Practical Intrusion-Tolerant Networking

by

Daniel S. Obenshain

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

November, 2015

Abstract

In networks such as the IP-based networks that run the Internet, nodes trust one another to properly execute routing and forwarding. When a node is compromised (i.e. Byzantine failure), this trust can be exploited by such compromised nodes to launch routing attacks that can disrupt communication throughout the network. In addition, a compromised node can drop, delay, reorder, replay, or duplicate messages, or inject its own messages into the network to consume resources. While these attacks are examples related to networking, in fact, a compromised node can perform any arbitrary action. Therefore, addressing this vulnerability requires an attack-agnostic approach that maintains network functionality even in the presence of compromised nodes.

We introduce the first practical solution for intrusion-tolerant networking. Our approach guarantees well-defined semantics to applications, rather than solely routing packets, and allows multiple different semantics to coexist. Specifically, we define

ABSTRACT

two semantics that fit the needs of many applications: one guarantees prioritized timely delivery, and the other guarantees reliable delivery. We introduce a Maximal Topology with Minimal Weights to prevent routing attacks, and provide generic support for source-based routing, limiting the power of the adversary. Specifically, we discuss two source-based routing techniques: K Node-Disjoint Paths, which is resilient to $K-1$ compromised nodes, and Constrained Flooding, which provides the optimal guarantee that it will deliver messages if there exists a correct path from source to destination. We also describe the resilient overlay architecture necessary for the deployment of these ideas and to make the solution holistic, allowing the resulting system to overcome benign faults as well as malicious and resource-consumption attacks in the underlying network. We present a formal specification of the guarantees and evaluate an implementation deployed on a global cloud spanning 12 data centers from East Asia to North America to Europe.

Advisor: Dr. Yair Amir

Readers: Dr. Yair Amir, Dr. Vladimir Braverman, Dr. Cristina Nita-Rotaru

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Yair Amir. Not only is he a boundless source of energy and great ideas, he also encourages me to look beyond the problem of the moment to try to answer the problems that will still be important many years down the road. His advice has gone beyond our academic collaborations and, over the years working together, he has become my friend.

I would also like to thank Prof. Vladimir "Vova" Braverman from Johns Hopkins University and Prof. Cristina Nita-Rotaru from Purdue University (now at Northeastern University), the other members of my thesis committee. Vova added a new depth to my algorithmic knowledge, a rare gift that will serve me well through the rest of my career. Cristina's probing questions, especially when discussing the security model of the system, helped me to specify things, not just to her, but also to myself.

This work would not have been possible without the help of Thomas Tantillo, my labmate and friend. We worked side-by-side implementing our protocols in Spines.

ACKNOWLEDGMENTS

He was a tireless sounding board for my ideas and, without his help, this work would not have been what it is today. He is the best programmer I have ever had the opportunity to work with and it has been an absolute priviledge to work with such an exceptional person.

I would also like to thank everyone in the Distributed Systems and Networking Lab. It has been a fantastic place to hatch new ideas and create new things. Amy Babay, Marco Platania, and Jeffrey DallaTezza have all been excellent colleagues and Im happy to have had the opportunity to work with them.

I want to thank John Schultz from Spread Concepts LLC for his help with the implementation. As the architect of Spines, he was instrumental in guiding us through the process of adding new protocols, and he was a great resource to have for all sorts of software-related questions. Every day that he visited the lab was a day to look forward to.

The experimental evaluation wouldn't have been possible without the help of the team at LTN Global Communications Inc. Many thanks are due to Dr. John Lane and Dr. Nilo Rivera for being willing to work with us and create an interface on the production monitoring system to allow us to deploy a shadow monitoring system. In addition, Dr. Jonathan Stanton, head of engineering, ensured that the global network and systems were all set up for us to do our experiments.

ACKNOWLEDGMENTS

Prof. Wyatt Lloyd from the University of Southern California and Dr. Daniel Peek from Facebook, Inc. both read early versions of this work and gave valuable feedback. I am immensely grateful to them for their help, without which this work would not be at the level that it currently is.

Kevin Jordan from Resurgo LLC saw the work at several stages and gave feedback on its usefulness in real scenarios. I am grateful for his enthusiastic encouragement for its eventual transition to production.

I have made many friendships at Hopkins during my time here. I want to especially thank Kelleher Guerin who, since first meeting him when I was a brand-new graduate student, has grown to be one of my closest friends. Alison Morrow, Amod Jog, Anand Malpani, Ayushi Sinha, Berhardt Fuerst, Chris Paxton, Christina Garman, Colin Lea, Ehsan Basfa, Fumbeya Marungo, Ian Miers, Iyah Chen, Jonathan Bohren, Julia Fedorova, Nikita Ivkin, Nikola Fuerst, Sarah Stamper, Sean Cantrell, Steve Chestnut, Steve Vozer, Manu Madhav, Megan Hindle, Michael Rushanan, Princy Parsana, Rachel Kreimer, Ravikrishnan Jayakumar, Rob Doverspike, and Roger Que, my time here at Hopkins has been more enjoyable for having known you.

My research was supported through DARPA grant N660001-1-2-4014 to Johns Hopkins University. I would like to thank Dr. Howard Shrobe for his vision in launching the Mission-oriented Resilient Clouds (MRC) project and his confidence

in my group, and to thank Dr. Robert Laddaga for facilitating transition efforts for our technology to the real world. Thanks are due to Laurisa Goergen as well, for her work behind the scenes to ensure that everything ran smoothly. I want to especially thank Daniel Adams, who was in countless meetings on these topics with interested parties and was a tireless champion for the work.

Finally, I would like to thank my family. My mother and father, Suzanne and Norman Obenshain have supported me and encouraged me all along the way. Along with my sister, Ellen Obenshain, they made home a place that I wanted to be. I couldn't wish for a more loving, caring, and supportive family. I also want to thank my grandparents Elizabeth "Betty" and Wallace Obenshain and my uncle Paul Obenshain for encouraging my education in every possible way.

Dedication

This thesis is dedicated to my family, who have encouraged me, loved me, and supported me every day.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In networks such as the IP-based networks that run the Internet, nodes trust one another to properly execute routing and forwarding. When a node is compromised (i.e. Byzantine failure), this trust can be exploited by such compromised nodes to launch routing attacks that can disrupt communication throughout the network. In addition, a compromised node can drop, delay, reorder, replay, or duplicate messages, or inject its own messages into the network to consume resources. While these attacks are examples related to networking, in fact, a compromised node can perform any arbitrary action. Existing protection mechanisms (such as IPsec [1]) are not designed to defend against insider attacks.

As networks become more essential to critical backend services in the financial,

medical, power, cloud, and military domains, the incentive to attack the network infrastructure itself grows. When the network is down, these services cannot function normally because communication between the networked components is impaired or prevented. Keeping the network functional at all times is of paramount importance. However, this is a challenging problem given that the networks underlying critical backend services can be compromised through many possible attack vectors (e.g. software vulnerabilities, social engineering, physical access), and successful compromises of secure, even air-gapped, systems have been reported [2–4].

Designing highly-resilient networks requires an overarching, attack-agnostic approach that maintains network functionality even in the presence of compromised nodes, rather than focusing on preventing compromises and detecting them as they surface. Previous work over three decades [5–13] has investigated how to route messages through networks with Byzantine failures. Cryptographic mechanisms are used to defeat common attacks such as spoofing, tampering, and replaying. Enforcing an upper-bound on the total number of nodes in the network [5, 7] defeats attacks that require the ability to add identities to the network (e.g. Sybil attack [14]). Source-based routing, where a message's source specifies the path across the network, is used [5, 8, 10] to decrease the dependence on potentially compromised forwarders. Simultaneously sending on K node-disjoint paths (or even flooding) [5, 13] can further

decrease this dependence. Fair network resource allocation among the nodes is preserved [5], preventing compromised source nodes from consuming a disproportionate share of resources. However, challenges such as the need for programmability in the middle of the network (lacking in IP-based networks), scalability, and relatively high overhead prevented practical solutions from being fully designed and deployed.

Several advancements in recent years create new opportunities that can be leveraged towards a practical solution for intrusion-tolerant networks. First, overlay networks [15, 16] provide a convenient, effective, and flexible way to gain the needed programmability, enabling the support for rich data dissemination semantics, a defined topology, and source-based routing, necessary for practical intrusion-tolerant networks. Second, the requirements of critical backend services have become stringent enough and important enough to justify paying the cost of deploying and using intrusion-tolerant networks. Existing solutions optimize bandwidth at the cost of network service interruptions that can last for tens of seconds [12] to minutes [7]. However, critical backend services (e.g. cloud monitoring and control) require high, and in some cases continuous, network availability. Thus, network service interruptions must be minimized, even at the cost of additional resource consumption (e.g. bandwidth, memory, processing). The stringent requirements of these services, and their importance, bring more expensive mechanisms, such as redundant sending or

even flooding, into the realm of possibility.

In this thesis, we introduce the first practical intrusion-tolerant network. It is designed to work in the presence of compromised (and colluding) nodes and provides well-defined semantics to applications, beyond best-effort message forwarding. We identify fundamental semantics (e.g. timeliness, reliability, bandwidth or combinations thereof) that fit the requirements of applications with different data dissemination needs (i.e. custom storing, forwarding, and delivery rules). Our solution allows multiple different data dissemination semantics to coexist in one infrastructure and allows the semantics to be selected on a per message basis. These different semantic-specific fairness and performance guarantees are provided to the applications using them even in the presence of Byzantine forwarders and Byzantine sources (or Byzantine flows). All the communication is performed on a Maximal Topology with Minimal Weights (MTMW), which explicitly indicates all nodes and edges in the network, as well as the corresponding minimal weight allowed on each edge. The MTMW prevents routing attacks (e.g. black hole [17] and wormhole [18]) because nodes cannot lower edge weights below the minimal weight for that edge without being detected by correct nodes, and prevents Sybil attacks because new nodes cannot be introduced into the MTMW without being detected by correct nodes. We focus on networks with a single administrative domain (e.g. overlay networks) in which such an MTMW can

be easily specified. We introduce a generic mechanism for source-based routing that allows the source to specify any subgraph of the network topology for dissemination, providing the ability to specify K Node-Disjoint Paths and Constrained Flooding, as well as any other dissemination scheme. We combine these principles with standard cryptography and buffer allocation to create a practical implementation, which we make available as open source.

However, this is only part of the story. A holistic solution also requires a resilient architecture (described in Section 4.1), to be resilient to benign faults and malicious and resource consuming attacks in the underlying network. We describe how, by using an overlay approach, diverse network providers, and multihoming on each overlay node, the resulting resilient architecture can overcome benign faults in the underlying networks, malicious problems such as BGP hijacking, and resource consumption attacks such as the Crossfire [19] and Coremelt [20] DDoS attacks. With such a resilient architecture we are protected against problems in the underlying network, but we are not protected against compromised nodes in the overlay itself. The ideas presented in this thesis combine with the resilient architecture to create a truly intrusion-tolerant network.

The proposed solution is motivated by the experience of the builders of a global specialized cloud service [21] and their need for robust monitoring and control of

the infrastructure. Clouds suffer from a circular dependency: the network relies on the monitoring and control system to track and mitigate problems, while the monitoring and control system relies on that same network to function. Network compromises have the ability to disrupt communication, effectively halting monitoring and control and leaving administrators largely blind and helpless. To address this serious threat, an intrusion-tolerant network that continues to deliver monitoring and control messages in the presence of compromises is required. However, simple message delivery is not sufficient; accurate monitoring requires timely delivery semantics, while control requires reliable delivery semantics.

Inspired by these requirements, we define two specific intrusion-tolerant data dissemination semantics: Priority Messaging with Source Fairness provides prioritized timely delivery and Reliable Messaging with Source-Destination Fairness provides reliable delivery. Many applications (including monitoring and control) are served well by one or the other. Applications that need more complex semantics (e.g. online vehicle fleet tracking) can be supported by augmenting the extensible network implementation with additional semantics.

We deploy the intrusion-tolerant network implementation as an overlay network on a global cloud that spans 12 data centers from East Asia to North America to Europe. We use the priority and reliable data dissemination semantics and specify

the MTMW based on the cloud topology, with minimal weights corresponding to round-trip latencies. We use two data dissemination techniques: K Node-Disjoint Paths, where messages are forwarded along K paths which do not share any nodes other than the source and destination, and Constrained Flooding, where messages are flooded on the edges of the MTMW (not flooded on the Internet), as source-based routing schemes because they provide a good spectrum of cost/resiliency trade-offs. K Node-Disjoint Paths provides resiliency to K-1 compromises for a tunable cost (which scales with K). Constrained Flooding provides optimal resiliency guarantees for a higher cost, succeeding even if there is only a single path of uncompromised nodes from source to destination. We evaluate the deployed network in two ways. First, we send realistic traffic across the deployed network to evaluate the network in the presence of both compromised and uncompromised nodes. Second, we use the deployment as a shadow monitoring system to carry the monitoring messages of the global cloud. It ran there for several months and was used in a limited production capacity.

The contributions of this work are as follows:

- We describe the resilient architecture necessary to address the fault model we are interested in, including an overlay approach, diverse network providers, and multihoming.

- We identify and describe the principles necessary for a practical intrusion-tolerant network: rich data dissemination semantics that ensure fairness and guarantee performance in the presence of network compromises, a Maximal Topology with Minimal Weights, and a generic mechanism for source-based routing.

- We define two specific intrusion-tolerant data dissemination semantics: Priority Messaging with Source Fairness and Reliable Messaging with Source-Destination Fairness.

- We deploy and evaluate the implementation of the first practical intrusion-tolerant network on a global cloud spanning 12 data centers from East Asia to North America to Europe, and present the results. This deployment ran there for several months and was used in a limited production capacity.

- We released this implementation as open source, as part of the Spines [22] overlay messaging toolkit. Spines has been available for 12 years; the new intrusion-tolerant version was publicly released in January 2015.

Note that in the cloud where we deploy the system, since the monitoring and control traffic is a small fraction of the total traffic, the overhead cost of intrusion-tolerant monitoring and control ranges from 0.2–3.0% of the total traffic.[1] We

[1]The messaging overhead ranges from 2–3x for K Node-Disjoint Paths (with K=2) to 15–30x

consider this a tolerable overhead cost because these important applications form the infrastructure of the cloud. However, we recognize that the relatively high per-message overhead is not justifiable for all applications. The interesting open question of how to leverage an intrusion-tolerant infrastructure to protect generic application messaging, without incurring the full cost of the techniques presented here, is beyond the scope of this thesis.

1.1 Problem Description

The problem this thesis addresses is how to construct a network that can continue to deliver messages with the required semantics when under attack from a sophisticated adversary, one who has even compromised some of the nodes in the network.

Any compromised nodes are controlled completely by the adversary and may perform any arbitrary action (Byzantine fault).

The adversary may also launch attacks at the level of the underlying network, such as sophisticated DDoS attacks or BGP hijacking attacks.

We assume that the adversary is not able to break the cryptographic mechanisms we use.

for Constrained Flooding on our topology, compared with secure single-path routing. The cloud monitoring and control traffic amounts to less than 0.1% of the overall traffic in the cloud, so the resulting overhead on the cloud ranges from 0.2–3.0%.

We assume that the adversary is not able to overwhelm the computational power of a correct node.

1.2 Solution Highlights

We describe the resilient architecture necessary to overcome the fault model we are interested in. This includes using an overlay approach, taking care to ensure that the overlay topology closely approximates the underlying physical topology. It includes the use of diverse network providers, to ensure that the failure of a single ISP will not cause the intrusion-tolerant network to fail. It includes the use of multihoming, meaning each overlay node will have simultaneous service from multiple network providers, in order to make the intrusion-tolerant network very resilient to problems in the underlying networks.

We describe the Maximal Topology with Minimal Weights, which puts constraints on the actions nodes can take with routing updates. Nodes may raise the weights of the edges they have with their neighbors in response to network problems and may lower the weights when those network problems resolve, but they may not alter the weights of other edges and may not reduce the weights of edges below the original, minimal weights.

We describe K Node-Disjoint Paths, which sends messages across the network

via K paths which do not share any intermediate nodes, making the communication resilient to $K-1$ compromised nodes. We extend this with a discussion of erasure coding to reduce the cost, and a discussion of how routing updates should be handled in an intrusion-tolerant setting. We also describe Constrained Flooding, where messages are sent on every edge in the network, giving the optimal guarantee that messages will be delivered as long as there exists a correct path from source to destination.

We define two specific intrusion-tolerant data dissemination semantics: Priority Messaging with Source Fairness and Reliable Messaging with Source-Destination Fairness. Priority Messaging provides a mechanism for a source to prioritize its own messages; in the case of network contention, a source's lower priority messages will be discarded in order to maintain timely delivery of its higher priority messages. In contrast, Reliable Messaging provides eventual-path end-to-end reliability across the network, and will stop the source via backpressure if messages are not getting through.

We deploy and evaluate the implementation of the first practical intrusion-tolerant network on a global cloud spanning 12 data centers from East Asia to North America to Europe, and present the results.

To prove the feasibility of our approach in a real situation, we deployed the system

as a shadow monitoring system for all monitoring messages in the cloud for several months. This shadow monitoring system was used in a limited production capacity.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2 we define the model we will use in the remainder of the work, including the network, the failures it can experience, and the powers of the adversary. In Chapter 3, we cover other work that is related to this topic. In Chapter 4, we describe the Intrusion-Tolerant Network and how messages pass across it, beginning with the Resilient Architecture in Section 4.1, continuing to the Maximum Topology with Minimal Weights in Section 4.2, before describing K Node-Disjoint Paths in Section 4.3 and Constrained Flooding in Section 4.4. In Chapter 5, we give the two different dissemination semantics we provide: Priority Messaging with Source-Based Fairness and Reliable Messaging with Source-Destination Fairness. In Chapter 6, we evaluate the implementation of our Intrusion-Tolerant Network and describe its deployment in a real system. Lastly, Chapter 7 concludes the thesis.

A reader interested in the implementation of the ideas presented in this thesis should concentrate on Chapter 2, Chapter 4, Section 5.1, and Chapter 6, especially Section 6.1.

A reader interested in understanding the guarantees given should concentrate on Chapter 2, Sections 4.3.3 and 4.4.1, and Chapter 5, with Figures 6.6 and 6.8 providing relevant illustration.

Chapter 2

Model

In this chapter we discuss the model we will be considering for the rest of this thesis. We begin with the network model: the nodes and edges we will consider and how messages are passed between them. We then continue to the failure model, which details the actions the attacker can take and the assumptions we make regarding the attacker.

2.1 Network Model

We consider a network of nodes and edges, where each node can act as both a source that injects new messages and a message forwarder. Each node has a set of neighbors, i.e. nodes with which it can communicate directly without any intermediate

nodes in the network. A node in the network may experience crash-recovery. Nodes have limited buffers for storing messages.

All communication is authenticated. A Public Key Infrastructure (PKI) is used, where the system administrator and each node in the network have a public/private key pair, and the public keys of the system administrator and each node are known by every node. Messages are signed by the source when entering the network and HMACs are used to protect messages on each link.

Communication between nodes takes place in a Resilient Architecture (further explained in Section 4.1). This resilient architecture enables the resulting system to be resilient to benign failures in the underlying network, BGP hijacking attacks, and sophisticated DDoS attacks in the underlying network.

Communication between nodes is also constrained to a specified topology, a Maximal Topology with Minimal Weights (MTMW). This specifies the nodes and edges in the network and the costs for each edge for the purposes of routing. Nodes can use routing updates to raise the costs of their direct edges in the case of network problems and lower the costs of their direct edges once those network problems resolve. Nodes cannot lower the cost of an edge below the associated Minimal Weight, and cannot change the cost of an edge they are not on. The MTMW is discussed in Section 4.2.

Messages between nodes travel via a link-level protocol: the Proof-of-Receipt (PoR) link. The link is reliable and will block, preventing new messages from being sent until enough old messages have been acknowledged to make progress. An HMAC scheme is used to authenticate messages, preventing message tampering. In addition, a nonce is included on each message and an ack must use that nonce to prove that the message was received. This prevents a compromised neighbor from launching a denial-of-service attack (as described in previous work [23]) by causing a correct node to send faster than current link conditions can handle.

This model is appropriate for overlay networks and for physical networks with a single administrative domain (such as ISPs). These networks allow full control of the network topology. We do not consider networks composed of many administrative domains (e.g. the Internet), because there is no way for an administrator to impose a single MTMW on those networks without coordination. However, note that our overlay approach can still support networks with many administrative domains by providing a single logical administrative domain over the underlying networks.

While this work is applicable to networks with any number of nodes, the intrusion-tolerant network principles make scaling to deployments with many hundreds of nodes impractical. In our deployment, we currently have 12 overlay nodes (data centers) achieving globe-spanning coverage, and we do not see a need to scale beyond several

tens of nodes (data centers) for good coverage of the globe. Each node can support many, many clients, so this is sufficient for our purposes.

We do not consider ad-hoc networks, peer-to-peer networks without access control, or wireless networks, since new members can announce themselves and join without being authenticated by the single offline administrator [14]. This model may be appropriate in certain wireless settings if care is taken to ensure the system is protected against transmission medium denial-of-service attacks. (via directed antennae, frequency hopping, etc.).

2.2 Threat Model

A *correct node* is one that executes the network protocols faithfully. In contrast, a *compromised node* is any node that is not correct. Compromised nodes can exhibit arbitrary behavior, often referred to as a Byzantine failure [24]. For the purposes of this work, we consider that such compromises may be sophisticated and difficult to detect. Rather than detecting and evicting compromised nodes, we provide guarantees even in the presence of such compromises.

Compromised nodes may become correct again through proactive recovery, where the node is brought back from a known-good state [25]. A network with failure detection and reactive recovery (an example of reactive recovery may be found

in Sousa et al. [26]) may also do the same, however, recall that we consider the compromises to be difficult to detect. Software diversity (e.g. [27–30]) may be used to lower the chance that one exploit will result in multiple compromised nodes in the network.

A *correct edge* is one that is able to pass messages freely in both directions. In contrast, a *failed edge* is any edge that is not correct. For example, in an overlay network, this can be due to a failure of an underlying Internet path, congestion on that underlying Internet path (resulting in large amounts of loss), DDoS attacks at the underlying network, or any other reason that messages cannot be passed freely in both directions. The multihoming aspect of the Resilient Architecture we use makes such failed edges less likely (see Section 4.1.3), but still possible.

Failed edges may become correct again when the underlying problem is resolved.

Note that a node which fails due to benign, not Byzantine, problems (e.g. a power failure) can be modeled as a collection of failed edges rather than a compromised node. This may be detected and routed around, as described in Section 4.3.2. Compromised nodes can also emulate the behavior of nodes which fail due to benign problems, in which case they will be treated the same.

A *correct path* is one consisting of correct nodes and correct edges, with no compromised nodes or failed edges.

Liveness. We do not assume a specific fractional bound on the number of compromised nodes in the network. However, as a liveness condition for a given source-based routing scheme, there must exist a correct path from source to destination in the dissemination subgraph chosen by that scheme. If this liveness condition is not met, the system will remain correct, but will not be live for that source-destination pair (it may be live for other source-destination pairs). Note that a network which has lost liveness for some source-destination pair may later regain it, through some combination of proactive recovery converting compromised nodes to correct nodes and the resolution of underlying problems converting failed edges to correct edges. *K Node-Disjoint Paths* requires that at least one of the K chosen paths from source to destination is correct. *Constrained Flooding* requires that there exists at least one correct path from source to destination anywhere in the network.

Attacker knowledge. We consider complete node compromises. Such an attacker has access to all of the private cryptographic material stored at that node and can use it to generate signed messages that will verify at correct nodes. The attacker has full knowledge of the network protocols, including static and dynamic parameters. The attacker can execute modified versions of the protocols' code and even other arbitrary code.

Attacker actions. We consider the Byzantine fault model, where compromised

nodes can take any action. The attacker can eavesdrop on the traffic between correct nodes and can launch message attacks at a compromised node that drop, delay, reorder, replay, duplicate, and misroute messages in the network. In addition, the attacker can distinguish between different message types and can try to forge messages or tamper with specific fields in the headers of individual messages. The attacker can try to disseminate incorrect routing updates. The attacker can acknowledge messages that were not actually received, can refuse to acknowledge messages that were received, and can forward messages more quickly than they would normally be forwarded. A node acting as a source can inject traffic at any rate and to any destination, introducing attacker-controlled flows to the network.

The attacker can even participate in the protocol in an attempt to subvert it. For example, the protocols in this work enforce fairness between different sources or flows. The attacker can deliberately forward messages unfairly in an attempt to subvert the system.

Attacker coalition. We consider collusion attacks, where attacker-controlled nodes establish an out-of-band connection and communicate to coordinate attacks. The attacker can even form their own network of attacker-controlled nodes.

Attacker cannot overwhelm the computational power of a correct node. We assume that a correct node will be able to keep up with processing incoming

messages. This can be supported by provisioning enough computational power to process the messages that are received, including verifying the cryptographic signatures involved (note that in Chapter 6 we deploy using RSA signatures, which require less computation to verify than to sign).

Attacker resources. Compromised nodes can have large amounts of network bandwidth, memory, and computation, potentially much more than correct nodes. However, we assume the attacker is computationally bounded and cannot break the cryptographic mechanisms used by our protocols.

Note that all the guarantees we provide depend on there being a correct path from source to destination. In the overlay deployment we envision, for example, with tens of datacenters, each multihomed over multiple Internet Service Providers, and arranged in a well-connected network (see Section 4.1 for more details), the attacker will require overwhelming resources to cut all the possibilities from source to destination. Of course, if the attacker is strong enough to cut the network, then no protocol can succeed.

Chapter 3

Related Work

Prior work has investigated securing Internet routing protocols (surveyed by Papadimitratos et al. [31]), such as integrating security into BGP [32], protecting OSPF with digital signatures [33], and using Public Key Infrastructure and secret keys to authenticate routing updates in generic networks [17]. These works provide security against external attacks, but do not provide intrusion tolerance.

Several works created Byzantine gossip and peer-to-peer (P2P) protocols to disseminate information. Fireflies [34] provides an intrusion-tolerant gossip protocol to maintain full membership information in the presence of Byzantine members, which is used to support a distributed hash table. Castro et al. [13] provide secure node ID assignment, secure maintenance of routing tables, and secure message forwarding for

a structured P2P network, assuming no more than a fraction of nodes are Byzantine. BAR Gossip [35] presents a P2P application on top of a Byzantine gossip protocol that provides predictable throughput and low latency for streaming media with high probability. In general, gossip and P2P protocols provide probabilistic message delivery, which is insufficient to support strong deterministic guarantees. In addition, these works assume the underlying network provides a clique of connectivity among the protocol participants, an assumption that can be violated by network compromises.

Other work has provided basic intrusion-tolerant messaging in limited network environments. Probing and flow conservation can be used to determine if routers are behaving maliciously [6, 36], but this places a limit on the location and number of compromises, and assumes correct routing behavior can always be determined. INSENS [11] provides intrusion-tolerant routing in wireless sensor networks by leveraging wireless-specific properties, e.g. compromised nodes have a limited broadcast range.

Previous work investigated routing messages in the presence of Byzantine failures. LITON [10] protects overlay network communication using on-demand node-disjoint routes and HMACs. ODSBR [8, 12] presents a source-based routing scheme that localizes faults to a specific link using disguised probing techniques and re-routes accordingly. Authenticated Adversarial Routing (AAR) [7] successfully routes mes-

sages if even one correct path exists between source and destination, however, the limitation of only a single flow and the large initialization overhead are barriers to practical deployment. These works address Byzantine forwarders, but not Byzantine sources. In addition, none of these works were deployed in practice and they focus solely on message delivery, as opposed to guaranteeing rich semantics.

The SCION work [37] provides a method to protect routing, even in the presence of some compromised nodes. It does this by allowing the source and destination to work together to select a path. SCION works based on one or a few trusted ISPs, which send messages out, hopping ISP to ISP, to the source and destination of a flow. The source and destination can then each find several disjoint paths back to the trusted root, through different sets of ISPs. They can then work together to select a path. This scheme enables SCION to be resilient to a single ISP failure that is not in the trusted root, because the source or destination can choose another path to the root. Similarly, for the same reason, the scheme is resilient to sophisticated DDoS attacks such as Crossfire [19] or Coremelt [20], because such an attack would have to be large enough to affect all the paths found back to the root, raising the bar for the attacker. This scheme also does not use BGP routing, making it resilient to BGP hijacking. However, as the SCION work is a "clean-slate" design, it is not feasible to deploy it on the Internet we have today.

The work most closely related to ours is Perlman's [5, 9], which provides authenticated link state routing in the presence of Byzantine failures. It floods routing updates with source-specific buffers to provide fairness and proposes using node-disjoint paths for data, but does not support arbitrary source-based routing schemes. Perlman bounds the number of nodes in the network to address Sybil attacks, whereas we specify the MTMW to defeat Sybil attacks and prevent routing attacks. The main difference with our work is that Perlman focuses solely on routing messages, and only provides delivery guarantees to applications at a higher level (end-to-end). In contrast, we guarantee multiple different data dissemination semantics to applications, such as end-to-end reliable delivery via back-pressure (Section 5.3), and provide this in the same layer, allowing our protocols to react much more quickly, providing better service. Finally, Perlman's work is in the context of the physical network layer, and has practical barriers to deployment in modern environments.

An important aspect of our work is the resilient architecture we use when deploying our system, further detailed in Section 4.1, which we briefly summarize here for the purpose of comparison. We use an overlay approach, which enables us to route around connectivity problems in the regular Internet [15, 16], and we use careful node placement to increase the chances that the same underlying physical link does not affect multiple overlay links. This enables the system to quickly react and route

around even state-of-the-art DDoS attacks such as the Crossfire [19] or Coremelt [20] attacks. To this, we advocate adding "course-grained diversity" in terms of which network provider a node connects to. This course-grained diversity can be carefully placed on the different nodes by using my own previous work [38, 39], which models the problem as a flow problem and uses Mixed-Integer Programming to maximize the chance that a non-compromised path exists from a source client to a destination client. Beyond this, we advocate using "multihoming" to enable each node to have simultaneous service from multiple network providers, ensuring that each logical overlay link can exchange messages as long as some choice of network provider for the first node on that link is able to communicate with some choice of network provider for the second node on that link. Lastly, we advocate using "fine-grained" diversity, as described in my previous work [25], to ensure that the same exploit cannot be used to compromise multiple overlay nodes.

We compare the salient pieces of related work in Table 3.1. ODSBR, LITON, AAR, and the work in this thesis can all be deployed at the overlay level, making them feasible to deploy on the Internet, while SCION advocates a "clean-slate" design and Perlman's work is in the context of the physical network layer. All the related work protects against link-level tampering with either HMACs or signatures. LITON, ODSBR, and AAR do not give any discussion of how to deploy such a system on

		LITON	ODSBR	AAR	SCION	Perlman	Our Work
Resilient Architecture	Feasibly (i.e. Internet) Deployable	✓	✓	✓			✓
	Protect against link-level tampering	✓	✓	✓	✓	✓	✓
	Protect against ISP failure				✓		✓
	Protect against sophisticated DDoS attack				✓		✓
	Protect against BGP hijacking				✓		✓
Dissemination	Overcomes Byzantine Forwarders	✓	✓	✓	✓	✓	✓
	Overcomes Byzantine Sources					✓	✓
Semantics	Guarantees Semantics						✓

Table 3.1: Comparison of Related Work

a resilient architecture (see Section 4.1), making them vulnerable to ISP failure, sophisticated DDoS attacks, and BGP hijacking in the underlying network. In contrast, SCION was built specifically to address those kinds of problems in the underlying network, and does address them. Perlman's work is in the context of a physical network, so, in contrast to an overlay approach, it cannot use multiple ISPs to be resilient to any one ISP failing. If clients on the network described in Perlman's work behave maliciously, they can launch sophisticated DDoS attacks that can consume links in the network. That work supports multiple disjoint paths, making this harder for the attacker, but not at the level we describe in Section 4.1. In a follow-up work [9], Perlman describes scaling that work, including sending between different networks over the Internet, which would be vulnerable to a BGP hijacking

attack if any intermediate network is not using that scheme. All the related work addresses Byzantine forwarders, but only ours and Perlman's address Byzantine sources. The other works provide message delivery, but do not provide semantics in the middle of the network in the way that this work does. Of the other works, Perlman's comes the closest in that the follow-up work discusses backpressure to provide reliability, but only in the context of routing between different networks, not in the context of routing within a network.

Chapter 4

The Intrusion-Tolerant Network

The Intrusion-Tolerant Network has three aspects: the resilient architecture which makes the network resilient to benign problems and attacks from outside the network, the dissemination methods which choose the paths across the network that each message will take, and the semantics that are provided for the messages via the dropping and forwarding rules at each node. In this chapter, we describe the resilient architecture and the dissemination methods. The protocols that provide the semantics are described in Chapter 5.

We first introduce the resilient architecture which our intrusion-tolerant network needs to address the fault model in which we are interested. This resilient architecture enables the system to overcome benign network problems and even attacks such as

sophisticated DDoS attacks and BGP hijacking attacks. While the purpose of this thesis is to create a network that tolerates intrusions *inside* the network, a complete and useful network must also tolerate problems which are not intrusions, including attacks from *outside* the network.

Second, we describe dissemination methods which are able to pass messages across the network *even if there are intrusions inside the network*. We introduce the Maximal Topology with Minimal Weights, which defines the topology over which the messages will travel. We describe K Node-Disjoint Paths and give the guarantees for it. In that scheme, we describe a way to use erasure coding to reduce the cost of sending messages. We also describe how routing updates are used in this context to route around benign problems in the network. Lastly, we describe Constrained Flooding and give its guarantees.

4.1 Resilient Architecture

The purpose of this thesis is to create an intrusion-tolerant network, which tolerates intrusions *inside* the network, but for such a network to be useful it must also be resilient to problems *outside* the network. We explain how to build such a resilient architecture in this section, with three salient points: using an overlay, using multiple network providers, and using multihoming.

It is possible to implement some of these ideas at the physical network level, perhaps by using multiple fiber connections and programmable routers. However, we assert that it is only practical to implement them at the overlay level. Overlays allow us to control the path messages take through the middle of the network, provide us with the programmability needed to implement our protocols, and allow us to leverage the existing Internet infrastructure, reducing cost.

In addition, it is necessary to use diverse network providers. A Byzantine Fault Tolerant (BFT) protocol which does not use software diversity to ensure that the same exploit will not work on all replicas is not useful (discussed in previous work [40], including my own [25]). In the same way, an intrusion-tolerant network built on top of a single network provider is also not useful, because if that network provider experiences a wide-scale failure, the entire intrusion-tolerant network will fail. Therefore, we must use multiple network providers to create our resilient architecture, and must go even further to use multihoming so that each node can simultaneously use multiple network providers. These considerations, when combined with the overlay approach, allow the resulting overlay network to be as strong as its strongest underlying network, or even stronger in some cases (e.g. if no underlying network reaches from source to destination, the overlay may still be able to use multiple underlying networks to reach from source to destination).

Furthermore, it is necessary to use software diversity between the different nodes of the network to ensure that an exploit which successfully compromises one node is unlikely to compromise another node. In addition, proactive recovery should be used to periodically rejuvenate nodes to remove undetected compromises, preventing the attacker from slowly compromising more and more of the system over time. Both are described in my prior work [25].

We end the section with a discussion of how this architecture is resilient to sophisticated DDoS attacks and BGP hijacking attacks.

4.1.1 Overlays

Overlay networks have been shown to improve the resilience of Internet communication by routing around problems at a higher level [15, 16]. If there is a problem with Internet connectivity, an overlay can bypass the problem by routing around it, sending the message on the Internet between different overlay nodes before finally delivering it at the destination. In order to provide resiliency, the overlay topology must be constructed with redundancy, to ensure that any source may send messages to any destination via multiple disjoint paths. Otherwise, a single point of failure may cut off communication. An example overlay is presented in Figure 4.1, with the overlay network and topology depicted on top and the underlying network depicted

underneath.

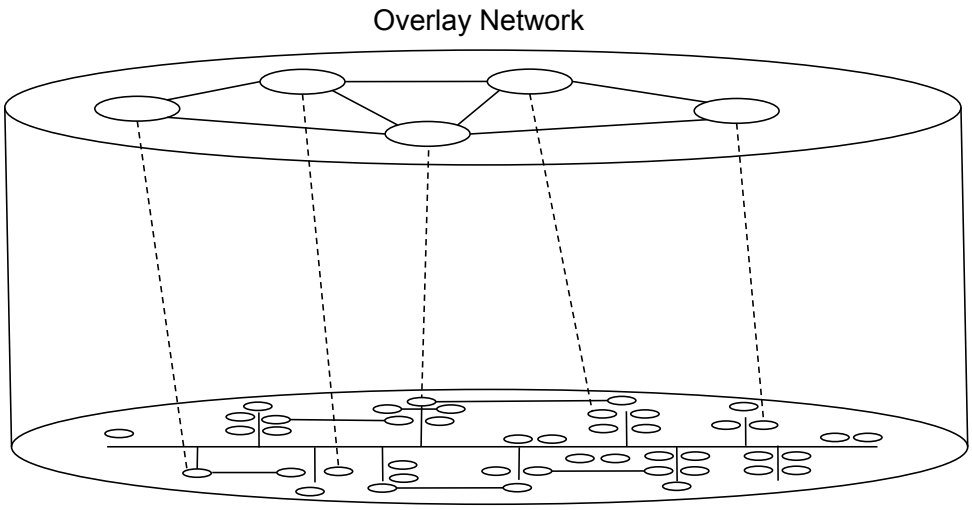

Figure 4.1: A depiction of an overlay network.

However, using an overlay with a redundant topology is not sufficient. We must also ensure that the redundancy represented in the overlay topology reflects actual physical redundancy in the underlying Internet infrastructure. This is done by placing the nodes in datacenters in strategic locations, where many ISPs meet. ISPs invest strongly in some small number of strategic datacenter locations, for example laying independent fiber connections between them. We are able to leverage that investment to cause our topology to follow, more or less, the underlying Internet structure. By deploying in this manner, we are able to increase the likelihood that the Internet links underlying the overlay links will be physically disjoint, reducing the risk of any

underlying failure affecting multiple overlay links. An example of this risk is given in Figure 4.2, where the left image (a) shows two overlay links which are entirely disjoint at the underlying level and the right image (b) shows two overlay links which partially overlap at the underlying level.

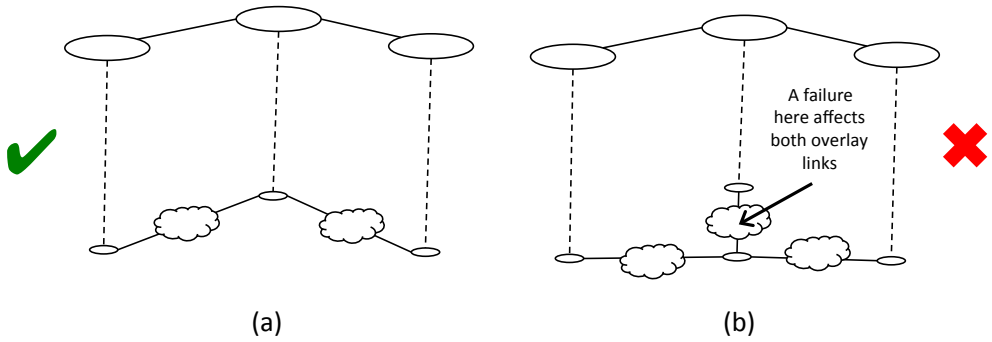

Figure 4.2: In (a), one underlying network failure results in one overlay network link failure. In (b), one underlying network failure can result in multiple overlay network link failures.

The topology should be constructed of short overlay links. For our purposes, we consider "short" to correspond to latencies on the order of 10 ms. Short overlay links are more reliable in general, as the messages have fewer opportunities to encounter the kind of congestion which would cause them to be delayed or dropped. In the same way, two overlay links to two different destinations have less opportunity to overlap with each other at the physical level if they are short, again reducing the risk of an underlying failure affecting multiple overlay links. Besides the resilience benefits

we gain, short overlay links also have performance implications: previous work [41] has showed that using short hops can greatly increase the quality of VoIP traffic.

Note that including too many overlay links in the topology, e.g. using a clique, can be detrimental. In that case, because many of the links will be long and will travel across the same physical regions crossed by other links, it is likely that links will overlap each other at the underlying layer. This increases the risk that an underlying failure will affect multiple overlay links, greatly reducing the usefulness of the overlay. In addition, as a practical consideration, each additional link adds cost to the system in terms of routing complexity and stored state, so additional unnecessary links should be avoided.

4.1.2 Use of Diverse Network Providers

To ensure that a wide-scale failure in one network provider (e.g. due to misconfiguration) does not take down our entire network, we must use diverse network providers. Otherwise, if we depend on only a single network provider and that network provider experiences a wide-scale failure, possibly due to an attack, it would take down our entire network.

My own previous work investigated how to apply diversity to such an overlay topology to make it more resilient by assigning one of a small set of software variants

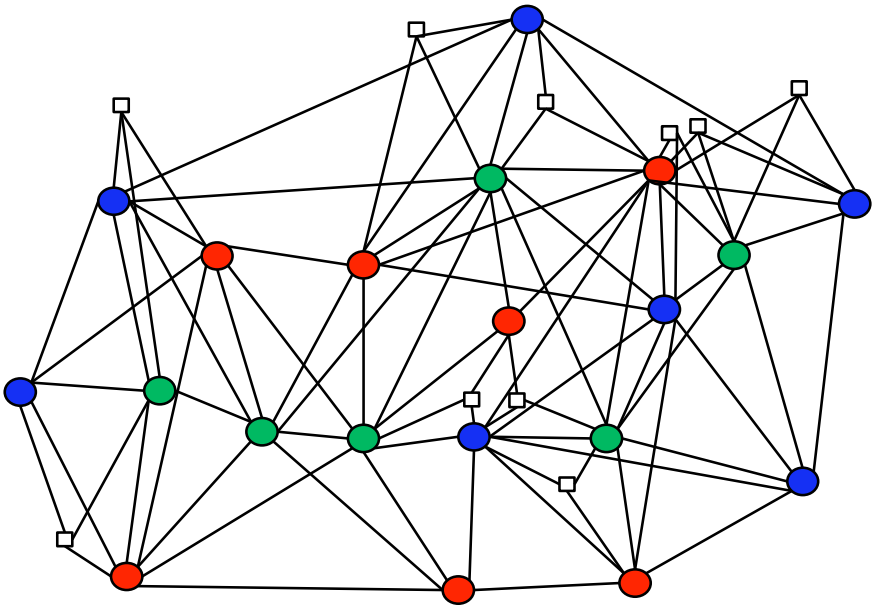

Figure 4.3: An overlay network using diverse network providers. Each color represents a different network provider.

to each node [38, 39]. In that work, each client connects to a few nodes in the network. Variants are assigned to the nodes in the network in order to maximize the expected client connectivity by maximizing the number of client pairs which are able to communicate under different failure scenarios (e.g. first variant compromised, second variant compromised, etc.). An example of this can be seen in Figure 4.3. The problem is reduced to a flow-based problem and solved with a Mixed Integer Programming (MIP) approach. If we consider the diverse variants to be network

providers (e.g. Comcast, Level 3, Verizon), we are able to use the same ideas to make our overlay network more resilient, even to the point of being resilient to one or more network providers going down.

4.1.3 Multihoming Using Diverse Network Providers

The use of diverse network providers is necessary, but using one network provider per overlay node means that the failure of a network provider will result in the failure of every overlay node using that network provider. This results in entire sections of the overlay becoming unusable. Instead, each node can have simultaneous service from multiple network providers, via "multihoming". An example of this can be seen in Figure 4.4, with multiple colors at a node representing simultaneous service from multiple network providers. Then, each logical overlay link can pass messages using any pair of service providers available, one at each endpoint. In practice, pairs that use the same provider at either end (the message is "on-net") are likely to be more resilient than other pairs, as on-net messages do not encounter BGP routing, but any pair can be used. As a result of this multihoming, the overlay link will be able to successfully pass messages even if only one such pair is able to communicate.

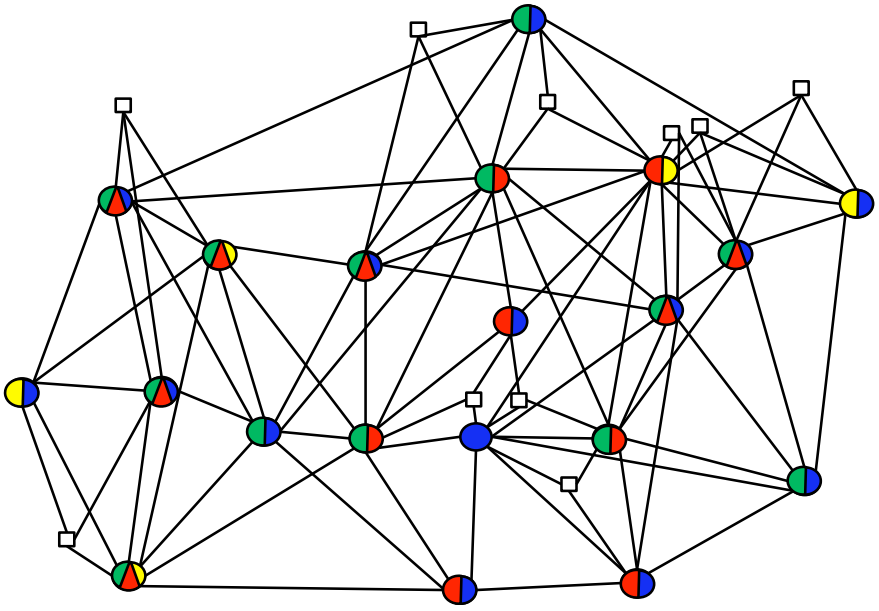

Figure 4.4: An overlay network using multihoming. Each color represents a different network provider; a node with multiple colors corresponds to that node being multihomed on all those network providers. Note that not all the nodes in the figure have the same number of network providers.

4.1.4 Bringing it Together

We have described the different components of our resilient architecture: an overlay network that matches well the underlying infrastructure, the use of diverse network providers, and the use of multihoming to let each overlay node sit on multiple network providers simultaneously. These must be used in combination to provide a resilient architecture.

An overlay approach by itself is insufficient. If the overlay is placed over a single ISP, a failure of that ISP will result in the failure of the overlay. Similarly, if the overlay is placed over multiple ISPs in a way that is vulnerable to BGP hijacking or Crossfire [19] or Coremelt [20], those attacks can disrupt communication across the overlay.

Multihoming by itself is insufficient. If a non-overlay approach only operating end-to-end from source to destination is used, there is no control over the route through the middle of the network. Even if this approach leverages multiple providers at either end, without controlling the route in the middle these routes are at the mercy of Internet paths and Internet re-routing, which previous work [15, 16] has shown to be problematic, since Internet reroutes can take 10s of seconds to minutes to converge in the event of a failure. Furthermore, all those routes could pass through the same physical area. This is made more likely by the fact that different network providers often use the same right-of-way to provide network service. So, then, all the routes could be subject to the same physical problems (e.g. a backhoe might destroy a trunk line containing fiber from each of the providers).

4.1.5 Attack Resilience

To understand how resilient the resulting topology is, here we consider how it overcomes a state-of-the-art DDoS attack and how it overcomes BGP hijacking attacks in the underlying network. Note that a naïve overlay, without the considerations outlined above, is not resilient to these attacks.

State-of-the-art DDoS attacks such as the Crossfire [19] or Coremelt [20] attacks can isolate a target area from the rest of the Internet by identifying and attacking the paths calculated by the Internet's routing algorithms. Specifically, this kind of attack can ensure that a link on each path leaving the target area is overwhelmed by many small attacker-controlled flows, resulting in very poor performance for legitimate flows. Normally, such poor performance on a link will eventually cause OSPF or BGP to route around the link, once enough consecutive control messages ("hello" messages for OSPF, "keepalive" messages for BGP) are missed. However, the attack can be made persistent by switching between different links on that path, to prevent OSPF or BGP from detecting the link as having failed. As a result, while no one link is unusable for long, the path as a whole is always unusable. This is illustrated in Figure 4.5. In the context of this thesis, these kinds of attacks can be used to attack the Internet paths found for the overlay links between different nodes. However, our resilient architecture makes it very difficult for an attacker to cut off communication between

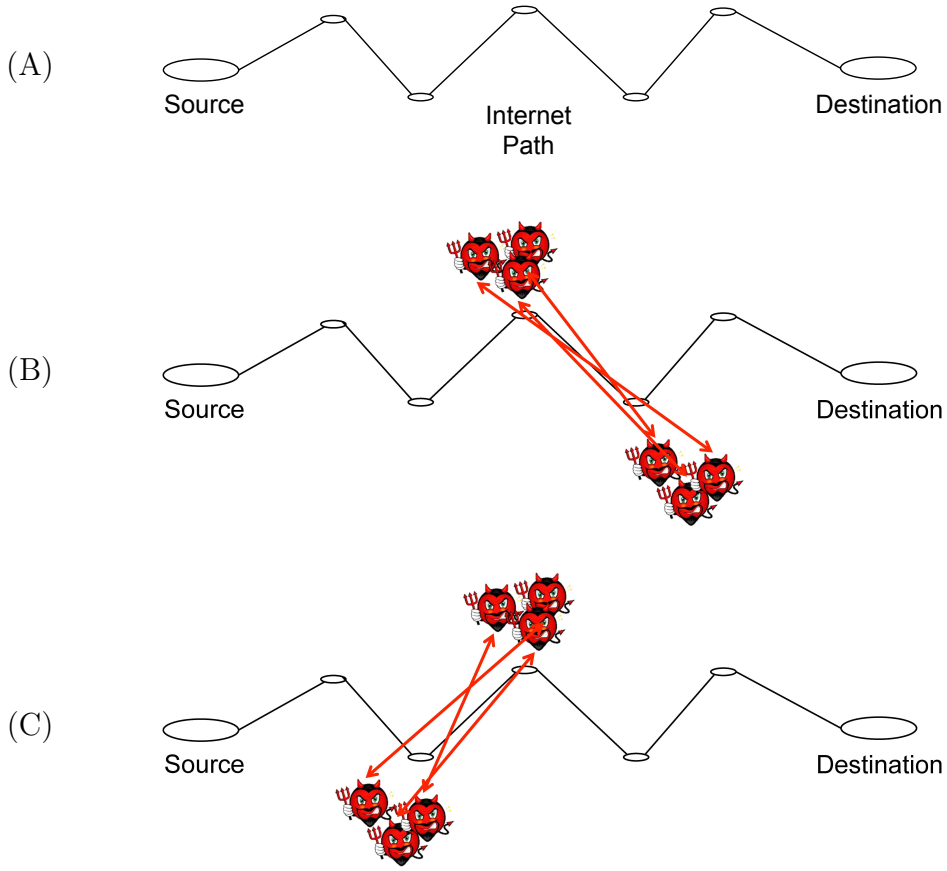

Figure 4.5: An illustration of Crossfire- and Coremelt-style DDoS attacks. By switching between (B) and (C) above, the attacker can cut off communication between source and destination, while simultaneously ensuring that the path will not be rerouted because no one link is unusable for long.

a source and a destination, via three aspects. First, because of the way the overlay is constructed, not only does each Internet link outage affect only a single overlay link, but because we choose our overlay links to be short, each overlay link contains only a few Internet links, constraining the attacker's ability to switch attacked links to avoid detection. Second, when such an attack succeeds, it will succeed only on a single provider; with multihoming, that overlay link will still be able to pass messages. Third, even if the attacker succeeds in cutting off communication on enough providers to cut that overlay link, the overlay will route through another part of the overlay. As a result, in order to cut the overlay network between source and destination, the attack must affect multiple overlay links (enough to cut the network), and attack each on multiple providers (enough to cut communication on that overlay link). This raises the bar for the attacker significantly.

In the event of a BGP hijacking attack, Internet routes which cross multiple network providers will be affected, but routes that stay within a single network provider should not be affected. Overlay links which have the same provider available on either end, then, will still be able to pass messages. The overlay nodes actively move a message from one network provider to another, by receiving the message on a link with one network provider and sending it on a link with a different network provider, as illustrated in Figure 4.6. Since the message does not use the routes

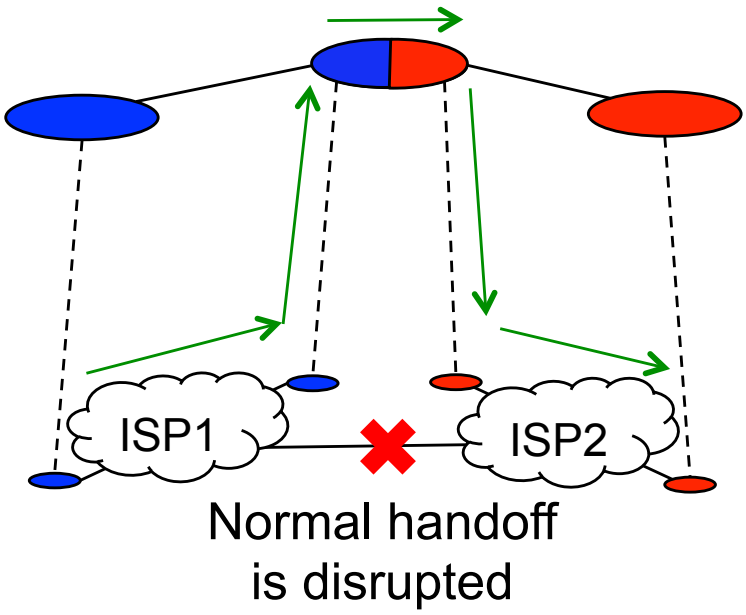

Figure 4.6: Switching between underlying networks (ISPs) at the overlay level, which overcomes BGP hijacking.

calculated by BGP, it is able to overcome the BGP hijacking attack. Similarly, if there is a malicious attack on OSPF routing in one of the underlying network providers, the use of multihoming ensures that the overlay network will use other providers, which should not be affected.

Of course, if the attack is severe enough to take down the whole Internet everywhere, no protocol will be able to succeed.

Note, however, that this resilient architecture is only resilient to attacks from the outside; it is not resilient to compromised nodes that are part of the overlay network (this is addressed in the remaining sections of this chapter, and the rest of this thesis).

4.2 Maximal Topology with Minimal Weights

Each node trusts an offline system administrator to initially distribute a signed Maximal Topology with Minimal Weights (MTMW). The MTMW specifies the nodes in the network, the edges between them, and the minimal weight allowed on each edge. Weights can represent any real-world cost (e.g. latency) and routing decisions minimize weight. Communication is limited to the topology in the MTMW, i.e. nodes only accept messages from their direct neighbors. Any pair of nodes that are not direct neighbors must communicate with each other via other nodes.

Each node monitors the edges with its neighbors, raises the weight of those edges when problems arise, lowers the weight of those edges when problems resolve, and disseminates signed routing updates.[1] A node is not allowed to change the weights of non-neighboring edges or decrease the weight of any edge below the minimal weight

[1]We use rate-limiting and overtaken-by-event techniques to protect the network by limiting the impact from spurious routing updates from compromised nodes.

specified in the MTMW. If a node attempts such an action, it is detected, that node is considered compromised, and that update is ignored. As a result, routing attacks (e.g. black hole [17] and wormhole [18]) are prevented, because routing updates which would otherwise have disproportionately attracted traffic towards the node that issued the update are disallowed and ignored.

To change the network topology, the offline system administrator can update and re-distribute the MTMW as needed, using a monotonically increasing sequence number. Nodes accept MTMWs with higher sequence numbers than the current MTMW. The offline system administrator can be converted to an online Certificate Authority, if desired.

4.3 K Node-Disjoint Paths

In the K Node-Disjoint Paths dissemination method, each message is sent across the network K times, via K distinct paths, such that no two paths share any nodes, other than the source and destination.

The K separate paths are specified on the message via a bitmask, with each bit in the bitmask corresponding to one edge in the topology. This enables very fast processing of messages: a node receiving a message will XOR the bitmask on the message with a reference bitmask specifying only its edges to immediately determine

on which outgoing edge or edges it should forward the message.

The K separate paths are computed by using Ford-Fulkerson [42], a max flow algorithm. This algorithm is illustrated in Figure 4.7. First, we begin with the topology, consisting of nodes and edges, with a weight (or cost) for each edge (A). Next, each undirected edge in the original topology is split into two directed edges, both with the same cost (B).

Since we desire not just edge-disjointness but rather node-disjointness, we apply a method from literature [43], splitting each node into two parts (C). Edges are arranged such that all edges destined to a node arrive at the first part of that node (e.g. A_1) and all edges leaving a node leave from the second part of that node (e.g. A_2). The exception to this is the new edge (with cost 0) that joins the two parts (e.g. from A_1 to A_2). This ensures that a path arriving at a node must traverse that edge, so any edge-disjoint set of paths on the altered topology must define a node-disjoint set of paths on the original topology.

Next, we run the Ford-Fulkerson algorithm on the resulting topology. For this, we add a residual edge (dashed line) opposite each edge. Each edge has a cost as before and the corresponding residual edge has the opposite cost (negative value). In addition, each edge has a capacity of 1, which is not currently consumed, and each residual edge has a capacity of 1, which is consumed. The resulting topology is

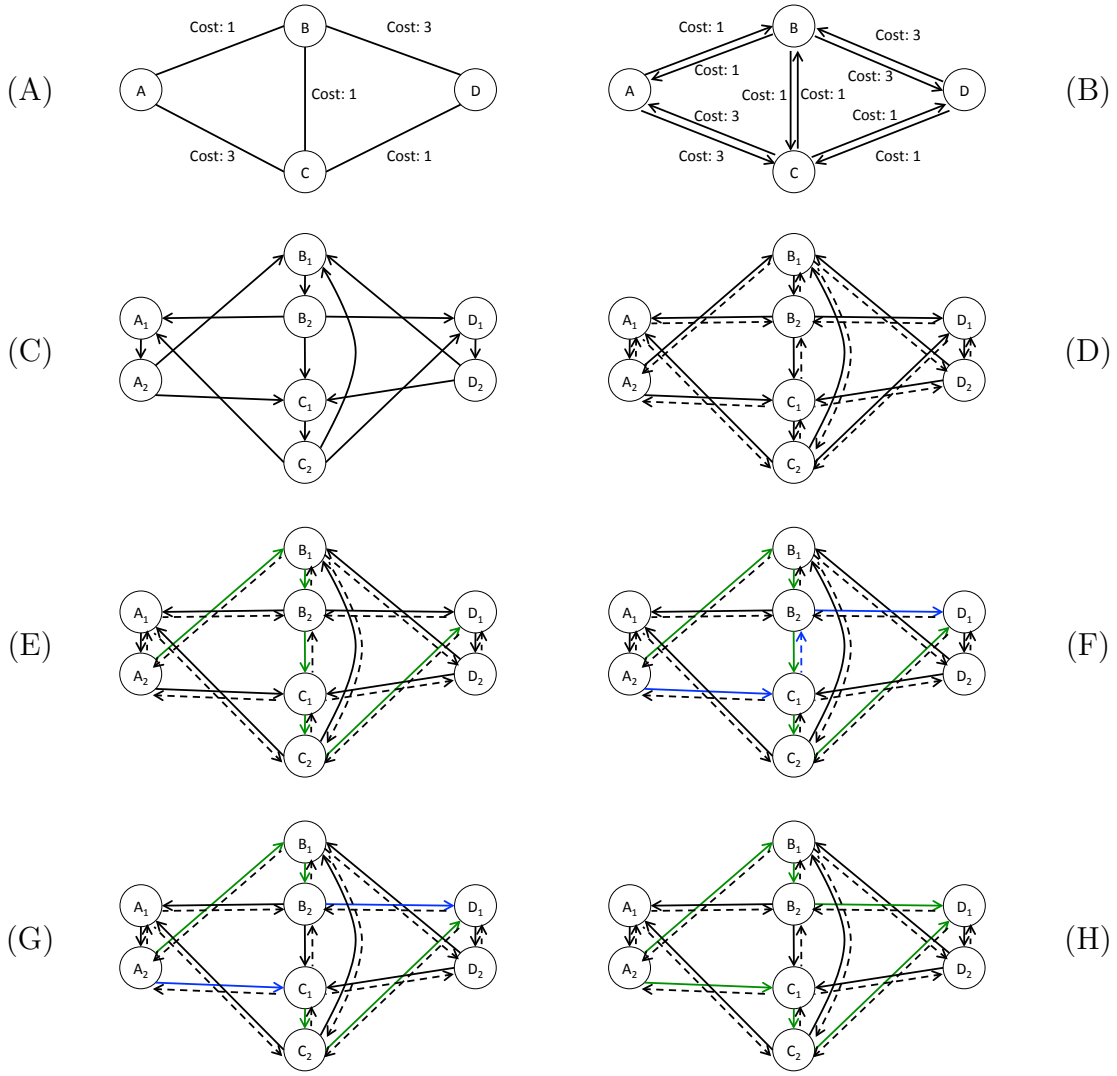

Figure 4.7: Steps when computing K Node-Disjoint Paths. Cost values are omitted after step (B) to avoid cluttering the figure.

depicted in (D), but costs and capacities are omitted to avoid cluttering the figure.

When running a shortest path algorithm on the resulting topology, we must run it from the second part of the source node (e.g. A_2) to the first part of the destination node (e.g. D_1), because there is only a single edge leaving the first part of the source node and there is only a single edge arriving at the second part of the destination node. In our implementation, we chose to use the Bellman-Ford algorithm, as it handles edge weights with negative values. Running the algorithm on this topology yields the path A_2, B_1, B_2, C_1, C_2, D_1, as can be seen in (E), for a total cost of 3. Each edge used is now in use (capacity 1 out of 1), and the corresponding residual edges are now available to be used (capacity 0 out of 1).

A second application of the shortest path algorithm yields the path A_2, C_1, B_2, D_1, for a total cost of $3 + (-1) + 3 = 5$, as seen in (F). Examining the solution, we see that both the edge from B_2 to C_1 and the corresponding residual edge have been used. We discard both from our solution and reset them, with the edge having capacity 0 of 1 and the residual edge having capacity 1 of 1 (G).

After the shortest path algorithm has been applied K times, the resulting edges are the solution (H). This solution has the property that the total cost across all K paths is minimized. These results are then cached and used for all messages from that source to that destination with that K, to avoid unnecessary computation.

For some applications, it would be preferable to minimize the worst of the K paths, balancing the paths. This would ensure that the worst case message route would be minimized. Unfortunately, such a solution is NP-hard.

4.3.1 Erasure Coding on K Node-Disjoint Paths

The cost per message can be reduced by the use of erasure coding. If K symbols are created using a scheme that requires n of these symbols to be recovered to recreate the original message, then these K symbols can be sent on K node-disjoint paths, creating a scheme resilient to $K - n$ compromises.

For example, if K is chosen to be three, a simple XOR scheme can be used to create three symbols: the first half of the message, the second half of the message, and the XOR of the first and second halves of the message. In this case, recovering any two of these symbols is sufficient to recover the message. These three symbols can be sent over three node-disjoint paths, ensuring the system is resilient to one compromise with an overhead of 50%. This is not for free, however; it ensures the message will arrive with the latency of the slower of the two correct paths.

4.3.2 Routing Updates

As network conditions change, nodes will issue routing updates. These updates must conform to the Maximum Topology with Minimal Weights. In response to network problems (e.g. loss on a link), a node will raise the weight of the corresponding link.

Other nodes in the network will use the worse of the weights reported by the nodes at either end of an edge when computing the K paths. This ensures that an edge between a correct node and a compromised node will be penalized with at least as much weight as the correct node has assigned to that edge.

Upon receiving a routing update, a node will verify that it is a valid update: checking the cryptographic signature, verifying that it conforms to the Maximal Topology with Minimal Weights, and verifying that the node issuing that update is only changing the weights of its own edges. Upon receiving a valid update, a node will invalidate its cache of stored K paths calculations. When this node introduces new data messages to the system (acting as a source), this will trigger a recalculation.

Because this recalculation is computationally expensive, routing updates are rate-limited. Correct nodes will not issue routing updates more often than some time interval, defined by the administrator. In addition, correct nodes will not forward routing updates from other nodes more often than that same time interval. If new

routing updates arrive while an old routing update is waiting to be forwarded, the updates are handled in overtaken-by-event fashion.

An interesting open problem considers whether this cache invalidation is necessary. If it is possible to efficiently compute whether a given routing update affects an existing K paths calculation, then it would be possible to invalidate the cache selectively, only discarding calculations that are no longer correct given the new edge weights.

4.3.3 Guarantees

This K Node-Disjoint Paths dissemination method enables the service to tolerate $K-1$ intrusions anywhere in the network and any number of failed edges, as long as there exist K node-disjoint paths across the topology after the failed edges are removed.

Note that, if fewer than K node-disjoint paths exist across the network after the failed edges are removed, the max flow algorithm will be unable to find the full K paths and will only be able to send with fewer paths, tolerating fewer intrusions.

Theorem 1. *For a network $\{V, E\}$ with a set of compromised nodes $C \in V$ such that $|C| < K$ and a set of failed edges $F \in E$ such that the network $\{V, E \setminus F\}$ supports K node-disjoint paths from source S to destination D, eventually the reroute updates will cause at least one of the K node-disjoint paths from S to D to be a correct path.*

Proof. Each of the failed edges F must either be between two correct nodes or not between two correct nodes. If it is not between two correct nodes we can disregard it, because at least one of the nodes involved in that edge is compromised and can halt communication on that edge regardless. A failed edge between two correct nodes will be detected as failed and both nodes will generate and flood a routing update including the failure of that edge.

The source will compute and use K node-disjoint paths. Since the number of compromised nodes is less than K, at least one of the paths must have no compromised nodes on it. If that path has a failed edge on it, that failed edge must have a correct path back to the source from at least one of the two nodes involved in that edge, ensuring that the source will eventually receive a routing update about that failed edge. Upon receiving that update, the source will compute a new set of K node-disjoint paths and continue. Note that, by the assumptions of the theorem, the source can always compute K node-disjoint paths across the network.

This process will terminate eventually, at which point at least one of the K node-disjoint paths will not have any compromised nodes on it. That same path must also have no failed edges on it, otherwise the process would not have terminated yet. So, at least one of the K node-disjoint paths must be correct, not having any compromised nodes or failed edges on it.

Note that this relies on routing stability and may not hold if the edges switch rapidly between correct and failed states. This can be addressed at the source by penalizing failed edges for longer and longer periods of time.

□

4.4 Constrained Flooding

Messages can also be sent across the network via flooding. In this case, every bit in the bitmask on that message is set. While K Node-Disjoint Paths can overcome $K-1$ compromises, Constrained Flooding provides a better guarantee: if there is a correct path from source to destination, messages will flow from source to destination. This scheme was first introduced in my previous work [44].

To understand why these guarantees differ, consider the topology in Figure 4.8. K Node-Disjoint Paths can be found from source to destination for values of K up to four (the red, blue, green, and purple paths), but each of these paths will fail as each contains a compromised node. K Node-Disjoint Paths with K greater than four is not possible on this topology. Constrained Flooding, however, will also send messages on the black edges in the middle of the graph, since Constrained Flooding uses all edges. In this way, messages are still able to flow from source to destination.

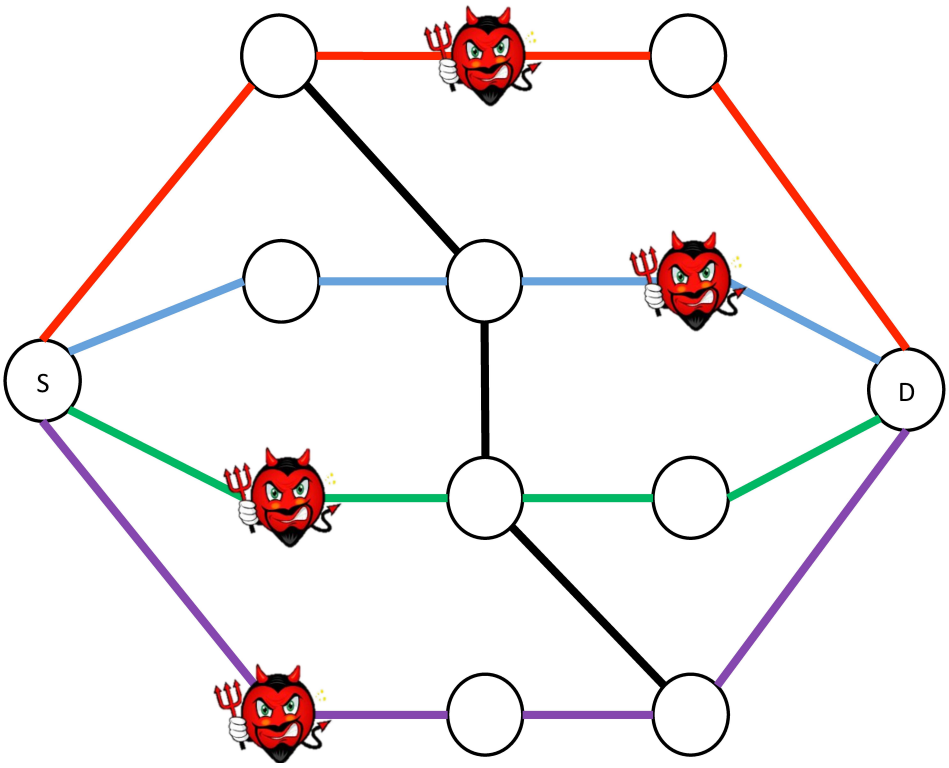

Figure 4.8: A situation where K-Paths fails but Constrained Flooding is able to succeed.

4.4.1 Guarantees

The Flooding dissemination method enables the service to tolerate any number of compromised nodes and failed edges, as long as there exists a correct path from source to destination.

Note that this guarantee is optimal. If a correct path does not exist then no protocol can succeed, because if a correct path does not exist, then a combination of

compromised nodes and failed edges have combined to cut the network. For simple message delivery, this is true both in our model (in which edges are correct if they are able to pass messages freely in both directions and failed otherwise, see Section 2.2) and in a weaker model that allows for edges that can pass messages in only one direction.

However, to provide the higher-level semantic guarantees in which we are interested and which we specify and prove in the next chapter we rely on bi-directional communication on the edges. Providing meaningful higher-level semantic guarantees in a model that allows for edges that pass messages in only one direction is beyond the scope of this thesis.

Chapter 5

Semantics and Protocols

In this chapter, we present two specific intrusion-tolerant data dissemination semantics: Priority Messaging with Source Fairness and Reliable Messaging with Source-Destination Fairness. The first provides prioritized timely delivery, and the second provides reliable delivery. These semantics ensure fairness and guarantee performance to the applications using them, even in the presence of compromised nodes. Many applications are served well by one of these two semantics, but if other semantics are desired, they can be added to the extensible implementation.

We first present the building blocks underlying these two different semantics. Next, we discuss Priority Messaging with Source Fairness and give the safety, liveness, and guaranteed throughput properties for it, for both the Flooding and K-Paths

variations. Then, we discuss Reliable Messaging with Source-Destination Fairness and give the safety, liveness, and guaranteed throughput properties for it, for both the Flooding and K-Paths variations.

5.1 Building Blocks

5.1.1 Proof-of-Receipt (PoR) Link.

We implement a link that assists the intrusion-tolerant messaging techniques by allowing them to abstract neighbor communication. The link provides TCP-fair reliable in-order communication and maintains cryptographic authentication and integrity for all messages sent on the link in a manner similar to DTLS [45]. The destination of the link is required to acknowledge messages with a proof-of-receipt, using a cumulative nonce method [23], to defeat denial-of-service attacks that acknowledge unreceived messages to drive the sender arbitrarily fast.

5.1.2 Cryptographic Mechanisms

To provide cryptographic authentication, we use the RSA [46] implementation from the OpenSSL library [47] for public-key cryptography. For the PoR Link, we use

the Diffie-Hellman [48] and HMAC [49] (using SHA-256 [50]) implementations from OpenSSL. Using their public/private keys, two direct neighbors run an authenticated Diffie-Hellman key exchange to establish a shared secret key for link-level message integrity.

5.1.3 Fairness

Fairness for bandwidth and message storage is enforced at correct nodes. For the purposes of the analysis below, we assume that each node can store b messages for each source (for Priority Messaging) or flow (for Reliable Messaging) in the network. Priority Messaging and Reliable Messaging use a round-robin scheme to serve actively sending entities fairly, and each maintains fair storage (either dynamically or statically, respectively).

5.2 Priority Messaging with Source Fairness

Priority Messaging with Source Fairness (for short, Priority Messaging) is motivated by the real-time demands of monitoring systems, which distribute continuous streams of messages across a network. Some messages are more important than others,

as they contribute more critical information to produce an up-to-date picture. When resources are constrained, Priority Messaging maintains real-time delivery of the more important messages by potentially never delivering some of the less important messages. Within these timeliness constraints, Priority Messaging is as reliable as possible.

Overview. We capture importance and timeliness information by requiring each source to assign a *priority* and an *expiration time* to each message it injects. Each message also contains the identity of its source, a monotonically increasing sequence number, and a digital signature of the message by its source. Only messages with a valid signature are accepted. Duplicate messages are identified by sequence number and simply discarded. Given the limited buffer size and capacity of each link, two policies are critical: the message forwarding policy to ensure link fairness across different contending sources and the message discarding policy to ensure timeliness for the delivered messages. A node is responsible for a message until it has expired or it has been sent, received, or dropped (to make space for higher priority messages) on each of the node's neighboring links specified in the bitmask on that message.

Source-based fairness. In traditional priority schemes, each node forwards the highest priority message it has across all sources. However, a compromised source can send all of its messages with highest priority, starving correct sources.

It is very challenging, if not impossible, to determine which (if any) of the sources are compromised, so we consider message priority independently for each source, never comparing priorities across sources. Nodes enforce fairness of both storage and network resources, independently for each link. Each link forwards the highest priority message it has from each source in a fair manner (e.g. round-robin).

Note that, since each correct node enforces fairness on all outgoing links, even if a correct node receives messages in an unfair way from a compromised neighbor, fairness is still maintained as it forwards those messages.

In contrast to static resource allocation schemes that reserve a predetermined amount of resources for each potential source in the network, Priority Messaging dynamically allocates link-level resources based on the number of active sources, i.e. the number of sources currently sending on a link. The allocated resources include both network bandwidth and message storage. By not preallocating resources, Priority Messaging enables full utilization of each link's capacity at all times. On each link between two correct nodes, each active source receives either the resources it requests or its fair share, whichever is smaller. Note that if a source is using less than its fair share, the unused resources are evenly reapportioned among the other active sources that are requesting more than their fair share. Moreover, allocating on a message-by-message granularity is crucial because it prevents malicious sources from

temporarily starving correct sources with a burst of traffic after a period of inactivity, which can happen in credit-based allocation schemes (e.g. [51, 52]).

Timeliness. Ideally, the latency of each message is only affected by the propagation delay of the underlying network. However, if the rate of messages entering a node is temporarily greater than the rate of messages leaving, messages will accumulate in queues, resulting in higher latency. In practice, many factors can cause this type of behavior, including a burst of traffic from a correct source, fluctuating link bandwidth, and a malicious source injecting spurious messages. In such scenarios, it is impossible to ensure that messages are delivered with low latency. Furthermore, incoming messages at a node with limited bandwidth and full message queues necessitate that some messages be dropped.

To provide real-time latency guarantees for the most important messages at all times, even in the presence of fully-contended bandwidth, Priority Messaging uses the following dropping rule: if message storage is full, an incoming message causes the lowest priority, oldest message from the source currently using the most storage to be dropped, making room for the newer message to be stored. Higher priority messages (of a given source) are sent first in order to ensure that they experience the lowest possible latency. Lower priority messages are either delivered with higher latency or dropped, i.e. never delivered, because newer, higher priority messages arrived before

these lower ones could be forwarded.

Message expiration. Since Priority Messaging does not provide ordered delivery, nodes cannot recognize duplicate messages simply by comparing a message's sequence number against a single monotonically increasing value, e.g. the highest sequence number received. Thus, a node must store the sequence numbers of all the messages it has received to recognize duplicates. The expiration time on each message prevents this storage from growing without bound.[1] Once a message expires, its sequence number is no longer needed; a compromised node cannot forge a new signature on an old message from a correct source with a different expiration time, and expired messages are promptly discarded. Note that using this scheme requires monotonically increasing clocks and some level of network synchronicity. Previous work has met these conditions with atomic clocks [53].

A benefit of message expiration is a more efficient use of network resources: intermediate forwarding nodes can disregard messages that have expired and would be discarded by the destination. The bandwidth reclaimed by this process can be used for other relevant messages (e.g. lower priority) that may otherwise have been dropped.

Note that, for priority levels where not all messages are received, the particular

[1]Enforcing an upper bound on expiration time puts a limit on sequence number storage; any source violating this bound is detected as compromised.

messages that are received may be adversarially chosen (but are legitimate messages). This is because compromised nodes may choose to forward certain messages around bottlenecks in the path of correct nodes, ensuring those messages reach the destination when they might otherwise have been dropped due to contention.[2]

We now formally define and prove the guarantees provided by Priority Messaging — safety, liveness, and guaranteed throughput — under the threat model presented in Chapter 2. Safety for Priority Messaging refers to achieving prioritized timely delivery for each correct source. Note that these guarantees hold separately for both K-Paths and Constrained Flooding.

5.2.1 Priority Flooding

Here, we both state and prove the guarantees of the Priority Messaging semantics over the Flooding dissemination method.

5.2.1.1 Safety

Safety in this context is equivalent to prioritized timely delivery.

Theorem 2 (Priority-Safe)**.** *If messages m_1 and m_2 are simultaneously accepted from a client into the network at a correct source S destined to a correct destination D,*

[2]Thanks to Dan Peek for this insight.

with m_1 having higher priority than m_2, then if both m_1 and m_2 are delivered at D along correct paths and none of S's messages are dropped due to resource contention while m_1 and m_2 are undelivered, then m_1 is delivered before m_2 at D.

Proof. In the context of this proof, "simultaneously accepted" implies that S does not communicate with any intermediate nodes in the time between accepting messages m_1 and m_2. Because of the Priority Messaging semantics, S will deliver m_1 before m_2 to each of its neighbors. Consider one of these neighbors X_1, along a correct path to D. X_1 will receive m_1 before m_2, and thus will also deliver m_1 before m_2 to the next node along the correct path, X_2. By induction, each node along the correct path will deliver m_1 before m_2, all the way to D.

Note that this proof relies on in-order delivery on each edge, a property provided by the PoR Link (see Section 5.1.1). Also, S has the ability to ensure that none of its messages are dropped by limiting its sending appropriately.

□

Theorem 3 (Timely-Safe)**.** *Consider a network of n nodes. If the network has no highest-priority message from a correct source S, then if S introduces a single highest-priority message m to a correct destination D, D will receive m within some time t. t is no greater than the minimum message transmission time along a correct path between S and D, including the time it takes for at most n-1 messages to be sent*

at each correct node along that path. Note that t is infinity if there is no correct path from S to D.

Proof. Consider a correct node X. When X receives m, by construction, m is S's only highest-priority message stored at X. Since X forwards messages from active sources in round-robin order, X will forward one of S's messages after no more than n-1 other messages are forwarded. When X forwards one of S's messages, it will forward m, because m is S's highest-priority message at X. The transmission time between when m arrives at X and m reaches a correct neighbor is the time to forward n-1 messages plus the edge propagation delay and any time needed for retransmissions.

Along a correct path between S and D, the total message transmission time for m to reach D is no greater than the sum of the message transmission times for m at nodes along the path. Since flooding explores all possible paths, m will arrive first at D along the path with the minimum total message transmission time. Note that in the case of no loss, no time is needed for retransmissions.

□

5.2.1.2 Liveness

As long as there exists a correct path from source to destination, messages for that source will be delivered.

Theorem 4 (Liveness)**.** *If there are undelivered messages from a correct source S to a correct destination D and there exists a correct path from S to D, then eventually at least one message from S will be delivered at D.*

Proof. Consider S and the next node along a correct path from S to D, X_1. As part of the flooding scheme, S will forward a message to each of its neighbors, including X_1. Similarly, once X_1 has received a message from S, at some point, due to the round-robin scheme, X_1 will forward it or another one of S's messages to each of its neighbors, including the next node along the correct path, X_2. By induction, each node on the path at some point will forward messages from S along the path until the messages reach the destination D. Thus, eventually messages will be delivered to D.

□

5.2.1.3 Guaranteed Throughput

Theorem 5 (Guaranteed Throughput)**.** *Consider a network of n nodes. If there exists a correct path from a correct source S to a correct destination D, and S sends only to D, and S is one of g correct sources actively sending, and there are f compromised sources actively sending, then the rate at which S can send to D is no less than $\frac{1}{f+g}$ times the minimum bandwidth over all edges in that correct path. In the worst case, this rate is $\frac{1}{n}$.*

Proof. Each node uses a round-robin scheme to determine which message to forward next on each edge. This round-robin scheme rotates between sources for which the node has unsent messages. If there are f compromised sources actively sending and g correct sources actively sending, then the round-robin scheme at an intermediate node considers at most $f + g$ sources and alternates between them. Each correct source will be served on a edge after at most $f + g - 1$ other sources are served, resulting in a correct source receiving $\frac{1}{f+g}$ of the bandwidth on that edge. Since that correct source receives $\frac{1}{f+g}$ of the bandwidth on each correct edge in the network, it receives $\frac{1}{f+g}$ of the total bandwidth in the network. This is true for every correct source. This value in the worst case becomes $\frac{1}{n}$ because $f + g \leq n$.

Note that when S sends to more than one destination, the guaranteed rate of $\frac{1}{f+g}$ is split between the d destinations at S's discretion.

□

5.2.2 Priority K-Paths

Here, we both state and prove the guarantees of the Priority Messaging semantics over the K-Paths dissemination method.

5.2.2.1 Safety

Safety in this context is equivalent to prioritized timely delivery. Priority-Safe in this case is the same as Priority-Safe for Priority Flooding (see Theorem 2).

Theorem 6 (Timely-Safe)**.** *Consider a network of n nodes. If the network has no highest-priority message from a correct source S, then if S introduces a single highest-priority message m to a correct destination D, D will receive m within some time t. t is no greater than the minimum message transmission time along one of the correct computed K paths between S and D, including the time it takes for at most n-1 messages to be sent at each correct node along that path. Note that t is infinity if none of the K paths from S to D are correct.*

Proof. Consider a correct node X. When X receives m, by construction, m is S's only highest-priority message stored at X. Since X forwards messages from active sources in round-robin order, X will forward one of S's messages after no more than n-1 other messages are forwarded. When X forwards one of S's messages, it will forward m, because m is S's highest-priority message at X. The transmission time between when m arrives at X and m reaches the next correct node on one of the computed K paths is the time to forward n-1 messages plus the edge propagation delay and any time needed for retransmissions.

Along a correct path between S and D, the total message transmission time

for m to reach D is no greater than the sum of the message transmission times for m at nodes along the path. Since messages are sent along each of the computed K paths, m will arrive first at D along the correct path with the minimum total message transmission time. Note that in the case of no loss, no time is needed for retransmissions.

□

5.2.2.2 Liveness

As long as at least one of the computed K paths is correct between a source and destination, messages for that source will be delivered.

Theorem 7 (Liveness)**.** *If there are undelivered messages from a correct source S to a correct destination D and at least one of the computed K paths is correct, then eventually at least one message from S will be delivered at D.*

Proof. Consider S and the next node, X_1, along one of the computed K paths from S to D that is correct. S will forward a message to X_1. Similarly, once X_1 has received a message from S, at some point, due to the round-robin scheme, X_1 will forward it or another one of S's messages to the next node, X_2, along the path specified in that message's bitmask that includes X_1. By induction, each node on the path at some point will forward messages from S along the path until the messages reach the

destination D. Thus, eventually messages will be delivered to D.

□

5.2.2.3 Guaranteed Throughput

Theorem 8 (Guaranteed Throughput)**.** *Consider a network of n nodes. If one of the computed K paths between a correct source S and a correct destination D is correct, and S sends only to D, and S is one of g correct sources actively sending, and there are f compromised sources actively sending, then the rate at which S can send to D is no less than $\frac{1}{f+g}$ times the minimum bandwidth over all edges in that correct path. In the worst case, this rate is $\frac{1}{n}$.*

Proof. This proof is equivalent to the proof of Theorem 5.

□

Theorem 9 (Timely-Safe)**.** *Consider a network of n nodes. If the network has no highest-priority message from a correct source S, then if S introduces a single highest-priority message m to a correct destination D with K node-disjoint paths specified in its bitmask, D will receive m within some time t. t is no greater than the minimum message transmission time along any of the K specified paths between S and D which are correct, including the time it takes for at most n-1 messages to be sent at each correct node along that path. Note that t is infinity if none of the K*

specified paths are correct.

Proof. Consider S and the next node, X_1, along one of the specified K paths from S to D that is correct. S will forward m to X_1 after no more than $n-1$ other messages, because different sources are served in round-robin order and S has no message originating at S with a higher priority than m. The transmission time between when m is introduced at S and m reaches X_1 is the time to forward no more than $n-1$ messages plus the edge propagation delay and any time needed for retransmissions.

Similarly, the same applies between X_1 and X_2, the next node along that same specified path. The transmission time between when m arrives at X_1 and m reaches X_2 is the time to forward no more than $n-1$ messages plus the edge propagation delay and any time needed for retransmissions.

Along a correct path between S and D, the total message transmission time for m to reach D is no greater than the sum of the message transmission times for m at nodes along the path. Since K-Paths uses all K paths simultaneously, t is no greater than the minimum message transmission time along any of the K specified paths between S and D which are correct. Note that in the case of no loss, no time is needed for retransmissions.

□

5.3 Reliable Messaging with Source-Destination Fairness

Reliable Messaging with Source-Destination Fairness (for short, Reliable Messaging) is motivated by the reliability demands of control messages, which contain critical information that changes the state of the system and may lead to inconsistencies if not delivered reliably to all destinations. Within these reliability constraints, Reliable Messaging is as timely as possible.

Overview. We provide reliable delivery by using acknowledgments and back-pressure. Each message contains the identity of its source, the identity of the destination, a monotonically increasing sequence number, and a digital signature of the message from its source. Only messages with a valid signature are accepted. Messages are delivered maintaining FIFO order with respect to the source that generated them. Each node stores uniquely received messages. Nodes only accept the message with the next expected sequence number for each flow, recognizing duplicates and trivially preventing replay attacks. Source-destination (flow) fairness is required to ensure that no correct flow is starved of resources. In order to provide end-to-end reliability, intermediate nodes maintain responsibility for messages until they are acknowledged by the destination. When a node's buffer for a particular flow fills, the

node creates back-pressure (all the way back to the source) to prevent new messages from entering the network for that flow. Reliable Messaging provides reliability even when intermediate nodes crash and recover.

Flow fairness. If messages are stored and forwarded based on their source, messages from a source to all destinations will share memory and bandwidth at intermediate nodes. If a correct source is sending to multiple destinations and one of those destinations is compromised, that compromised destination can refuse to acknowledge messages, resulting in messages from that source being blocked (due to the reliable semantics), even to correct destination. Thus, in Reliable Messaging, we consider messages as part of a source-destination flow, rather than associated solely with the source.

Reliable Messaging uses a dynamic allocation scheme for network bandwidth among the active source-destination flows on each link, and fairness is ensured by serving each active flow in a round-robin fashion. However, Reliable Messaging cannot use a dynamic allocation scheme for message storage at each node. In Priority Messaging, a source is always guaranteed its fair share of storage because the algorithm will drop lower priority, older messages from other sources that are over their fair share to make room. In contrast, in Reliable Messaging, stored messages cannot be dropped, even to enforce fairness, because messages are guaranteed to be reliably

delivered. In order to ensure fairness, we must use static storage allocation, reserving storage for every possible flow in the network.

Note that, as with Priority Messaging, if a correct node receives messages in an unfair way from a compromised neighbor, fairness is still maintained as it forwards those messages.

End-to-end acknowledgements. End-to-end acknowledgements are periodically generated (with a configurable end-to-end timeout), signed, and flooded back through the network by each destination, indicating in a single message the highest in-order sequence number received from every source node in the network. End-to-end acknowledgements allow intermediate nodes to discard acknowledged messages, making room for new ones. End-to-end acknowledgements operate on an overtaken-by-event basis, with intermediate nodes only storing the latest one from each destination. In order to prevent compromised nodes from spamming end-to-end acknowledgements to consume network bandwidth, a correct node only forwards end-to-end acknowledgements from a destination that indicate progress, and forwards them no more often than the end-to-end timeout. Only the direct links with compromised nodes can receive end-to-end acknowledgements more frequently than the specified timeout. Note that a compromised node can introduce messages destined to all nodes, causing each of them to generate one end-to-end acknowledgement per timeout. However,

additional compromised nodes will not cause more end-to-end acknowledgements to be sent between correct nodes, because a correct destination will only send one end-to-end acknowledgement per timeout. Thus, the overhead when all nodes are active destinations must be considered when choosing this timeout (see Table 6.4).

Neighbor acknowledgements. Neighbor acknowledgements, where a node notifies its neighbors of messages it has received, can be used. This optimization improves bandwidth usage by allowing nodes to avoid sending messages to neighbors that already have them. See Figure 6.3 for an experimental evaluation of this optimization.

Back-pressure. If end-to-end acknowledgements do not arrive in a timely manner for a flow, message storage for that flow at intermediate nodes will fill, forcing those nodes to create *back-pressure*. Correct intermediate nodes will not accept new messages for that flow until back-pressure is relieved, even to the point that the source node will stop accepting messages from the client.

Crash-recovery resiliency. Upon recovery, a node must communicate with its neighbors to rejoin the system. The recovering node notifies each of its neighbors and correct neighbors will respond by sending it the latest end-to-end acknowledgement from each destination, signed by each destination. Once it has received an end-to-end acknowledgement, the recovering node can receive and forward messages, starting from

the first message after the messages covered by that end-to-end acknowledgement, for flows to the destination that generated that end-to-end acknowledgement. Because the end-to-end acknowledgement is signed, a compromised neighbor cannot cause the rejuvenating node to progress further than it should. Note that if a compromised neighbor provides a valid but out-of-date end-to-end acknowledgement, the recovering node will receive and forward old messages, but upon receiving an up-to-date end-to-end acknowledgement from a correct neighbor the recovering node will update its state and forward new messages. This ensures that messages will flow even when the only correct path from source to destination is an eventual path. Similar functionality could be provided via stable storage.

We now formally define and prove the guarantees provided by Reliable Messaging — safety, liveness, and guaranteed throughput — under the threat model presented in Chapter 2. For Reliable Messaging, safety corresponds to reliable in-order delivery for each correct flow. Note that the guarantees hold separately for both K-Paths and Constrained Flooding.

5.3.1 Reliable Flooding

Here, we both state and prove the guarantees of the Reliable Messaging semantics over the Flooding dissemination method.

5.3.1.1 Safety

Safety in this context is equivalent to reliable, in-order delivery separately for the messages on each flow.

Theorem 10 (Safety)**.** *If a correct source node S accepts i messages destined to some correct destination node D, then the first $i - b$ messages have all been reliably delivered in order at D, where b is the size of the buffer for one flow at a node.*

Proof. When the source S accepts a message from a client, it places that message in its buffer for that flow, and S will not accept more messages than it has room for in the buffer (b). S will forward the message across the network, and intermediate nodes will place the message in their buffers as well. At an intermediate node X, there are two ways that the message will leave its buffers.

The first is if X receives an end-to-end acknowledgement that covers that message. Since the end-to-end acknowledgement is protected with a signature, the only way X can receive such a message is if the correct destination D generated one. D will only generate such an acknowledgement if it has received all messages up to and including that one, in order, since it is a correct node. Note that the end-to-end acknowledgement is flooded, so it will arrive along any correct path.

The second is if X loses its state, which may happen through a benign fault and subsequent recovery, or a benign fault combined with a disk failure (if the nodes

write their state to disk) and subsequent recovery, or by being compromised and rejuvenated. In this case, as specified in Section 5.3 above in the discussion of crash-recovery resiliency, correct neighbors will provide the recovering node with the latest end-to-end acknowledgements and will then provide it with the relevant messages after that point, starting with the first message after the messages covered by each end-to-end acknowledgement. Because the end-to-end acknowledgement is signed, compromised neighbors cannot cause a recovering node to progress further forward than it should. A compromised neighbor can provide an out-of-date end-to-end acknowledgement to cause X to forward old messages, but since X is on a correct path it must have at least one correct neighbor and X will receive an up-to-date end-to-end acknowledgement from that neighbor, causing X to update its state and begin forwarding current messages.

D (a correct node) ensures reliable in-order message delivery using its own buffer. Thus, the first $i - b$ messages must have all been reliably delivered in order at D.

□

5.3.1.2 Liveness

As long as there exists a correct path from source to destination, messages for that flow will be delivered.

Theorem 11 (Liveness)**.** *If there are undelivered messages from a correct source S to a correct destination D and there exists a correct path from S to D, then eventually at least one message from S will be delivered at D.*

Proof. Consider S and the next node along a correct path from S to D, X_1. As part of the flooding scheme, S will forward each message it has to each of its neighbors, including X_1. Similarly, once X_1 has received a message, at some point, due to the round-robin scheme, X_1 will forward the message to each of its neighbors, including the next node along the correct path, X_2. By induction, each node on the path at some point will forward messages along the path until the messages reach the destination D. Thus, eventually messages will be delivered to D.

Note that if X_i receives an end-to-end acknowledgement covering a message before it forwards that message, X_i will not forward the message. But in this case, D has already received that message, as indicated by this end-to-end acknowledgement, implying that liveness is still met.

Note that if X_i fails and recovers, as specified in Section 5.3 above in the discussion of crash-recovery resiliency, correct neighbors will provide the recovering node with the latest end-to-end acknowledgment and continue sending from that point forward. X_i is on a correct path, so it must have a correct neighbor who can provide the up-to-date end-to-end acknowledgements. Thus, as long as there is a correct path

from source to destination, even if it is an eventual path, messages will be delivered.

□

5.3.1.3 Guaranteed Throughput

Theorem 12 (Guaranteed Throughput)**.** *Consider a network of n nodes. If there exists a correct path from a correct source S to a correct destination D, and S is one of g correct sources actively sending, and there are f compromised sources actively sending, then the rate at which S can send to D is no less than $\frac{1}{(f+g)*(n-1)}$ times the minimum bandwidth over all the edges in that correct path. In the worst case, this rate is $\frac{1}{n*(n-1)}$.*

Proof. Each node uses a round-robin scheme to determine which message to forward next on each edge. This round-robin scheme rotates between flows for which the node has unsent messages. If there are f compromised sources actively sending and g correct sources actively sending, then the round-robin scheme at an intermediate node considers at most $(f+g)*(n-1)$ flows and alternates between them. Each correct flow will be served on a edge after at most $(f+g)*(n-1)-1$ other flows are served, resulting in a correct flow receiving $\frac{1}{(f+g)*(n-1)}$ of the bandwidth on that edge. Since that correct flow receives $\frac{1}{(f+g)*(n-1)}$ of the bandwidth on each correct edge in the network, it receives $\frac{1}{(f+g)*(n-1)}$ of the total bandwidth in the network. This is true

for every correct flow. In the worst case this value becomes $\frac{1}{n*(n-1)}$ because $f + g \leq n$.

Note that this assumes that end-to-end acknowledgements are sent sufficiently frequently (sufficiently small end-to-end timeout) or the buffer b is sufficiently large that flows are not blocked due to back-pressure, waiting for their destination to acknowledge via end-to-end acknowledgement.

□

5.3.2 Reliable K-Paths

Here, we both state and prove the guarantees of the Reliable Messaging semantics over the K-Paths dissemination method.

Note that, in the case of a reroute, unacknowledged messages may need to be resent. In our implementation, we use the same sequence number, but use a bitmask created from the superset of the two sets of K Node-Disjoint Paths. Intermediate nodes honor the new message and forward it accordingly. This is safe because the message bitmask can only increase a bounded number of times before it becomes a flooded message, at which point intermediate nodes will no longer honor new versions of it.

5.3.2.1 Safety

Safety in this case is the same as safety for Reliable Flooding (see Theorem 10).

5.3.2.2 Liveness

As long as at least one of the specified K paths is correct between a source and destination, messages for that flow will be delivered.

Theorem 13 (Liveness)**.** *If there are undelivered messages from a correct source S to a correct destination D and at least one of the specified K paths is correct, then eventually at least one message from S will be delivered at D.*

Proof. Consider S and the next node, X_1, along one of the K paths specified on the messages from S to D and that path is correct. S will forward each message it has to X_1. Similarly, once X_1 has received a message, at some point, due to the round-robin scheme, X_1 will forward the message to the next node along the path, X_2. By induction, each node on the path at some point will forward messages along the path until the messages reach the destination D. Thus, eventually messages will be delivered to D.

Note that if X_i receives an end-to-end acknowledgement covering a message before it forwards that message, X_i will not forward the message. But in this case, D has

already received that message, as indicated by this end-to-end acknowledgement, implying that liveness is still met.

Note that if X_i fails and recovers, as specified in Section 5.3 above in the discussion of crash-recovery resiliency, correct neighbors will provide the recovering node with the latest end-to-end acknowledgments and the previous node on the path (of the K paths) will continue sending from that point forward, starting with the first message after the messages covered by each end-to-end acknowledgement. X_i is on a correct path, so it must have a correct neighbor who can provide the up-to-date end-to-end acknowledgements. Thus, as long as one of the specified K paths is correct, even if it is an eventual path, messages will be delivered.

□

5.3.2.3 Guaranteed Throughput

Theorem 14 (Guaranteed Throughput)**.** *Consider a network of n nodes. If one of the computed K paths between a correct source S and a correct destination D is correct, and S is one of g correct sources actively sending, and there are f compromised sources actively sending, then the rate at which S can send to D is no less than $\frac{1}{(f+g)*(n-1)}$ times the minimum bandwidth over all the edges in that correct path. In the worst case, this rate is $\frac{1}{n*(n-1)}$.*

Proof. This proof is equivalent to the proof of Theorem 12.

□

Chapter 6

Deployment and Evaluation

We deploy the intrusion-tolerant network implementation on a global cloud [21] and evaluate the deployment in two ways. First, we send realistic traffic similar to actual traffic patterns observed in the cloud to evaluate the network performance in the presence of compromised nodes. Second, we use the deployment to carry the monitoring messages of the global cloud, and discuss the experience.

6.1 Deployment Environment

We deploy the implementation as an overlay network on a global cloud (Figure 6.1) that spans 12 data centers from East Asia to North America to Europe. We do not report the specific latency on each edge for proprietary considerations. This

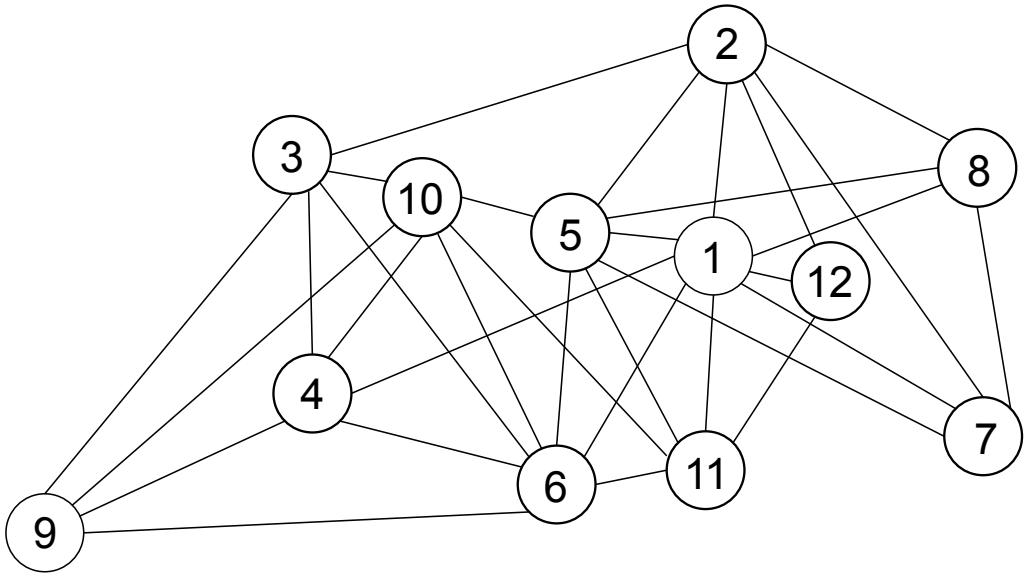

Figure 6.1: The global cloud topology, spanning from East Asia to North America to Europe.

topology contains sufficient redundancy to support at least three node-disjoint paths between any two nodes. The deployed network uses the following parameters: 150 ms end-to-end acknowledgement timeout, 100 ms neighbor acknowledgement timeout, 10 Mbps link bandwidth capacity, 1024-bit RSA signatures, and HMACs that use SHA-256.

We evaluate the deployment's performance and overhead (Section 6.2) and resilience to attacks (Section 6.3) by sending realistic traffic similar to monitoring traffic patterns observed in the cloud, with most messages below 3500 bytes. In addition, we evaluate the deployment using production traffic by protecting all the monitoring messages in the cloud (Section 6.4).

	Priority (Mbps)			Reliable (Mbps)		
	Flood	**K=1**	**K=2**	**Flood**	**K=1**	**K=2**
(a)	125	480	425	125	395	395
(b)	45	85	80	40	85	80

Table 6.1: Maximum goodput measured with: (a) no cryptography, (b) HMACs and signatures.

6.2 Performance and Overhead

We evaluate the performance and overhead of the deployed network in benign environments.

Cryptographic impact on performance. Table 6.1 shows the maximum performance for one active flow in the network, obtained in a controlled laboratory environment matching the topology of Figure 6.1, with artificial latencies matching that same topology. In (a), cryptographic mechanisms are disabled. In (b), they are enabled.

Since our performance is strictly CPU bound when using cryptography, adding additional hardware by sharding the incoming traffic across multiple cores or even multiple machines would enable us to reach performance comparable with (a) in Table 6.1. We note that 3% of overall traffic (as mentioned in Chapter 1) on a 10Gbps link is 300 Mbps, which (a) in Table 6.1 shows we can support, with additional hardware to support the cryptography.

Sensitivity to message size. Figure 6.2 shows the effect of message size on

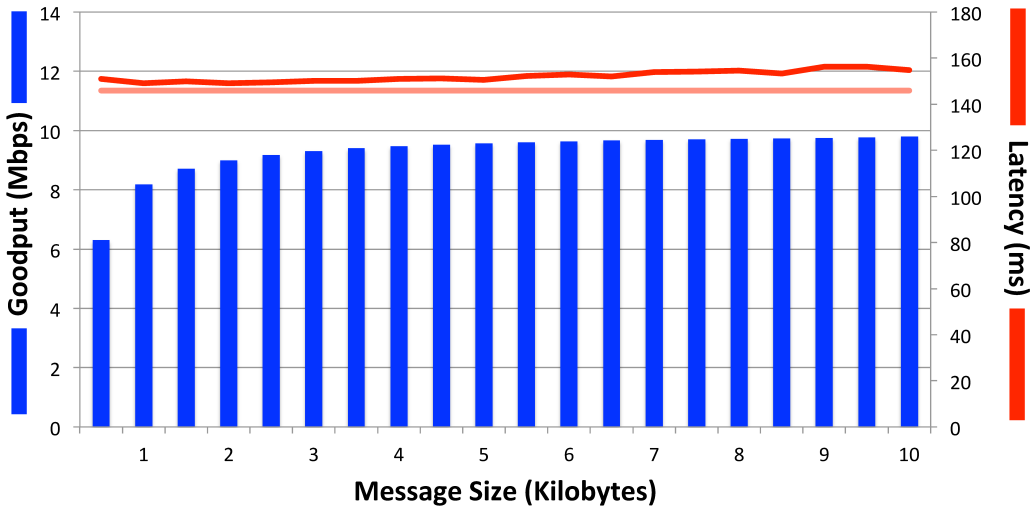

Figure 6.2: Effect of message size on goodput and latency. The bar chart (left axis) measures goodput and the line graph (right axis) measures latency. The flat line represents the propagation delay from the source to the destination.

goodput and latency for one active flow sending at 10 Mbps. As message size increases, a smaller fraction of each message is consumed by the cryptographic mechanisms and protocol headers, resulting in higher goodput. Larger messages also increase the latency, since each intermediate node must receive the entire message to verify the signature before forwarding it.

Communication cost. The messaging cost on the network depends on the topology and can vary widely. Tables 6.2 and 6.3 show the analytical and experimental costs of source-based routing schemes and data dissemination semantics. When we mention shortest path, we refer to the lowest latency path. The cost associated with sending messages corresponds to the number of edges those messages traverse.

Dissemination Method	**Avg. # Hops**	**Scaled Cost**	**Avg. Path Lat. (ms)**
K=1	1.9	1.0	41.4
K=2	4.4	2.3	43.5
K=3	6.6	3.5	46.6
Naïve Flooding	64.0	34.1	—
Engineered Flooding	32.0	17.0	—

Table 6.2: The analytical cost of source-based routing schemes on the topology. The scaled cost is the average number of hops normalized by the cost of K=1. The average path latency is also shown.

Protocol	**Avg. # Hops**	**Scaled Cost**
Priority Flooding	35.8	19.0
Reliable Flooding (w/o End-to-End acknowledgements)	31.3	16.7
Reliable Flooding	16.3	8.7

Table 6.3: The measured cost on the topology. The scaled cost is the average number of hops normalized by the cost of K=1.

Analytical comparison: In Table 6.2, we consider all source-destination pairs and report the average cost. For K-Paths with K=1, equivalent to single-path routing, the average number of hops between two nodes is 1.9 and the average path latency, the latency of the shortest path, is 41.4 ms. For K=2 (resp. 3), the total number of hops across the paths increases by more than double (resp. triple). Since multiple shortest (latency-wise) paths do not exist, the average latency across the paths increases. Recall that since Priority Flooding and Reliable Flooding leverage semantic-specific information for their optimizations, we cannot calculate the analytical cost. Instead, we show the analytical cost of Naïve Flooding, where messages traverse each edge in

both directions, and Engineered Flooding, where techniques (such as random delay) are used so that messages traverse each edge only once. Since flooding schemes are not path-based, average path latency is not reported.

Experimental comparison: We measure and compare the costs of Priority Flooding, Reliable Flooding without end-to-end acknowledgements, and Reliable Flooding (Table 6.3). We also compare these costs to the analytical costs of the flooding schemes in Table 6.2. The experiments use five randomly selected flows (node 9 to node 11, node 4 to node 5, node 7 to node 9, node 1 to node 10, and node 3 to node 8) each sending at 10 Mbps to create network contention. We omit experimental costs for K-Paths because they are very similar to their analytical costs.

Note that Reliable Flooding without end-to-end acknowledgements is not a correct protocol. End-to-end acknowledgements are required to inform intermediate nodes when it is safe to discard a packet. In this case, solely to evaluate the decrease in cost due to neighbor acknowledgements, the intermediate nodes will instead discard a packet once all their neighbors have acknowledged it. This is not a correct intrusion-tolerant protocol and would be blocked by even a single compromised or failed node in the system.

The cost of Priority Flooding is between the costs of Naïve Flooding and Engineered Flooding because messages traverse some (but not all) edges in only one direction.

Since timeliness is vital for Priority Messaging, a random delay is infeasible and only the natural latency of the network can prevent messages from flowing twice on a given edge.

In Reliable Flooding, neighbor acknowledgements eliminate the need to forward messages to neighbors that have already acknowledged them, resulting in messages traversing each edge twice, once, or not at all. End-to-end acknowledgements eliminate the need to forward messages that have already been acknowledged by the destination. As a result, messages traverse each edge twice, once, or not at all, and may even not arrive at some intermediate nodes because their neighboring nodes may receive the corresponding end-to-end acknowledgement before forwarding the message. The cost of Reliable Flooding without end-to-end acknowledgements is comparable to the cost of Engineered Flooding (due to the neighbor acknowledgements), showing the benefits of neighbor coordination. The cost of Reliable Flooding is significantly lower; the end-to-end acknowledgements provide global knowledge to nodes, giving the power of flooding for a much cheaper cost.

Aggregate goodput. In Figure 6.3, we give the experimental goodput for Naïve Flooding, Priority Flooding, Reliable Flooding without end-to-end acknowledgements, and Reliable Flooding. Each flow is represented by a single colored line, with the aggregate throughput of the network represented by a thin blue line. The flows are

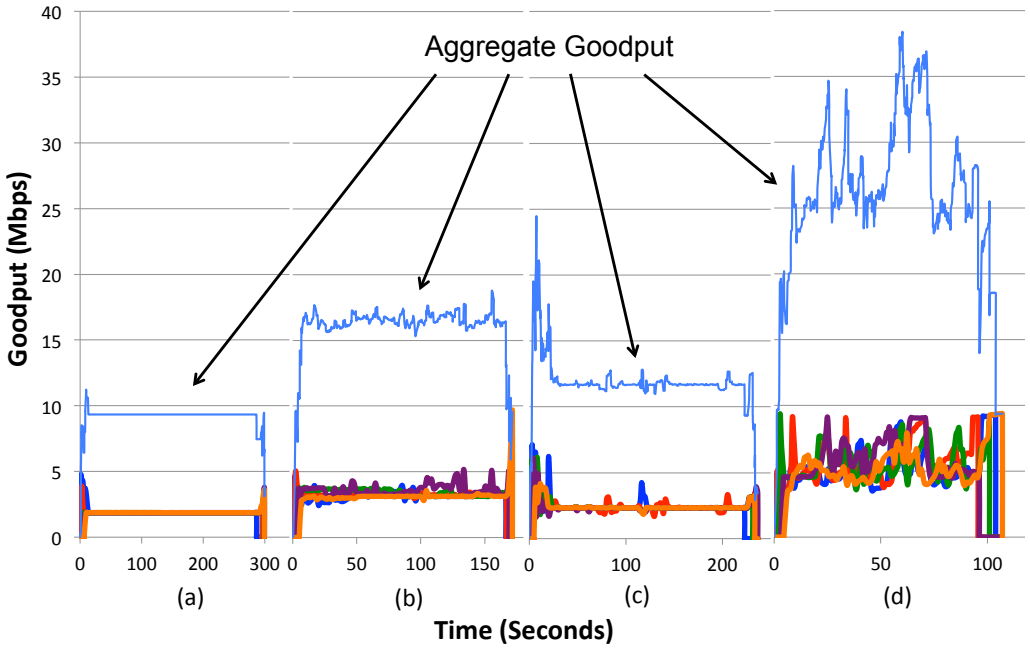

Figure 6.3: Experimental goodput for: (a) Naïve Flooding, (b) Priority Flooding, (c) Reliable Flooding (no end-to-end acknowledgements), and (d) Reliable Flooding. Despite additional instability, aggregate and individual flow goodput increases.

the same as those specified above for Table 6.3.

In Naïve Flooding (Figure 6.3a), every message travels on every edge twice (once in each direction). We see the overhead of our protocols in the difference between the aggregate throughput and the 10 Mbps bandwidth cap on each edge. We can also see that the system is exactly fair, with each of the five flows getting one fifth of the bandwidth. By comparing these results with the other three results (Figure 6.3b, Figure 6.3c, and Figure 6.3d), we see that all of them meet or exceed the fair share at all times.

In Priority Flooding (Figure 6.3b), messages travel across the network using the Priority Flooding protocol specified earlier (see Section 5.2.1). In this case, a node will not send a message back to a neighbor from which it has already received that message. In addition, if a message is dropped due to contention, it may not travel on that edge at all. By comparing Figure 6.3b with Naïve Flooding (Figure 6.3a), we see that these optimizations increase the aggregate throughput.

In Reliable Flooding without end-to-end acknowledgements (Figure 6.3c), messages travel across the network using the Reliable Flooding protocol specified earlier (see Section 5.3.1), but without the end-to-end acknowledgements. Note that this is not a correct protocol: the end-to-end acknowledgements are required for the protocol to make progress. Here, solely to evaluate the benefit of neighbor acknowledgements

without end-to-end acknowledgements, a node will instead discard a message once all of its neighbors have acknowledged it. A node will not forward a message to a neighbor who has given it that message, nor will it forward a message to a neighbor who has already acknowledged that message. By comparing this with Naïve Flooding (Figure 6.3a), we see the improvement in aggregate goodput due to neighbor acknowledgements.

Comparing Figure 6.3b and Figure 6.3c, we see that Priority Flooding has higher aggregate goodput than Reliable Flooding without end-to-end acknowledgements, even though they are both using local knowledge to avoid sending messages to neighbors who already have those messages. This is because Priority Flooding may drop different messages on different paths, possibly resulting in higher aggregate goodput. In the experiment shown, each flow was sending at 10 Mbps, while the bandwidth cap on each link was 10 Mbps, resulting in a lot of contention. That contention led Priority Flooding to drop many messages, although different messages were dropped on different paths, providing opportunities for more messages to arrive at the destination in aggregate, leading to higher overall goodput. In other words, a message that arrives at the destination does not necessarily arrive at all nodes in the network. In contrast, Reliable Flooding enforces reliable semantics and therefore will not drop messages. While Reliable Flooding without end-to-end acknowledgements

will reduce bandwidth usage by not sending a message to a neighbor it definitely knows already has that message, for a given flow it will always choose to send to that neighbor the next unacknowledged message, in order, for which it has not received a definite acknowledgement, leaving fewer opportunities for different sets of messages to arrive via different paths. In other words, a message that arrives at the destination will definitely arrive at all nodes in the network. As a result, Priority Flooding achieves higher aggregate goodput than Reliable Flooding without end-to-end acknowledgements.

In Reliable Flooding (Figure 6.3d), messages travel across the network using the Reliable Flooding protocol specified earlier (see Section 5.3.1). With the full Reliable Flooding protocol, including end-to-end acknowledgments, there is a potential for an end-to-end acknowledgement to overtake a message, so that message may not even be disseminated to large parts of the network topology, dramatically reducing the number of edges that message will cross. This is in addition to the benefits of neighbor acknowledgements previously discussed. As a result, messages travel on each edge twice, once, or not at all, and may even not arrive at some nodes in the network. This results in significantly higher goodput, as can be seen by comparing with Reliable Flooding without end-to-end acknowledgements (Figure 6.3c). However, since the effect of the end-to-end acknowledgements depends on the timing of messages in the

	# of Active Destination Nodes				
Timeout	**1**	**5**	**10**	**20**	**50**
20 ms	0.390	1.952	3.904	7.808	19.52
100 ms	0.078	0.390	0.781	1.562	3.904
500 ms	0.016	0.078	0.156	0.312	0.781

Table 6.4: E2E acknowledgement overhead (Mbps) on all edges.

network, the goodput is more chaotic.

Note that Reliable Flooding (Figure 6.3d) is not strictly better than Priority Flooding (Figure 6.3b), as it does not provide the same level of timeliness.

Sensitivity to acknowledgement rate. Although the end-to-end acknowledgements provide a significant optimization, they introduce network overhead. Destinations flood end-to-end acknowledgements once per end-to-end acknowledgement timeout, and the choice of this timeout presents a trade-off between overhead and responsiveness. Longer timeouts preserve more bandwidth for data messages, but the network takes longer to clear back-pressure. Table 6.4 shows the overhead with different timeouts and numbers of active destination nodes.

6.3 Resilience to Attacks

We evaluate the performance of the deployment under instrumented attacks. The reported experiments use Constrained Flooding because, in terms of contention, it is

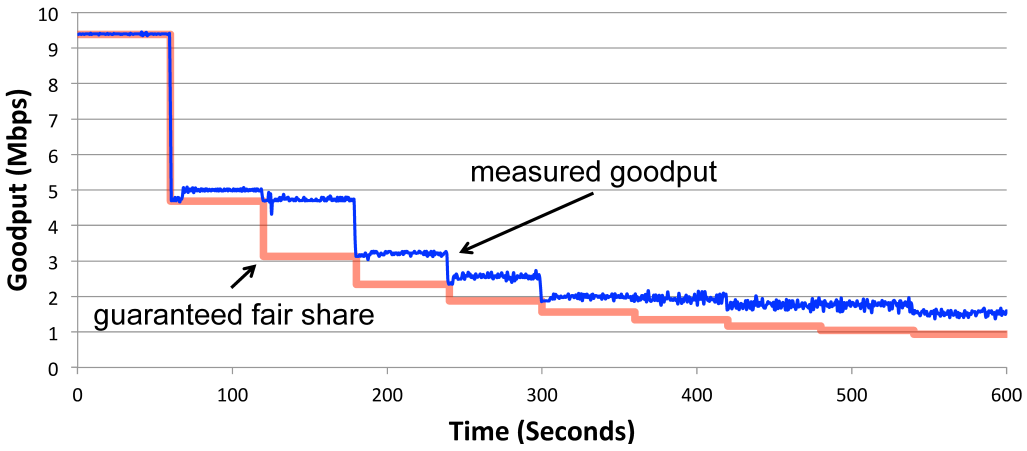

Figure 6.4: Performance of one Priority Flooding flow (thin line) impacted by an increasing number of additional active sources. The thick line shows the guaranteed fair share.

a worse scenario than K-Paths.

6.3.1 Priority Messaging

Figure 6.4 shows the performance of a single Priority Flooding flow sending at 10 Mbps in the presence of an increasing number of randomly selected active source nodes also sending at 10 Mbps (an additional source node is started every 60 seconds). At each interval, we report the measured performance (thin line) and the guaranteed fair share (thick line) based on the number of active sources (Theorem 5). The measured goodput outperforms the minimum guaranteed value because not all edges are in full contention at all times.

Figure 6.5a shows the goodput for a Priority Flooding flow sending less than its

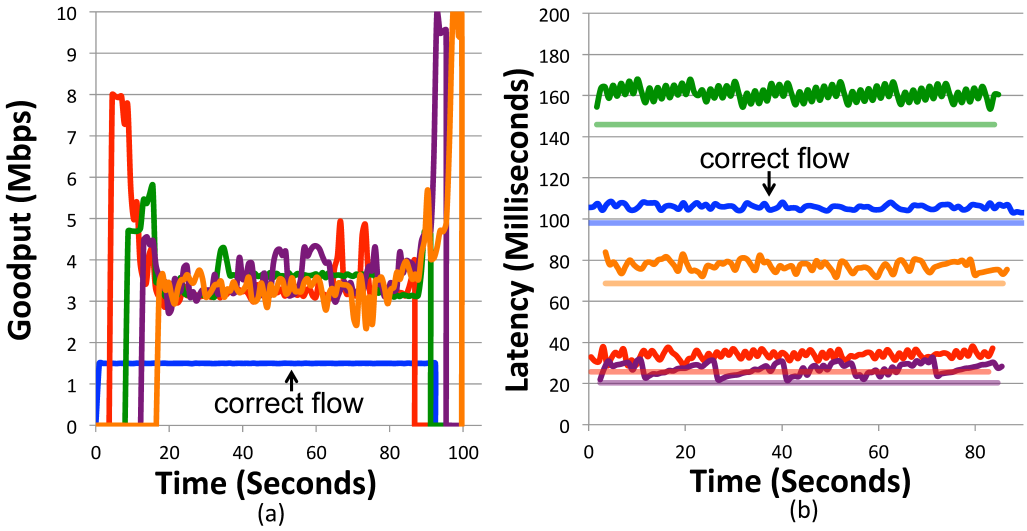

Figure 6.5: Priority Flooding (a) goodput and (b) latency.

fair share. A correct flow (node 9 to node 11) sends at a rate of 1.6 Mbps and four compromised flows send at 10 Mbps. The goodput of the correct flow is not affected because 1.6 Mbps is less than its fair share with four other active flows in the network. The remaining bandwidth is shared evenly among the other flows. These results are consistent with the guarantees of Priority Flooding (Theorem 5).

Figure 6.5b shows the sampled latency experienced by these five flows and compares it to the minimum propagation delay between the source and destination (flat line). While messages from all five flows experience latency close to the propagation delay, the latency of the correct flow's messages (second from the top) are closer to the propagation delay because it sends less than its fair share, so its messages do not wait in queues.

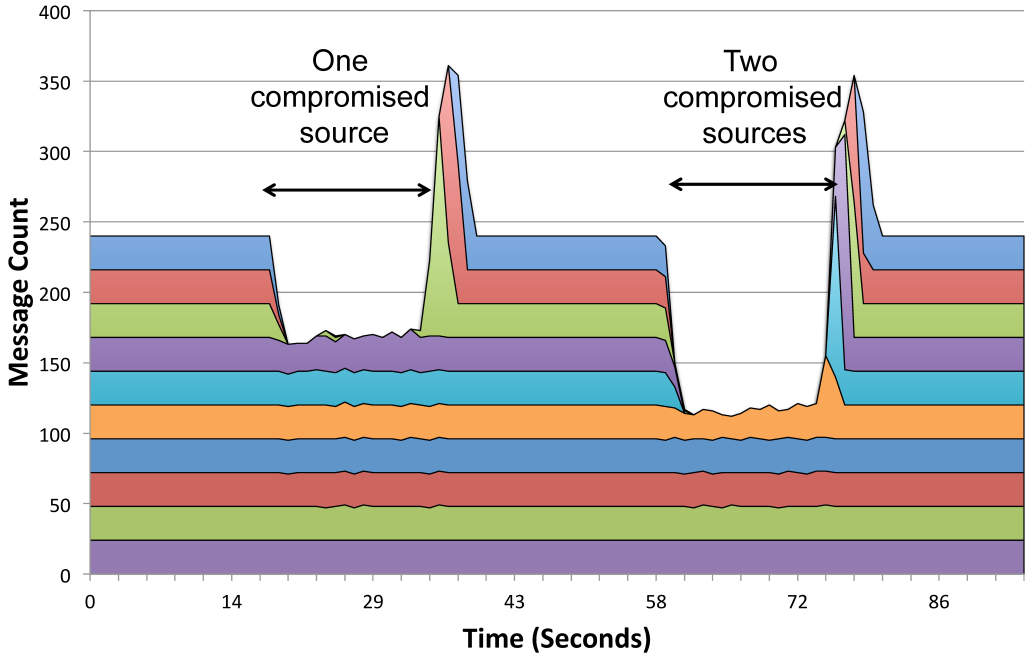

Figure 6.6: Priority Flooding under message spamming attack. When compromised nodes attempt to saturate the network with highest-priority messages, the correct node's higher priority messages (lower bands) are preserved.

These experiments show that, even with many compromised senders in the network, the system still provides usable goodput to the correct sender.

In Figure 6.6, a correct Priority Flooding flow (node 7 to node 9) sends at a rate of 7 Mbps and evenly distributes its messages across ten priority levels (ten colored bands). With no attack, all messages from all priority levels are received at the destination. When one compromised node attempts to saturate the network with highest-priority messages, the correct node's higher priority messages are preserved at the expense of its lower priority messages. After the compromised node stops, relieving

network contention, the lower priority messages still in storage at intermediate nodes are forwarded, resulting in a burst of traffic. As can be seen in the graph, this storage is cleared in order by priority (i.e. an entire priority level is cleared before starting the next lower level). The behavior is similar when two compromised nodes attempt to saturate the network with highest-priority messages, with the correct flow receiving less bandwidth (and fewer priority levels) accordingly. The ripples in the bands observed during network contention are due to message queueing, which can cause a newer higher priority message to arrive at a node before an older lower priority message is forwarded. Since higher priority messages are forwarded first, this results in message reordering. Note that in all cases, the correct flow achieves the bandwidth it requests or at least its fair share, meeting the guarantees (Theorem 5).

6.3.2 Reliable Messaging

Figure 6.7 shows the performance of a single Reliable Messaging flow (node 7 to node 9) for both Constrained Flooding and K-Paths, with various loss rates applied to all links in the network. This experiment is run in an emulated environment matching the topology and latencies of the real cloud to accurately control the injected loss. The flow is able to maintain performance, even under high loss. The flow shown starts in Europe and travels across North America, all the way to East Asia. This is

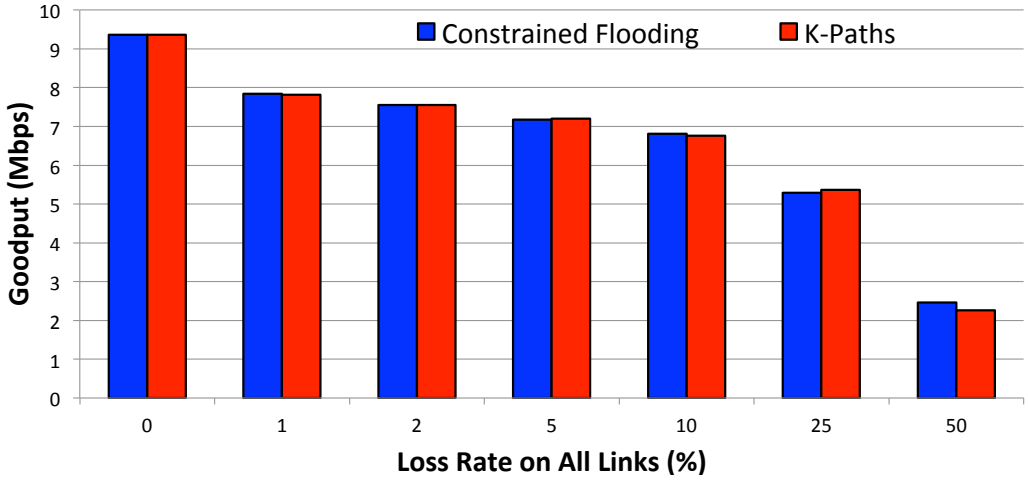

Figure 6.7: Performance of one Reliable Messaging flow with loss rates applied to all links in the topology.

the worst-case flow for loss in the topology because the shortest path (latency-wise) between source and destination contains four hops. In fact, this is one of the worst-case flows on the globe for backbone traffic because the distance is approximately half the circumference of the earth. TCP fairness is disabled on the PoR Link for this experiment. While disabling TCP fairness is generally inadvisable, it is helpful to defend against loss-based attacks.

Figure 6.8 shows the performance of a single Reliable Flooding flow sending at 10 Mbps over the course of two events: two compromised flows attempting to saturate the network, and a crash-recovery of intermediate nodes that causes a network partition between source and destination. The guaranteed fair share (thick line) is shown for reference in all cases. Throughout the experiment, the flow's goodput meets the

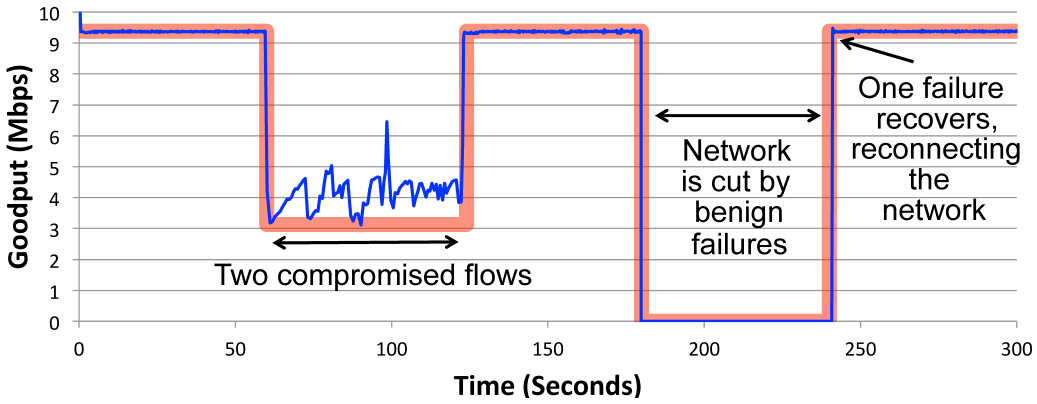

Figure 6.8: Performance of one Reliable Flooding flow (thin line), impacted by two compromised flows and by crashes that cut the network. All messages were delivered reliably.

guarantee (Theorem 12). During contention, the end-to-end acknowledgements cause network instability (see Figure 6.3d), but result in goodput even higher than the guarantee.

Experimental comparison with compromised nodes: We measure and compare the performance of Reliable Flooding when there are compromised nodes in the network. In this case, nodes 2 and 6 are compromised, ensuring that no source or destination is compromised, to provide an apples-to-apples comparison with the previous experiments. For this experiment, each compromised node behaves correctly at the link level to maintain communication with its neighbors and participates in the protocol in an initial handshake period, but does not forward data, does not send hop-by-hop acknowledgements, does not forward end-to-end acknowledgements, and

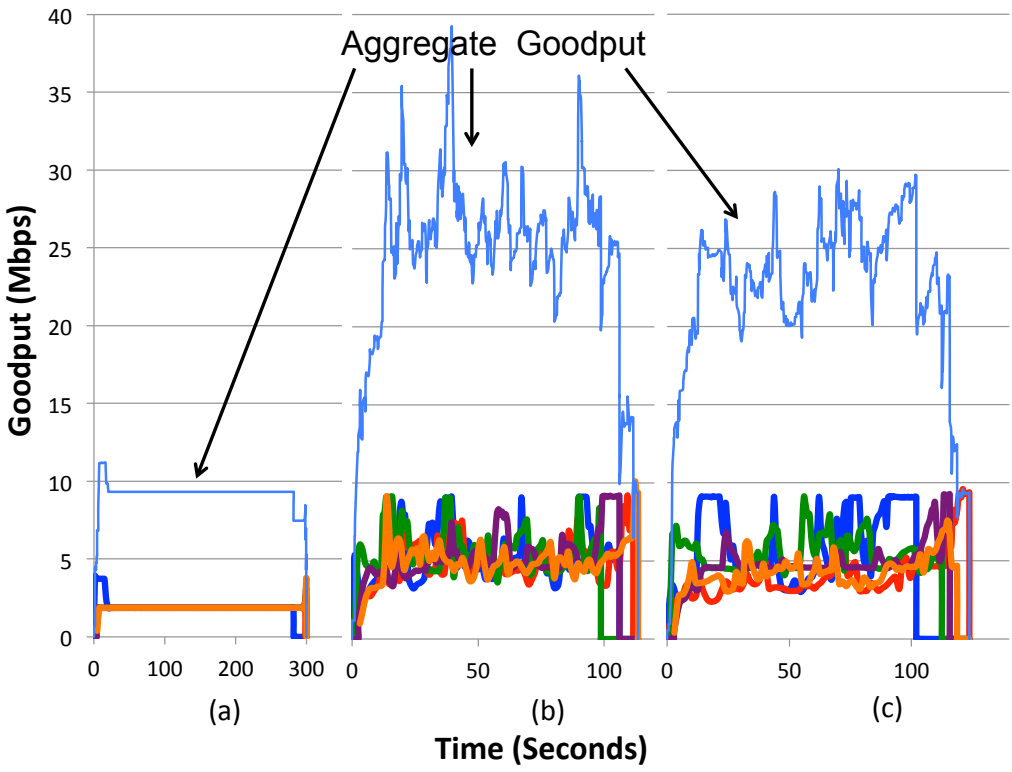

Figure 6.9: Experimental goodput for: (a) Naïve Flooding, (b) Reliable Flooding with all correct nodes, (c) Reliable Flooding with two compromised nodes. In (c), all flows meet or exceed the guarantees (compare (a) and (c)) and aggregate throughput has decreased due to the presence of the compromised nodes (compare (b) and (c)).

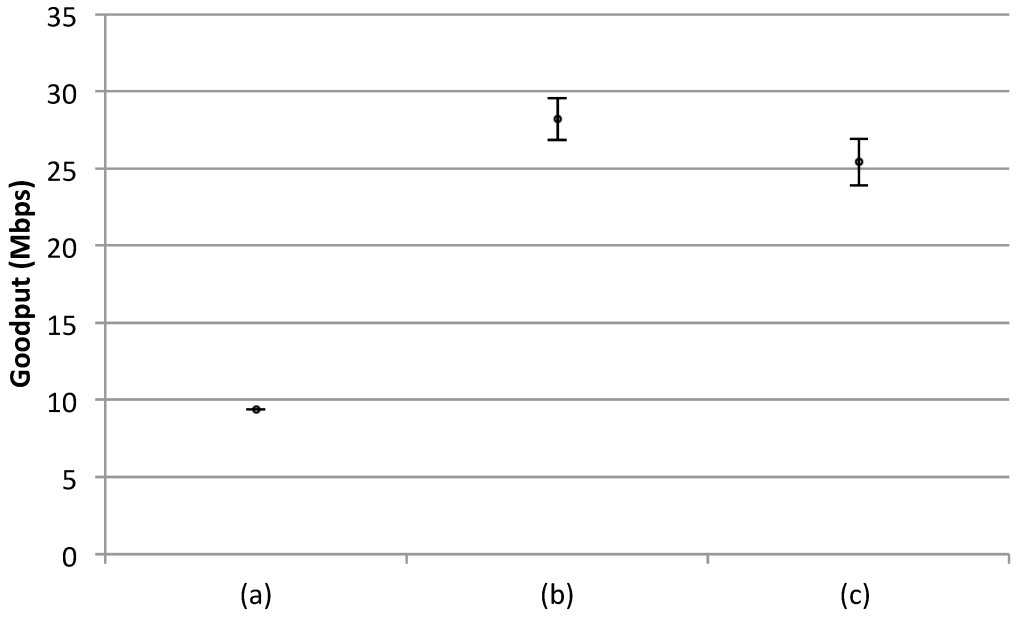

Figure 6.10: The average goodput for three scenarios: (a) Naïve Flooding, (b) Reliable Flooding with all correct nodes, (c) Reliable Flooding with two compromised nodes. Each scenario was run five times and the average aggregate throughput was computed over the time when all flows were active; the data presented here is the mean and standard deviation of these five values.

does not generate its own end-to-end acknowledgements. Note that this is only one of many, many possible behaviors for the compromised nodes. We choose this particular behavior as it does not involve the compromised nodes attempting to overwhelm the system and is difficult to detect as being compromised behavior rather than benign loss in the system (as opposed to, for example, sending malformed messages, which can be detected).

Figure 6.9 shows the corresponding goodput for three experiments. Figure 6.9a and 6.9b are different runs of the same experiments as in Figure 6.3a and 6.3d, and are replicated here for comparison. Figure 6.9c shows the goodput when nodes 2 and 6 are compromised. As can be seen when comparing Figure 6.9a and 6.9c, the compromised nodes do not prevent good flows from meeting or exceeding their fair share. When comparing Figure 6.9b and 6.9c, we see that the aggregate throughput with the compromised nodes is similar to, but not as high as, the case with all correct nodes. Nodes 2 and 6 are centrally-located in the network, so their compromise reduces the network resources available to the five flows, causing the drop in throughput. Figure 6.10 gives the average and standard deviation of the aggregate throughput across five different runs of these same three experiments, to show the same comparison.

Unlike our other experiments, the experiments with compromised nodes were

performed in a research lab with emulated latency, rather than on the wide-area network. This was done to eliminate the possibility that the compromised nodes could adversely affect production traffic.

6.4 Shadow Monitoring System

To prove the feasibility of our approach in a real situation, we used the deployment to carry the monitoring messages of the global cloud. The monitoring messages provide a real-time view of the cloud, updating every 1–3 seconds depending on the type of information. This view contains detailed information regarding the status of data centers, the network characteristics (e.g. latency, bandwidth, loss rate) of links between data centers, the status of cloud access points (i.e. clients), and the service characteristics that each client-generated task receives.

The deployment ran for several months. It was used in a limited production capacity: monitoring messages carried by the deployed network were processed and displayed in a graphical user interface that showed a real-time view of the cloud. Other alarm and log related features were not implemented. The monitoring messages used Priority Messaging because it meets the real-time requirements. Throughout the deployment, K-Paths (with K=2) and Constrained Flooding were used to evaluate their applicability.

The deployed network was able to provide the same timely delivery of monitoring messages as the original network (resulting in an equivalent real-time view of the cloud). Based on this experience, and talking with the builders of this cloud, we find the higher cost (even for Constrained Flooding) an acceptable price to pay for critical messages, given the strong guarantees gained. Further, it may even be worthwhile to use K-Paths (K=2) for the data of some select high-value applications.

Chapter 7

Conclusion

We introduced and described the first practical intrusion-tolerant network. Our solution builds on three principles: rich data dissemination semantics that ensure fairness and guarantee performance in the presence of network compromises, a Maximal Topology with Minimal Weights, and a generic mechanism for source-based routing. We defined two specific intrusion-tolerant data dissemination semantics: Priority Messaging with Source Fairness and Reliable Messaging with Source-Destination Fairness. We also discussed the resilient architecture necessary for the deployment of these ideas, using an overlay approach, diverse network providers, and multihoming on each overlay node.

We deployed the implementation on a global cloud spanning 12 data centers from

East Asia to North America to Europe. We evaluated the performance and resilience to attacks, and showed that, in all cases, the performance met or exceeded the guarantees. The implementation is publicly available as part of the Spines messaging toolkit (www.spines.org).

Bibliography

[1] S. Kent and R. Atkinson, "Security architecture for the internet protocol," Internet Requests for Comments, RFC Editor, RFC 4301, December 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4301.txt

[2] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[3] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi, "Duqu: Analysis, detection, and lessons learned," in *Proceedings of the 5th European Workshop on System Security*, April 2012.

[4] ——, "The cousins of stuxnet: Duqu, flame, and gauss," *Future Internet*, vol. 4, no. 4, pp. 971–1003, November 2012.

[5] R. Perlman, "Network layer protocols with Byzantine robustness," Ph.D. dissertation, Massachusetts Institute of Technology, 1989.

[6] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting disruptive routers: A distributed network monitoring approach," *IEEE Network*, vol. 12, no. 5, pp. 50–60, May 1998.

[7] Y. Amir, P. Bunn, and R. Ostrovsky, "Authenticated adversarial routing," in *Proceedings of the 6th Theory of Cryptography Conference*, March 2009, pp. 163–182.

[8] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to Byzantine failures," in *Proceedings of the 1st ACM Workshop on Wireless Security*. ACM, September 2002, pp. 21–30.

[9] R. Perlman, "Routing with Byzantine robustness," Sun Microsystems, Inc., Tech. Rep. SMLI TR-2005-146, September 2005.

[10] R. Obelheiro and J. Fraga, "A lightweight intrusion-tolerant overlay network," in *Proceedings of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, April 2006, pp. 8–15.

[11] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29, no. 2, pp. 216–230, January 2006.

[12] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: An on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks," *ACM Transactions on Information and System Security*, vol. 10, no. 4, pp. 6:1–6:35, January 2008.

[13] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 299–314, December 2002.

[14] J. R. Douceur, "The sybil attack," in *Peer-to-peer Systems*. Springer, 2002, pp. 251–260.

[15] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, October 2001, pp. 131–145.

[16] Y. Amir and C. Danilov, "Reliable communication in overlay networks," in *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, June 2003, pp. 511–520.

[17] G. Finn, "Reducing the vulnerability of dynamic computer networks," USC Information Sciences Institute, Tech. Rep. ISI-RR-88-201, June 1988.

[18] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2003, pp. 1976–1986.

[19] M. S. Kang, S. B. Lee, and V. Gligor, "The crossfire attack," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 127–141.

[20] A. Studer and A. Perrig, "The coremelt attack," in *Proceedings of the 14th European Conference on Research in Computer Security*, September 2009, pp. 37–52.

[21] "LTN global communications," http://www.ltnglobal.com, accessed: 2015-11-08.

[22] "Spines overlay messaging system," http://www.spines.org, accessed: 2015-11-08.

[23] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 71–78, October 1999.

[24] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982.

[25] M. Platania, D. Obenshain, T. Tantillo, R. Sharma, and Y. Amir, "Towards a practical survivable intrusion tolerant replication system," in *Proceedings of the 33rd IEEE International Symposium on Reliable Distributed Systems*, Oct 2014, pp. 242–252.

[26] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Resilient intrusion tolerance through proactive and reactive recovery," in *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing*, December 2007, pp. 373–380.

[27] M. Franz, "E unibus pluram: massive-scale software diversity as a defense mechanism," in *Proceedings of the 2010 Workshop on New Security Paradigms*, September 2010, pp. 7–16.

[28] T. Jackson, B. Salamat, A. Homescu, K. Manivannan, G. Wagner, A. Gal, S. Brunthaler, C. Wimmer, and M. Franz, "Compiler-generated software diversity," in *Moving Target Defense*. Springer, 2011, pp. 77–98.

[29] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, "Profile-guided automated software diversity," in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization*, February 2013, pp. 1–11.

[30] S. Forrest, A. Somayaji, and D. Ackley, "Building diverse computer systems," in *Proceedings of the 6th Workshop on Hot Topics in Operating Systems*, May 1997, pp. 67–72.

[31] P. Papadimitratos and Z. J. Haas, "Securing the internet routing infrastructure," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 60–68, October 2002.

[32] B. Kumar and J. Crowcroft, "Integrating security in inter-domain routing protocols," *ACM SIGCOMM Computer Communication Review*, vol. 23, no. 5, pp. 36–51, 1993.

[33] S. L. Murphy and M. Badger, "Digital signature protection of the OSPF routing protocol," in *Proceedings of the 1996 Symposium on Network and Distributed System Security*, February 1996, pp. 93–102.

[34] H. Johansen, A. Allavena, and R. van Renesse, "Fireflies: Scalable support for intrusion-tolerant network overlays," in *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*, April 2006, pp. 3–13.

[35] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "BAR gossip," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, November 2006, pp. 191–204.

[36] S. Cheung and K. N. Levitt, "Protecting routing infrastructures from denial of service using cooperative intrusion detection," in *Proceedings of the 1997 Workshop on New Security Paradigms*, September 1997, pp. 94–106.

[37] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. Andersen, "SCION: Scalability, control, and isolation on next-generation networks," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, May 2011, pp. 212–227.

[38] A. Newell, D. Obenshain, T. Tantillo, C. Nita-Rotaru, and Y. Amir, "Increasing network resiliency by optimally assigning diverse variants to routing nodes," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2013, pp. 1–12.

[39] ——, "Increasing network resiliency by optimally assigning diverse variants to routing nodes," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–14, November 2014.

[40] L. Zhou, F. B. Schneider, and R. Van Renesse, "Coca: A secure distributed online certification authority," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 329–368, November 2002.

[41] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "1-800-overlays: using overlay networks to improve voip quality," in *Proceedings of the 15th*

International Workshop on Network and Operating Systems Support for Digital Audio and Video, June 2005, pp. 51–56.

[42] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[43] J. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974.

[44] D. Obenshain, T. Tantillo, A. Newell, C. Nita-Rotaru, and Y. Amir, "Intrusion-tolerant cloud monitoring and control," in *Proceedings of the 6th Workshop on Large-Scale Distributed Systems and Middleware*, July 2012.

[45] E. Rescorla and N. Modadugu, "Datagram transport layer security," Internet Requests for Comments, RFC Editor, RFC 4347, April 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4347.txt

[46] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[47] "OpenSSL project," http://www.openssl.org, accessed: 2015-11-08.

[48] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[49] H. Krawczyk, R. Canetti, and M. Bellare, "HMAC: Keyed-hashing for message authentication," Internet Requests for Comments, RFC Editor, RFC 2104, February 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2104.txt

[50] "FIPS PUB 180-4: Secure hash standard," *US Department of Commerce, National Institute of Standards and Technology*, 2012.

[51] H. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing," in *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, October 1994, pp. 101–114.

[52] C.-Q. Yang and A. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *Network, IEEE*, vol. 9, no. 4, pp. 34–45, Jul 1995.

[53] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's

globally distributed database," *ACM Transactions on Computer Systems*, vol. 31, no. 3, pp. 8:1–8:22, August 2013.

Vita

Daniel Obenshain received a BS degree in Computer Science from Caltech in 2011, where he also received Student Life and Master's Award. When he came to Johns Hopkins he received a Beauchamp Fellowship. He earned an MSE degree at Johns Hopkins in Computer Science in 2013 and received the department's Outstanding Teaching Award the same year. He was a member of the Distributed Systems and Networks lab and his research interests include distributed systems and intrusion tolerant systems. Daniel Obenshain will start as a Research Scientist at Facebook in January 2016.