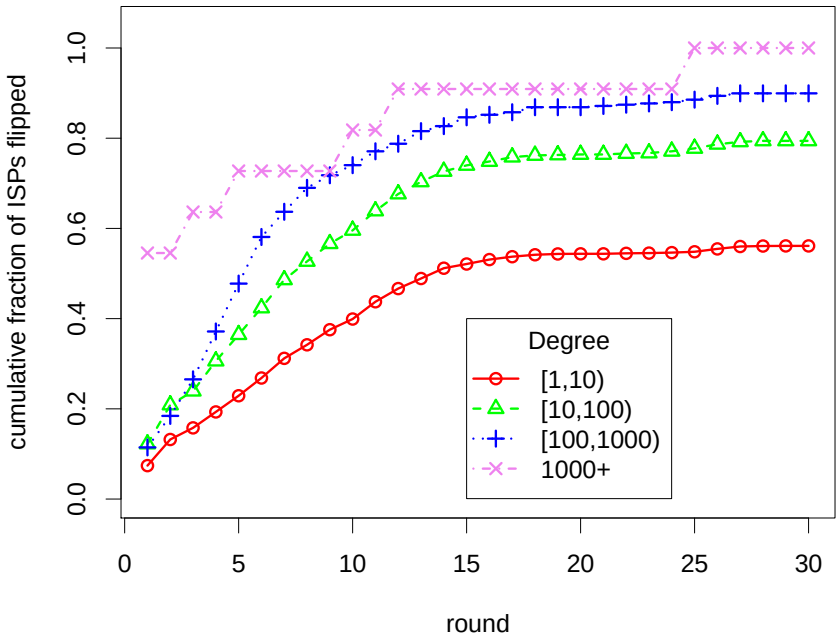



Figure 2.5: Cumulative fraction of ISPs that deploy S*BGP by degree

### 2.6.4 Longer secure paths sustain deployment

In Figure 2.4 we observed rapid, sustained deployment of S*BGP in the first 17 iterations. This happens because longer secure paths are created as more ASes deploy S*BGP, thus creating incentives for S*BGP at ASes that are far away from the early adopters:

**Figure 2.6:** We once again encounter AS 8359 from Figure 2.3. We show how AS 8359's decision to deploy S*BGP in round 4 allows a new ISP (AS 6731) to compete for traffic. In round 5 AS 6731 sees a large increase in utility by becoming secure. This occurs, in part, because AS 6731 can now entice six of the early adopters to route through him on a total of 69 newly-secure paths. Indeed, when AS 6731 becomes secure, he continues the chain reaction set in motion by AS 8359; for instance, in round 7 (not shown), AS 6371's neighbor AS 41209 becomes secure in order to offer Sprint a new, secure four-hop path to one of 41209's own stubs.

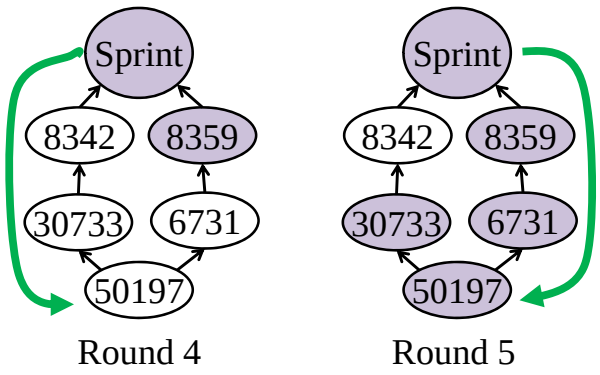



Figure 2.6: A newly created four-hop secure path

### 2.6.5 Keeping up with the competition

Two behaviors drive S*BGP deployment in a DIAMOND. First, an ISP becomes secure to steal traffic from a competitor, and then the competitor becomes secure in order to regain the lost traffic. We can watch this happening for the ISPs from Figure 2.3 and 2.6:

**Figure 2.7:** We show the utilities of ISPs 8359, 6731, and 8342 in each round, normalized by *starting utility i.e.,* the utility before the deployment process began (when all ASes, including the early adopters, were still insecure). As we saw in Figure 2.3, AS 8359 deploys S*BGP in round 4 in order to regain traffic he lost to his secure competitors; here we see that in round 4, AS 8359 has lost 3% of his starting utility. Once AS 8359 deploys S*BGP, his utility jumps up to more than 125% of his starting utility, but these gains in utility are only temporary, disappearing around round 15. The same is true in round 6 for AS 6371 from Figure 2.6. By round 15, 60% ISPs in the AS graph are already secure (Figure 2.4), and our ISPs can no longer use security to differentiate themselves, causing their utility to return to within 3% of their starting utility.

This is also true more generally:

**Figure 2.8:** For each round $i$, we show the median utility and median projected utility for ISPs that become secure in round $i+1$, each normalized by starting utility. (Recall from (2.3) that these ISPs have projected utility at least $1+\theta$ times their utility in round $i$.) In the first 9 rounds, ISPs mainly deploy S*BGP to steal traffic from competitors; that is, their projected utility in the

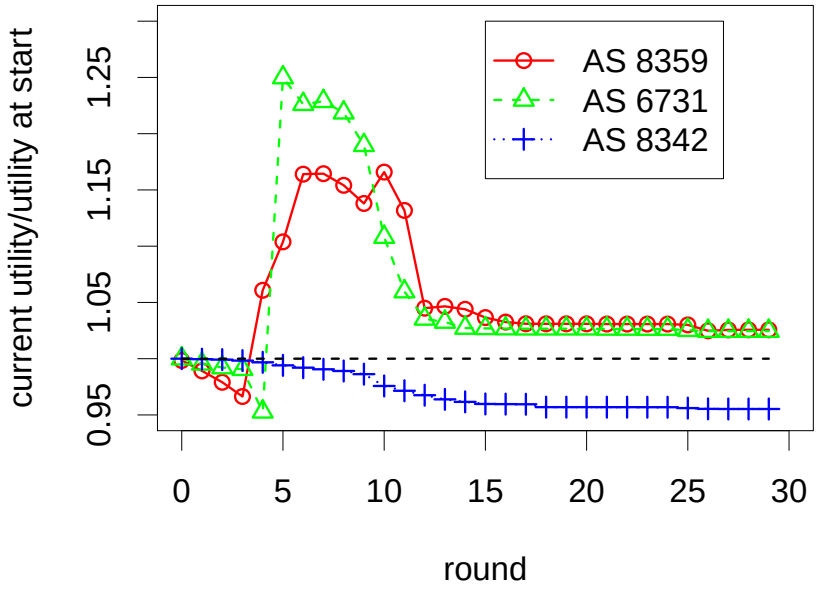



Figure 2.7: Normalized utility of ISPs in Fig. 2.3 and 2.6

round before they deploy S*BGP is at least $1+\theta = 105\%$ times their starting utility. However, as deployment progresses, ASes increasingly deploy S*BGP in order to recover lost traffic and return to their starting utility; that is, in rounds 10-20 ISP utility drops to at least $\theta = 5\%$ less than starting utility, while projected utility approaches starting utility (y=1).

### 2.6.6 Is S*BGP deployment a zero-sum game?

Our model of S*BGP deployment is indeed a zero-sum game; we assume that ISPs compete over a fixed set of customer traffic. Thus, when the vast majority of ASes have deployed S*BGP, ISPs can no longer use security to distinguish themselves from their competitors (Figure 2.7). At the termination of this case study, only 8% of ISPs have an increase in utility of more than $\theta = 5\%$ over their starting utility. On the other hand, 85% of ASes now benefit from a (mostly) secure Internet. Furthermore, like ASes 8359 and 6731 in Figure 2.7, many of these secure ASes enjoyed a prolonged period of increased utility that could potentially help defray

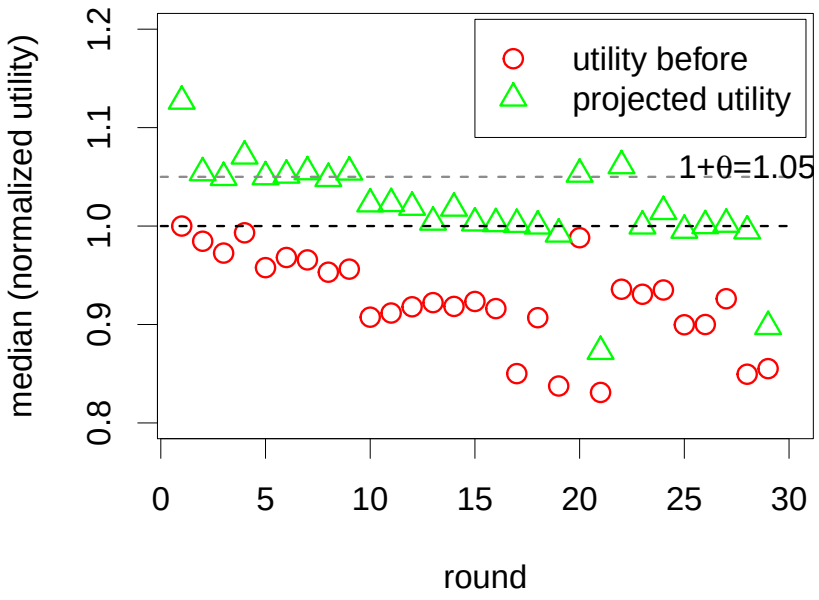



Figure 2.8: Projected and actual utility before deploying S*BGP normalized by starting utility

the costs of deploying S*BGP.

**It is better to deploy S*BGP.** One might argue that a cynical ISP might preempt the process by *never* deploying S*BGP. However, a closer look shows that its almost always in the ISPs interest to deploy S*BGP. ISPs that deploy S*BGP usually return to their starting utility or slightly above, whereas ISPs that do *not* deploy S*BGP lose traffic in the long term. For instance, AS 8342 in Figure 2.6 never deploys S*BGP. As shown in Figure 2.7, when the deployment process terminates, AS 8342 has lost 4% of its starting utility. Indeed, another look at the data (not shown) shows that the ISPs that remain insecure when the process terminates lose on average 13% of their starting utility!

## 2.7 Choosing early adopters

Next, we consider choosing the set of ASes that should be targeted to become early adopters of S*BGP.

### 2.7.1 It's hard to choose early adopters

Ideally, we would like to choose the *optimal* set of early adopters that could cause the maximum number of other ASes to deploy S*BGP. However, this is NP-hard by presenting a reduction to the 'set cover' problem (proof in [42]):

**Theorem 2.7.1.** *For an AS graph $G(V,E)$ and a parameter $1 \leq k \leq |V|$, finding a set of early adopter ASes of size k that maximizes the number of ASes that are secure when the deployment process terminates is NP-hard. Approximating the solution within a constant factor is also NP-hard [41].*

As such, we use simulations[4] of the deployment process to investigate heuristic approaches for choosing early adopters, including AS degree (*e.g.,* Tier 1s) and volume of traffic originated by an AS (*e.g.,* content providers).

### 2.7.2 The parameter space

We consider how the choice of early adopters is impacted by assumptions on (1) whether or not stubs running simplex S*BGP break ties based on security, (2) the AS graph, and (3) traffic volumes sourced by CPs.

**Outgoing utility.** Also, recall that we have two models of ISP utility (Section 2.4.3). In this section, we dive into the details of the *outgoing utility* model because it has the following very nice property:

**Theorem 2.7.2.** *In the outgoing utility model, a secure node will never have an incentive to turn off S*BGP [41].*

As a consequence of this theorem (proof in [42]), it immediately follows that (a) every simulation must terminate, and (b) we can significantly reduce compute time by *not* computing

[4]Since there is no sampling involved, there is no variability between simulations run with the same set of parameters.

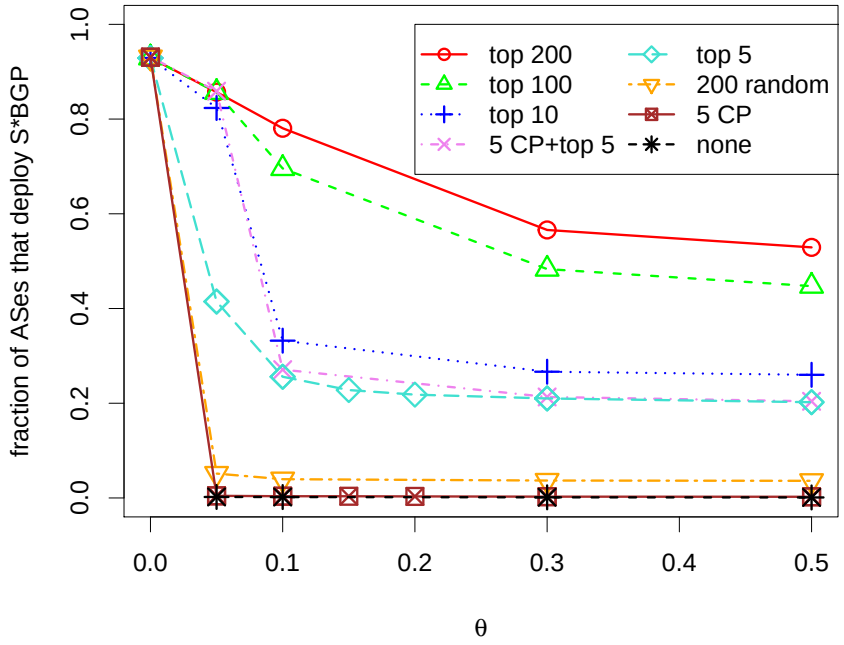



Figure 2.9: Fraction of ASes that deploy S*BGP for varying θ and early adopters

projected utility for ISPs that are already secure. (We discuss complications that arise from the incoming utility model in Section 2.8.)

**Deployment threshold θ.** Our update rule (Equation 2.3) is such that ISPs change their actions if they can increase utility by at least θ. Thus, to gain insight into how 'difficult' it is to convince ISPs to deploy S*BGP, we assume that each ISP uses the same threshold θ, and sweep through different values of θ (but see also Section 2.9.2).

### 2.7.3 Comparing sets of early adopters

We next explore the influence of different early adopters:

**Figure 2.9:** We show the fraction of ASes that adopt S*BGP for different values of θ. We consider no early adopters, the top 5-200 ISPs in terms of degree, the five CPs, five CPs in combination with the top five ISPs, and 200 random ISPs.

**There are incentives to deploy S*BGP.** For low values of $\theta < 5\%$, we observe that there is sufficient competition over customer traffic to transition 85% of ASes to S*BGP. Moreover,

this holds for almost every set of early adopters we considered. (Note that in the unrealistic case where $\theta = 0$, we see widespread S*BGP deployment even with *no* early adopters, because we assume the stubs break ties in favor of secure paths. But see also Section 2.7.7.) Furthermore, we find that the five CPs have approximately the same amount of influence as the case where there are no early adopters; we investigate this in more detail in Section 2.7.8.

**Some ISPs always remain insecure.** We find 20% of the 6K ISPs in the AS graph [8, 21] never deploy S*BGP, because they are never subject to competition for customer traffic. This highlights two important issues: (1) some ISPs may never become secure (*e.g.,* ASes whose customers are exclusively single-homed) (2) S*BGP and BGP will coexist in the long term.

**Choice of early adopters is critical.** For higher values of $\theta \geq 10\%$, it becomes important to choose ISPs with high customer degree as early adopters. In fact, Figure 2.9 shows a set of 200 random ASes has significantly lower influence than a set containing only the five top ASes in terms of degree. For large values of $\theta \geq 30\%$, a larger set of high-degree early adopters is required, with the top 200 ASes in terms of degree causing 53% of the ASes to deploy S*BGP for $\theta = 50\%$. However, to put this observation in some perspective, recall that $\theta = 30\%$ suggests that the cost of S*BGP deployment exceeds 30% of an ISP's profit margin from transiting customer traffic.

### 2.7.4 How much security do we get?

We count the number of secure paths at the end of the deployment process, as a measure of the efficacy of S*BGP deployment. (Of course, this is *not* a perfect measure of the AS graph's resiliency to attack; quantifying this requires approaches similar to [19, 44], an important direction for future work.)

**Figure 2.10:** We show the fraction of the $(36K)^2$ paths between ASes that are secure, for the different sets of early adopters. As expected, we find that the fraction of secure path is only slightly lower than $f^2$, where $f$ is the fraction of ASes that have deployed S*BGP. (The $f^2$ follows from the fact that for a path to be secure, both its source AS and its destination AS

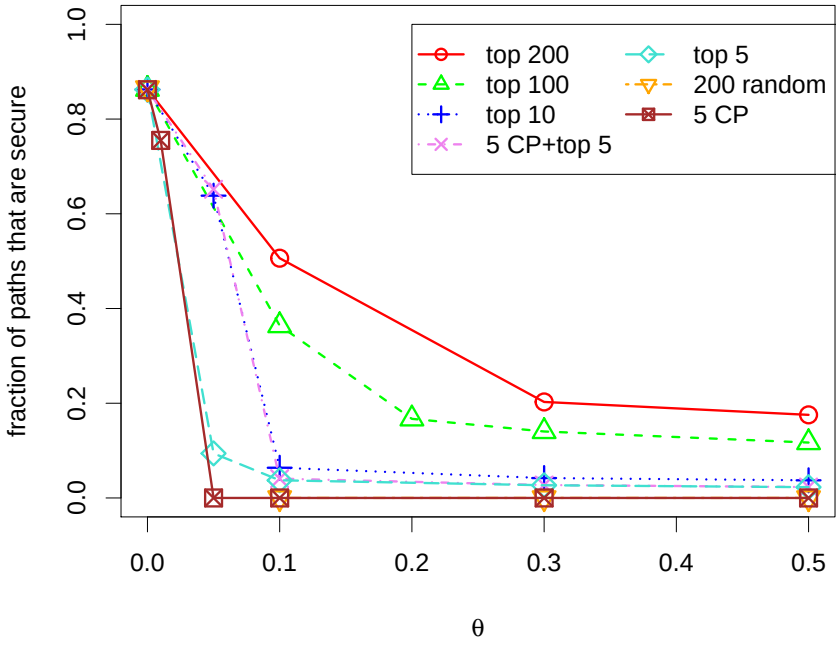



Figure 2.10: Fraction of paths on the Internet that are secure

must be secure.) Indeed, the fact the number of secure paths is only 4% lower than $f^2$ suggests that the majority of secure paths are likely to be quite short.

### 2.7.5 Market pressure vs. simplex S*BGP

The cause of global S*BGP deployment differs for low and high values of the deployment threshold θ:

**Figure 2.11:** We show the fraction of ISPs (not ASes) that deploy S*BGP for the early adopter sets and varying values of θ. For low values of θ, market pressure drives a large fraction of ISPs to deploy S*BGP. In contrast, for higher values of θ very few ISPs deploy S*BGP, even for large sets of well-connected early adopters. In these cases, most of the deployment is driven by ISPs upgrading their stub customers to simplex S*BGP. For example, for the top 200 ISPs, when $\theta = 50\%$, only a small fraction of secure ASes (4%) deploy S*BGP because of market pressure, the vast majority (96%) are stubs running simplex S*BGP.

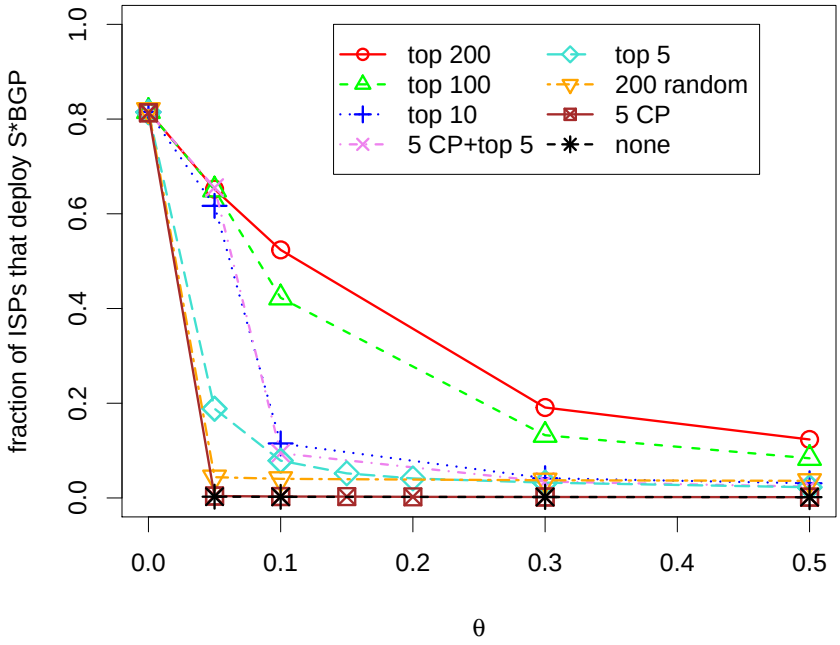



Figure 2.11: Fraction of ISPs that deploy S*BGP for varying θ and early adopters

### 2.7.6 The source of competition: Tie break sets

Recall that the *tiebreak set* is the set of paths on which an AS employs the security criterion to select paths to a destination AS (Section 2.3.1). A tiebreak set with multiple paths presents opportunities for ISPs to compete over traffic from the source AS.

**Figure 2.12:** We show distribution of tiebreak set size for all source-destination pairs of ASes. (This result holds for the AS graph [8, 21] under the assumption that ASes use the routing policies of Appendix A.) Noting that this graph has a log-log scale, observe that tiebreak sets are typically very small. ISPs have slightly larger tiebreak sets than stubs: an average of 1.30 for ISPs and 1.16 for stubs. Moreover, only 20% tiebreak sets contain more than a single path.

This striking observation suggests that even a very limited amount of competition suffices to drive S*BGP deployment for low θ. Furthermore, we speculate that this might also explain why there is limited market pressure for S*BGP deployment at ISPs when $\theta > 10\%$.

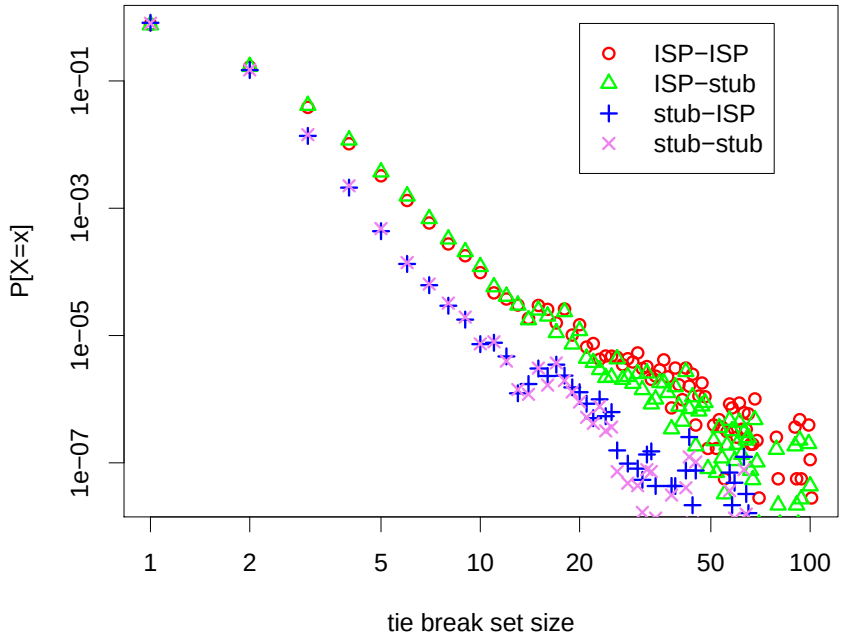



Figure 2.12: Probability density function of tie break set size in the AS graph [8, 21] for different source-destination pairs (log-log scale)

### 2.7.7 Stubs don't need to break ties on security

So far, we have focused on the case where secure stubs break ties in favor of secure paths. Indeed, given that stubs typically make up the majority of secure ASes, one might expect that their routing decisions can have a major impact of the success of the S*BGP deployment process. Surprisingly, we find that this is not the case. Indeed, our results are insensitive to this assumption, for $\theta > 0$ and regardless of the choice of early adopter (Figure 2.13). We explain this by observing that stubs both (a) have small tiebreak sets, and (b) transit no traffic.

**Security need only effect a fraction of routing decisions!** Thus, only 15% of ASes (*i.e.,* the ISPs) need to break ties in favor of secure routes, and only 23% of ISP tiebreak sets contain more than one path. Combining these observations, we find that S*BGP deployment can progress even if only $0.15 \times 0.23 = 3.5\%$ of routing decisions are effected by security considerations!

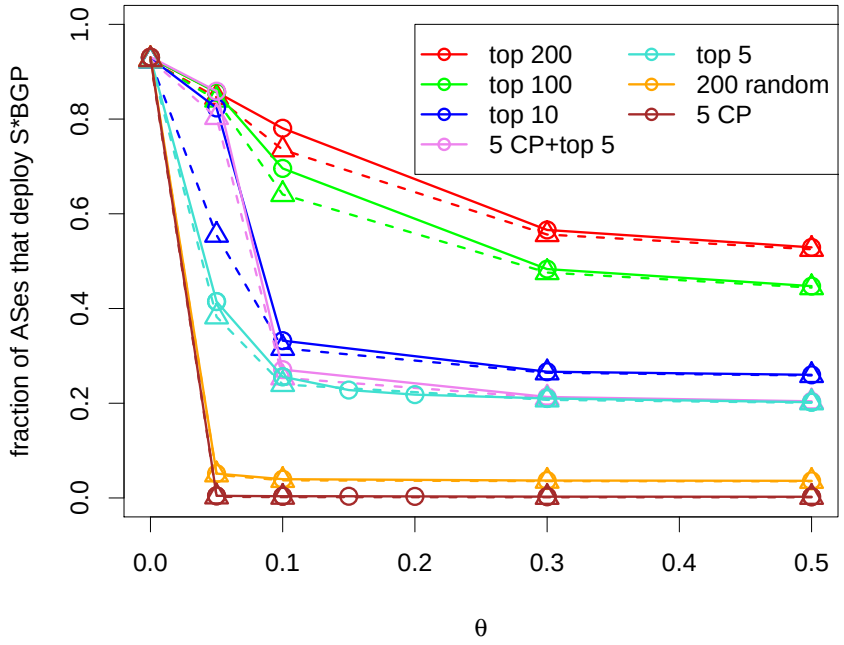



Figure 2.13: Fraction of ASes that deploy S*BGP for different early adopters (dashed lines stubs do not prefer secure paths)

### 2.7.8 Robustness to traffic and connectivity

#### Varying parameters

To understand the sensitivity of our results we varied the following parameters:

**1. Originated traffic volumes.** We swept through different values $x = \{10\%, 20\%, 33\%, 50\%\}$ for the fraction of traffic originated by the five CPs (Section 2.4.1); recent work suggests a reasonable range is $x =$10-20% [73, 75] .

**2. Traffic destinations.** Initially, we assume ASes uniformly spread their traffic across all potential destinations. We test the robustness of our results to this assumption by modeling traffic locality. We model locality by assuming ASes send traffic proportional to $1/k$ to destination ASes that are $k$ hops away.

**3. Connectivity of content providers.** Published AS-level topologies are known to have poor visibility into peering links at the edge of the AS-level topology [94]. This is particularly problematic for CPs, who in recent years, have shifted towards peering with many other ASes

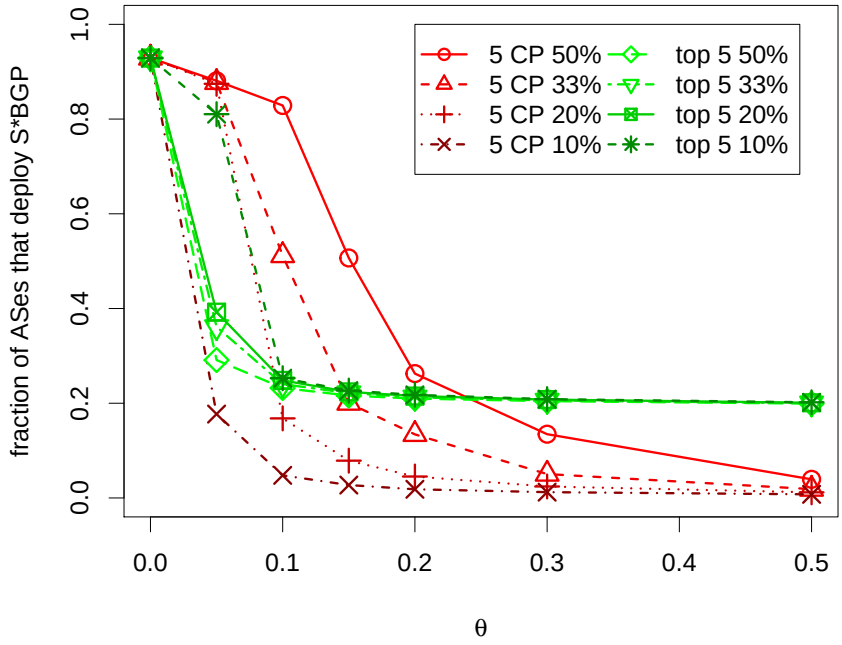



Figure 2.14: Fraction of ASes that deploy S*BGP for the five content providers and five Tier 1s in the peering-heavy, augmented topology

to cut down content delivery costs [29, 39] . Indeed, while the CPs known to have short path lengths [95], their average path length in our AS graph (with routing policies as in Appendix A) was 2.7 hops or more. Thus, for sensitivity analysis, we created a peering-heavy AS graph with 19.7K artificial peering edges from the five CPs to 80% of ASes found to be present at IXPs [8]. In our augmented AS graph, (described in Appendix C) the average path length of the CPs dropped to about 2, and their degree increased to be as high as the largest Tier 1 ISPs.

**Impact of traffic volumes and connectivity**

**Figure 2.14:** We sweep through values of θ to compare the five CPs and top five Tier 1s as early adopters for (a) different traffic volumes, (b) in our augmented AS graph vs. the original graph. We find the following:

*1. Originated traffic volumes vs. degree.* Surprisingly, when the five CPs source $x = 10\%$ of traffic, they are much less effective as early adopters than the top five Tier 1 ASes. Even though in the augmented topology the Tier 1s and CPs have about equal degree, the dominant

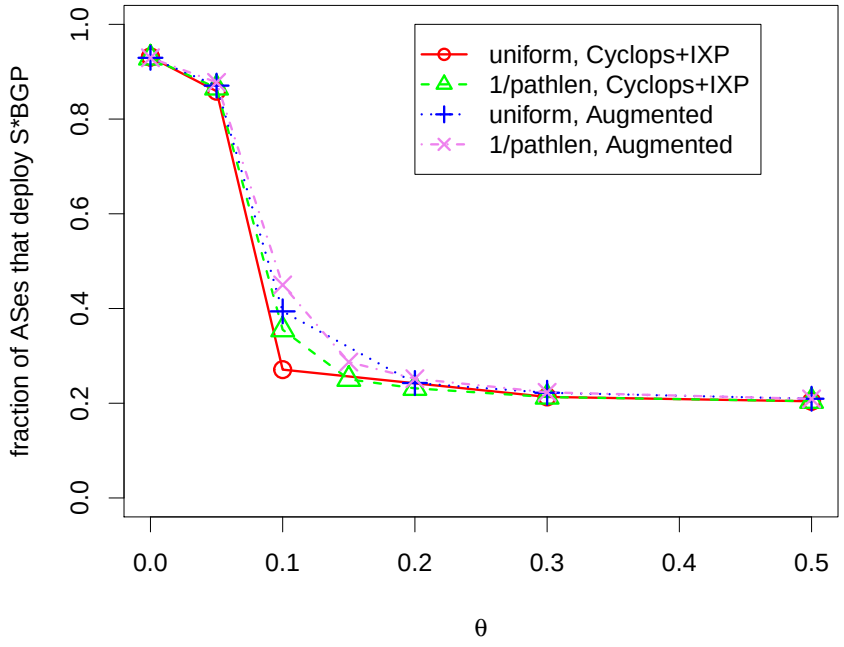



Figure 2.15: Fraction of ASes that deploy S*BGP with the five content providers and five Tier 1s as early adopters for both the Cyclops+IXP [8, 21] and peering-heavy, augmented topology

factor here is traffic; even though the CPs *originate* 10% of traffic, the Tier 1s still *transit* 2-9X times more traffic. As $x$ increases to 50%, the Tier 1s only transit 0.3-1.2X more traffic than is originated by the CPs. Thus, the CPs tend to have more influence for lower values of $\theta \leq 10\%$.

*2. Localized interdomain traffic.* We consider the number of ASes that deploy S*BGP with the five content providers and five Tier 1s as early adopters for both the original graph and the peering-heavy, augmented topology for a range of values of $\theta$ (Figure 2.15). We compare the model where ASes send traffic uniformly to all destinations to the case where ASes send more traffic to destinations within a few hops of themselves. For both the original and augmented topology, our results are robust even when ASes direct most of their traffic to nearby destinations.

*3. Impact of peering-heavy structure on simplex S*BGP.* Figure 2.14 indicates the five Tier 1 consistently outperform the CPs as early adopters when $\theta \geq 0.3$. The explanation for this is simple; Tier 1s have a large number of stub customers that they immediately upgrade to

simplex S*BGP. This suggests that having CPs to upgrade their *stub peers* to simplex S*BGP could potentially drive S*BGP deployment further.

### 2.7.9 Summary and recommendations

We make two key observations regarding selection of early adopters. First, only a small number of ISPs suffice as early adopters when deployment thresholds θ are small. Second, to withstand high θ, Tier 1 ASes should be targeted. This is due to the high volumes of traffic they transit and the many stubs they upgrade to simplex S*BGP. Finally, we note that our results hold even if more than 96% of routing decisions are insensitive to security considerations!

## 2.8 Other Complications

Intuition suggests that a secure ISP will observe increased utility because secure ASes transit traffic through it. While this is true in the *outgoing utility* model (Theorem 2.7.2), it turns out that this is *not* the case for the *incoming utility* model. We now discuss complications that might arise because we require S*BGP to play a role in route selection.

### 2.8.1 Buyer's Remorse: Turning off S*BGP

We present an example of a severe obstacle to S*BGP deployment: an secure ISP that has incentive to turn *off* S*BGP. The idea here is that when an ISP $n$ becomes secure, some of $n$'s incoming traffic might change its path, and enter $n$'s network along peer/provider edges instead of customer edges, thus reducing $n$'s utility. This causes the secure ISP's utility to satisfy Equation 2.3, resulting in the ISP opting to undeploy S*BGP.

**Figure 2.16:** We show that AS 4755, a Telecom provider in India, has an incentive to turn off S*BGP in its network. We assume content providers have $w_{CP} = 821$ which corresponds to 10% of Internet traffic originating at the big five CPs (including Akamai's AS 20940).

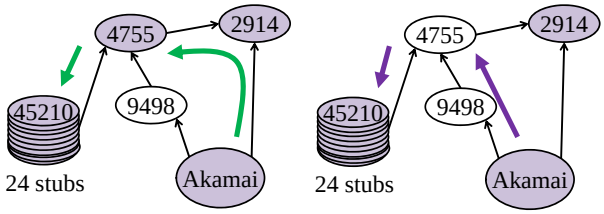



Figure 2.16: AS 4755 incentives turn off S*BGP

In the state $S$ on the left, Akamai, AS 4755, and NTT (AS 2914) are secure, the stub customers of these two secure ISPs run simplex S*BGP, and all other ASes are insecure. Here, AS 4755 transits traffic sourced by Akamai from his provider NTT AS 2914, to a collection of twenty-four of its stub customers (including AS 45210). Akamai's traffic does *not* increase AS 4755's utility because it arrives at AS 4755 along a provider edge.

In the state $(\neg S_{4755}, S_{-4755})$ on the right, AS 4755 turns S*BGP off. If we assume that stubs running simplex S*BGP do *not* break ties based on security, then the only ASes that could potentially change their routes are the secure ASes 20940 and 2914. Notice that when AS 4755 turns S*BGP off, Akamai's AS 20940 has *no secure route* to AS 4755's stub customers (including AS 45210). As such, Akamai will run his usual tie break algorithms, which in our simulation came up in favor of AS 9498, a customer of AS 4755. Because Akamai's traffic is now enters AS 4755 on customer edges, AS 4755's incoming utility *increases* by a factor of 205% per each of the 24 stub destinations.

**Turning off the entire network.** Our simulations confirmed that, apart from Akamai changing its chosen path these twenty-four stubs, all other ASes use the same routes in state $S$ and state $(\neg S_{4755}, S_{-4755})$. This means that AS 4755 has an incentive to turn off S*BGP in his *entire network*; no routes other than those ones Akamai uses to reach the twenty-four stubs are impacted by his decision. Indeed, we found that the utility of AS 4755 increase by a total of 0.5% (over all destinations) when he turns off S*BGP!

**Turning off a destination.** AS 4775 could just as well turn off S*BGP on a *per destination* basis, *i.e.,* by refusing to propagate S*BGP announcements for the twenty-four stubs in Figure 2.16, and sending insecure BGP messages for these destinations instead.

### 2.8.2 How common are these examples?

At this point, the reader may be wondering how often an AS might have incentives to turn off S*BGP.

**Turning off an entire network?** Figure 2.16 proves that cases where an ISP has an incentive to turn off S*BGP in its *entire network* do exist in realistic AS-level topologies [21]. However, we speculate that such examples will occur infrequently in practice. While we cannot provide any concrete evidence of this, our speculation follows from the fact that an ISP $n$ obtains utility from many destinations. Thus, even if $n$ has increased its utility by turning OFF S*BGP for destinations that are part of subgraphs like Figure 2.16, he will usually obtain higher utility by turning ON S*BGP for the other destinations that are not part of such subgraphs. (In Figure 2.16, this does not happen because the state $S$ is such that only a very small group of ASes are secure; thus, no routes other than the ones pictured are effected by AS 4755's decision to turn off S*BGP.)

**Turning off a destination is likely.** On the other hand, it is quite easy to find examples of *specific destinations* for which an ISP might want to turn off S*BGP. Indeed, a search through the AS graph found that at least 10% of the 5,992 ISPs could find themselves in a state where they have incentives to turn off S*BGP for at least one destination!

## 2.9 Discussion of our model

The wide range of parameters involved in modeling S*BGP deployment means that our model (Section 2.4) cannot be *predictive* of S*BGP deployment in practice. Instead, our model was designed to (a) capture a few of the most crucial issues that might drive S*BGP deployment, while (b) taking the approach that simplicity is preferable to complexity.

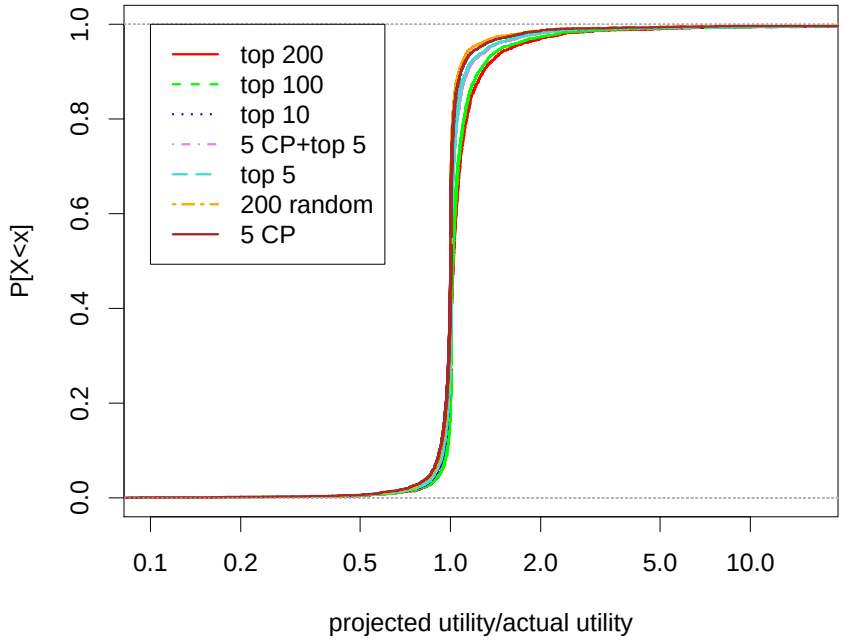



Figure 2.17: Projected utility normalized by actual utility after an AS deploys S*BGP

### 2.9.1 Myopic best response

For simplicity, we used a *myopic best-response* update rule that is standard in the game-theory literature [52]. In Section 2.6.6, we discussed the consequences of the fact that ISPs only act to improve their utility in the next round, rather than in long run. Another potential issue is that our update rule ignores the possibility that *multiple* ASes could deploy S*BGP in the transition from a round $i$ to round $i+1$, resulting in the gap between the projected utility, and the actual utility in the subsequent round. Fortunately, our simulations show projected utility $u_n(\neg S_n, S_{-n})$ is usually an excellent estimate of actual utility in the subsequent round. For example, in the case study of Section 2.6, 80% of ISPs overestimate their utility by less than 2%, 90% of ISPs overestimate by less than 6.7%. This observation also holds more generally across simulations:

**Figure 2.17:** We show the projected utility normalized by the utility nodes observed once they deployed S*BGP for different sets of early adopters when $\theta = 0$. In most cases ASes projected utilities that are within a few percentage points of what they actually received in the next round.

### 2.9.2 Computing utility locally

Because we lack information about interdomain traffic flows in the Internet, our model uses weighted counts of the subtrees of ASes routing through ISP $n$ as a stand-in for traffic volumes, and thus ISP utility. While computing these subtrees in our model requires *global* information that would be unavailable to the average ISP (*e.g.,* the state $S$, the AS graph topology, routing policies), in practice, an ISP can just compute its utility by locally observing traffic flows through its network.

**Computing projected utility.** Computing projected utility $u_n(\neg S_n, S_{-n})$ in practice is significantly more complex. While projected utility gives an accurate estimate of actual utility when it is computed using global information, ISPs may inaccurately estimate their projected utility when using only local information. Our model can accommodate these inaccuracies by rolling them into the deployment threshold $\theta$. (That is, if projected utility is off by a factor of $\pm\varepsilon$, model this with threshold $\theta \pm \varepsilon$.) Thus, while our approach was to sweep through a common value of $\theta$ for every ISP (Section 2.7.2), extensions might capture inaccurate estimates of projected utility by randomizing $\theta$, or even by systematically modeling an ISP's estimation process to obtain a measure for how it impacts $\theta$.

**Practical mechanisms for projecting future traffic patterns.** Because S*BGP deployment can impact route selection, it is crucial to develop mechanisms that allow ISPs predict how security will impact traffic patterns through it's network. Moreover, if ISPs could use such mechanisms to estimate projected utility, they would also be an important driver for S*BGP deployment. For example, an ISP might set up a router that listens to S*BGP messages from neighboring ASes, and then use these message to predict how becoming secure might impact its neighbors' route selections. A more sophisticated mechanism could use extended "shadow configurations" with neighboring ASes [4] to gain visibility into how traffic flows might change.

### 2.9.3 Alternate routing policies and actions

**Routing policies.** Because our model of ISP utility depends on traffic volumes (Section 2.4.3), we need a model for how traffic flows in the Internet. In practice, traffic flow is determined by the local routing policies used by each AS, which are arbitrary and not publicly known. Thus, we use a standard model of routing policies (Appendix A) based on business relationship and path length [11, 44].

Routing policies are likely to impact our results by determining (a) AS path lengths (longer AS paths mean it is harder to secure routes), and (b) tiebreak set size (Section 2.7.6). For example, we speculate that considering shortest path routing policy would lead to overly optimistic results; shortest-path routing certainly leads to shorter AS paths, and possibly also to larger tiebreak sets. On the other hand, if a large fraction of multihomed ASes always use one provider as primary and the other as backup (irrespective of the AS path lengths *etc.*) then our current analysis is likely to be overly optimistic. (Of course, modeling this is difficult given a dearth of empirical data on backup paths).

**Choosing routing policies.** An AS might cleverly choose its routing policies to maximize utility. However, the following suggests that this is intractable:

**Theorem 2.9.1.** *When all other ASes' routing policies are as in Appendix A, it is NP hard for any AS n to find the routing policy that maximizes its utility (in both the incoming and outgoing utility models). Moreover, approximating the optimal routing policy within any constant factor is also NP-hard. [41]*

The proof (in [42]) shows that this is NP-hard even if $n$ has a single route to the destination, and must only choose the set of neighbors to which it announces the route. (Thus, the problem is tractable when the node's neighbors set is of constant size.)

**Per-link S*BGP deployment.** An ISP might be able to optimize its utility by turning ON S*BGP on a *per link* basis, *i.e.,* with only a subset of its neighbors. (For instance, a second look at Figure 2.16 suggests that AS 4775 improve his utility by turning off S*BGP on the link

to his provider AS 2914.) Once again, this is intractable when an ISP has a large number of neighbors (Proof in [42] ):

**Theorem 2.9.2.** *Given an AS graph and state S, it is NP-hard to choose the set of neighbors for which ISP n should deploy S*BGP so as to maximize its incoming utility. Moreover, approximating the optimum within any constant factor is also NP hard [41].*

**Lying and cheating.** While it is well known that an AS can increase the amount of traffic it transits by manipulating its BGP messages [17], we avoided this issue because our focus is on technology adoption by economically-motivated ASes, not BGP manipulations by malicious or misconfigured ASes.

### 2.9.4 Other extensions to our model

**Static AS graph.** Our model of interdomain routing assumes that the AS graph does not change. Because the time-scale of the deployment process can be quite large (*e.g.,* years), extensions to our model might also model the evolution of the AS graph with time, and possible incorporate issues like the addition of new edges if secure ASes manage to sign up new customers.

**Mapping revenue to traffic volume.** Our model of ISP utility is based on the idea that revenue is related to the *total volume* of customer traffic the ISP transits. In practice, ISPs may use a variety of pricing policies, *e.g.,* by volume, flat rates based on discrete units of capacity. Thus, extensions might consider collecting empirical data on pricing policies to more accurately map revenue to traffic volumes.

## 2.10 Related work

**Social networks.** The diffusion of new technologies in social networks has been well studied in economics and game theory (*e.g.,* [66, 87] and references therein). The idea that players

will myopically best-respond if their utility exceeds a threshold is standard in this literature (*cf.,* our update rule (2.3)). However, in a social network, a player's utility depends only on its immediate *neighbors*, while in our setting it depends on the set of secure *paths*. Thus, while [66] finds approximation algorithms for choosing an optimal set of early adopters, this is NP-hard in our setting (Theorem 2.7.1).

**Protocol adoption in the Internet.** The idea that competition over customer traffic can drive technology adoption in the Internet has appeared in many places in the literature [24, 101]. Ratnasamy *et al.* [101] suggest using competition for customer traffic to drive protocol deployment (*e.g.,* IPv6) at ISPs by creating new mechanisms for directing traffic to ASes with IPv6. Leveraging competition is much simpler with S*BGP, since it directly influences routing decisions without requiring adoption of new mechanisms.

Multiple studies [58, 60, 108] consider the role of converters (*e.g.,* IPv4-IPv6 gateways) on protocol deployment. While S*BGP must certainly be backwards compatible with BGP, the fact that security guarantees only hold for fully-secure paths (Section 2.3.1) means that there is no reason to convert BGP messages to S*BGP messages. Thus, we do not expect converters to drive S*BGP deployment.

Guérin and Hosanagar [48] consider the role of quality differentials in IPv6 migration. They observe that if content delivery quality is higher for native IPv6 services that content providers will have incentive to deploy IPv6 for clients running the newer protocol. While this work sets the bar for IPv6 performance, achieving the required performance increases for migration remains an engineering task. In contrast, the incentives we observe for S*BGP adoption are inherent in BGP's influence on route selection.

**S*BGP adoption.** Perhaps most relevant is Chang *et al.*'s comparative study on the adoptability of secure interdomain routing protocols [19]. Like [19], we also consider how early adopters create local incentives for other ASes to deploy S*BGP. However, our study focuses on how S*BGP deployment can be driven by (a) simplex S*BGP deployment at stubs, and (b) the requirement that security plays a role in routing decisions. Furthermore, in [19] ISP

utility depends on the security benefits offered by the partially-deployed protocol. Thus, the utility function in [19] depends on possible attacker strategies (*i.e.,* path shortening attacks) and attacker location (*i.e.,* random, or biased towards small ISPs). In contrast, our model of utility is based solely on economics (*i.e.,* customer traffic transited). Thus, we show that global S*BGP deployment is possible even if ISPs' local deployment decisions are *not* driven by security concerns. Also, complementary to our work is [10]'s forward-looking proposal that argues that extra mechanisms (*e.g.,* secure data-plane monitoring) can be added to S*BGP to get around the problem of partially-secure paths (Appendix B). Finally, we note both our work and [10, 19] find that ensuring that Tier 1 ASes deploy S*BGP is crucial, a fact that is not surprising in light of the highly-skewed degree distribution of the AS graph.

## 2.11 Summary

Our results indicate that there is hope for S*BGP deployment. We have argued for (1) simplex S*BGP to secure stubs, (2) convincing but a small, but influential, set of ASes to be early adopters of S*BGP, and (3) ensuring that S*BGP influences traffic by requiring ASes to (at minimum) break ties between equally-good paths based on security.

We have shown that, if deployment cost $\theta$ is low, our proposal can successfully transition a majority of ASes to S*BGP. The transition is driven by market pressure created when ISPs deploy S*BGP in order draw revenue-generating traffic into their networks. We also pointed out unexplored challenges that result from S*BGP's influence of route selection (*e.g.,* ISPs may have incentives to disable S*BGP).

**Recommendations.** We hope that this work motivates the standardization and research communities to devote their efforts along three key lines. First, effort should be spent to engineer a lightweight simplex S*BGP. Second, with security impacting route selection, ISPs will need tools to forecast how S*BGP deployment will impact traffic patterns (*e.g.,* using "shadow configurations", inspired by [4], with cooperative neighboring ASes) so they can provision their

networks appropriately. Finally, our results suggest that S*BGP and BGP will coexist in the long term. Thus, effort should be devoted to ensure that S*BGP and BGP can coexist without introducing new vulnerabilities into the interdomain routing system.

# Chapter 3

# Empirical Study of Failures in a Data Center Network

## 3.1 Introduction

Demand for dynamic scaling and benefits from economies of scale are driving the creation of mega data centers to host a broad range of services such as Web search, e-commerce, storage backup, video streaming, high-performance computing, and data analytics. To host these applications, data center networks need to be scalable, efficient, fault tolerant, and easy-to-manage. Recognizing this need, the research community has proposed several architectures to improve scalability and performance of data center networks [2, 3, 46, 49, 50, 68, 88]. However, the issue of reliability has remained unaddressed, mainly due to a dearth of available empirical data on failures in these networks.

In this chapter, we study data center network reliability by analyzing network error logs collected for over a year from thousands of network devices across tens of geographically distributed data centers. Our goals for this analysis are two-fold. First, we seek to characterize network failure patterns in data centers and understand overall reliability of the network. Second, we want to leverage lessons learned from this study to guide the design of future data

center networks.

Motivated by issues encountered by network operators, we study network reliability along three dimensions:

- **Characterizing the most failure prone network elements.** To achieve high availability amidst multiple failure sources such as hardware, software, and human errors, operators need to focus on fixing the most unreliable devices and links in the network. To this end, we characterize failures to identify network elements with high impact on network reliability e.g., those that fail with high frequency or that incur a high downtime.

- **Estimating the impact of failures.** Given limited resources at hand, operators need to prioritize *severe* incidents for troubleshooting based on their impact to end-users and applications. In general, however, it is difficult to accurately quantify a failure's impact from error logs, and annotations provided by operators in trouble tickets tend to be ambiguous. Thus, as a first step, we estimate failure impact by correlating event logs with recent network traffic observed on links involved in the event. Note that logged events do not necessarily result in a service outage because of failure-mitigation techniques such as network redundancy [1] and replication of compute and data [35, 117], typically deployed in data centers.

- **Analyzing the effectiveness of network redundancy.** Ideally, operators want to mask all failures before applications experience any disruption. Current data center networks typically provide 1:1 redundancy to allow traffic to flow along an alternate route when a device or link becomes unavailable [1]. However, this redundancy comes at a high cost—both monetary expenses and management overheads—to maintain a large number of network devices and links in the multi-rooted tree topology. To analyze its effectiveness, we compare traffic on a per-link basis during failure events to traffic across all links in the network redundancy group where the failure occurred.

For our study, we leverage multiple monitoring tools put in place by our network operators. We utilize data sources that provide both a static view (e.g., router configuration files, device

procurement data) and a dynamic view (e.g., SNMP polling, syslog, trouble tickets) of the network. Analyzing these data sources, however, poses several challenges. First, since these logs track low level network events, they do not necessarily imply application performance impact or service outage. Second, we need to separate failures that potentially impact network connectivity from high volume and often noisy network logs e.g., warnings and error messages even when the device is functional. Finally, analyzing the effectiveness of network redundancy requires correlating multiple data sources across redundant devices and links. Through our analysis, we aim to address these challenges to characterize network failures, estimate the failure impact, and analyze the effectiveness of network redundancy in data centers.

### 3.1.1 Key observations

We make several key observations from our study:

- **Data center networks are reliable.** We find that overall the data center network exhibits high reliability with more than four 9's of availability for about 80% of the links and for about 60% of the devices in the network (Section 3.4.5).

- **Low-cost, commodity switches are highly reliable.** We find that Top of Rack switches (ToRs) and aggregation switches exhibit the highest reliability in the network with failure rates of about 5% and 10%, respectively. This observation supports network design proposals that aim to build data center networks using low cost, commodity switches [3, 46, 88] (Section 3.4.3).

- **Load balancers experience a high number of software faults.** We observe 1 in 5 load balancers exhibit a failure (Section 3.4.3) and that they experience many transient software faults (Section 3.4.7).

- **Failures potentially cause loss of a large number of small packets.** By correlating network traffic with link failure events, we estimate the amount of packets and data lost during

failures. We find that most failures lose a large number of packets relative to the number of lost bytes (Section 3.5), likely due to loss of protocol-specific keep alive messages or ACKs.

- **Network redundancy helps, but it is not entirely effective.** Ideally, network redundancy should completely mask all failures from applications. However, we observe that network redundancy is only able to reduce the median impact of failures (in terms of lost bytes or packets) by up to 40% (Section 3.5.1).

**Limitations.** As with any large-scale empirical study, our results are subject to several limitations. First, the best-effort nature of failure reporting may lead to missed events or multiply-logged events. While we perform data cleaning (Section 3) to filter the noise, some events may still be lost due to software faults (e.g., firmware errors) or disconnections (e.g., under correlated failures). Second, human bias may arise in failure annotations (e.g., root cause). This concern is alleviated to an extent by verification with operators, and scale and diversity of our network logs. Third, network errors do not always impact network traffic or service availability, due to several factors such as in-built redundancy at network, data, and application layers. Thus, our failure rates should not be interpreted as impacting applications. Overall, we hope that this study contributes to a deeper understanding of network reliability in data centers.

**Chapter organization.** The rest of this chapter is organized as follows. Section 3.2 presents our network architecture and workload characteristics. Data sources and methodology are described in Section 3.3. We characterize failures over a year within our data centers in Section 3.4. We estimate the impact of failures on applications and the effectiveness of network redundancy in masking them in Section 3.5. Finally we discuss implications of our study for future data center networks in Section 3.6. We present related work in Section 3.7 and conclude in Section 3.8.

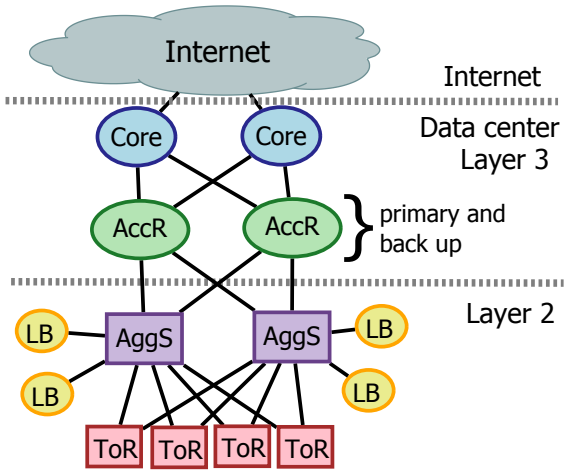



Figure 3.1: A conventional data center network architecture adapted from figure by Cisco [46]. The device naming convention is summarized in Table 3.1.

## 3.2 Background

Our study focuses on characterizing failure events within our organization's set of data centers. We next give an overview of data center networks and workload characteristics.

### 3.2.1 Data center network architecture

Figure 3.1 illustrates an example of a partial data center network architecture [1]. In the network, rack-mounted servers are connected (or dual-homed) to a Top of Rack (ToR) switch usually via a 1 Gbps link. The ToR is in turn connected to a primary and back up aggregation switch (AggS) for redundancy. Each redundant pair of AggS aggregates traffic from tens of ToRs which is then forwarded to the access routers (AccR). The access routers aggregate traffic from up to several thousand servers and route it to core routers that connect to the rest of the data center network and Internet.

All links in our data centers use Ethernet as the link layer protocol and physical connections are a mix of copper and fiber cables. The servers are partitioned into virtual LANs (VLANs) to limit overheads (e.g., ARP broadcasts, packet flooding) and to isolate different applications

Table 3.1: Summary of device abbreviations

| Type | Devices | Description |
|---|---|---|
| AggS | AggS-1, AggS-2 | Aggregation switches |
| LB | LB-1, LB-2, LB-3 | Load balancers |
| ToR | ToR-1, ToR-2, ToR-3 | Top of Rack switches |
| AccR | - | Access routers |
| Core | - | Core routers |

hosted in the network. At each layer of the data center network topology, with the exception of a subset of ToRs, 1:1 redundancy is built into the network topology to mitigate failures. As part of our study, we evaluate the effectiveness of redundancy in masking failures when one (or more) components fail, and analyze how the tree topology affects failure characteristics e.g., correlated failures.

In addition to routers and switches, our network contains many middle boxes such as load balancers and firewalls. Redundant pairs of load balancers (LBs) connect to each aggregation switch and perform mapping between static IP addresses (exposed to clients through DNS) and dynamic IP addresses of the servers that process user requests. Some applications require programming the load balancers and upgrading their software and configuration to support different functionalities.

**Network composition.** The device-level breakdown of our network is as follows. ToRs are the most prevalent device type in our network comprising approximately 75% of devices. LBs are the next most prevalent at 10% of devices. The remaining 15% of devices are AggS, Core and AccR. We observe the effects of prevalent ToRs in Section 3.4.4, where despite being highly reliable, ToRs account for a large amount of downtime. LBs on the other hand account for few devices but are extremely failure prone, making them a leading contributor of failures (Section 3.4.4).

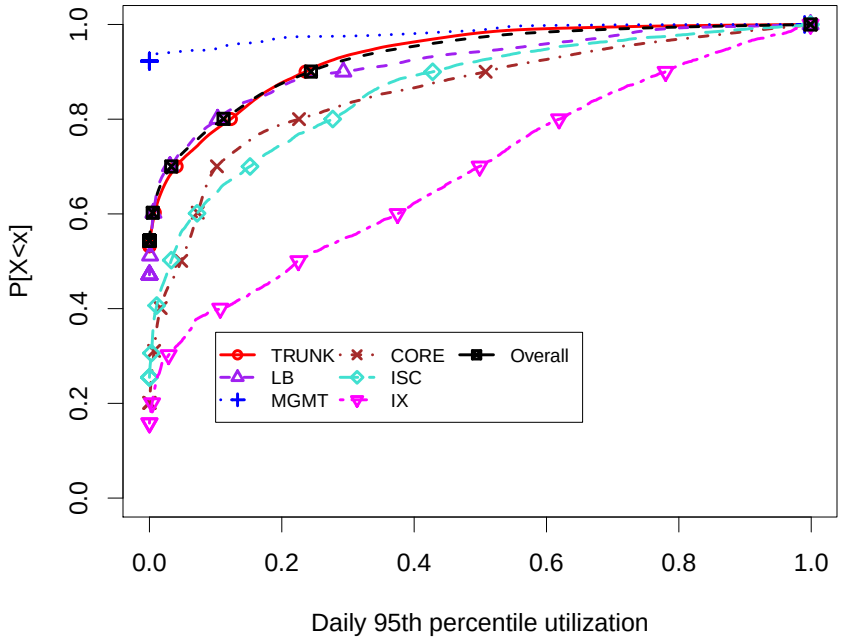



Figure 3.2: The daily 95th percentile utilization as computed using five-minute traffic averages (in bytes)

### 3.2.2 Data center workload characteristics

Our network is used in the delivery of many online applications. As a result, it is subject to many well known properties of data center traffic; in particular the prevalence of a large volume of short-lived latency-sensitive "mice" flows and a few long-lived throughput-sensitive "elephant" flows that make up the majority of bytes transferred in the network. These properties have also been observed by others [5, 12, 46].

**Network utilization.** Figure 3.2 shows the daily 95th percentile utilization as computed using five-minute traffic averages (in bytes). We divide links into six categories based on their role in the network (summarized in Table 3.2). TRUNK and LB links which reside lower in the network topology are least utilized with 90% of TRUNK links observing less than 24% utilization. Links higher in the topology such as CORE links observe higher utilization with 90% of CORE links observing less than 51% utilization. Finally, links that connect data centers

Table 3.2: Summary of link types

| Type | Description |
|---|---|
| TRUNK | connect ToRs to AggS and AggS to AccR |
| LB | connect load balancers to AggS |
| MGMT | management interfaces |
| CORE | connect routers (AccR, Core) in the network core |
| ISC | connect primary and back up switches/routers |
| IX | connect data centers (wide area network) |

(IX) are the most utilized with 35% observing utilization of more than 45%. Similar to prior studies of data center network traffic [12], we observe higher utilization at upper layers of the topology as a result of aggregation and high bandwidth oversubscription [46]. Note that since the traffic measurement is at a granularity of five minute averages, it is likely to smooth the effect of short-lived traffic spikes on link utilization.

## 3.3 Methodology and data sets

The network operators collect data from multiple sources to track the performance and health of the network. We leverage these existing data sets in our analysis of network failures. In this section, we first describe the data sets and then the steps we took to extract failures of network elements.

### 3.3.1 Existing data sets

The data sets used in our analysis are a subset of what is collected by the network operators. We describe these data sets in turn:

- **Network event logs (SNMP/syslog).** We consider logs derived from syslog, SNMP traps and polling, collected by our network operators. The operators filter the logs to reduce the

number of transient events and produce a smaller set of *actionable* events. One of the filtering rules excludes link failures reported by servers connected to ToRs as these links are extremely prone to spurious port flapping (e.g., more than 100,000 events per hour across the network). Of the filtered events, 90% are assigned to NOC tickets that must be investigated for troubleshooting. These event logs contain information about what type of network element experienced the event, what type of event it was, a small amount of descriptive text (machine generated) and an ID number for any NOC tickets relating to the event. For this study we analyzed a year's worth of events from October 2009 to September 2010.

- **NOC Tickets.** To track the resolution of issues, the operators employ a ticketing system. Tickets contain information about when and how events were discovered as well as when they were resolved. Additional descriptive tags are applied to tickets describing the cause of the problem, if any specific device was at fault, as well as a "diary" logging steps taken by the operators as they worked to resolve the issue.

- **Network traffic data.** Data transferred on network interfaces is logged using SNMP polling. This data includes five minute averages of bytes and packets into and out of each network interface.

- **Network topology data.** Given the sensitive nature of network topology and device procurement data, we used a static snapshot of our network encompassing thousands of devices and tens of thousands of interfaces spread across tens of data centers.

### 3.3.2 Defining and identifying failures

When studying failures, it is important to understand what types of logged events constitute a "failure". Previous studies have looked at failures as defined by pre-existing measurement frameworks such as syslog messages [115], OSPF [109, 119] or IS-IS listeners [82]. These approaches benefit from a consistent definition of failure, but tend to be ambiguous when trying to determine whether a failure had impact or not. Syslog messages in particular can be spurious

with network devices sending multiple notifications even though a link *is* operational. For multiple devices, we observed this type of behavior after the device was initially deployed and the router software went into an erroneous state. For some devices, this effect was severe, with one device sending 250 syslog "link down" events *per hour* for 2.5 months (with no impact on applications) before it was noticed and mitigated.

We mine network event logs collected over a year to extract events relating to device and link failures. Initially, we extract all logged "down" events for network devices and links. This leads us to define two types of failures:

**Link failures:** A link failure occurs when the connection between two devices (on specific interfaces) is down. These events are detected by SNMP monitoring on interface state of devices.

**Device failures:** A device failure occurs when the device is not functioning for routing/forwarding traffic. These events can be caused by a variety of factors such as a device being powered down for maintenance or crashing due to hardware errors.

We refer to each logged event as a "failure" to understand the occurrence of low level failure events in our network. As a result, we may observe multiple component notifications related to a single high level failure or a correlated event e.g., a AggS failure resulting in down events for its incident ToR links. We also correlate failure events with network traffic logs to filter *failures with impact* that potentially result in loss of traffic (Section 3.3.4); we leave analyzing application performance and availability under network failures, to future work.

### 3.3.3 Cleaning the data

We observed two key inconsistencies in the network event logs stemming from redundant monitoring servers being deployed. First, a single element (link or device) may experience multiple "down" events simultaneously. Second, an element may experience another down event before the previous down event has been resolved. We perform two passes of cleaning over the data to resolve these inconsistencies. First, multiple down events on the same element that start at

the same time are grouped together. If they do not end at the same time, the earlier of their end times is taken. In the case of down events that occur for an element that is already down, we group these events together, taking the earliest down time of the events in the group. For failures that are grouped in this way we take the earliest end time for the failure. We take the earliest failure end times because of occasional instances where events were not marked as resolved until long after their apparent resolution.

### 3.3.4 Identifying failures with impact

As previously stated, one of our goals is to identify failures that potentially impact end-users and applications. Since we did not have access to application monitoring logs, we cannot precisely quantify application impact such as throughput loss or increased response times. Therefore, we instead *estimate* the impact of failures on network traffic.

To estimate traffic impact, we correlate each link failure with traffic observed on the link in the recent past before the time of failure. We leverage five minute traffic averages for each link that failed and compare the median traffic on the link in the time window preceding the failure event and the median traffic during the failure event. We say a failure has impacted network traffic if the median traffic during the failure is less than the traffic before the failure. Since many of the failures we observe have short durations (less than ten minutes) and our polling interval is five minutes, we do not require that traffic on the link go down to zero during the failure. We analyze the failure impact in detail in Section 3.5.

Table 3.3 summarizes the impact of link failures we observe. We separate links that were transferring no data before the failure into two categories, "inactive" (no data before or during failure) and "provisioning" (no data before, some data transferred during failure). (Note that these categories are inferred based only on traffic observations.) The majority of failures we observe are on links that are inactive (e.g., a new device being deployed), followed by link failures with impact. We also observe a significant fraction of link failure notifications where no impact was observed (e.g., devices experiencing software errors at the end of the deployment

Table 3.3: Summary of logged link events

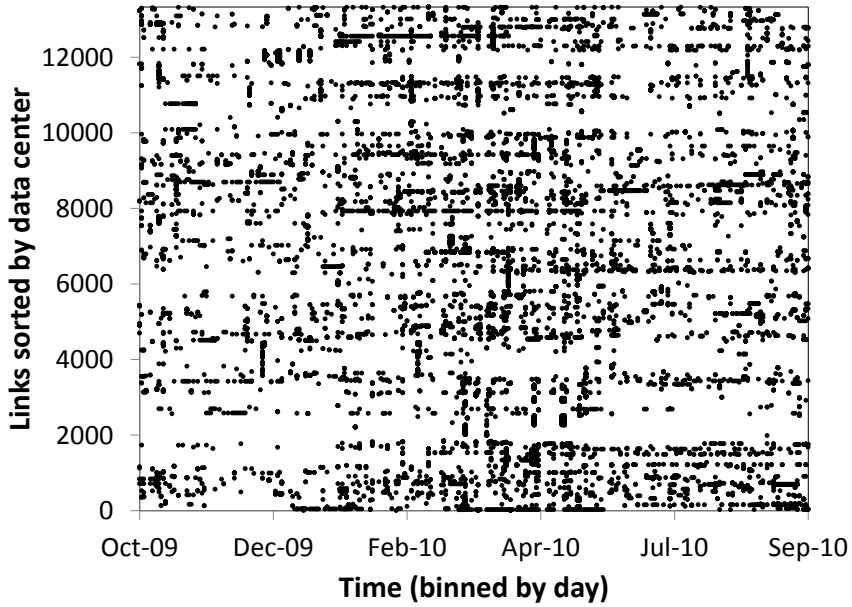
| Category | Percent | Events |
|---|---|---|
| All | 100.0 | 46,676 |
| Inactive | 41.2 | 19,253 |
| Provisioning | 1.0 | 477 |
| No impact | 17.9 | 8,339 |
| Impact | 28.6 | 13,330 |
| No traffic data | 11.3 | 5,277 |

process).

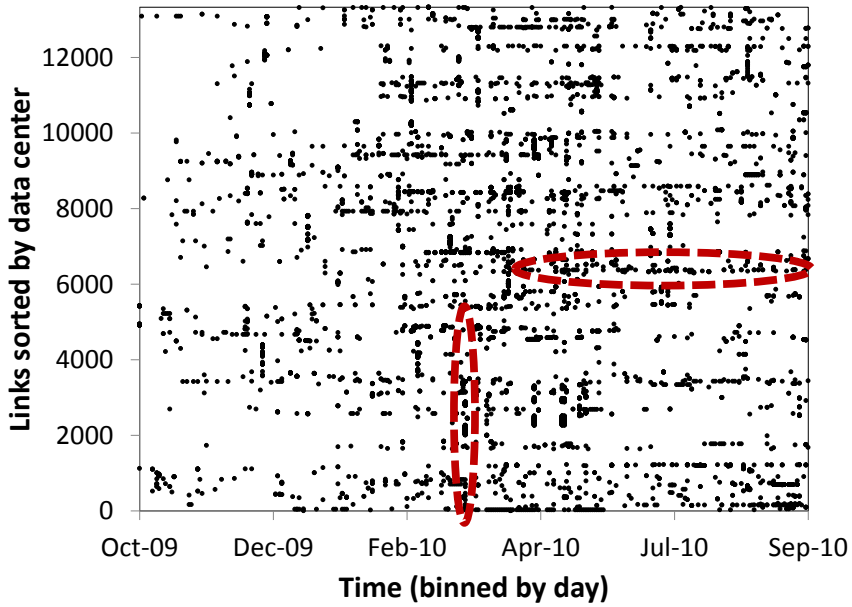
For link failures, verifying that the failure caused impact to network traffic enables us to eliminate many spurious notifications from our analysis and focus on events that had a measurable impact on network traffic. However, since we do not have application level monitoring, we are unable to determine if these events impacted applications or if there were faults that impacted applications that we did not observe.

For device failures, we perform additional steps to filter spurious failure messages (e.g., down messages caused by software bugs when the device is in fact up). If a device is down, neighboring devices connected to it will observe failures on inter-connecting links. For each device down notification, we verify that at least one link failure with impact has been noted for links incident on the device within a time window of five minutes. This simple sanity check significantly reduces the number of device failures we observe. Note that if the neighbors of a device fail simultaneously e.g., due to a correlated failure, we may not observe a link-down message for that device.

For the remainder of our analysis, unless stated otherwise, we consider only failure events that impacted network traffic.




(a) All failures




(b) Failures with impact

Figure 3.3: Overview of all link failures (a) and link failures with impact on network traffic (b) on links with at least one failure

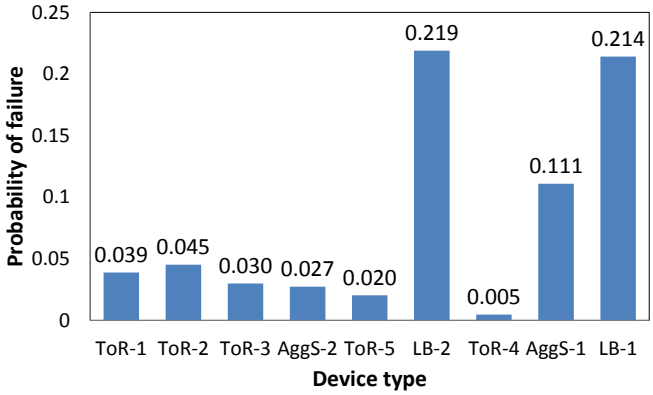



Figure 3.4: Probability of device failure in one year for device types with population size of at least 300

## 3.4 Failure Analysis

### 3.4.1 Failure event panorama

Figure 3.3 illustrates how failures are distributed across our measurement period and across data centers in our network. It shows plots for links that experience at least one failure, both for all failures and those with potential impact; the y-axis is sorted by data center and the x-axis is binned by day. Each point indicates that the link (y) experienced at least one failure on a given day (x).

**All failures vs. failures with impact.** We first compare the view of all failures (Figure 3.3 (a)) to failures having impact (Figure 3.3 (b)). Links that experience failures impacting network traffic are only about one third of the population of links that experience failures. We do not observe significant widespread failures in either plot, with failures tending to cluster within data centers, or even on interfaces of a single device.

**Widespread failures:** Vertical bands indicate failures that were spatially widespread. Upon further investigation, we find that these tend to be related to software upgrades. For example, the vertical band on March 25 was due to an upgrade of load balancer software that spanned multiple data centers. In the case of planned upgrades, the network operators are able to take

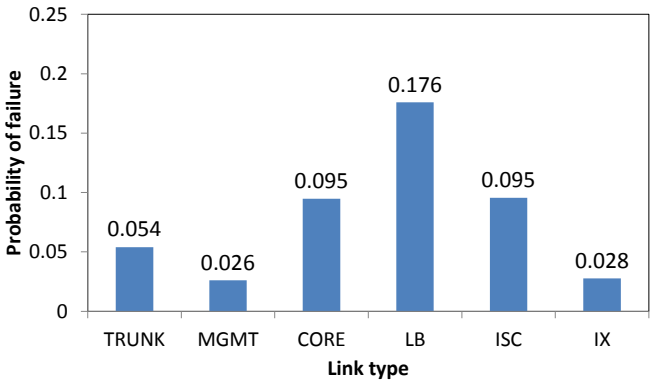



Figure 3.5: Probability of a failure impacting network traffic in one year for interface types with population size of at least 500

Table 3.4: Failures per time unit

| Failures per day: | Mean | Median | 95% | COV |
|---|---|---|---|---|
| Devices | 5.2 | 3.0 | 14.7 | 1.3 |
| Links | 40.8 | 18.5 | 136.0 | 1.9 |

precautions so that the disruptions do not impact applications.

**Long-lived failures:** Horizontal bands indicate link failures on a common link or device over time. These tend to be caused by problems such as firmware or device unreliability (wider bands indicate multiple interfaces failed on a single device). We observe horizontal bands with regular spacing between link failure events. In one case, these events occurred weekly and were investigated in independent NOC tickets. As a result of the time lag, the operators did not correlate these events and dismissed each notification as spurious since they occurred in isolation and did not impact performance. This underscores the importance of network health monitoring tools that track failures over time and alert operators to spatio-temporal patterns which may not be easily recognized using local views alone.

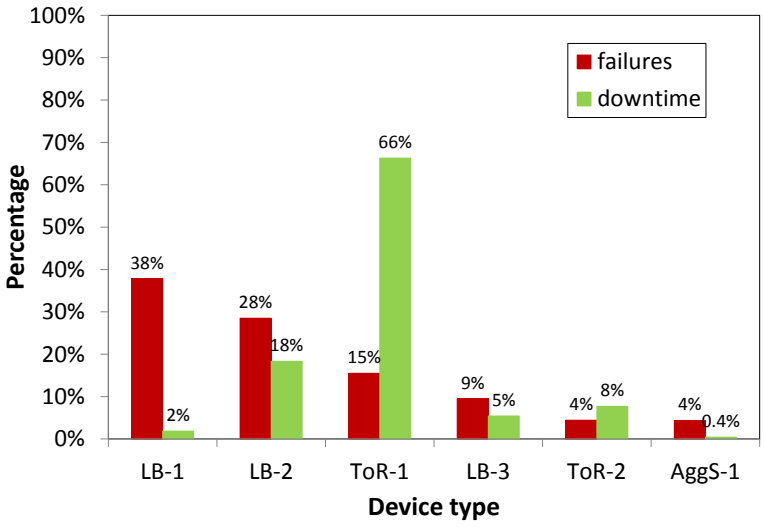



Figure 3.6: Percent of failures and downtime per device type

### 3.4.2 Daily volume of failures

We now consider the daily frequency of failures of devices and links. Table 3.4 summarizes the occurrences of link and device failures per day during our measurement period. Links experience about an order of magnitude more failures than devices. On a daily basis, device and link failures occur with high variability, having COV of 1.3 and 1.9, respectively. (COV $>$ 1 is considered high variability.)

**Link failures are variable and bursty.** Link failures exhibit high variability in their rate of occurrence. We observed bursts of link failures caused by protocol issues (e.g., UDLD [23]) and device issues (e.g., power cycling load balancers).

**Device failures are usually caused by maintenance.** While device failures are less frequent than link failures, they also occur in bursts at the daily level. We discovered that periods with high frequency of device failures are caused by large scale maintenance (e.g., on all ToRs connected to a common AggS).

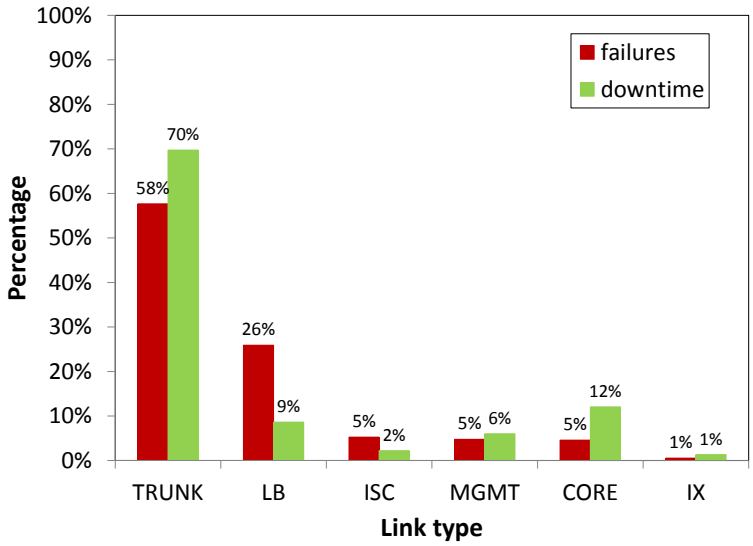



Figure 3.7: Percent of failures and downtime per link type

### 3.4.3 Probability of failure

We next consider the probability of failure for network elements. This value is computed by dividing the number of devices of a given type that observe failures by the total device population of the given type. This gives the probability of failure in our one year measurement period. We observe (Figure 3.4) that in terms of overall reliability, ToRs have the lowest failure rates whereas LBs have the highest failure rate. (Tables 3.1 and 3.2 summarize the abbreviated link and device names.)

**Load balancers have the highest failure probability.** Figure 3.4 shows the failure probability for device types with population size of at least 300. In terms of overall failure probability, load balancers (LB-1, LB-2) are the least reliable with a 1 in 5 chance of experiencing failure. Since our definition of failure can include incidents where devices are power cycled during *planned* maintenance, we emphasize here that not all of these failures are unexpected. Our analysis of load balancer logs revealed several causes of these transient problems such as software bugs, configuration errors, and hardware faults related to ASIC and memory.

**ToRs have low failure rates.** ToRs have among the lowest failure rate across all devices.

This observation suggests that low-cost, commodity switches are not necessarily less reliable than their expensive, higher capacity counterparts and bodes well for data center networking proposals that focus on using commodity switches to build flat data center networks [3, 46, 88].

We next turn our attention to the probability of link failures at different layers in our network topology.

**Load balancer links have the highest rate of logged failures.** Figure 3.5 shows the failure probability for interface types with a population size of at least 500. Similar to our observation with devices, links forwarding load balancer traffic are most likely to experience failures (e.g., as a result of failures on LB devices).

Links higher in the network topology (CORE) and links connecting primary and back up of the same device (ISC) are the second most likely to fail, each with an almost 1 in 10 chance of failure. However, these events are more likely to be masked by network redundancy (Section 3.5.2). In contrast, links lower in the topology (TRUNK) only have about a 5% failure rate.

**Management and inter-data center links have lowest failure rate.** Links connecting data centers (IX) and for managing devices have high reliability with fewer than 3% of each of these link types failing. This observation is important because these links are the most utilized and least utilized, respectively (cf. Figure 3.2). Links connecting data centers are critical to our network and hence back up links are maintained to ensure that failure of a subset of links does not impact the end-to-end performance.

### 3.4.4 Aggregate impact of failures

In the previous section, we considered the reliability of individual links and devices. We next turn our attention to the aggregate impact of each population in terms of total number of failure events and total downtime. Figure 3.6 presents the percentage of failures and downtime for the different device types.

**Load balancers have the most failures but ToRs have the most downtime.** LBs have

Table 3.5: Summary of failures per device (for devices that experience at least one failure)

| Device type | Mean | Median | 99% | COV |
|---|---|---|---|---|
| LB-1 | 11.4 | 1.0 | 426.0 | 5.1 |
| LB-2 | 4.0 | 1.0 | 189.0 | 5.1 |
| ToR-1 | 1.2 | 1.0 | 4.0 | 0.7 |
| LB-3 | 3.0 | 1.0 | 15.0 | 1.1 |
| ToR-2 | 1.1 | 1.0 | 5.0 | 0.5 |
| AggS-1 | 1.7 | 1.0 | 23.0 | 1.7 |
| Overall | 2.5 | 1.0 | 11.0 | 6.8 |

the highest number of failures of any device type. Of our top six devices in terms of failures, half are load balancers. However, LBs do not experience the most downtime which is dominated instead by ToRs. This is counterintuitive since, as we have seen, ToRs have very low failure probabilities. There are three factors at play here: (1) LBs are subject to more frequent software faults and upgrades (Section 3.4.7) (2) ToRs are the most prevalent device type in the network (Section 3.2.1), increasing their aggregate effect on failure events and downtime (3) ToRs are not a high priority component for repair because of in-built failover techniques, such as replicating data and compute across multiple racks, that aim to maintain high service availability despite failures.

We next analyze the aggregate number of failures and downtime for network links. Figure 3.7 shows the normalized number of failures and downtime for the six most failure prone link types.

**Load balancer links experience many failure events but relatively small downtime.** Load balancer links experience the second highest number of failures, followed by ISC, MGMT and CORE links which all experience approximately 5% of failures. Note that despite LB links being second most frequent in terms of number of failures, they exhibit less downtime than CORE links (which, in contrast, experience 5X fewer failures). This result suggests that

failures for LBs are short-lived and intermittent caused by transient software bugs, rather than more severe hardware issues. We investigate these issues in detail in Section 3.4.7.

We observe that the total number of failures and downtime are dominated by LBs and ToRs, respectively. We next consider how many failures each element experiences. Table 3.5 shows the mean, median, 99th percentile and COV for the number of failures observed per device over a year (for devices that experience at least one failure).

**Load balancer failures dominated by few failure prone devices.** We observe that individual LBs experience a highly variable number of failures with a few outlier LB devices experiencing more than 400 failures. ToRs, on the other hand, experience little variability in terms of the number of failures with most ToRs experiencing between 1 and 4 failures. We make similar observations for links, where LB links experience very high variability relative to others.

### 3.4.5 Properties of failures

We next consider the properties of failures for network element types that experienced the highest number of events.

#### Time to repair

This section considers the time to repair (or duration) for failures, computed as the time between a down notification for a network element and when it is reported as being back online. It is not always the case that an operator had to intervene to resolve the failure. In particular, for short duration failures, it is likely that the fault was resolved automatically (e.g., root guard in the spanning tree protocol can temporarily disable a port [22]). In the case of link failures, our SNMP polling interval of four minutes results in a grouping of durations around four minutes indicating that many link failures are resolved automatically without operator intervention. Finally, for long-lived failures, the failure durations may be skewed by when the NOC tickets were closed by network operators. For example, some incident tickets may not be termed as 'resolved' even if normal operation has been restored, until a hardware replacement arrives in

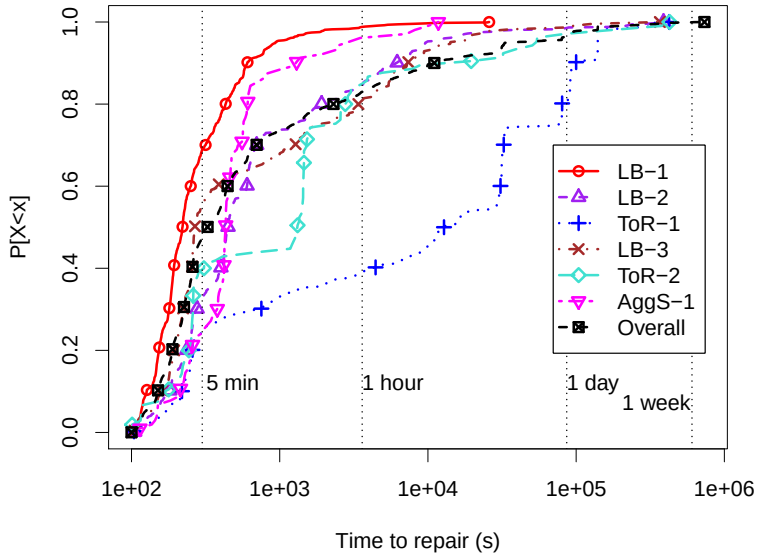



Figure 3.8: Time to repair for devices

stock.

**Load balancers experience short-lived failures.** We first look at the duration of device failures. Figure 3.8 shows the CDF of time to repair for device types with the most failures. We observe that LB-1 and LB-3 load balancers experience the shortest failures with median time to repair of 3.7 and 4.5 minutes, respectively, indicating that most of their faults are short-lived.

**ToRs experience correlated failures.** When considering time to repair for devices, we observe a correlated failure pattern for ToRs. Specifically, these devices tend to have several discrete "steps" in the CDF of their failure durations. These steps correspond to spikes in specific duration values. On analyzing the failure logs, we find that these spikes are due to groups of ToRs that connect to the same (or pair of) AggS going down at the same time (e.g., due to maintenance or AggS failure).

**Inter-data center links take the longest to repair.** Figure 3.9 shows the distribution of time to repair for different link types. The majority of link failures are resolved within five minutes, with the exception of links between data centers which take longer to repair. This is because

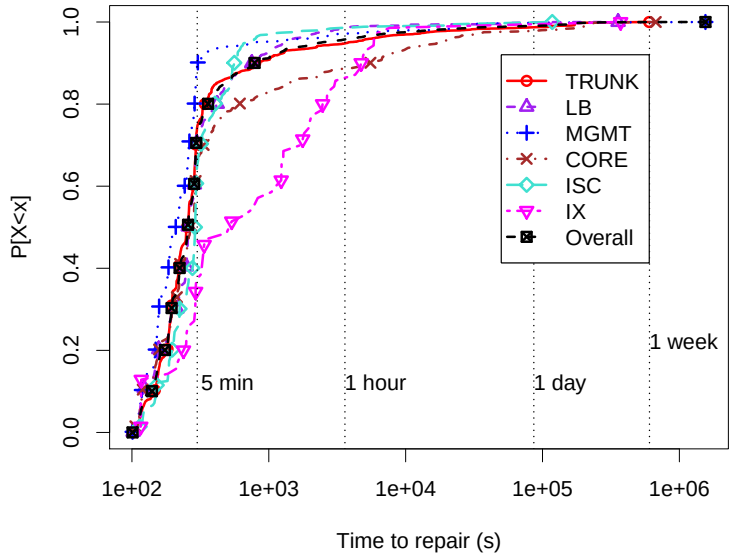



Figure 3.9: Time to repair for links

links between data centers require coordination between technicians in multiple locations to identify and resolve faults as well as additional time to repair cables that may be in remote locations.

### Time between failures

We next consider the time between failure events. Since time between failure requires a network element to have observed more than a single failure event, this metric is most relevant to elements that are failure prone. Specifically, note that more than half of all elements have only a single failure (cf. Table 3.5), so the devices and links we consider here are in the minority.

**Load balancer failures are bursty.** Figure 3.10 shows the distribution of time between failures for devices. LBs tend to have the shortest time between failures, with a median of 8.6 minutes and 16.4 minutes for LB-1 and LB-2, respectively. Recall that failure events for these two LBs are dominated by a small number of devices that experience numerous failures (cf. Table 3.5). This small number of failure prone devices has a high impact on time between failure, especially since more than half of the LB-1 and LB-2 devices experience only a single

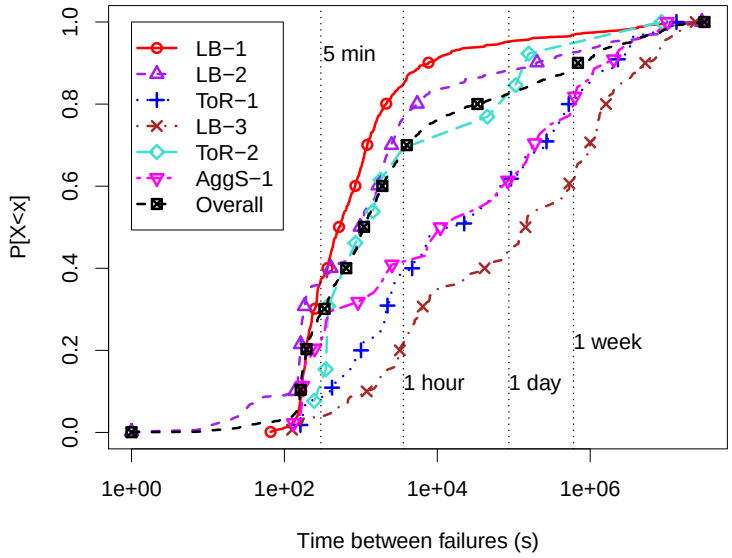



Figure 3.10: Time between failures for devices

failure.

In contrast to LB-1 and LB-2, devices like ToR-1 and AggS-1 have median time between failure of multiple hours and LB-3 has median time between failure of more than a day. We note that the LB-3 device is a newer version of the LB-1 and LB-2 devices and it exhibits higher reliability in terms of time between failures.

**Link flapping is absent from the *actionable* network logs.** Figure 3.11 presents the distribution of time between failures for the different link types. On an average, link failures tend to be separated by a period of about one week. Recall that our methodology leverages actionable information, as determined by network operators. This significantly reduces our observations of spurious link down events and observations of link flapping that do not impact network connectivity.

MGMT, CORE and ISC links are the most reliable in terms of time between failures, with most link failures on CORE and ISC links occurring more than an hour apart. Links between data centers experience the shortest time between failures. However, note that links connecting data centers have a very low failure probability. Therefore, while most links do not fail, the few

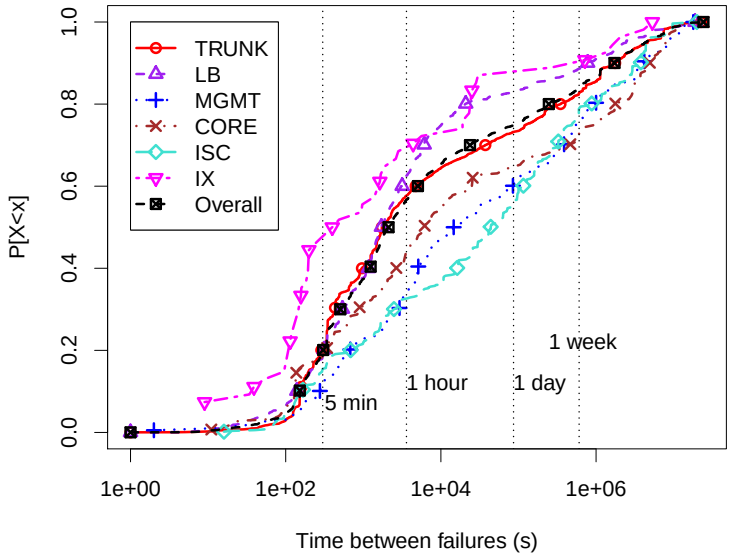



Figure 3.11: Time between failures for links

that do tend to fail within a short time period of prior failures. In reality, multiple inter-data center link failures in close succession are more likely to be investigated as part of the same troubleshooting window by the network operators.

### Reliability of network elements

We conclude our analysis of failure properties by quantifying the aggregate downtime of network elements. We define annualized downtime as the sum of the duration of all failures observed by a network element. For link failures, we consider failures that impacted network traffic, but highlight that a subset of these failures are due to planned maintenance. Additionally, redundancy in terms of network, application, and data in our system implies that this downtime *cannot* be interpreted as a measure of application-level availability. Figure 3.12 summarizes the annual downtime for devices that experienced failures during our study.

**Data center networks experience high availability.** With the exception of ToR-1 devices, all devices have a median annual downtime of less than 30 minutes. Despite experiencing the highest number of failures, LB-1 devices have the lowest annual downtime. This is due

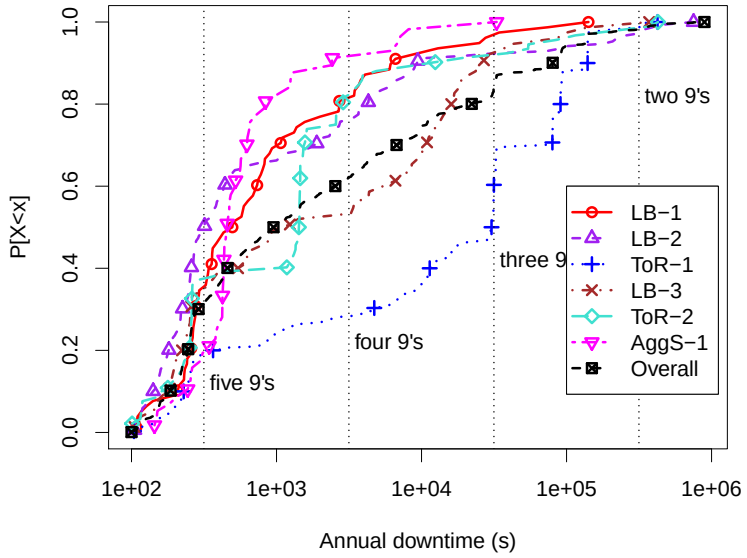



Figure 3.12: Annualized downtime for devices

to many of their failures being short-lived. Overall, devices experience higher than four 9's of reliability with the exception of ToRs, where long lived correlated failures cause ToRs to have higher downtime; recall, however, that only 3.9% of ToR-1s experience any failures (cf. Figure 3.4).

Annual downtime for the different link types are shown in Figure 3.13. The median yearly downtime for all link types, with the exception of links connecting data centers is less than 10 minutes. This duration is smaller than the annual downtime of 24-72 minutes reported by Turner *et al.* when considering an academic WAN [115]. Links between data centers are the exception because, as observed previously, failures on links connecting data centers take longer to resolve than failures for other link types. Overall, links have high availability with the majority of links (except those connecting data centers) having higher than four 9's of reliability.

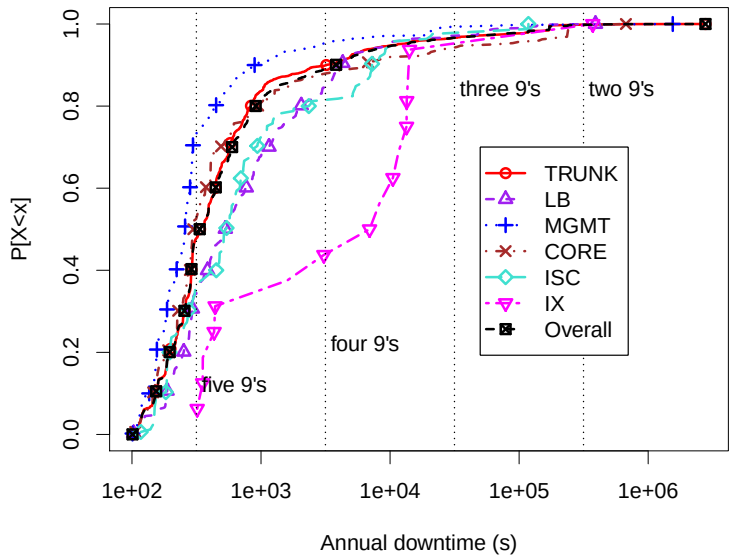



Figure 3.13: Annualized downtime for links

### 3.4.6 Grouping link failures

We now consider correlations between link failures. We also analyzed correlated failures for devices, but except for a few instances of ToRs failing together, grouped device failures are extremely rare (not shown).

To group correlated failures, we need to define what it means for failures to be correlated. First, we require that link failures occur in the same data center to be considered related (since it can be the case that links in multiple data centers fail close together in time but are in fact unrelated). Second, we require failures to occur within a predefined time threshold of each other to be considered correlated. When combining failures into groups, it is important to pick an appropriate threshold for grouping failures. If the threshold is too small, correlated failures may be split into many smaller events. If the threshold is too large, many unrelated failures will be combined into one larger group.

We considered the number of failures for different threshold values. Beyond grouping simultaneous events, which reduces the number of link failures by a factor of two, we did not

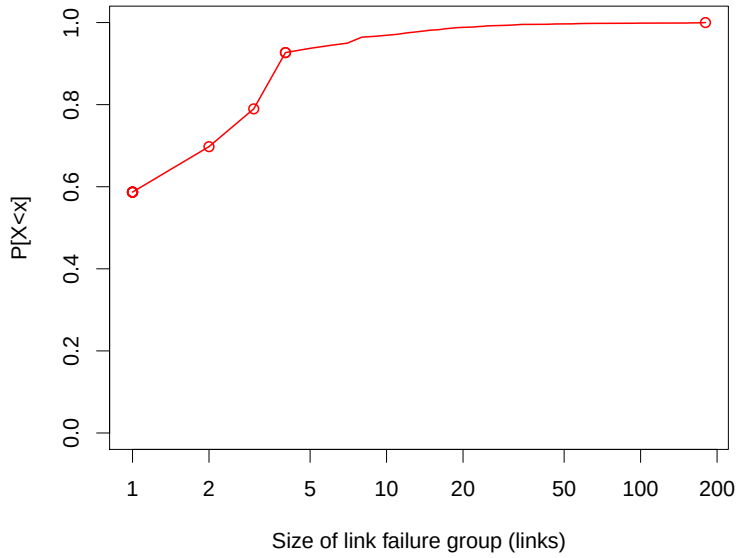



Figure 3.14: Number of links involved in link failure groups

see significant changes by increasing the threshold.

**Link failures tend to be isolated.** The size of failure groups produced by our grouping method is shown in Figure 3.14. We see that just over half of failure events are isolated with 41% of groups containing more than one failure. Large groups of correlated link failures are rare with only 10% of failure groups containing more than four failures. We observed two failure groups with the maximum failure group size of 180 links. These were caused by maintenance to an AggS connected to a large number of ToRs.

### 3.4.7 Root causes of failures

Finally, we analyze the types of problems associated with device and link failures. We initially tried to determine the root cause of failure events by mining diaries associated with NOC tickets. However, the diaries often considered multiple potential causes for failure before arriving at the final root cause, which made mining the text impractical. Because of this complication, we chose to leverage the "problem type" field of the NOC tickets which allows operators to place tickets into categories based on the cause of the problem. Table 3.6 gives examples of

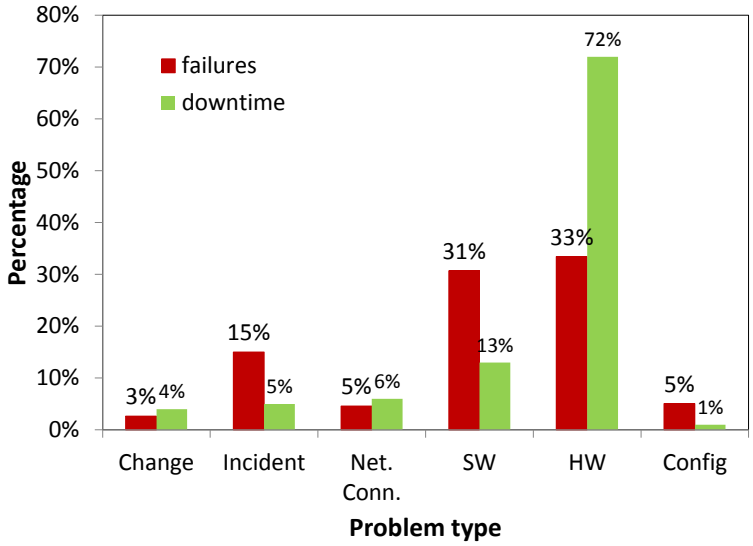



Figure 3.15: Device problem types

the types of problems that are put into each of the categories.

**Hardware problems take longer to mitigate.** Figure 3.15 considers the top problem types in terms of number of failures and total downtime for devices. Software and hardware faults dominate in terms of number of failures for devices. However, when considering downtime, the balance shifts and hardware problems have the most downtime. This shift between the number of failures and the total downtime may be attributed to software errors being alleviated by tasks that take less time to complete, such as power cycling, patching or upgrading software. In contrast, hardware errors may require a device to be replaced resulting in longer repair times.

**Load balancers affected by software problems.** We examined what types of errors dominated for the most failure prone device types (not shown). The LB-1 load balancer, which tends to have short, frequent failures and accounts for most failures (but relatively low downtime), mainly experiences software problems. Hardware problems dominate for the remaining device types. We observe that LB-3, despite also being a load balancer, sees much fewer software issues than LB-1 and LB-2 devices, suggesting higher stability in the newer model of the device.

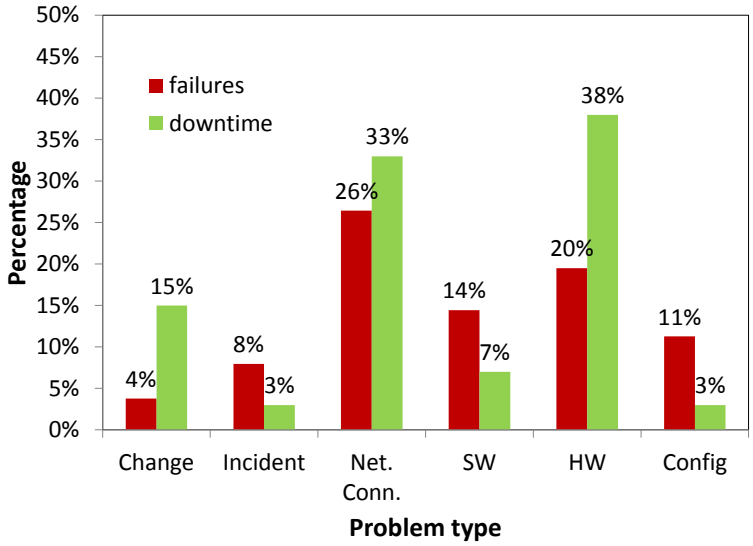



Figure 3.16: Link problem types

**Link failures are dominated by connection and hardware problems.** Figure 3.16 shows the total number of failures and total downtime attributed to different causes for link failures. In contrast to device failures, link failures are dominated by network connection errors, followed by hardware and software issues. In terms of downtime, software errors incur much less downtime per failure than hardware and network connection problems. This suggests software problems lead to sporadic short-lived failures (e.g., a software bug causing a spurious link down notification) as opposed to severe network connectivity and hardware related problems.

## 3.5 Estimating failure impact

In this section, we estimate the impact of link failures. In the absence of application performance data, we aim to quantify the impact of failures in terms of lost network traffic. In particular, we estimate the amount of traffic that would have been routed on a failed link had it been available for the duration of the failure.

In general, it is difficult to precisely quantify how much data was *actually* lost during a failure because of two complications. First, flows may successfully be re-routed to use alternate

Table 3.6: Examples of problem types

| Problem Type | Example causes or explanations |
|---|---|
| Change | Device deployment, UPS maintenance |
| Incident | OS reboot (watchdog timer expired) |
| Network Connection | OSPF convergence, UDLD errors, Cabling, Carrier signaling/timing issues |
| Software | IOS hotfixes, BIOS upgrade |
| Hardware | Power supply/fan, Replacement of line card/chassis/optical adapter |
| Configuration | VPN tunneling, Primary-backup failover, IP/MPLS routing |

routes after a link failure and protocols (e.g., TCP) have in-built retransmission mechanisms. Second, for long-lived failures, traffic variations (e.g., traffic bursts, diurnal workloads) mean that the link may not have carried the same amount of data even if it was active. Therefore, we propose a simple metric to approximate the magnitude of traffic lost due to failures, based on the available data sources.

To estimate the impact of link failures on network traffic (both in terms of bytes and packets), we first compute the median number of packets (or bytes) on the link in the hours preceding the failure event, $med_b$, and the median packets (or bytes) during the failure $med_d$. We then compute the amount of data (in terms of packets or bytes) that was *potentially* lost during the failure event as:

$$loss = (med_b - med_d) \times duration$$

where $duration$ denotes how long the failure lasted. We use median traffic instead of average to avoid outlier effects.

As described in Section 3.2, the network traffic in a typical data center may be classified into short-lived, latency-sensitive "mice" flows and long-lived, throughput-sensitive "elephant" flows. Packet loss is much more likely to adversely affect "mice" flows where the loss of

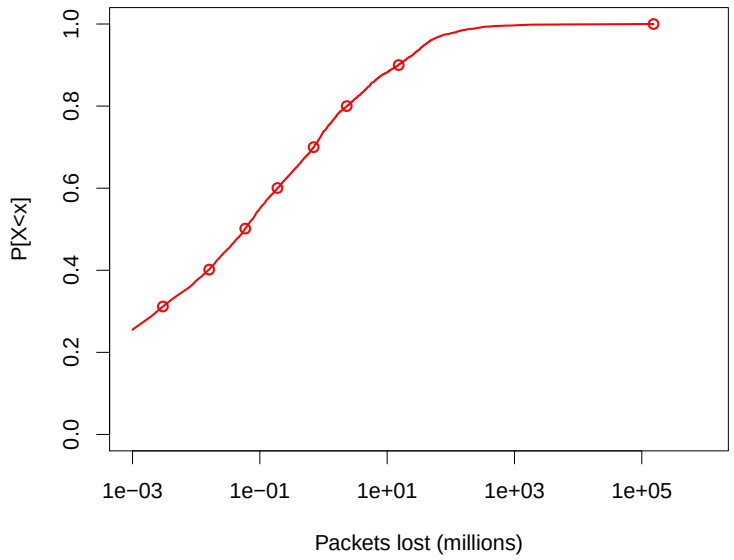



Figure 3.17: Estimated packet loss during failure events

an ACK may cause TCP to perform a timed out retransmission. In contrast, loss in traffic throughput is more critical for "elephant" flows.

**Link failures incur loss of many packets, but relatively few bytes.** For link failures, few bytes are estimated to be lost relative to the number of packets. We observe the estimated median number of packets lost during failures is 59K (Figure 3.17) but the estimated median number of bytes lost is only 25MB (Figure 3.18). Thus, the average size of lost packets is 423 bytes. Prior work on data center network traffic observed that packet sizes tend to be bimodal with modes around 200B and 1,400B [12]. This suggests that packets lost during failures are mostly part of the lower mode, consisting of keep alive packets used by applications (e.g., MYSQL, HTTP) or ACKs [12].

### 3.5.1 Is redundancy effective in reducing impact?

In a well-designed network, we expect most failures to be masked by redundant groups of devices and links. We evaluate this expectation by considering median traffic during a link failure (in packets or bytes) normalized by median traffic before the failure: $med_d/med_b$; for brevity,

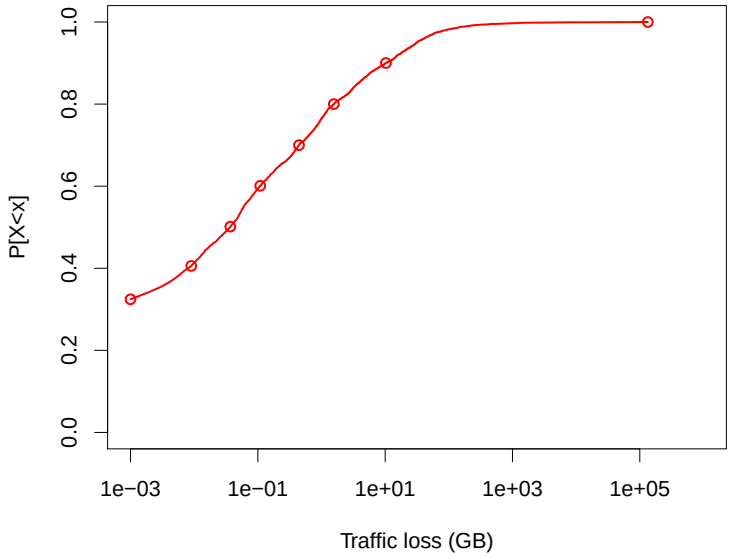



Figure 3.18: Estimated traffic loss during failure events

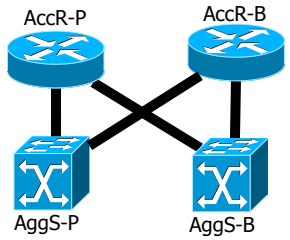



Figure 3.19: An example redundancy group between a primary (P) and backup (B) aggregation switch (AggS) and access router (AccR)

we refer to this quantity as "normalized traffic". The effectiveness of redundancy is estimated by computing this ratio on a per-link basis, as well as across all links in the redundancy group where the failure occurred. An example of a redundancy group is shown in Figure 3.19. If a failure has been masked completely, this ratio will be close to one across a redundancy group i.e., traffic during failure was equal to traffic before the failure.

**Network redundancy helps, but it is not entirely effective.** Figure 3.20 shows the distribution of normalized byte volumes for individual links and redundancy groups. Redundancy groups are effective at moving the ratio of traffic carried during failures closer to one with 25% of events experiencing no impact on network traffic at the redundancy group level. Also, the

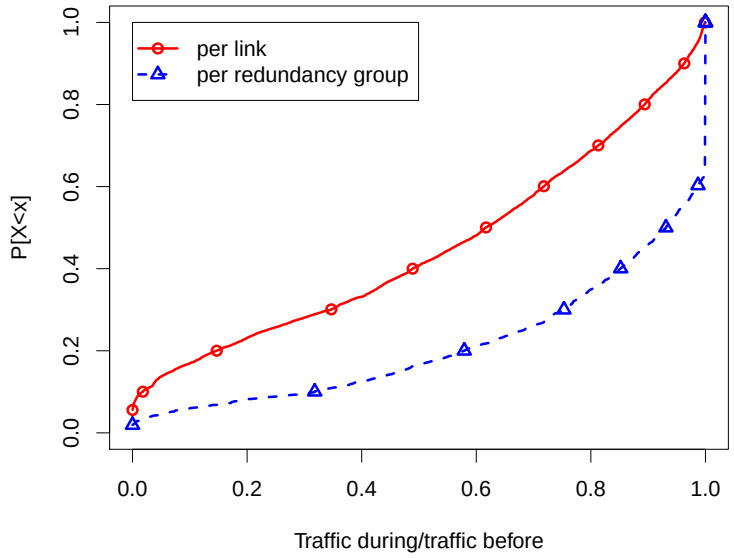



Figure 3.20: Normalized traffic (bytes) during failure events per link as well as within redundancy groups

median traffic carried at the redundancy group level is 93% as compared with 65% per link. This is an improvement of 43% in median traffic as a result of network redundancy. We make a similar observation when considering packet volumes (not shown) .

There are several reasons why redundancy may not be 100% effective in eliminating the impact of failures on network traffic. First, bugs in fail-over mechanisms can arise if there is uncertainty as to which link or component is the back up (e.g., traffic may be regularly traversing the back up link [16]). Second, if the redundant components are not configured correctly, they will not be able to re-route traffic away from the failed component. For example, we observed the same configuration error made on both the primary and back up of a network connection because of a typo in the configuration script. Further, protocol issues such as TCP backoff, timeouts, and spanning tree reconfigurations may result in loss of traffic.

### 3.5.2 Redundancy at different layers of the network topology

This section analyzes the effectiveness of network redundancy across different layers in the network topology. We logically divide links based on their location in the topology. Location

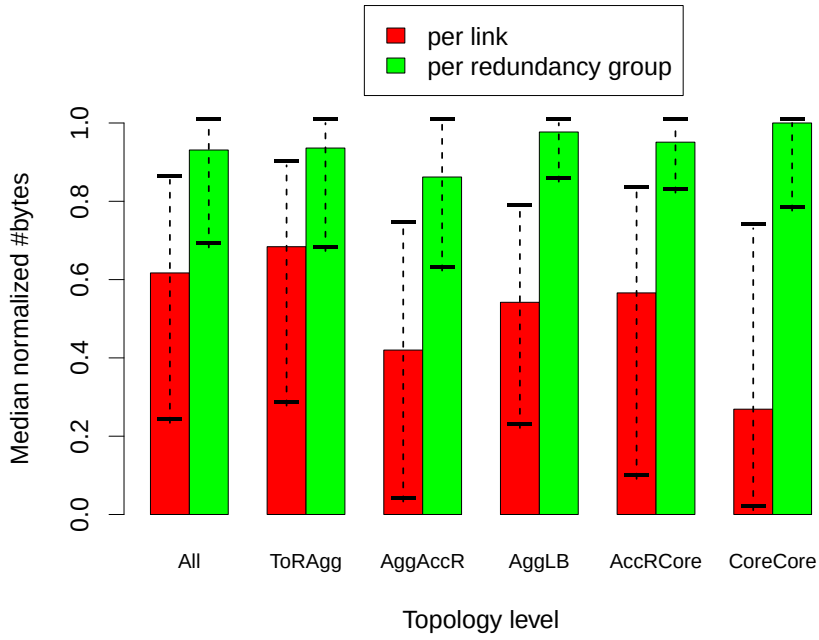



Figure 3.21: Normalized bytes (quartiles) during failure events per link and across redundancy group compared across layers in the topology

is determined based on the types of devices connected by the link (e.g., a CoreCore link connects two core routers). Figure 3.21 plots quartiles of normalized traffic (in bytes) for links at different layers of the network topology.

**Links highest in the topology benefit most from redundancy.** A reliable network core is critical to traffic flow in data centers. We observe that redundancy is effective at ensuring that failures between core devices have a minimal impact. In the core of the network, the median traffic carried during failure drops to 27% per link but remains at 100% when considered across a redundancy group. Links between aggregation switches and access routers (AggAccR) experience the next highest benefit from redundancy where the median traffic carried per link during failure drops to 42% per link but remains at 86% across redundancy groups.

**Links from ToRs to aggregation switches benefit the least from redundancy, but have low failure impact.** Links near the edge of the data center topology benefit the least from redundancy, where the median traffic carried during failure increases from 68% on links to

94% within redundancy groups for links connecting ToRs to AggS. However, we observe that on a per link basis, these links do not experience significant impact from failures so there is less room for redundancy to benefit them.

## 3.6 Discussion

In this section, we discuss implications of our study for the design of data center networks and future directions on characterizing data center reliability.

**Low-end switches exhibit high reliability.** Low-cost, commodity switches in our data centers experience the lowest failure rate with a failure probability of less than 5% annually for all types of ToR switches and AggS-2. However, due to their much larger population, the ToRs still rank third in terms of number of failures and dominate in terms of total downtime. Since ToR failures are considered the norm rather than the exception (and are typically masked by redundancy in application, data, and network layers), ToRs have a low priority for repair relative to other outage types. This suggests that proposals to leverage commodity switches to build flat data center networks [3, 46, 88] will be able to provide good reliability. However, as populations of these devices rise, the absolute number of failures observed will inevitably increase.

**Improve reliability of middleboxes.** Our analysis of network failure events highlights the role that middle boxes such as load balancers play in the overall reliability of the network. While there have been many studies on improving performance and scalability of large-scale networks [2, 3, 46, 49, 50, 88], only a few studies focus on management of middle boxes in data center networks [61]. Middle boxes such as load balancers are a critical part of data center networks that need to be taken into account when developing new routing frameworks. Further, the development of better management and debugging tools would help alleviate software and configuration faults frequently experienced by load balancers. Finally, software load balancers running on commodity servers can be explored to provide cost-effective, reliable alternatives

to expensive and proprietary hardware solutions.

**Improve the effectiveness of network redundancy.** We observe that network redundancies in our system are 40% effective at masking the impact of network failures. One cause of this is due to configuration issues that lead to redundancy being ineffective at masking failure. For instance, we observed an instance where the same typo was made when configuring interfaces on both the primary and back up of a load balancer connection to an aggregation switch. As a result, the back up link was subject to the same flaw as the primary. This type of error occurs when operators configure large numbers of devices, and highlights the importance of automated configuration and validation tools (e.g., [20]).

**Separate control plane from data plane.** Our analysis of NOC tickets reveals that in several cases, the loss of keep alive messages resulted in disconnection of portchannels, which are virtual links that bundle multiple physical interfaces to increase aggregate link speed. For some of these cases, we manually correlated loss of control packets with application-level logs that showed traffic bursts in the application on the egress path. This interference between application and control traffic is undesirable. Software Defined Networking (SDN) proposals such as OpenFlow [84] present a solution to this problem by maintaining state in a logically centralized controller, thus eliminating keep alive messages in the data plane. In the context of proposals that leverage location independent addressing (e.g., [46, 88]), this separation between control (e.g., ARP and DHCP requests, directory service lookups [46]) and data plane becomes even more crucial to avoid impact to hosted applications.

## 3.7 Related work

Previous studies of network failures have considered application-level [62, 96] or network connectivity [74, 82, 109, 115, 119] failures. There also have been several studies on understanding hardware reliability in the context of cloud computing [35, 106, 107, 117].

**Application failures.** Padmanabhan *et al.* consider failures from the perspective of Web

clients [96]. They observe that the majority of failures occur during the TCP handshake as a result of end-to-end connectivity issues. They also find that Web access failures are dominated by server-side issues. These findings highlight the importance of studying failures in data centers hosting Web services.

Netmedic aims to diagnose application failures in enterprise networks [62]. By taking into account state of components that fail together (as opposed to grouping all components that fail together), it is able to limit the number of incorrect correlations between failures and components.

**Network failures.** There have been many studies of network failures in wide area and enterprise networks [74, 82, 109, 115, 119] but none consider network element failures in large-scale data centers.

Shaikh *et al* study properties of OSPF Link State Advertisement (LSA) traffic in a data center connected to a corporate network via leased lines [109]. Watson *et al* also study stability of OSPF by analyzing LSA messages in a regional ISP network [119]. Both studies observe significant instability and flapping as a result of external routing protocols (e.g., BGP). Unlike these studies, we do not observe link flapping owing to our data sources being geared towards actionable events.

Markopolou *et al.* use IS-IS listeners to characterize failures in an ISP backbone [82]. The authors classify failures as either router related or optical related by correlating time and impacted network components. They find that 70% of their failures involve only a single link. We similarly observe that the majority of failures in our data centers are isolated.

More recently, Turner *et al.* consider failures in an academic WAN using syslog messages generated by IS-IS [115]. Unlike previous studies [82, 109, 119], the authors leverage existing syslog, e-mail notifications, and router configuration data to study network failures. Consistent with prior studies that focus on OSPF [109, 119], the authors observe link flapping. They also observe longer time to repair on wide area links, similar to our observations for wide area links connecting data centers.

**Network configuration tools.** Many tools have been developed to help protect against configuration errors as a cause of network failures and performance degradation [4, 20, 33, 81, 114]. Many of these tools allow network operators to determine the impacts of potential network changes [4, 33, 114]. For example, Tariq *et al.* enable CDNs to evaluate changes to response time distributions prior to applying configuration changes [114]. Alimi *et al.* develop Shadow Configurations as a mechanism for operators to test router configuration changes on live network traffic before deploying them throughout the network [4]. Other tools detect constraint violations to prevent configuration errors. Feamster and Balakrishnan leverage static analysis to prevent BGP configuration errors [33] and Chen *et al.* [20] provide a database-like abstraction that enforces constraints on network connectivity to prevent operator errors. Mahimkar *et al.* take a different approach and develop MERCURY, a tool that enables operators to reason about impacts of configuration changes after they have been made [81]. MERCURY detects network upgrade events from configuration and work flow logs and then identifies persistent behavior changes caused by these upgrades [81].

**Fault detection and diagnosis.** Many related papers focus on identifying and diagnosing network faults after they have occurred, either because of misconfigurations or other causes [14, 98, 124, 125]. The challenge of deriving meaningful events from noisy syslog messages is tackled by Qiu *et al.*, who leverage data mining to distill numerous syslog messages into a smaller set of *actionable* events [98]. Bodik *et al.* develop fingerprints of previous network failures that can be used to help operations teams identify reoccurring failures and quickly apply known solutions [14]. A system for root cause analysis, G-RCA, is proposed by Yan *et al.* [125]. Their system identifies relevant topology components and applies machine learning to determine root cause of the network event. Using console logs to debug large scale systems is proposed by Xu *et al.* [124]. They use source code analysis to parse console logs and employ machine learning to identify console logs that indicate abnormal system state.

**Failures in cloud computing.** The interest in cloud computing has increased focus on understanding component failures, as even a small failure rate can manifest itself in a high number

of failures in large-scale systems. Previous work has looked at failures of DRAM [107], storage [35, 106] and server nodes [117], but there has not been any study on network component failures in data centers. Ford *et al.* consider the availability of distributed storage and observe that the majority of failures involving more than ten storage nodes are localized within a single rack [35]. We also observe spatial correlations but they occur higher in the network topology, where we see multiple ToRs associated with the same aggregation switch having correlated failures.

Complementary to our work, Benson *et al.* mine threads from customer service forums of an IaaS cloud provider [13]. They report on problems users face when using IaaS and observe that problems related to managing virtual resources and performance of computing instances that require involvement of cloud administrators, increase over time.

## 3.8 Summary

In this chapter, we have presented the first large-scale analysis of network failure events in data centers. We focused our analysis on characterizing failures of network links and devices, estimating their failure impact, and analyzing the effectiveness of network redundancy in masking failures. To undertake this analysis, we developed a methodology that correlates network traffic logs with logs of actionable events, to filter a large volume of non-impacting failures due to spurious notifications and errors in logging software.

Our study is part of a larger project, NetWiser, on understanding reliability in data centers to aid research efforts on improving network availability and designing new network architectures. Based on our study, we find that commodity switches exhibit high reliability which supports current proposals to design flat networks using commodity components [3, 46, 68, 88]. We also highlight the importance of studies to better manage middle boxes such as load balancers, as they exhibit high failure rates. Finally, more investigation is needed to analyze and improve the effectiveness of redundancy at both network and application layers.

# Chapter 4

# Conclusions

The Internet plays an important role in many aspects of our lives. With services such as health care and power grid depending on it, the Internet is truly a critical resource. As a consequence, it is important that the network infrastructure provides dependable service to the many applications relying on it. This thesis addresses the challenge of improving the dependability of network infrastructure. We consider the dual challenges of (1) ensuring data in transit cannot be intercepted or dropped by a malicious entity and (2) ensuring that traffic is not impacted by unreliability of network components. We tackle these challenges from both the core and edge of the network with sensitivity to challenges faced in each setting.

In the core of the network, we challenge conventional wisdom, and show that there can be economic incentives to deploy new protocols, such as S*BGP. We present a three step strategy for deploying S*BGP that harnesses S*BGP's impact on routing. The key idea of our strategy is to drive revenue-generating network traffic toward ISPs that have deployed S*BGP, thus creating economic incentives for deployment. This work illustrates a potential deployment path for control plane protocols that validate routing messages, and opens many avenues for future work on incremental deployment of protocols like S*BGP.

At the edge of the network, we shed light on the hitherto unexplored topic of data center network reliability. We characterize failures of network components over a year in a data center

network comprised of tens of data centers hosting numerous popular online services. Our results support recent proposals to build data center networks out of commodity components, and emphasize the importance of work on improving reliability of middle boxes in data centers.

The work of this thesis has benefited from dialog with the relevant stakeholders on the Internet: standardization bodies, network operators and large content providers. This has allowed the work to have real world impact in the form of an FCC working group [53], and improved root cause analysis at a large content provider. This dialog also points to many areas for future research on incremental deployment challenges for S*BGP and taking a more holistic approach to studying network reliability.

## 4.1 Future work.

### 4.1.1 Incremental deployment challenges on the Internet

Deploying new protocols on the Internet presents many challenges, both technical and political in nature. We now summarize some of these potential directions for future work.

**How can we detect and quantify the impact of routing attacks and misconfiguration on the Internet?** Measuring the benefits of S*BGP over time requires benchmarking the frequency and impact of anomalous routing events (*e.g.,* misconfiguration, attacks, censorship) on the Internet. However, there is a dearth of metrics for accomplishing this task on an ongoing basis. This issue can be addressed by leveraging publicly available repositories of routing data (*e.g.,* RouteViews [93]) to detect anomalous routing events (*e.g.,* building on techniques proposed in [11, 64]). We will further use publicly available network traffic measurements (*e.g.,* iPlane [80]) to quantify the impact of anomalous routing events on network traffic.

**What are the potential risks of a single cryptographic root of trust for secure routing?** A key stumbling block in the deployment of S*BGP has been the requirement for a cryptographic root of trust, referred to as the Resource Public Key Infrastructure (RPKI), to manage cryptographic keys used by networks. RPKI is now on the horizon with initial deployments

taking place at the regional Internet registries [6, 103]. Ideally, these regional RPKIs will be united under a single cryptographic root of trust to ensure consistency and synchronization of data across the regions, but deciding on a single root of trust remains a contentious issue. Specifically, this root of trust will be able to revoke cryptographic keys, a powerful tool for Internet censorship as well as a key piece of leverage in negotiations. Empirical data about the AS graph and business relationships can play a role in terms of quantifying the impact of a single root of trust on the Internet's structure.

**What are the costs associated with secure routing protocols?** In Chapter 2, we propose a strategy to create economic incentives for deploying S*BGP on the Internet. However, the magnitude of these incentives depends on the costs of deploying S*BGP which are largely unknown. Since S*BGP is still in the standardization process, it is difficult to ascertain the precise financial costs of deployment. Thus, in collaboration with a Tier 1 ISP, we are investigating the computational overheads of different S*BGP deployment scenarios in their network in order to approximate the financial costs of deployment (*i.e.,* the more overhead, the more expensive the upgrade will be).

### 4.1.2 Building more dependable networks

Chapter 3 presents an empirical study of failures in a data center networks. This work is an important first step to improving the reliability of networks. With this goal in mind, future work involves taking a holistic view of network reliability and considering the following problems:

**Failures at all layers of the protocol stack.** In Chapter 3 we focus on failures between network interfaces and correlate with network traffic measurements to infer impact. However, it is important to study failures at multiple layers of the network protocol stack to better understand their impact. Specifically, correlating failures between interfaces, with application layer availability can help us understand how these low level failures translate into user facing impact. Further, we can also study application level failures that *do not* manifest as failures between interfaces (*e.g.,* errors in network or transport protocols) to understand how layers of

the protocol impact the reliability of services.

**Failures in different types of networks.** Chapter 3 focuses on reliability of one type of network: data center networks. However, it is important to understand the reliability of different types of networks such as campus/enterprise networks and even cellular data networks. These environments present different challenges to reliability in terms of the technologies and topologies employed by the networks. They are also subject to different workload patterns (*e.g.,* e-mail vs. mobile Web browsing) which impacts how users will perceive the impact of network failures.

**The role of network administrators in errors.** Our study of network reliability highlighted the role of human factors in terms of configuration errors (*e.g.,* typos in configuration scripts). To address these problems there has been a wealth of research designing better network configuration tools in order to reduce the impact of human error on networks (*e.g.,* [4, 20, 33, 81, 114]). However, the relationship between human actions and network reliability has not been well studied. By correlating logs of network changes (*e.g.,* using techniques from [69]) with logs of network failures we can pin point the most failure prone human activities and ensure configuration tools address these issues.

# Bibliography


[1] Cisco: Data center: Load balancing data center services, 2004. www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns668/net_implementation_white_paper0900aecd8053495a.html.

[2] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly. Symbiotic routing in future data centers. In *Proc. of ACM SIGCOMM*, 2010.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. of ACM SIGCOMM*, 2008.

[4] R. Alimi, Y. Wang, and Y. Yang. Shadow configuration as a network management primitive. In *Proc. of ACM SIGCOMM*, 2008.

[5] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. DCTCP: Efficient packet transport for the commoditized data center. In *Proc. of ACM SIGCOMM*, 2010.

[6] ARIN. ARIN resource certification. https://www.arin.net/resources/rpki.html.

[7] M. Arrington. Facebook COO: 175 million people log into Facebook every day, 2011. http://techcrunch.com/2010/02/01/facebook-coo-sheryl-sandberg-world-economic-forum-davos/.

[8] B. Augustin, B. Krishnamurthy, and W. Willinger. IXPs: Mapped? In *Proc. of the ACM Internet Measurement Conference*, 2009.

[9] R. Austein, G. Huston, S. Kent, and M. Lepinski. Secure inter-domain routing: Manifests for the Resource Public Key Infrastructure, 2010. `http://tools.ietf.org/html/draft-ietf-sidr-rpki-manifests-09`.

[10] I. Avramopoulos, M. Suchara, and J. Rexford. How small groups can secure interdomain routing. Technical report, Princeton University Computer Science, 2007.

[11] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the Internet. In *Proc. of ACM SIGCOMM*, 2007.

[12] T. Benson, A. Akella, and D. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of the ACM Internet Measurement Conference*, 2010.

[13] T. Benson, S. Sahu, A. Akella, and A. Shaikh. A first look at problems in the cloud. In *Proc. of HotCloud*, 2010.

[14] P. Bodik, M. Goldszmidt, A. Fox, D. Woodard, and H. Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *Proc. of Eurosys*, 2010.

[15] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *RSA CryptoBytes*, 2003.

[16] J. Brodkin. Amazon EC2 outage calls 'availability zones' into question, 2011. `http://www.networkworld.com/news/2011/042111-amazon-ec2-zones.html`.

[17] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proc. of the IEEE*, 2010.

[18] K. Butler, P. McDaniel, and W. Aiello. Optimizing BGP security by exploiting path stability. In *Proc. of the ACM Conference on Computer and Communications Security*, 2006.

[19] H. Chang, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocol. In *Proc. of ACM SIGCOMM*, 2006.

[20] X. Chen, Y. Mao, Z. Mao, and K. van de Merwe. Declarative configuration management for complex and dynamic networks. In *Proc. of ACM CoNEXT*, 2010.

[21] Y. Chi, R. Oliveira, and L. Zhang. Cyclops: The Internet AS-level observatory. *ACM SIGCOMM Computer Communication Review*, 2008.

[22] Cisco. Spanning tree protocol root guard enhancement. `http://www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800ae96b.shtml`.

[23] Cisco. Unidirectional link detection (UDLD). `http://www.cisco.com/en/US/tech/tk866/tsd_technology_support_sub-protocol_home.html`.

[24] D. Clark, J. Wroclawski, K. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's Internet. *Transactions on Networking*, 2005.

[25] J. Clark. Windows Azure suffers worldwide outage. *ZD.Net*, 2012.

[26] J. Cowie. Rensys blog: China's 18-minute mystery. `http://www.renesys.com/blog/2010/11/chinas-18-minute-mystery.shtml`.

[27] A. Dainotti, C. Squarcella, E. Aben, K. Claffy, M. Chiesa, M. Russo, and A. Pescape. Analysis of country-wide Internet outages caused by censorship. In *Proc. of the ACM Internet Measurement Conference*, 2011.

[28] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[29] A. Dhamdhere and C. Dovrolis. The Internet is flat: Modeling the transition from a transit hierarchy to a peering mesh. In *Proc. of ACM CoNEXT*, 2010.

[30] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, and kc claffy. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review*, 2007.

[31] X. Dimitropoulos and G. Riley. Efficient large-scale BGP simulations. *Elsevier Computer Networks, Special Issue on Network Modeling and Simulation*, 2006.

[32] N. Falliere, L. Murchu, and E. Chien. W32.Stuxnet Dossier. *Symentec Security Response*, 2011.

[33] N. Feamster and H. Balakrishnan. Detecting BGP configuration faults with static analysis. In *Proc. of ACM SIGCOMM*, 2005.

[34] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. In *Proc. of ACM SIGMETRICS*, 2004.

[35] D. Ford, F. Labelle, F. Popovici, M. Stokely, V. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[36] L. Gao and J. Rexford. Stable Internet routing without global coordination. *Transactions on Networking*, 2001.

[37] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube traffic characterization: A view from the edge. In *Proc. of the ACM Internet Measurement Conference*, 2007.

[38] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Characterizing user sessions on YouTube. In *Proc. of Multimedia Computing and Networking*, 2008.

[39] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening Internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *Proc. of the Passive and Active Measurement Conference*, April 2008.

[40] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proc. of ACM SIGCOMM*, 2011.

[41] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. In *Proc. of ACM SIGCOMM*, 2011.

[42] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. Full version. Technical report, Boston University, 2011. `http://www.cs.toronto.edu/~phillipa/papers/SBGPtrans_full.pdf`.

[43] P. Gill, M. Schapira, and S. Goldberg. Modeling on quicksand: Dealing with the scarcity of ground truth in interdomain routing data. *ACM SIGCOMM Computer Communication Review*, 2012.

[44] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols. In *Proc. of ACM SIGCOMM*, 2010.

[45] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proc. of the Network and Distributed System Security Symposium*, 2003.

[46] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *Proc. of ACM SIGCOMM*, 2009.

[47] T. Griffin, F. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *Transactions on Networking*, 2002.

[48] R. Guérin and K. Hosanagar. Fostering IPv6 migration through network quality differentials. *ACM SIGCOMM Computer Communication Review*, 2010.

[49] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu. DCell: A scalable and fault-tolerant network structure for data centers. In *Proc. of ACM SIGCOMM*, 2008.

[50] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu. BCube: A high performance, server-centric network architecture for modular data centers. In *Proc. of ACM SIGCOMM*, 2009.

[51] C. Harris. Data center outages generate big losses. *Information Week*, 2011.

[52] S. Hart. Adaptive heuristics. *Econometrica*, 2005.

[53] S. Hartman. CSRIC III working group descriptions and proposed co-chairs and leadership, 2011. `http://transition.fcc.gov/pshs/advisory/csric3/wg-descriptions_v1.pdf`.

[54] Y. Hu, A. Perrig, and M. Sirbu. SPV: secure path vector routing for securing BGP. In *Proc. of ACM SIGCOMM*, 2004.

[55] IETF. Secure inter-domain routing (SIDR). `http://datatracker.ietf.org/wg/sidr/charter/`.

[56] M. Jackson and L. Yariv. Diffusion on social networks. In *Conferences des journees Louis-Andre Gerard-Varet Public Economy Theory Meeting*, 2005.

[57] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical report, University of Michigan, 2000.

[58] Y. Jin, S. Sen, R. Guerin, K. Hosanagar, and Z. Zhang. Dynamics of competition between incumbent and emerging network technologies. In *Proc. of NetEcon*, 2008.

[59] J. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkataramani. Consensus routing: The Internet as a distributed system. In *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[60] D. Joseph, N. Shetty, J. Chuang, and I. Stoica. Modeling the adoption of new network architectures. In *Proc. of ACM CoNEXT*, 2007.

[61] D. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *Proc. of ACM SIGCOMM*, 2008.

[62] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *Proc. of ACM SIGCOMM*, 2010.

[63] S. Kandula, S. Sengupta, A. Greenberg, and P. Patel. The nature of datacenter traffic: Measurements and analysis. In *Proc. of the ACM Internet Measurement Conference*, 2009.

[64] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks*, 2008.

[65] J. Karlin, S. Forrest, and J. Rexford. Nation-state routing: Censorship, wiretapping, and BGP. *Computing Research Repository (CoRR)*, abs/0903.3218, 2009.

[66] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proc. of ACM SIGKDD*, 2003.

[67] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 2000.

[68] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A scalable ethernet architecture for large enterprises. In *Proc. of ACM SIGCOMM*, 2008.

[69] H. Kim, T. Benson, N. Feamster, and A. Akella. The evolution of network configuration: A tale of two campuses. In *Proc. of the ACM Internet Measurement Conference*, 2011.

[70] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for

large-scale production networks. In *Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[71] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J. Cui, and A. Percus. Sampling large internet topologies for simulation purposes. *Computer Networks (Elsevier)*, 2007.

[72] J. Kurose and K. Ross. *Computer Networking: A top-down approach featuring the Internet*. Addison Wesley, 2005.

[73] C. Labovitz. Arbor blog: Battle of the hyper giants. `http://asert.arbornetworks.com/2010/04/the-battle-of-the-hyper-giants-part-i-2/`.

[74] C. Labovitz and A. Ahuja. Experimental study of internet stability and wide-area backbone failures. In *Proc. of the 29th Annual International Symposium on Fault-Tolerant Computing*, 1999.

[75] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proc. of ACM SIGCOMM*, 2010.

[76] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Shang. PHAS: Prefix hijack alert system. In *Proc. of the 15th USENIX Security Symposium*, 2006.

[77] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480 (Informational), February 2012.

[78] M. Lepinski and S. Turner. BGPSEC protocol specification, 2011. `http://tools.ietf.org/html/draft-lepinski-bgpsec-overview-00`.

[79] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proc. of EUROCRYPT*, 2004.

[80] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc. of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 2006.

[81] A. Mahimkar, H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proc. of ACM SIGCOMM*, 2010.

[82] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational IP backbone network. *IEEE/ACM Transactions on Networking*, 2008.

[83] C. Duffy Marsan. U.S. plots major upgrade to Internet router security. *Network World*, 2009.

[84] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*, 2008.

[85] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. In *Proc. of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2001.

[86] S.A. Misel. "Wow, AS7007!". Merit NANOG Archive, apr 1997. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html`.

[87] S. Morris. Contagion. *Review of Economics Studies*, 2003.

[88] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *Proc. of ACM SIGCOMM*, 2009.

[89] J. Naous, M. Walfish, A. Nicolosi, D. Mazieres, M. Miller, and A. Seehra. Verifying and enforcing network paths with ICING. In *Proc. of ACM CoNEXT*, 2011.

[90] Merit Networks. Internet routing registry. `www.irr.net`.

[91] D. Nicol, S. Smith, and M. Zhao. Evaluation of efficient security for BGP route announcements using parallel simulation. *Simulation practice and theory journal, Special issue on modeling and simulation of distributed systems and networks*, 2004.

[92] OECD. Oecd broadband portal. `http://www.oecd.org/document/54/0,3746,en_2649_34225_38690102_1_1_1_1,00.html`.

[93] University of Oregon. Route views project. `http://www.routeviews.org/`.

[94] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. Quantifying the completeness of the observed internet AS-level structure. *UCLA Computer Science Department - Techical Report TR-080026-2008*, Sept 2008.

[95] Fixed Orbit. `http://www.fixedorbit.com/metrics.htm`.

[96] V. Padmanabhan, S. Ramabhadran, S. Agarwal, and J. Padhye. A study of end-to-end web access failures. In *Proc. of ACM CoNEXT*, 2006.

[97] B. Premore. *An analysis of convergence properties of the border gateway protocol usingdiscrete event simulation.* PhD thesis, Dartmouth College, 2003.

[98] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu. What happened in my network? Mining network events from router syslogs. In *Proc. of the ACM Internet Measurement Conference*, 2010.

[99] T. Qiu, L. Ji, D. Pei, J. Wang, J. Xu, and H. Ballani. Locating prefix hijackers using LOCK. In *Proc. of the 18th USENIX Security Symposium*, 2009.

[100] B. Quoitin and S. Uhlig. Modeling the routing of an autonomous system with CBGP. *IEEE Network Magazine*, 2005.

[101] S Ratnasamy, S. Shenker, and S McCanne. Towards an evolvable Internet architecture. In *Proc. of ACM SIGCOMM*, 2005.

[102] Rensys Blog. Pakistan hijacks YouTube. http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml.

[103] RIPE. RIPE NCC Resource Certification. http://www.ripe.net/certification/.

[104] E. Rogers. *Diffusion of Innovations, 5th Edition*. Free Press, 2003.

[105] Sandvine. Fall 2010 global internet phenomena, 2010.

[106] B. Schroeder and G. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proc. of the 5th USENIX Conference on File and Storage Technologies*, 2007.

[107] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM errors in the wild: A large-scale field study. In *Proc. of ACM SIGMETRICS*, 2009.

[108] S. Sen, Y. Jin, R. Guerin, and K. Hosanagar. Modeling the dynamics of network technology adoption and the role of converters. *IEEE/ACM Transations on Networking*, 2010.

[109] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb. A case study of OSPF behavior in a large enterprise network. In *Proc. of the ACM Internet Measurement Workshop*, 2002.

[110] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the test-bed? In *Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[111] A. Singla, C. Hong, L. Popa, and P. Godfrey. Jellyfish: Networking data centers randomly. In *Proc. of the 9th USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[112] S. Steinberg. Over half of Americans now get their news online, 2011. http://www.rollingstone.com/culture/blogs/gear-up/over-half-of-americans-now-get-their-news-online-20110727.

[113] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security mechanisms for BGP. In *Proc. of the 1st USENIX Symposium on Networked Systems Design and Implementation*, 2004.

[114] M. Bin Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering "What-if" deployment and configuration questions with WISE. In *Proc. of ACM SIGCOMM*, 2008.

[115] D. Turner, K. Levchenko, A. Snoeren, and S. Savage. California fault lines: Understanding the causes and impact of network failures. In *Proc. of ACM SIGCOMM*, 2010.

[116] T. Valente. *Network Models of the Diffusion of Innovations (Quantitative Methods in Communication Subseries)*. Hampton Press, 1995.

[117] K. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *Proc. of the ACM Symposium on Cloud Computing*, 2010.

[118] T. Wan, E. Kranakis, and P. van Oorschot. Pretty secure BGP (psBGP). In *Proc. of the 12th Annual Network and Distributed System Security Symposium*, 2005.

[119] D. Watson, F. Jahanian, and C. Labovitz. Experiences with monitoring OSPF on a regional service provider network. In *Proc. of the 23rd International Conference on Distributed Computing Systems*, 2003.

[120] R White. Deployment considerations for secure origin BGP (soBGP). draft-white-sobgp-bgp-deployment-01.txt, June 2003, expired.

[121] M. Wojciechowski. Border gateway protocol modeling and simulation. Master's thesis, University of Warsaw, 2008.

[122] J. Wu, Y. Zhang, Z. M. Mao, and K. Shin. Internet routing resilience to failures: Analysis and implications. In *Proc. of ACM CoNEXT*, 2007.

[123] Y. Xiang, Z. Wang, X. Yin, and J. Wu. Argus: An accurate and agile system to detecting IP prefix hijacking. In *Proc. of the IEEE International Conference on Network Protocols*, 2011.

[124] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordon. Detecting large-scale system problems by mining console logs. In *Proc. of the ACM Symposium on Operating Systems Principles*, 2009.

[125] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates. G-RCA: A generic root cause analysis platform for service quality management in large ip networks. In *Proc. of ACM CoNEXT*, 2010.

[126] H. Peyton Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 2001.

[127] M. Yu, J. Rexford, M. Freedman, and J. Wang. Scalable flow-based networking with DIFANE. In *Proc. of ACM SIGCOMM*, 2010.

[128] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. Kumar Gunda, and J. Currey. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In *Proc. of the 8th USENIX Symposium on Operating Systems Design and Implementation*, 2008.

[129] E. Zegura, K. Calvert, and S. Bhattarcharjee. How to model an internetwork. In *Infocom*, 1996.

[130] Z. Zhang, Y. Zhang, Y. Hu, Z. Mao, and R. Bush. iSPY: detecting IP prefix hijacking on my own. In *Proc. of ACM SIGCOMM*, 2008.

[131] M. Zhao, S. Smith, and D. Nichol. Aggregated path authentication for efficient BGP security. In *Proc. of the 12th ACM Conference on Computer and Communications Security*, 2005.

[132] M. Zhao, S. Smith, and D. Nicol. Evaluating the performance impact of PKI on BGP security. In *4th annual PKI R&D workshop*, 2005.

[133] M. Zhao, W. Zhao, A. Gurney, A. Haeberlen, M. Sherr, and B. Loo. Private and verifiable interdomain routing decisions. In *Proc. of ACM SIGCOMM*, 2012.

# Appendix A

# A model of routing with BGP.

We follow [47] by assuming that each AS $a$ computes paths to a given destination AS $d$ based a *ranking* on outgoing paths, and an *export policy* specifing the set of neighbors to which a given path should be announced.

**Rankings.** AS $a$ selects a path to $d$ from the set of simple paths it learns from its neighbors as follows:

**LP Local Preference.** Paths are ranked based on their next hop: customer is chosen over peer which is chosen over provider.

**SP Shortest Paths.** Among the paths with the highest local preference, prefer the shortest ones.

**SecP Secure Paths.** If there are multiple such paths, and node $a$ is secure, then prefer the secure paths.

**TB Tie Break.** If there are multiple such paths, node $a$ breaks ties: if $b$ is the next hop on the path, choose the path where hash, $H(a,b)$ is the lowest.[1]

[1] In practice, this is done using the distance between routers and router IDs. Since we do not incorporate this information in our model we use a randomized tie break which prevents certain ASes from "always winning".

This standard model of local preference [36] captures the idea that an AS has incentives to prefer routing through a customer (that pays it) over a peer (no money is exchanged) over a provider (that it must pay).

**Export Policies.** This standard model of export policies captures the idea that an AS will only load its network with transit traffic if its customer pays it to do so [36]:

**GR2** AS $b$ announces a path via AS $c$ to AS $a$ iff at least one of $a$ and $c$ are customers of $b$.

# Appendix B

# Attacks on partially secure paths

We show how preferring partially secure paths over insecure paths can introduce *new* attack vectors that do not exist even without S*BGP:

**Figure B.1:** Suppose that only ASes $p$ and $q$ are secure, and that malicious AS $m$ falsely announces the path $(m,v)$, and suppose that $p$'s tiebreak algorithm prefers paths through $r$ over paths through $q$. Then, $p$ has a choice between two paths; a partially-secure false path $(p,q,m,v)$, and and insecure true path $(p,r,s,v)$. If no AS used S*BGP, $p$ would have chosen the true path (per his tiebreak algorithm); if $p$ prefers partially secure paths, he will be fooled into routing to AS $m$.

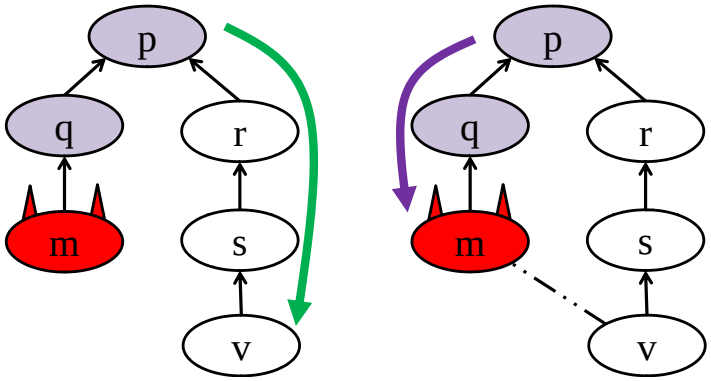



Figure B.1: A new attack vector

# Appendix C

# AS Graph Sensitivity Analysis.

**Incompleteness of AS-level topologies.** It is widely reported that AS-level graphs of the Internet are incomplete [94], but a ground truth for AS-level connectivity remains elusive. This is especially problematic for large content providers that primarily peer with large numbers of ASes to drive down costs for both themselves and the networks they peer with. Since peering links are only exported to customers [94] and content providers do not generally have customers, this is potentially a large blind spot in the existing data. Indeed, we observed that average path lengths for the five CPs in the AS graph (according to the routing policies of Appendix A) were around 2.7-3.5 hops whereas they are reported to be much lower, around 2.2-2.4 hops [95]. The reported value we consider is the Knodes index [95], which uses public and private BGP data to measure the number of hops that must be traversed between IP addresses in a given network and all other IP addresses.

**Creating the augmented graph.** To understand how incompleteness of the AS-level toplogy impacts our results, we developed an augmented topology with particular focus on more accurate connectivity for the five CPs. Our strategy for augmenting the AS-level topology leveraged recent research on IXPs that finds that many CPs are joining IXPs and peering with a large fraction of their members [8].

We used the Cyclops AS graph from Dec. 9, 2010 with additional peering edges from [8].

Table C.1: Summary of AS graphs

| Graph | ASes | peering | customer-provider |
|---|---|---|---|
| Cyclops+IXP [8, 21] | 36,964 | 58,829 | 72,848 |
| Augmented graph | 36,966 | 77,380 | 72,848 |

Table C.2: Average path length from the five content providers to all other destinations

| AS | Cyclops | Augmented | Knodes |
|---|---|---|---|
| 15169 | 2.7 | 2.1 | 2.2 |
| 8075 | 2.8 | 2.1 | 2.3 |
| 20940 | 3.6 | 2.2 | 2.2 |
| 22822 | 6.9 | 6.8 | 2.3 |
| 32934 | 3.5 | 2.1 | 2.4 |

In the Cyclops graph, the content providers have some customers, mainly owing to company acquisitions (e.g., YouTube's AS 35361 is a customer of Google). Since the CPs do not generally provide transit to other ASes, we remove these 79 customer ASes from the graph. We summarize the resulting Cyclops+IXP graph in Table C.1.

Starting with this graph, the five CPs were then connected randomly to ASes present at IXPs [8] until their average path length to *all destinations* on the Internet decreased to around 2.1-2.2 hops. The path lengths in the original and augmented topologies as well as the reported Knodes index [95] (an approximation of path length) are shown in Table C.2.

**Properties of the augmented graph.** In our augmented AS graph, the five CPs have higher degree than even the largest Tier 1s (summarized in Table C.3). However, unlike the Tier 1s the five CPs edges are primarily peering edges. The five CPs also do not provide transit. Note also that the path lengths for LimeLight (AS 22822) are longer than paths observed by other ASes. This may be caused by the AS-level topology not being particularly complete for LimeLight or more likely, that Limelight's routing policies are very different from those of Appendix A.

Table C.3: Degree of five CPs in original and augmented graph with Tier 1s for comparison

| Cyclops+IXP | Cust. | Peer | Prov. | Total |
|---|---|---|---|---|
| Tier 1s | | | | |
| 174 | 2,928 | 377 | 12 | 3,317 |
| 3356 | 2,936 | 119 | 0 | 3,055 |
| 7018 | 2,407 | 135 | 0 | 2,542 |
| 701 | 2,069 | 72 | 0 | 2,141 |
| 1239 | 1,243 | 90 | 0 | 1,333 |
| CPs | | | | |
| 15169 | 0 | 244 | 3 | 247 |
| 8075 | 0 | 90 | 2 | 92 |
| 32934 | 0 | 49 | 4 | 53 |
| 20940 | 0 | 81 | 31 | 112 |
| 22822 | 0 | 567 | 10 | 577 |
| Augmented | Cust. | Peer | Prov. | Total |
| 15169 | 0 | 3,931 | 3 | 3,934 |
| 8075 | 0 | 3,927 | 2 | 3,929 |
| 32934 | 0 | 3,922 | 4 | 3,926 |
| 20940 | 0 | 3,895 | 31 | 3,926 |
| 22822 | 0 | 3,917 | 10 | 3,927 |