Leveraging Network Maps to Improve Evaluations of Overlay System Performance and Security

A Thesis
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Master of Science
in Computer Science

By

Christopher Wacek, B.A.

Washington, DC
December 16, 2013

Many thanks my advisor Micah Sherr for going through all the revisions, to the rest of the committee for their advice and suggestions, and most importantly, to my parents for their continual help and support.

LEVERAGING NETWORK MAPS TO IMPROVE EVALUATIONS OF OVERLAY SYSTEM PERFORMANCE AND SECURITY

Christopher Wacek, B.A.

Thesis Advisor: Micah Sherr

ABSTRACT

Distributed systems often communicate through *overlay networks*, which use custom addressing and protocols to communicate between participating nodes at the application layer, but route those custom messages over the standard network infrastructure. Overlay networks enable application system designers to focus on the intended operation of their system distinct from the network layer. This can have several benefits: improvements at lower levels of the technology stack can be assimilated without modifying the application layer protocol, and modeling application behavior is easier because the protocol doesn't depend on network interactions.

However, while this enables overlay networks to be easily studied and modeled, this very abstraction can make it difficult to understand how their interaction with the network underlay affects them. Ideally the application's behavior would be completely isolated from the network layer, but in practice this is rarely the case. For instance, application layer modeling cannot easily predict exactly how a widespread deployment will behave; security and performance can both be affected by the path overlay networks take through the underlay network. This can make conscientious operators of overlay networks hesistant to make large modifications to their protocol for fear its interaction with lower layers, once distributed across the internet, will have unintended effects. For instance, the Tor Project, which manages the Tor anonymity network [18], is relatively conservative with respect to protocol changes, in part because of fears that a change might affect anonymity through

some unexpected interaction with the underlying network, whether due to routing or performance.

The goal of this thesis is to introduce *network maps* which can be used to effectively evaluate overlay network technologies with respect to both performance and anonymity within evaluation platforms that provide a safe environment for experimentation. Safe evaluation environments are critical in that they permit modification of core protocols without affecting active system users. We discuss the advantages and disadvantages posed by different classes of evaluation platforms and how they can interface with our proposed *network maps*.

We present a series of techniques for constructing these *network maps* which combine network information from disparate sources into large graphs which represent the global internet. For each type of network data, we discuss the sources from which they can be obtained and the types of inaccuracies they can introduce in network evaluations. Given the set of available data, we propose methods for constructing *network maps* by combining these sources of information.

We develop maps at two granularity levels, then present several case studies which use the proposed mapping techniques in combination with several platforms to perform security and performance evaluations of the Tor anonymity network, including a consideration of the effects of modifications to the Tor protocol. The first study investigates the performance and security implications of a number of modifications to Tor's relay selection strategy. We show that while Tor's existing strategy is highly effective, there are opportunities for performance improvement from layered selection strategies. A second study researches the level and prevalence of the threat posed to Tor users by network level adversaries, showing that Tor users are highly vulnerable – perhaps more so than previously thought – against network adversaries.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The advent of the Internet has revolutionized the environment in which computer software is developed and operated. In recent years the ability to rapidly interconnect software across the globe has driven an increasing trend towards applications whose core functionality depends on interconnectivity and the ability to communicate with central infrastructure. The rise of this type of application was enabled through the usability of the global network infrastructure, most critically the global adoption of a common suite of protocols designed to provide point-to-point communication between arbitrary hosts on the network. These protocols, headlined by the TCP/IP stack, have been researched for many years and are well understood in the computer science community.

Traditionally, network-connected applications operated in a client-server model, where information was either pushed to or pulled from a central platform. Now a new trend is growing, where applications do not simply connect to a central system, but instead form distributed systems where application instances act in concert with each other. These distributed systems are used for a wide variety of purposes: BitTorrent enables scalable distribution of large files, Skype's supernodes help voice calls navigate network infrastructure, and Tor [18] provides a network that enables users to browse the web anonymously.

Distributed application present new challenges for their developers and operators. They operate in disparate environments and locations, and their behavior can be affected significantly by those differences. Simultaneously, the distributed nature of these applications makes it difficult for operators to observe the conditions and behavior of each node; in many

cases the developers and operators may be separate entities with limited ability to communicate. This was the case with Skype, where supernode operators were simply normal Skype users who have been "promoted"[1].

These applications form and operate within what is called a "network overlay", a virtual network which contains only nodes that speak the application protocol. These network overlays often treat the underlying network transparently and do not attempt to affect or understand network level interactions. This simplifies the development and creation of the application, but does not mean that the interactions do not exist. Those interactions may affect the security and performance of the overlay in various ways.

For instance, it has been suggested that Tor could change the way clients select the relays they use for anonymous paths in order to permit quality of service (QoS) policies for different applications [62]. Users who use chat services might then optimize their relay selection to improve the latency of their connection. Other users who are concerned that certain world governments might attempt to track their communications might select relays to avoid the countries in question. The modifications required for the Tor protocols to enable such a QoS selection are not complicated – indeed that they function as intended could be evaluated with little difficulty. However, enabling QoS functionality in Tor might cause anonymity to suffer by increasing the ability of an adversary to perform traffic correlation attacks. The extent to which the adversary is empowered cannot be understood without considering how traffic is routed across the underlay network. Evaluation at the overlay application layer alone provides no insights, because the attack is affected by the interaction with the underlying network.

These two factors – that the overlay is affected by the underlying network and that it is difficult to observe the operation of distributed remote nodes – means that emulation

[1]This is actually no longer true [14], but was part of the original design.

and simulation become very attractive methods for evaluating application changes. However, simulation also presents its challenges, namely that an effective evaluation requires simulating not only the application behavior, but the environment in which it operates. As a result, a major component of evaluating overlay systems becomes determining how to properly simulate the environment, specifically how to arrange the network graph such that it approximates what might be expected in the real world.

This thesis suggests that existing data about the Internet can be leveraged to produce *network maps* that model the operating domain of a network overlay system. These network maps are representations of the internet at varying levels of fidelity, designed to enable emulation and simulation to answer difficult questions about network overlay systems. That is, these maps are intended to help elucidate the interactions between the overlay systems and the underlying network in globally distributed systems.

Network maps of this type can allow developers and researchers to sidestep the difficult question of how to appropriately simulate networks on which to test their overlay systems. An actual scaled down representation of the internet can be used, enabling the ability to conduct evaluations not just of the application layer, but of how that application layer interacts with the underlying network to affect performance and security. In the case of relay selection for Tor described earlier, a network map might show the Tor relays that could be selected for an anonymous path and the network entities that could observe the links upon that path. In Chapter 5, we describe building a map of this very type.

In this work we suggest a general framework for developing network maps from extant datasets, focused on evaluating overlay network performance and security. We then describe two evaluations that use these techniques to build maps and enable performance and security analyses of the Tor anonymity network. In these evaluations we show how different network maps can be constructed for different purposes – this specificity allows

the size of the network map to accommodate different resource constraints and needs. Some of the specific contributions of this work include:

- A general framework for developing network maps for evaluating overlay networks from available datasets;
- The most complete and accurate network representation of the live Tor network to date;
- An evaluation of several proposed relay selection strategies for the Tor network;
- An evaluation of the traffic corrlation threat posed by network adversaries to Tor users.

While our evaluations focus heavily on Tor, similar approaches could be applied to other overlay systems. For instance, a company that uses Skype for business communication might wish to understand how frequently Skype calls are routed through countries which might eavesdrop on their communications. A routing security focused map could enable this type of evaluation. We suggest that network maps are a generalized technique for evaluating network overlay systems in a safe environment, where the interaction with the underlying network is of specific interest.

Our evaluation of relay selection strategies in Tor shows that Tor's default selection strategy is highly effective because it aggressively weights for bandwidth, while other proposed strategies provide little to no improvement in performance. Our study of traffic correlation suggests that Tor users may be more vulnerable than previously believed to traffic correlation against relatively strong adversaries – an adversary who controls a single autonomous system (AS) has a 50% probability of being able to observe both ends of an Internet Relay Chat (IRC) users connection within 44 days. We describe these findings in detail in Chapters 4 and 5.

Our discussion begins with an overview of the data and evaluation platforms that can be used to support this type of evaluation.

CHAPTER 2

BACKGROUND AND RELATED WORK

Prior to delving into the process of constructing network maps for overlay network evaluation, it is important to understand the wide variety and scope of data available. Since computers were first networked, a significant amount of time and effort has been put into the development of simulation and emulation tools which can be used to evaluate the operation of those networks. In the process, researchers have attempted to produce datasets about the networks we use, which can enable experimentation and research using those tools. This section surveys the types of datasets and tools which prove useful in the evaluation of network overlay systems.

2.1 TYPES OF NETWORK DATA

Many of the protocols that comprise the Internet are driven by and produce data about their operation: from BGP route advertisements we can glean information not just about how traffic will traverse the network, but also about how network entities may be related. Other data are produced as output from tools primarily used for diagnosing issues, such as `ping`, `traceroute`, and port scanners.

In the construction of network maps, we use many sources of data about the Internet loosely categorized into two types: *network structure data* and *metadata*. The accuracy of evaluations performed using our network map methodology necessarily depends on the quality of this data. If we are to perform accurate evaluations we need real data about

the network conditions on the Internet. In this section we describe the difference between *network structure data* and *metadata* and highlight some of the data sources available in each.

2.1.1 Network Structure Data

The structure of the Internet is essentially a graph. Vertexes, represented by hosts, routers, and other computing entities are connected by a web of connections which form the edges of the graph. Network structure data describe the shape and properties of this graph. Building an effective network map requires information about the vertices and the edges, including appropriate attributes for each of them. Tools such as ZMap [20] can be used to rapidly enumerate the vertices of this graph and identify characteristics of each of the end hosts.

One of the sources of both node and link data is *IP Trace Data* created by establishing servers at many distributed vantage points and running `traceroute`[1] to a large number of destinations from each one. CAIDA's Archipelago [7] infrastructure is designed specifically to produce this type of data: three teams of 18 monitors collectively probe the entire routed IPv4 /24 network space once every few days. CAIDA publishes the results of their data gathering efforts in the *IPv4 /24 Routed Topology dataset*, which provides the output of the 10.1 billion traceroutes performed since June 2007.

This type of traceroute data is valuable because of the nature in which it is created: each trace represents the actual path traversed by a packet across the Internet with the intermediate time taken to transit between each router. This means that *IP Trace Data* can be used to provide indications of routing relationships (further discussed in Section 2.1.2) and to estimate the latencies across network links.

[1] Many of the projects gathering this type of information do not use the UNIX `traceroute` tool, but a more advanced active measurement tool capable of using multiple different protocols, such as `scamper` [43].

CAIDA is not the only entity to gather this type of information. iPlane [44] similarly runs traceroutes from a set of distributed vantage points; some of the vantage points they use are located on the PlanetLab [58] global research network. PlanetLab can also be used to collect this type of data independently.

Another important type of network structure data is the physical characteristics of the links identified through IP trace data, such as their bandwidth, latency and loss rates. Some of this – notably latency and drop rates – can be gleaned from the trace data directly. However, the bandwidth of a link is not usually provided through trace data due to the type of measurement, the distance of the measurer from the link in question, and the quantity of interposed links. As a result, other techniques have been applied to measure network throughput:

The venerable "packet pair" technique can be used to measure the bottleneck throughput of an end-to-end link by observing the increase in interarrival time between two closely spaced packets sent over that link [38]. However, this technique is not terribly effective for measuring the throughput of arbitrary links on the Internet because it can only measure the bottleneck value between the source and sink.

iPlane attempts to measure the bandwidth of each /24 network in the Internet by passively participating in BitTorrent [11] swarms and taking the median observed uplink and downlink speeds to hosts within that /24. This technique may be effective, but can by definition only measure class C networks containing an active BitTorrent user and assumes that the BitTorrent client is not throttled.

Another approach to obtaining measurements of bandwidth available to different areas of the Internet is to simply ask. The popular speed test website SpeedTest.net provides the free NetIndex dataset [54] which aggregates data gathered through bandwidth tests for each country or region where a test has been run. This connection speed survey may provide more direct answers about the speed which a region is *capable* of achieving, but

suffers from a self selection problem in that those with very low bandwidth and very high bandwidth are more likely to run speedtests (for different reasons). The vast majority of people who are satisfied with their basic service may not have any reason to run speed tests.

In general, measuring the throughput of an arbitrary Internet link is difficult. We have described some techniques which can provide a rough approximation – the best estimate is likely to combine all of the available information.

At a higher level of granularity than physical links and routers, Border Gateway Protocol (BGP) route advertisements can provide information about the structure of the network graph. BGP advertisements describe the connections between ASs, although they do not contain data about the latency or bandwidth of those links. The RouteViews project [73] collects and archives advertised BGP routes to every internet prefix from several vantage points.

2.1.2 METADATA

The structure of the physical links which form the network graph of the Internet is necessary but not sufficient to understand how traffic flows and is routed. The Internet (from 'Internetwork') is comprised of smaller ASs connected at peering locations. Metadata about these networks may provide additional detail about network ownership, segmentation and routing. For instance, protocols have been developed to inform hosts on the network how to route packets to distant networks; the metadata produced by those protocols can allow researchers to reproduce the routing structures as they existed on the network at a given point in time by defining who owns networks and how they are related.

Routing information is one sort of metadata which can augment the ability of researchers to perform studies about the network; the analysis of Tor routing security

described in Chapter 5 relies heavily on a network map annotated with ownership metadata.

Other metadata can provide valuable information to an evaluation. Network ownership is critically important in routing and may also be of interest to a security evaluation, while knowledge of relationships between networks can help inform how traffic will flow across them. In the following section we describe different types of metadata in greater detail.

Ownership Attribution

The standard protocol for inter-network routing as occurs on the Internet is Border Gateway Protocol (BGP). At a high level, BGP provides routing to all Internet addresses based on shortest path. However, it has been shown that business relationships exert a strong influence on BGP routes [27]; ISPs are often unwilling to transit traffic for which they are neither paying or receiving payment. Since routing is heavily dependent upon business relationships, ownership attribution for segments of the network becomes important. Additionally, ownership information is useful in its own right if the location of specific entities is important, as would be the case with an evaluation of how frequently overlay traffic crosses zones in which it could be intercepted by foreign governments.

Autonomous System (AS) numbers are assigned to networks by the Internet Assigned Numbers Authority, which delegates to the regional Network Information Centers (NICs). While the assignments can be drawn directly from these databases, a number of groups provide IP to AS number mapping services. MaxMind includes IP to AS number tranlation as part of its geolocation services [41]. Cymru, a security focused non-profit, also provides a mapping with additional details such as the router originating routes for that AS and registration information [12].

IP to AS mapping is only the first step in performing ownership attribution on a host. Many autonomomous systems are in fact owned by the same or closely related organiza-

tions. These siblings occur through mergers and acquisitions, or simply because an organization maintains multiple geographically separated networks. Identifying these siblings is an important step in inferring routes between autonomous systems.

A powerful source for identifying related organizations is the WHOIS database maintained by each of the regional Internet registries. The records in these databases describe organization names, contact information, and addresses for each of their assigned AS numbers; this information can be used to identify autonomous systems owned by related entities.

Internet Exchanges are another type of entity which operates network segments within the Internet, but which are distinct from an autonomous system. Internet exchanges operate facilities known as internet exchange points (IXPs) in which many ASs come together and exchange traffic. In many ways, IXP can be considered as very large routers, with the caveat that in theory they are capable of observing all traffic which passes through their infrastructure. The IXP Mapping Project [4] gathers data about IXP across the Internet, and seeks to identify the ASs which peer at each IXP.

Relationship Data

The attribution data previously described are raw data; they are obtained largely intact from various Internet authorities. Another type of data is produced by various organizations who process and combine raw data to constuct datasets which provide additional information. For example, CAIDA provides access to their IPv4 Routed /24 AS Links dataset [8], which identifies links between ASs through processing of the data from their IPv4 /24 Topology data. iPlane provides a similar set of data, drawn from their traceroutes, although at a slightly lower level of granularity.

Another constructed dataset which can be of immense use in constructing network maps is an AS relationship database. This type of data attempts to identify the business

relationships between autonomous systems, which generally fall into three categories: provider/customer, peer, and sibling. CAIDA provides a dataset containing this type of data [10]; however, when constructing network maps from disparate sources the relationship data is often incomplete and must be augmented as we show in Section 5.2.

2.2 Experimentation Platforms

Effectively evaluating network overlays requires considering the activity of a globally distributed application in which the ability to control activity may be limited and even the view of the activities that are occurring may be obscured. This can make it difficult to perform research and impossible to control results between experiments.

To address this problem, researchers have developed a wide range of *simulation* tools which provide approximations of application and network behavior for the purposes of repeatable experimentation. These tools vary significantly in the fidelity they provide along different axes of application behavior. For instance, general network simulation tools tend to provide high fidelity depictions of the network but may have less complete application models. On the other hand, specialized application simulators may be accurate in simulating the behavior of an application or protocol, but may be less effective at understanding network interaction.

Tools all along this spectrum can be valuable to researchers, depending on their purposes. In this section we highlight a variety of different types of simulation tools, some of which are used in the case studies found in Chapters 4 and 5.

2.2.1 Application-layer Simulators

Application-layer simulators represent what can be the fastest, but lowest-fidelity, method for evaluating network systems. These are application-specific software components which

attempt to emulate the application behavior, but do not consider the details of network interactions.

BRAIDS [32] is a discrete event application simulator designed by Jansen *et al.* to simulate Tor behavior. It is a discrete event simulator written in Java, highly specialized to the purpose for which it was designed – evaluating the effectiveness of incentive mechanisms in Tor. Network communication is highly abstracted, with messages simply scheduled to be delivered to other nodes at the appropriate time.

p2psim is a discrete event application simulator designed by Gil *et al.* at MIT [28]. The goal of p2psim was to provide a simulation platform within which many different peer-to-peer protocols could be analyzed and compared easily. p2psim has pre-built modules for the Vivaldi network coordinate system [13] and the Chord [67] and Kademlia [46] distributed hash tables. It provides a basic network interaction model where communication occurs through remote procedure calls (RPCs), with delays inserted as appropriate for the network topology chosen.

TorPS is specifically designed to simulate Tor path selection, using historical network data to recreate the conditions under which clients operated in the past and then executes path selection algorithms over those conditions given user actions [37]. TorPS includes a model of the Tor relays and their past states, a model of user behavior, and a model of the Tor client. For each sample simulation, it takes streams produced by the user model and network states from the network model and uses them as input to the client model, which chooses circuits and assigns streams to them. In Chapter 5 we use simulation output from TorPS in conjuction with a network map produced using one of our proposed methodologies to evaluate routing security in Tor.

2.2.2 NETWORK SIMULATORS

Network simulators could be described as simulation *platforms* in that they attempt to simulate the network behavior in a detailed manner and allow application simulators to be built above them. The network stack is constructed to emulate the model followed by most network stacks, with layers approximating those of the OSI model [79] passing data between them.

NS-3 was developed by a large consortium of researchers [31] and may be the most well known network simulation platform in existence. The goal of the NS-3 project is to build a strong simulation core which can be used to simulate a wide variety of network types, ranging from point-to-point links to WiMAX wireless signals. NS-3 network simulation is highly detailed and highly complex, with the goal of accurately recreating the behavior of real networks. As such, it constructs complete simulated network stacks including most of the levels of the OSI model. The physical channel, network device, and protocol layers are each distinct simulated elements.

Network simulators provide a realistic network model but require that applications be written within their framework, which can add complexity. This can mean that these types of tools are better suited for prototyping new applications or even layer one and two protocols rather than evaluating existing ones where significant advanced functionality may need to be rewritten.

2.2.3 NETWORK EMULATORS

Network emulators have an entirely different focus from network simulators, in that they are focused primarily on the application being evaluated rather than the network. They do this by using indirection to reroute packets on the actual network infrastructure on modern operating systems, then running instances of actual applications on the host.

This approach has a number of advantages for evaluating existing applications, the most significant of which is that researchers do not need to modify the application to use it in the simulation. This saves the time and effort required to build an effective simulation model of the application and includes any bugs and corner cases that exist in the original application.

ModelNet was developed by Vahdat *et al.* at Duke in 2001 [74]. ModelNet provides emulated networks using a pair (or more) machines operating in two roles: *network emulator* and *application host*. The *application host* has multiple network interface aliases, and runs many instances of unmodified network-capable applications with each of them bound to a specific alias. Network traffic is sent to the *network emulator*, which maintains an in-kernel map of the emulated network as discrete links each with separately configurable link speed, latency and loss rate. ModelNet is capable of scaling until resources are exhausted (e.g. the *application host* has saturated its CPU) although performance artifacts may appear before fatal errors occur. In later work, Bauer *et al.* developed a series of wrappers dubbed ExperimenTor intended to simplify running experiments on the Tor anonymity system within a ModelNet environment [5].

Common Open Research Emulator (CORE) is a real-time network emulator developed by the Naval Research Lab (NRL) [1]. It allows multiple applications to connect to the network independently using a FreeBSD kernel patch which allows multiple virtual instances of the network stack to be run concurrently. These virtual network stacks can pass packets between each other based on the presence and characteristics of simulated links. This architecture permits many processes to operate in lightweight 'paravirtualized' environments, but may suffer from the same resource limitations as ModelNet.

Mininet is a network virtualizer designed to enable applications to run within a paravirtualized environment [39]. It has a particular focus in enabling the development of OpenFlow software defined networking prototypes – software developed on Mininet can be deployed directly on OpenFlow hardward switches. Its original goal was to simulate

"1000 nodes in a laptop" so its scalability at present is limited, although it is actively under development.

2.2.4 Hybrid Approaches

Hybrid approaches are those which combine aspects of each of the methods described above. For instance, the Shadow simulator developed by Rob Jansen, *et al.* has similar benefits to network emulators but is able to scale without incurring artifacts [33]. The tradeoff is that simulation can no longer occur in real time.

Shadow is a discrete-event network application simulator which uses a plugin model to implement different application types. However, these plugins can be complex C applications, which are simulated by running a single instance of the application code and establishing a memory image for each discrete process within the simulator. Each time a process needs to act, its memory image is swapped in and execution continues where it was last halted. Networking system calls are intercepted by Shadow and redirected through a virtualized network stack assigned to that node.

Scallion [33], a Tor plugin for Shadow, allows researchers to run nearly unmodified versions of Tor within the Shadow simulator and has been shown to be effective when simulating thousands of hosts.

2.3 Additional Related Work

Other researchers have struggled with the difficulty of designing network topologies for use in simulation environments. Initially, researchers generated topologies based on models of how the Internet was structurally connected, but without empirical data. One of the early generators was the Transit-Stub model, part of the Georgia Tech Internetwork Topology Models [78], which considers two types of nodes: "transit" and "stub". Networks are

formed by clusters of stub nodes around transit nodes. These networks are interconnected by links between transit nodes.

Adaptations from this early model included "Tiers", which conceptualizes the Internet as a hierarchical structure connected by a minimum spanning tree [19]. The tiers represent the Wide Area Networks (WANs), Metropolitan Area Networks (MANs) and Local Area Networks (LANs) from which the Internet is constructed. However, Tangmunarunkid *et al.* showed that the Internet is in fact far less structurally rigid than the forms that either of these generators enforce, especially at lower levels of the hierarchy [70]. In fact, Floyd and Paxson note that developing models of the Internet is exceedingly difficult in general because it changes rapidly, presenting a moving target [24].

We take a different approach: rather than attempt to replicate the hierarchical structure of the internet using randomized generators, we seek to develop models from data that provide us with a snapshot of the Internet *at a given point in time*. This type of topology generation can provide reasonably accurate network maps for specific applications. Jansen *et al.* [35] construct a network graph that clusters hosts into geographical regions, assigns upstream and downstream bandwidths and loss rates to hosts using measurements from Ookla Net Index, and utilizes link latency and jitter data obtained from iPlane [44]. In the work with Moore *et al.* [48] we use bandwidth data from Ookla Net Index to approximate the bandwidths available to Tor relays and clients in a study of Tor congestion, but do not consider network latency or loss rates. As described further in Chapter 4 Wacek *et al.* [75] leverage the fact that the Tor network publishes bandwidth information for each relay and use that to set bandwidths for relays in the network.

The general network maps that we present in this work differ from the model suggested by Jansen, *et al.*, by seeking to include more detail about the network structure and ownership, and by being flexible to additional types of data. Rather than segregating into geographic regions, our models are capable of (1) delineating at the level of routers within

the network and (2) maintaining network ownership and other metadata throughout the network. Additionally, our models are flexible and can be pruned to reduce size by including only information relevant pertinent to the research.

CHAPTER 3

NETWORK MAPS

This thesis proposes constructing two different types of network maps which can be applied to different research questions focusing on either the performance of overlay networks or their security. These maps are not mutually exclusive, but focus the structural data and metadata to specifically support queries about routing security and performance, respectively, while maintaining a structure that is minimal in size and complexity.

In each case, the foundation of the map is based on related, and even sometimes the same datasets. However, the methods and procedures used differ.

3.1 PERFORMANCE-FOCUSED MAPS

An effective network topology for evaluating how modifications to overlay protocols affect performance must have high fidelity with regard to basic network characteristics such as latency and bandwidth. Effectively, we desire a "scale" model of the actual network that accurately reflects the network's bandwidths and latencies, as well as the locations of different points of interest, such as clients, destinations, and overlay network relays.

The Internet is a collection of individual point-to-point links between routers. Those routers belong to ASs, which may or may not cross country borders. Several previous attempts to build effective network maps for emulation have attempted to abstract link information at either the AS or country level. Unfortunately these abstractions frequently lose information because important information is contained *within the abstraction*. For

instance, much of the latency between points in two different ASs is contained in the individual ASs, rather than the links that connect them. This occurs because the abstraction (AS in this case) is too large, and encompasses too large an area.

To obtain a level of abstraction whic provides the desired granularity, we construct the performance-focused network map at the level of of a *point-of-presence* (PoP), where a PoP is roughly intended to represent a "node" on the Internet. A node may have multiple interfaces or IP addresses; in fact, a node could subsume a number of very close actual machines, such as a number of routers in a single rack. This allows us to compress the size of the network graph without losing significant information because we only compress nodes that are very close to each other.

Once nodes in the network are identified, we perform a series of iterative steps intended to prune and compact the model to reduce the burden placed on the emulation platform of choice. This is an important factor for large networks. While some simulators remain accurate at large scale, they suffer in terms of performance. Eliminating unnecessary components from the graph is prudent to ensure that simulations finish in a reasonable amount of time.

We now describe the steps taken in practice to build the foundations of a performance-focused network map.

1. *Latency normalization.* To construct the map, we utilize `traceroute` data from CAIDA [9].[1] CAIDA collects and makes available `traceroute` measurements to most of the Internet's /24 prefixes from geographically and topologically diverse vantage points. We normalize the traceroutes by removing all *negative* latency hops; these occur when the traceroute data indicate that the latency required for reaching a node b in a path sequence $a \rightarrow b \rightarrow c$ is *greater* than that required to reach node c in

[1] iPlane [44] provides a similar `traceroute` dataset; we use CAIDA because, as of this writing, its data were drawn from a wider distribution of sources and destinations.

the same path.[2] We normalize such occurrences by setting the time required to reach b to be the average of the times required to reach a and c.

2. *PoP grouping.* Using the cleansed CAIDA data, we group IP addresses into PoPs using a simple *nearness heuristic*: IPs within 2.5 ms of each other, within the same /24 network, and belonging to the same AS are assigned to a single point-of-presence (PoP). These grouping rules preserve the AS paths in our topology and reduce its size significantly while still maintaining meaningful inter-PoP latencies.

   The nearness heuristic may result in multiple "edges" between two PoPs. This occurs when the CAIDA datasets contain `traceroute` measurements for multiple $(src, sink)$ pairs, where src and $sink$ are IP addresses belonging to the two respective PoPs. To ensure that only one "edge" exists between PoPs in our model, we assign the latency of each PoP-level link to be the median latency over all the $(src, sink)$ links. Alternately, we could store the variance over these latencies or a distribution of them; this would permit evalution platforms to vary latency each time a link is traversed.

3. *Attaching Points of Interest.* We define a Point of Interest (PoI) as any network location which is intended as a network endpoint. This may include clients, destinations, or network service points; in Chapter 4 we describe a case study which defined Tor clients, relays, and destination as PoIs and placed them in the network graph according to heuristics specific to Tor.

4. *Graph pruning and compaction.* To reduce the size of our model and make it practical for experimentation, we prune unimportant nodes and edges. First, we perform All-Pairs-Shortest-Paths over the PoIs and retain only the nodes and edges that appear on the shortest paths. We use shortest path as an approximation for routing through

[2] Such inconsistencies likely occur due to jitter and other transient network effects that take place during successive ICMP echo requests belonging to the same traceroute query.

the network map. Existing work has demonstrated that the Internet generally obeys shortest path routing policies, with some notable exceptions [25]. Conceptually, this removes the portion of the Internet from our model that does not "participate" in our modeled overlay network. Second, we iteratively replace all segments $a \leftrightarrow b \leftrightarrow c$, *where* b *has degree two*; if $w(a \leftrightarrow b)$ and $w(b \leftrightarrow c)$ are the respective costs of links $a \leftrightarrow b$ and $b \leftrightarrow c$, we remove b from our model and insert a new edge $a \leftrightarrow c$ with cost $w(a \leftrightarrow c) = w(a \leftrightarrow b) + w(b \leftrightarrow c)$.

The resulting model represents a reduced map of the Internet built directly from `traceroute` data that contains all of the PoIs that are of interest to an evaluation. However, this map does not contain all of the required elements for evaluating an overlay system, many of which are specific to the system being evaluated. We describe the additional components necessary for a performance evaluation of the Tor network in Chapter 4.

3.2 Routing Security Maps

One of the ways that overlay networks can be affected by the underlying network is through the routes that application packets take. While these routes are transparent to the application, it can present concerns. For instance, the ability of adverasries to observe application level traffic is dependent upon whether traffic is likely to flow through the networks under their control. If it does, adversaries may be able to eavesdrop on Skype calls or deanonymize Tor traffic.

We term this issue "overlay routing security" and have devised a method for constructing a network map which can assist in evaluating the routing security of an application. In Chapter 5 we apply this method to an evaluation of network level adversaries in Tor.

When evaluating routing security, the entities and organizations through which traffic is routed are more critical than the performance characteristics of those links.[3] As a result, when constructing an Internet map for that purpose, our goal is to highlight the segments of the network infrastructure that could be controlled by potential adveraries: network links, routers, and the facilities that host this equipment. The primary type of network ownership on the Internet comes in the form of ASs, the interlinked networks that comprise the Internet.

Since ASs represent the core of ownership on the Internet, we begin building our map by building a graph of AS from two sources. First, we consider the endpoints of links contained within BGP paths gathered from geographically distributed *RouteViews* routers [73]. RouteViews data provides a "ground truth" about how ASs route traffic; the data contains the actual BGP paths to each IPv4 prefix as advertised to a RouteViews router. However, RouteViews may not give a complete set of AS links because each datapoint originates from a fixed location.

To fill out the AS graph, we supplement RouteViews data with additional AS links identified from traceroutes in the CAIDA *IPv4 Routed /24 AS Links Dataset* [9]. The CAIDA data improves the connectivity of our graph because it is gathered from more than three times as many vantage points across the Internet. While these links do not have explicit routing information, they do implicitly provide routing information – if a traceroute traversed a particular link, then the link was part of an actual Internet route at that point.

Having constructed our AS graph, we need to determine how traffic would be routed through it. Since BGP routing is dependent upon relationships between ASs, the next step is to obtain a near-complete set of relationship information for the AS links contained in our graph. This type of relationship information can be difficult to obtain because it is usu-

[3]Except of course for the case where performance characteristics of the underlying network affects routing. We leave those situations aside for now.

ally confidential. However, a number of algorithms have been developed which attempt to map relationships by assigning directionality to links. The seminal work was by Gao [26]. Other records about organizational ownership can be used to identify certain types of relationships; specifically in the case of siblings we can identify similarities in the registration records for different ASs and use those to determine organizations which are likely components of a single larger entity. Our approach applies a multistep process which applies these heuristics iteratively to obtain a complete, reasonably accurate, set of relationships.

1. Apply Gao's heuristic algorithm to the network graph. This algorithm provides a baseline set of relationships that covers all ASs within the graph. This is important since the more accurate datasets we apply on top may not cover all links within the graph.

2. Overlay data drawn from available AS relationships datasets, replacing any relationships previously identified.

3. Heuristically identify sibling organizations from WHOIS records based on organization name, address, and contact similarity. Use identified siblings to correct any sibling relationships misclassified as peer or provider/customer.

This process effectively augments the majority of links within our AS network graph with relationship information.

We may wish to additionally annotate the graph with the locations of IXPs, since they represent another type of network entity which could affect routing security. To do so, we identify links which pass through IXPs using data from the IXP Mapping Project [4] and annotate them as traversing an IXP.

Similar to our performance focused network map, using a routing security focused map for evaluations of a network overlay requires adding additional information specific to the

overlay in question. We describe some of these elements in Chapter 5, including how we determine the location of Tor clients and destinations within the graph.

These two methods describe the general procedures for building foundational maps which can be used for evaluating performance and routing security. Maps of this type can provide a structure on which researchers can build targeted maps to evaluate questions about specific overlay networks. In drawing from actual internet data to construct these foundations, we enable researchers to evaluate overlays systems as if they were deployed at scale. The next two sections present evaluations of performance and routing security in the Tor network that we performed using our network maps as a foundation.

CHAPTER 4

EVALUATING RELAY SELECTION IN TOR

Tor is the third-generation onion routing network and provides anonymous communication to TCP-based applications [18]. Tor clients select a source-routed *circuit* of precisely three Tor *routers* (sometimes called relays) by querying any one of several authoritative *directories*. After constructing a circuit, clients forward traffic through their circuits using a layered encryption scheme based on onion routing [29]. Upon receipt of a fixed-size unit of transmission (a *cell*), each router along a circuit adds or removes a layer of encryption, depending on the cell's direction.

While early onion routing systems initially specified that clients should select routers uniformly at random [68], it became necessary to attempt to balance Tor's traffic load over the available router bandwidth as the anonymity network's popularity increased. Tor performs load balancing by weighting router selection in proportion to each router's perceived bandwidth capacity. Tor supports the use of trusted Bandwidth Authorities which are responsible for actively probing the Tor routers and estimating each router's capacity [57] to augment each routers self-reported capacity. Additional constraints are placed on router selection, including the use of entry guards [55] for the first hop to defend against the predecessor attack [77] and exit policies that specify the destination addresses and ports allowed by an exit router's operator. Recent work has also suggested selection algorithms that incorporate users' trust over various parts of the network [36].

Despite Tor's popularity with several hundreds of thousands of daily Tor users [30], one of the primary roadblocks to wide-scale Tor adoption continues to be its poor performance.

Prior work [17] has examined a number of factors that contribute to Tor's performance problems, including undesirable inter-circuit interference due to TCP's congestion control [60], suboptimal flow control at the application layer [3], and imperfect load balancing which causes lower bandwidth routers to handle too much traffic.

Tor exhibits high latencies partly due to the manner in which clients select relays for their anonymous *circuits* (a path of three Tor relays, selected in proportion to their bandwidth). For example, a large fraction of Tor's volunteer-operated relays are located in the United States or Germany [47, 71], requiring a typical client's traffic to make at least one transoceanic trip.

Existing work has proposed methods of creating lower latency anonymous circuits by carefully selecting relays to reduce either link latencies [62, 63] or the geographic distance covered by anonymous paths [2]. Other work has proposed that Tor clients be given the ability to *tune* the selection of relays in a manner that allows clients to achieve greater performance (by weighting more heavily toward high-bandwidth routers) or greater anonymity (by weighting selection more uniformly at random) [64]. Follow-up work by Murdoch and Watson have found that Tor's default router selection algorithm offers a good trade-off between performance and anonymity [50]. Still additional work has attempted to decrease latency by avoiding circuits that have high levels of congestion [76].

Despite this growing body of research on path selection techniques for Tor, none of the existing proposals have been evaluated under conditions that accurately reflect those that would be found on a live anonymity network. To illustrate, studies [33, 50, 64, 69] have shown that performance gains achieved under modeling and simulation are not present when the system is tested under more realistic conditions. Although a particular algorithm may show advantageous effects – even in the live network – when adopted by a small number of clients and/or relays, the technique may have unexpected negative consequences on the network when adopted en masse.

There is thus a need to move beyond the consideration of solely local effects ("*if I adopt this algorithm, will it improve my performance and anonymity?*") and consider potential impacts on the network in toto ("*what are the effects if a large number of clients/relays adopts this strategy?*"). In this case study, we use the performance-focused network map described in Section 3.1 to enable more comprehensive performance and anonymity analyses.

4.1 Existing Tor Models

A large volume of existing work attempts to approximate the live network's behavior, often as a means to evaluate a refinement to the network's protocols or configuration. At a high-level, efforts at modeling Tor can roughly be organized into three categories: *analytic*, *simulation*, and *emulation*.

Analytic methods [6, 50] allow researchers to evaluate refinements to Tor's protocols. However, accurately and analytically modeling complex network effects (e.g., congestion, jitter, etc.) remains an open problem. Indeed, existing analytic approaches often ignore network effects entirely.

There are also a number of available Tor simulators [34, 50, 52, 53]. As with the analytic methods, existing simulators fail to fully capture Tor's complexities. (To highlight this complexity, we note that recent versions of Tor have more than 200 configuration options.) In general, it is difficult to assess how well simulated behavior predicts the behavior of Tor in an actual deployment.

More recently, Moore *et al.* [48] and AlSabah *et al.* [3] use the ExperimenTor emulator [5] to respectively examine alternative rate limiting and congestion/flow control policies for Tor. However, their topologies do not model the live Tor network's latency, geography, or AS distributions. Similarly, Jansen and Hopper [33] introduce the Shadow frame-

work for executing (slightly modified) Tor code on a synthetic network. They sample the live network's bandwidth distribution, but configure geographic locations and latencies only according to a sample of several PlanetLab nodes.

To the best of our knowledge, ours is the first work that attempts to model the distributions of latencies, bandwidths, relay types, AS assignments, and geographies found on the live Tor network. We make use of the performance focused network map described in Section 3.1 under simulation (for scalability) and emulation (for realism).

4.2 Modeling

The foundation of the analysis performed in this case study is the performance-focused network map described in Section 3.1, using Tor relays, clients, and destinations as the PoIs in the graph.

4.2.1 Configuring the Model

We attach these endpoints to the graph as follows:

- *Tor relays.* We identify the Tor relays on the live Tor network whose IP addresses are in the same /24 network as some PoP in our model. We add the matching relays to our model, and mark the corresponding PoPs as a PoI.

- *Clients and destinations.* Prior research [21] has identified popular Tor client and destination ASs. We add clients and destinations to our model at the PoPs that belong to the popular client and destination (resp.) ASs and mark their PoPs as PoIs. (Note that based on our grouping heuristic, a PoP belongs to exactly one AS.)

We are able to effectively model 1524 relays in our network map, which constitutes a large proportion of the Tor network. While we were unable to model the full Tor network

(since we lacked the necessary `traceroute` information), it is worth noting that the 1524 Tor relays in our graph handle 71.3% of all traffic on the live network. We apply the remainder of the algorithm described in Section 3.1 as usual, resulting in a large network network focused on Tor. To set up our evaluation, we additionally configure several Tor specific elements.

To generate a scaled-down topology that is faithful to the bandwidth distribution of the live Tor network, we sample router bandwidths from the live Tor network as follows. We first take a list of all routers in a current Tor consensus and sort the list by the routers' observed bandwidths, as reported in each router's descriptor. We sample routers uniformly from this sorted list to precisely select the desired number of routers.[1]

Since Tor allows router operators to configure rate limits using a token bucket rate-limiting mechanism, we also sample each router's rate-limiting configurations (i.e., the `BandwidthRate` and `BandwidthBurst` options), which are also advertised within each router's descriptor.

Lastly, it is necessary to configure the directory authorities in the emulated network to advertise the correct bandwidth weights for each router. These weights ensure that clients select routers in the proper proportions. As described in Section 4.1 the live Tor network uses a set of Bandwidth Authorities to measure and compute these bandwidth weights. In our emulated network, we take a more simple approach: Each router is configured with an estimated bandwidth capacity according to its observed bandwidth value given in its live descriptor. The emulated directories then use these observed bandwidth values to compute a set of bandwidth weights to be used by clients for router selection in our subsequent experiments.

[1]While we note that this procedure allows us to approximate the bandwidth distribution of the live Tor network, it may slightly underestimate the bandwidths since the live network's measurements are affected by latencies (in addition to relays' actual bandwidths).

We assign unlimited bandwidths to the clients and server PoIs in our models so that they do not create bottlenecks. Although this may be slightly unrealistic, we note that except for very bandwidth-limited clients, performance bottlenecks occur in the Tor network itself, not at the sender or receiver. The "last-mile" latencies for servers and clients are assigned to be the median latency of the links within the PoP they are attached to, if available in our network map. If not, the latency is set to 10ms.

We run a single Tor client for each client PoI within our topology. Each Tor client uses different configuration options depending on the selection strategy being evaluated; however, there are a number of standard configuration options that we apply for our emulation experiments. We disable the use of entry guards, since the Guard designation relies on components such as uptime which are difficult to apply to short-term experiments. We also use the `MaxCircuitDirtiness` and `LearnCircuitBuildTimeout` parameters to increase the frequency with which new circuits are requested and to prevent historical data from being used to choose circuits.

Destinations are handled by a single server listening on all designated destination IP addresses.

We produce two models using the above techniques: one with 1524 relays and another with 100 relays (Figures 4.1 and 4.2, respectively). The 1524-relay topology contains every relay that could be mapped from the live Tor network. The 100-relay model was constructed by down-sampling from the 1524 model, while preserving the bandwidth profiles and relay type (i.e., guard, exit, etc.) distributions from the larger model. We use the larger model for simulations of circuit building events (Section 4.5), and the smaller model in our emulation environment (which requires a more manageable topology size). Our experimental emulation (Section 4.6) uses 50 of the 100 possible relays to increase the ratio of clients-to-relays and better approximate the performance offered by the live Tor network; we will refer to it as the 50-relay model.

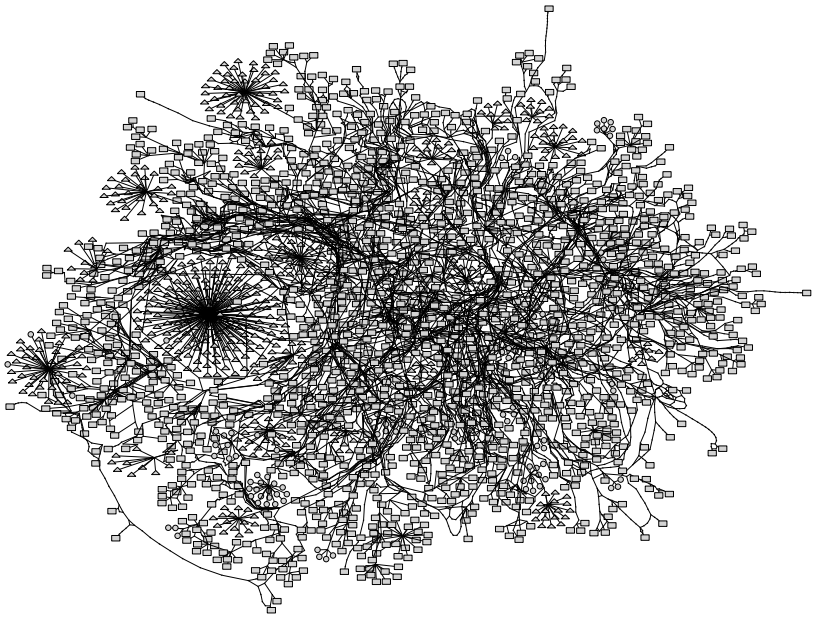

Figure 4.1: 1524-relay model of the Tor network.

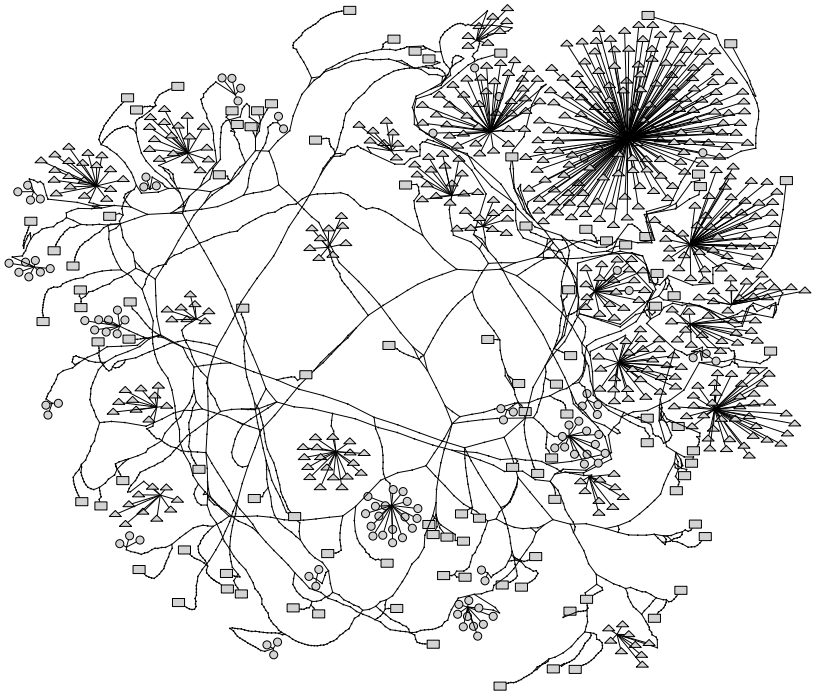

Figure 4.2: 100-relay model of the Tor network. 50 of the 100 relays are active during experimentation.

Network/Model	Relays	Non-Exit Guards	Exit Guards	Middle	Non-Guard Exits
Live Tor network	2642	579 [22%, 40%]	239 [9%, 31%]	1208 [46%, 18%]	616 [23%, 10%]
1524-relay model	1524	389 [26%, 43%]	154 [10%, 31%]	650 [43%, 18%]	332 [22%, 9%]
50-relay model	50	13 [26%, 64%]	4 [8%, 17%]	22 [44%, 15%]	11 [22%, 4%]

Figure 4.3: Distribution of relays in the live Tor network and our 1524- and 50-relay models, by count. The percentage of the network by count and the percentage of the network by bandwidth are respectively indicated in brackets.

4.2.2 Model Verification

We next verify our two models by demonstrating that they share several important characteristics with the live Tor network, including the bandwidth capacities, types, and geographic distribution of relays. We also discuss how our model of client behavior approximates real-worldTor client behavior.

Tor biases relay selection in part based on relay type (e.g., guard, exit, etc.). Since relay selection affects both the performance and anonymity properties of Tor circuits, to properly evaluate performance and anonymity, we desire models that reflect the same proportions of relay types as the live Tor network. Table 4.3 shows that our topologies reflect the numeric distribution of non-exit guards, exit guards, middle relay, and non-guard exits that occur on the live Tor network. We reasonably approximate the bandwidth handled by those classes of relays in our 1524-relay model, but see a modest shift in bandwidth capacity from Exit Guards to Non-Exit Guards in our 50-relay model.

Figures 4.4, 4.5, and 4.6 respectively show the global distribution of Tor relays for the full Tor network, our 1524-relay model, and our 50-relay model. We use the GeoIP [41] service to map relays to geographic locations based on their IP addresses. Our down-sampled set of 1524-relays maintains similar geographic characteristics to the full set of Tor relays. The 50-relay model used for emulation unavoidably loses some fidelity due to

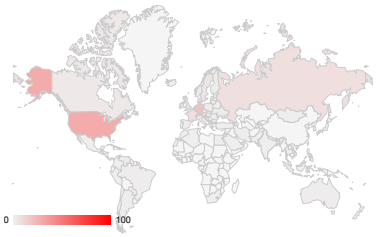

Figure 4.4: Geographic distribution of Tor relays in the full Tor network.

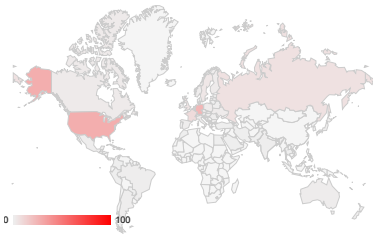

Figure 4.5: Geographic distribution of Tor relays in our 1524-relay topology.

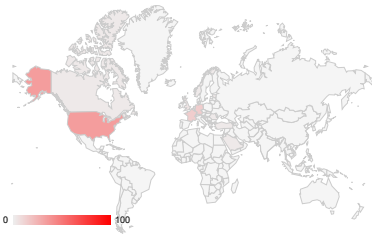

Figure 4.6: Geographic distribution of the 50 emulated relays.

down-sampling, but still retains a diverse geographic distribution that covers sixteen countries.

Figure 4.7 plots the cumulative distributions of bandwidth capacities for relays in the live Tor network as well as our 1524- and 50-relay models. Applying the two sample Kolmogorov-Smirnov test (a statistical measure for comparing the similarity between two empirical distributions), we find a Kolmogorov/Smirnov (K-S) statistic of 0.050 between the 1524-relay model and the live network, and a K-S statistic of 0.065 between the 50-relay model and the live network. This strongly indicates that the bandwidth distributions of our models closely match that of the live Tor network.

Tor's anonymity is affected by the network's AS topology [21]. An AS that exists both on the ingress path – between a client and the first relay – and the egress path – between the exit relay and the destination – can apply known timing attacks [40] to link the two segments and discover the identities of both the sender and receiver. To accurately assess the anonymity offered by various relay selection policies, our models should therefore exhibit AS distributions that closely match that of the live Tor network.

A histogram of AS memberships for the live network and our 1524-relay topology is shown in Figure 4.8. For ease of presentation, AS numbers have been replaced with indexes,

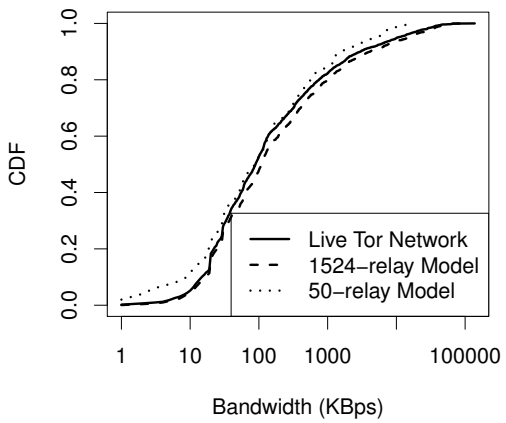

Figure 4.7: CDF of bandwidths as reported by the actual Tor network, and our 1524- and 50-relay models.

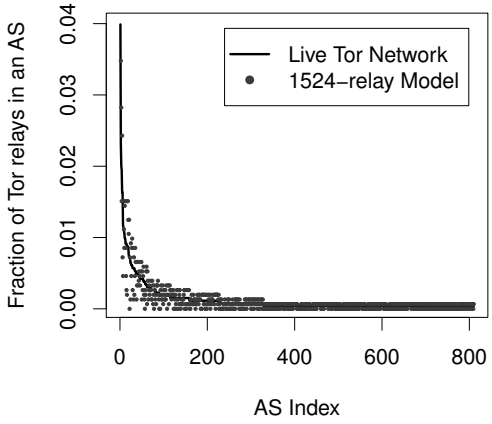

Figure 4.8: Histogram of relays' AS membership for the live Tor network and our 1524-relay model.

sorted by the count of constituent relays for the live Tor network. As can be seen from the figure, our model accurately reflects the live Tor network's distribution of ASs. Comparing against the live Tor network, the K-S statistics for the 1524-relay model is 0.046.

While the AS distribution in the 1524-relay model closely resembles that of the live Tor network, the 50-relay model is not particularly representative (here, the K-S statistic is 0.153). This "loss in fidelity" results from the small size of our 50-relay sample, relative to the number of relays on the live network. We discuss this limitation in more detail in Section 4.2.3. However, we note that our *security results* (in which we investigate how often an AS appears on both a circuit's ingress and egress segments) are based on simulations over the larger 1524-relay topology (Section 4.5). Our *performance* analyses, which are less dependent on AS topologies, are conducted using emulation over the 50-relay model (Section 4.6).

To reflect realistic workloads, we model two types of Tor clients. *Interactive* (also called *web*) clients repeat a fetch-sleep cycle where they access content for five minutes and sleep for up to one minute. While in the fetch stage of this cycle, clients request files (i.e., "web pages") between 100KB and 500KB in size, which approximates the average web page size (320KB) as reported by Google [66]. Between each fetch, clients wait for up to 11 seconds to simulate the behavior of someone browsing the web (i.e., they do not click links continuously, but pause to decide where to navigate next).

In contrast, *bulk* clients download continuously, and request files between 1MB and 5MB in size. Bulk clients roughly approximate the behavior of file sharers on the Tor network. To match existing studies [47] of behavior on the live Tor network, 3% of the clients are configured to be bulk; the remaining 97% are interactive.

To create workloads that capture the latency of Tor connections, each client additionally runs a low-bandwidth "echo" client that sends a single Tor cell once a second through the Tor network.

Our models also include *destination* nodes, which are the targets of anonymous communication. They serve HTTP requests and respond to "echo" messages.

4.2.3 LIMITATIONS

Our goal is to use our network map methodology as the basis for models that accurately represent the live Tor network's bandwidth, relay type, geographic, and AS distributions. Despite the verification measures described in the previous step, there is an inherent loss of fidelity due to down-sampling and the inability to perfectly represent client behavior. Our technique has some limitations, particularly wiht regard to client behavior and experiment scale. We discuss these limitations now:

By design, Tor makes it difficult to capture the behavior of the network's users. Existing studies [21, 47] of client behavior rely on sampled data from specially instrumented Tor

guard and exit relays that record usage statistics. We utilize the results of these studies to place clients and destinations in our network map. Unfortunately, these studies are becoming somewhat dated [21, 47]. We chose not to repeat the experiments described in the studies due to privacy concerns; as others have noted [65], recording client behavior on the live Tor network runs contrary to the anonymity network's principles and has the potential to put the network's users at risk. Although our datasets may not perfectly match current behavior, our placement of clients conforms to high-level statistics reported by the Tor Metrics Portal [71].

To maximize realism, we use the ExperimenTor [5] emulator in which unmodified Tor binaries communicate over a virtual network topology. However, the ability to scale our emulation is limited by our CPU and bandwidth capacities. Since we cannot emulate the hundreds of thousands of users who are estimated to use Tor [42], we instead opt to capture the level of congestion that occurs on the live Tor network. To do this, we adjust the number of Tor clients, and tune their behavior by changing how often they request pages to approximate the performance characteristics of the Tor network with a reduced number of clients (see Section 4.6).

4.3 Selection Algorithms

Using our network map, we evaluate the performance and anonymity of various relay selection strategies *under realistic network conditions*. In this section we enumerate existing and novel relay selection algorithms (Section 4.3), describe how we integrate the relay selection techniques into Tor (Section 4.3.1), and present metrics for measuring anonymity and performance (Section 4.4).

We consider the following relay selection algorithms:

- *Tor.* Conceptually, Tor's relay selection algorithm weighs relays proportionally according to their bandwidth [16, 18]. Murdoch and Watson have found that such a strategy offers good load balancing properties while providing reasonable anonymity [50]. In practice, however, Tor uses a slightly more complex weighting strategy that de-emphasizes unstable and/or new relays in favor of more longstanding routers. Additionally, Tor biases against selecting guard relays except as entry points, and exit relays except at egress locations.

- *Snader/Borisov.* Snader and Borisov [64] propose a refinement to Tor's algorithm that allows the sender to "tune" the degree to which selection is biased in favor of bandwidth. They introduce a family of functions

$$f_s(x) = \begin{cases} \frac{1-2^{sx}}{1-2^s} & \text{if } s \neq 0 \\ x & \text{if } s = 0 \end{cases}$$

  where s is a parameter that trades off between anonymity (selecting relays uniformly at random) and performance (biasing more heavily in favor of bandwidth). Given a list of relays sorted by their bandwidths, the Snader/Borisov (SB) algorithm selects the relay at index $\lfloor n \cdot f_s(x) \rfloor$, where x is chosen uniformly at random from $[0, 1)$ and n is the number of relays. In the remainder of this thesis, we denote the SB strategy with some fixed value of s as *SB-s*.

- *Unweighted Tor.* As a point of comparison, we include an *Unweighted Tor* selection strategy where clients build paths by choosing relays uniformly at random without replacement from the set of available relays provided by Tor. Clients using *Unweighted Tor* will only choose paths terminating at relays with accepting exit policies; they also are subject to any other constraints imposed by Tor.

- *Coordinate.* Sherr *et al.* [62, 63] propose latency-aware *link-based* relay selection strategies. In their approach, relays participate in a virtual coordinate embedding

system [13]. (To avoid potential anonymity attacks, neither clients nor destinations participate in the coordinate system.) The Euclidean distance between any two relays' virtual coordinates serves as an indicator of the latency between the pair. By summing the virtual distances between relays' advertised coordinates, clients can estimate the latencies of potential anonymous circuits *before they are instantiated*.

We implement two variants of coordinate-based routing. In the *Coordinates* strategy, clients select — but do not instantiate — k candidate paths where the relays in each path are selected using the *Unweighted Tor* methodology. Clients compute the expected latency of each of their k candidates paths, and select the path with the lowest estimated latency.

We also introduce a hybrid *Tor+Coordinates* strategy. Here, clients select k candidate paths *using Tor's default bandwidth-weighted relay selection strategy*. Clients then compute the expected latencies of the k candidate paths and instantiate the path with the lowest expected latency.

An evaluation of several potential values showed that setting $k = 3$ offered the best trade-off between increased performance and the time spent identifying the best path.

- *LASTor.* The *LASTor* [2] system selects relays in a manner that (1) reduces the probability that an autonomous system will appear on both sides of the anonymous circuit and (2) reduces path latencies by using geographic distance as an estimate for latency. (*LASTor* uses the GeoIP service to map network addresses to physical locations.) All possible candidate paths between a client and a destination are weighted based on their great circle distance (i.e., the distance measured over a spherical representation of Earth), and a path is selected that seeks to minimize that weight. To make this computationally tractable, relays are clustered into gridsquares based on latitude and longitude, and paths are calculated through these gridsquares. In addition, *LASTor*

makes use of iPlane datasets [44, 45] to avoid selecting paths where there exists an AS at both ends that could correlate traffic across the anonymous path.

- *Congestion-aware selection.* This technique, proposed by Wang *et al.* [76], seeks to intelligently select Tor circuits with the lowest levels of congestion. Congestion measurements for a given circuit are obtained by opportunistically sampling roundtrip times across that circuit and subtracting the lowest recorded roundtrip time. Both circuit building events and application connections are used to measure circuits with little additional overhead.

  Based on these congestion measurements, Wang *et al.* propose two *immediate* and one *long-term* path selection techniques. We applied the two immediate techniques together but omit the long-term algorithm entirely, as it was found to have negligible impact by the paper's authors [76]. The immediate techniques are as follows: (a) when choosing a circuit to use, randomly choose three of the available circuits, then select the one with the lowest measured congestion time, and (b) if at any point, the mean of the last five measured congestion times on a given circuit is more than 0.5 seconds, switch to another circuit.

4.3.1 Integrating Selection Algorithms into Tor

We implement the described selection algorithms within Tor version 0.2.3.0-alpha. For the *Coordinates* and *Tor+Coordinates* algorithms, we implement additional Tor cell types to support `ping` messages between Tor instances. Ping targets are selected uniformly at random from the list of running relays once every three seconds. A TLS connection is established with that target, and ping requests and responses are exchanged. The initiating relay uses the minimum ping response received to update its coordinate using the distributed Vivaldi algorithm [13]. We also modify Tor to include coordinate information

in relay descriptors, enabling clients to collect the necessary information to estimate the latencies of potential circuits.

Our implementations of the *LASTor* and *Congestion-aware* protocols use the Python TorCtl controller interface to select and instantiate paths according to the specifications outlined by Akhoondi *et al.* [2] and Wang *et al.* [76], respectively. For *LASTor*, we statically designate the latitude and longitude of our 50-emulated relays based on their real-world locations, obtained through IP-geolocation with the GeoIP City database, resulting in 38 geographic clusters. We do not implement the AS avoidance portion of *LASTor* since it relies upon iPlane Nano data [45] for BGP routing policies which may not map accurately to the routing on our experimental topology. Akhoondi *et al.* [2] showed that AS awareness *increased* the latency of selected paths, and hence we expect our *LASTor* performance results to be slightly optimistic.

Our *Congestion-aware* implementation obtains opportunistic measurements from three sources: the time taken to extend the circuit to the third relay; the time taken for application connection requests and acknowledgment; and a special `PINGED` cell sent once to measure the roundtrip time immediately after circuit construction.

4.4 METRICS

Our goal is to understand the implications of running the above relay selection strategies on the actual Tor network.

Our *performance* metrics are: **throughput**; **time-to-first-byte** (TTFB) (the time required for clients to fetch the first byte of a document); and **average ping time** (P-RTT) (the median roundtrip-time of sixty 100-byte pings).

Our *anonymity* metrics include: (1) the fraction of instantiated anonymous paths in which the same AS appears on both sides of the path, as proposed by Edman and

Syverson [21]; (2) the Shannon entropy over the distribution of relays in the entry and exit positions of paths [15, 61]; and (3) the Gini coefficient over those relays, as proposed by Snader and Borisov [64]. The Gini coefficient is a measure of equality (equality of selection probability, in this case) used frequently in economics. A Gini coefficient of 0 represents perfect selection equality (i.e., all routers are chosen with equal frequency), while a coefficient of 1 represents perfect inequality (i.e., only one router is always chosen). Note that because each path requires that a distinct entry and exit relay be selected, a Gini coefficient of 1 is effectively impossible to attain; the ceiling is 0.98 for our 50 relay emulation environment.

We compute Shannon entropy and the Gini coefficient over only the first and last relays in a given path. Tor instantiates paths containing three relays; the middle relay communicates only with the entry and exit relays using TLS encryption. An adversary observing the middle relay obtains little information of value, while one who observes both the entry and exit does not need to see the middle relay to break anonymity. Thus the distribution of entries and exits is most critical to anonymity. We desire relay selection strategies that produce high entropy and low Gini coefficients, as this prevents a subset of relays from observing a disproportionate amount of the network's traffic.

Using our models of the live Tor network, we next evaluate the anonymity (Section 4.5) and performance (Section 4.6) properties of proposed relay selection strategies.

4.5 Evaluation: Anonymity of Relay Selection

We simulate path selection on our 1524-relay model of the live Tor network. The simulator is based on actual Tor code (version 0.2.2.33) and uses Tor's relay selection functions. Our simulator implements only Tor's relay selection logic and does not simulate the actual

construction of paths, the transmission of data, or network effects such as congestion.[2] The performance of various relay selection policies, which is heavily dependent on network effects, is studied under full-network emulation in Section 4.6. Here, we focus our simulation experiments on measuring the AS diversity of paths as well as the distribution of selected relays.

We modify Tor's relay selection logic to support the *SB-s* and *LASTor* strategies. Since we do not simulate network conditions, we do not consider the *Coordinates*, *Tor+Coordinates*, or *Congestion-aware* strategies, each of whose behavior is dependent upon those conditions.

For each tested strategy, we simulate 5 million paths. Using the client and destination AS distributions reported by Edman and Syverson [21], we assign clients and destinations to the ASs for each path. We then check whether the same AS appears both on the path from the client to the guard relay as well as on the path from the exit relay to the destination.

If so, we weigh the result by the probability that this client and destination pair would be chosen (again, using Edman and Syverson's AS distribution). Effectively, this method yields the percentage of vulnerable paths, assuming clients and destinations are distributed as they were in Edman and Syverson's study. Our *LASTor* implementation does not include its AS avoidance strategy, resulting in a higher percentage of vulnerable paths than is likely to occur in a deployed implementation. The entropy and Gini coefficient results for *LASTor* are unaffected.

Table 4.1 shows the percentage of vulnerable paths for the various relay selection strategies, as well as the Gini coefficient and entropy over the distribution of selected relays. These metrics should not be taken as direct indicators of the strength of a particular anonymity technique, but rather as a mechanism for comparing the security properties

[2]Our simulator is based on the framework described by Elahi *et al.* [22].

Relay selection strategy	% vulnerable paths	Gini coef.	Entropy
Unweighted Tor	24.34	0.530	9.65
Tor	27.39	0.891	7.68
SB-3	24.66	0.662	9.32
SB-6	24.99	0.776	8.53
SB-9	26.01	0.841	7.58
SB-12	26.84	0.878	6.68
SB-15	27.42	0.900	5.95
LASTor	24.94	0.644	9.38

Table 4.1: Percentage of vulnerable paths, Shannon entropy and Gini coefficient for various relay selection strategies, under simulation using the 1524-relay model.

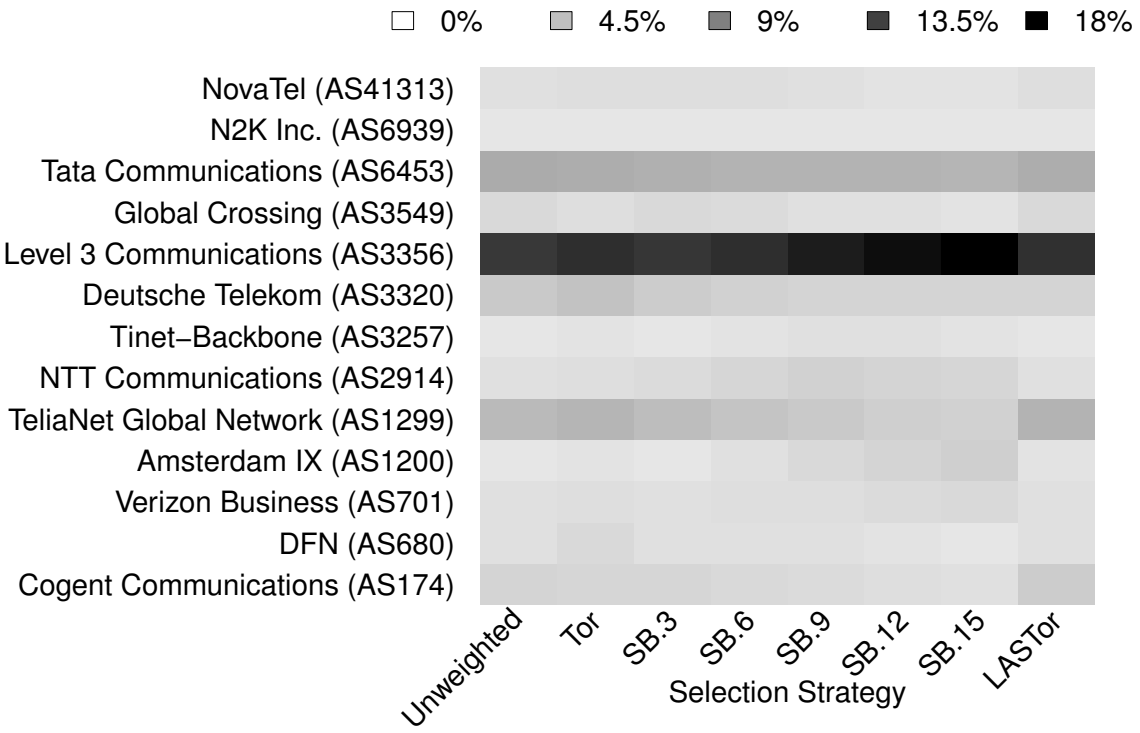

Figure 4.9: The ASs that most often appeared on both sides of anonymous paths, for each selection strategy. Shading indicates the percentage of paths on which the AS appeared.

of different strategies. The *Unweighted Tor* strategy offers the smallest percentage of vulnerable paths. This is unsurprising, since randomly selecting relays increases the diversity of paths. For the *Snader-Borisov* paths, increasing the value of s (i.e., biasing more heavily in favor of performance) increases the percentage of vulnerable paths, as the distribution of selected relays becomes less uniform. This effect is best captured by the increase in the Gini coefficient (indicating increasingly uneven distributions) as s increases.

Overall, in all cases, the relay selection strategy did not significantly increase the percentage of vulnerable paths. However, we note that the prevalence of a small number of ASs on both sides of the anonymous circuits cause approximately one quarter of the circuits to be vulnerable. This is slightly higher than the value reported in Edman and Syverson's study [21] (approximately 18% for Tor's default strategy). The potential increase in vulnerability may be due to topological changes on the Internet since their study was conducted in 2009. Additionally, we note that while our Internet model is based on empirical `traceroute` data, Edman and Syverson estimate AS paths using Qiu's inference algorithm [59] applied to RouteViews [73] data. In Figure 4.9, we report the ASs that most commonly occurred on either side of a path, and the rate at which that occurred for each selection strategy.

4.6 Evaluation: Performance Effects of Relay Selection

To measure the performance of relay selection, we run a modified version of Tor in an emulated network. We use ExperimenTor [5], executing all Tor instances on a 12-core 2.8 GHz Xeon X5660 machine with 64 GB of memory, running Ubuntu 11.10 with the 2.6.38 Linux kernel. ExperimenTor's ModelNet [74] virtual network backend runs inside of a FreeBSD 6.3 virtual machine connected to the host emulator with a direct 10 GbE link. Our experimental configuration uses the 50-relay model described in Section 4.2. We

carefully monitor the CPU, memory, and network utilization of both machines to ensure that resource constraints do not introduce any artifacts into our experiments.

We run each experiment for 2.5 hours to allow the system to stabilize and record results only from the last 90 minutes of each experiment. This allows the coordinates to stabilize for the *Coordinates* and *Tor+Coordinates* strategies, and the bandwidth weightings to properly adjust for *SB-s* and *Tor*.

As discussed in Section 4.2.3, establishing an appropriate level of traffic is difficult when modeling the Tor network. Our goal is to select a level of congestion that is on par with that found on the live Tor network.

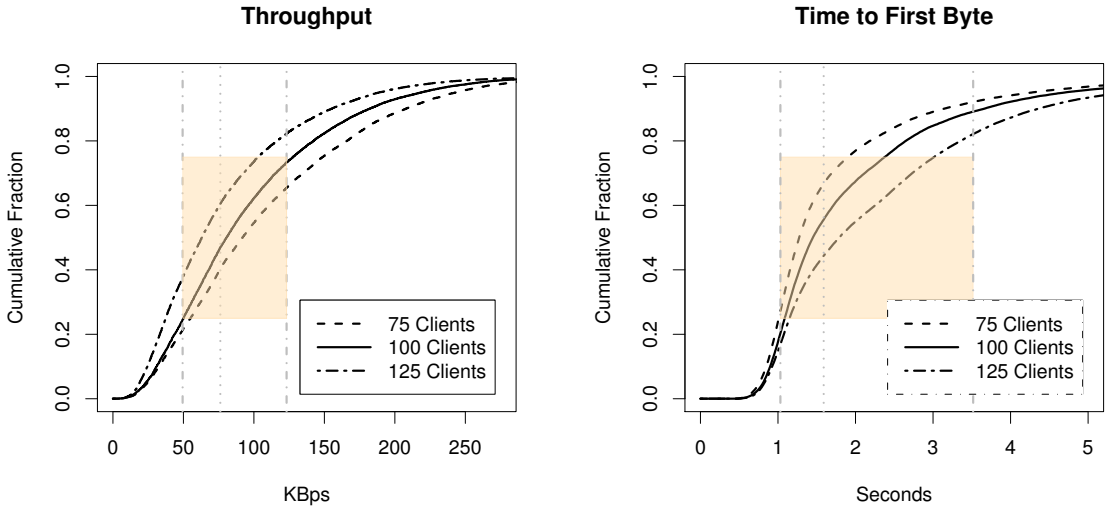

Figure 4.10: Cumulative distribution of throughput (left) and time-to-first-byte (right) of paths in our emulated environment with 75, 100, and 125 clients. The yellow highlighted boxes depict the interquartile ranges of performance of the live Tor network, as reported by the Tor Metrics Portal. To match the live network, the performance curve should intersect the lower-left and upper-right corners of the highlighted region, and intersect the dashed vertical line (the median of the live Tor network) when the cumulative fraction is 0.5.

We alter the number of clients to tune the level of congestion in our emulated Tor network. To select a level of congestion that matches Tor, we compare the throughput and time-to-first-byte experienced in our emulation to performance data collected from the live network in March 2012 [71]. As shown in Figure 4.10, with 100 active clients (three of which are *bulk* clients), our emulated throughput matches that of the Tor network almost exactly, while our time-to-first-byte is 32% faster at the third quartile, but only 6% faster at the median and 5% slower at the first quartile.

While our use of 100 active clients approximates the performance of the current live Tor network, we also evaluate relay selection strategies under both less and more severe congestion conditions. Specifically, we emulate 25 and 175 clients in the "low" and "high" congestion configurations, respectively, to evaluate performance for possible future conditions on the Tor network.

We first consider a homogeneous network in which all clients adopt identical relay selection strategies. Figure 4.11 shows the cumulative distribution of measured client throughput, time-to-first-byte, and P-RTT for a network experiencing a medium level of congestion. (For readability, we adopt the convention of listing the labels in the figures' keys in order of the corresponding curves' median values, while maintaining the same line types between figures.) There is a distinct difference in performance between the selection strategies that use bandwidth to influence their selection and those that do not. The *Tor+Coordinates* and *Congestion-aware* strategies achieve a median throughput of 85 KBps, outperforming all other selection strategies, although *Tor* (81 KBps) is only 4.7% worse. Strategies that apply either little or no weight to bandwidth perform poorly: *Unweighted Tor* nets a median throughput of just 22 KBps, while *LASTor* has a median throughput of 24 KBps.

The latency metrics follow a similar trend. Here *Congestion-aware* performs the best with a median time-to-first-byte of 1.321 seconds, 8.5% better than *Tor*, and 10.7% better

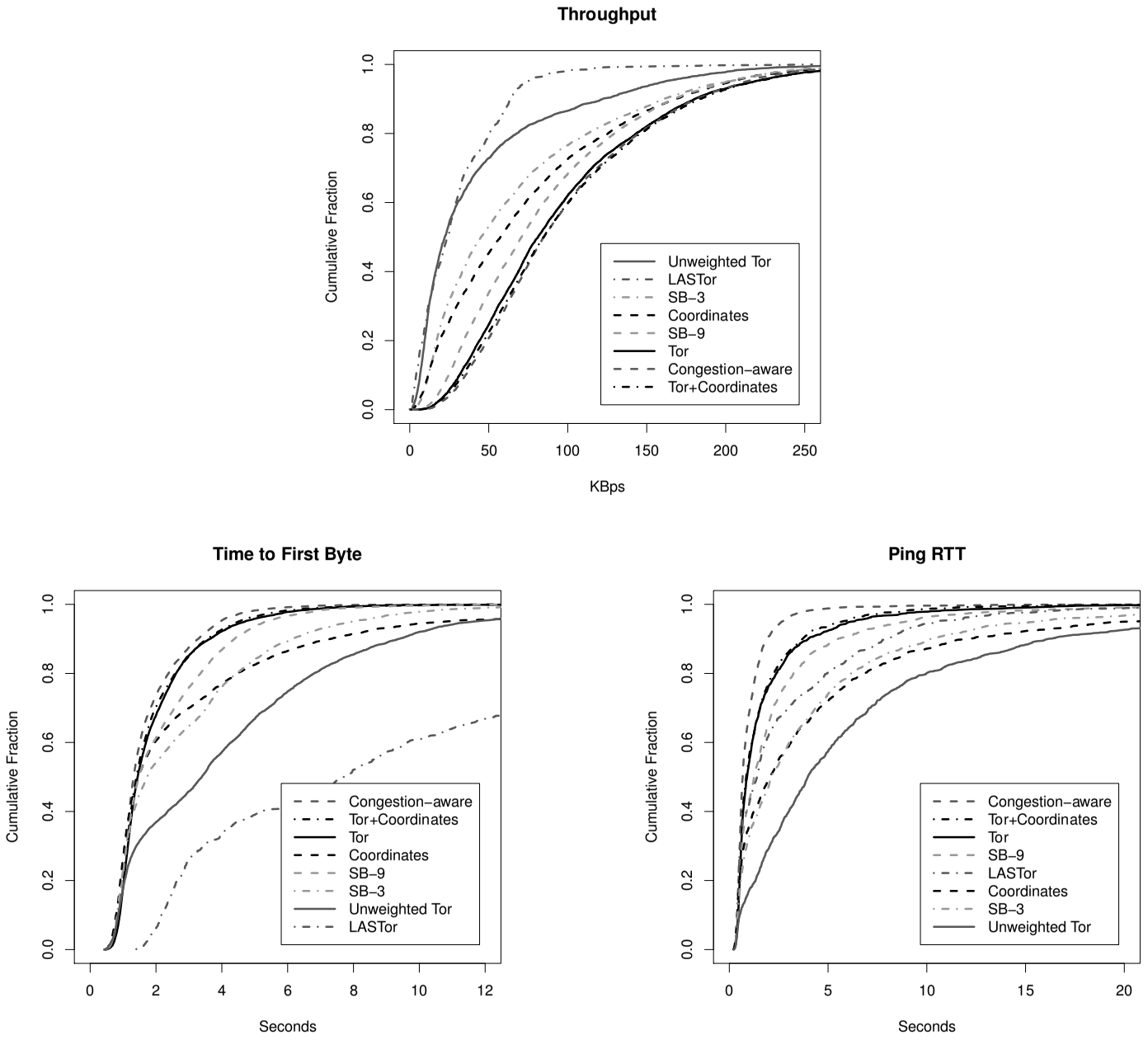

Figure 4.11: Cumulative distribution of measured client throughput (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with medium congestion.

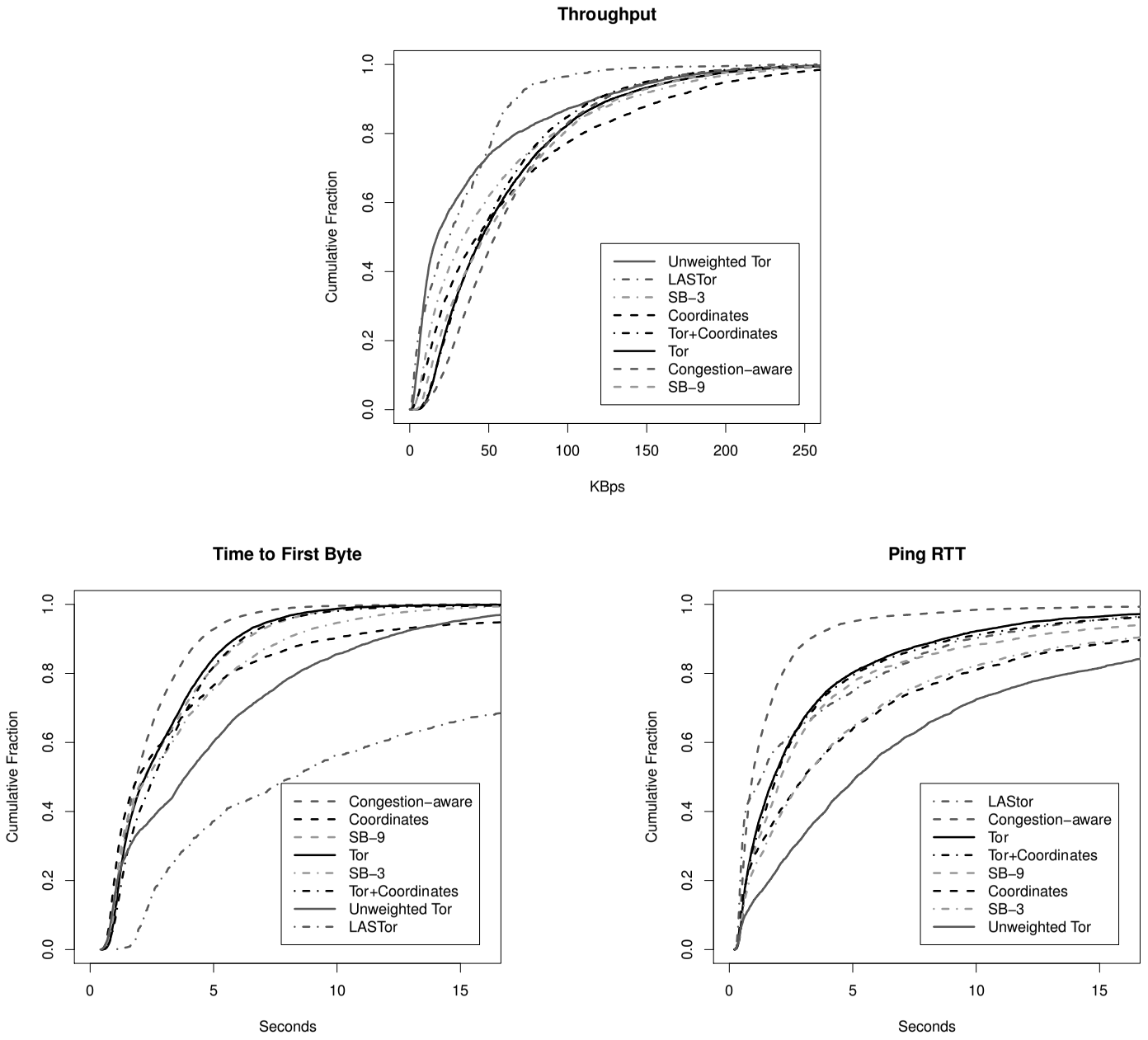

Figure 4.12: Cumulative distribution of measured client throughput (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with high congestion.

than *SB-9*. Notably, while *LASTor* remains poor at time-to-first-byte, it actually performs reasonably in P-RTT; its median P-RTT of 1.41 seconds is only 12% worse than *SB-9* and much better than that of *Unweighted Tor*, *Coordinates* and *SB-3*. This is unusual considering both metrics measure latency. We suggest that the difference is due to a distinction that strongly affects *LASTor*: time-to-first-byte incorporates TCP connect time, while P-RTT does not begin measuring until a connection is established. Since *LASTor* optimizes the latency across the full path between client and destination, the destination must be known *before a Tor circuit can be established*. Time-to-first-byte captures the time required for *LASTor* to pick and extend an appropriate path each time a new application request is received. By contrast, Tor normally maintains a set of established circuits and simply routes new traffic over one of them.

In Figures 4.12 and 4.13, we explore the performance of the relay selection strategies in networks with resp. high and low congestion. In the highly congested environment, throughput and time-to-first-byte suffer across the board. *Tor*, *SB-9* and *Tor+Coordinates* perform similarly, with median throughput of 46 KBps, 47 KBps, and 44 KBps respectively. Somewhat surprisingly, *Coordinates* also does reasonably well, with a median throughput of 43 KBps. The other less bandwidth-focused strategies produce median throughput of less than 33 KBps. *Congestion-aware*, by virtue of its focus on avoiding congestion, performs the best by a considerable margin with a median throughput of 53.8 KBps.

The *Congestion-aware* strategy continues to be effective when performance is measured in time-to-first-byte. *Congestion-aware* outperforms all other strategies with a median time-to-first-byte of 1.87 seconds, 14%, 21%, and 27% faster than *SB-9*, *Tor*, and *Tor+Coordinates* respectively. Similarly, *LASTor* continues to have impressive P-RTT times (although not throughput or time-to-first-byte): under high congestion, its 1.31 seconds are the second lowest at the median, behind only *Congestion-aware*.

In a low-congestion environment (Figure 4.13), throughput is considerably higher, and time-to-first-byte lower, due to reduced traffic and congestion.

Bandwidth remains an important factor in path selection, with a clear delineation between strategies that weight heavily for bandwidth and those that do not. The former category is led by *SB-9* which exhibits 45% better throughput than *Tor*. With the lower level of congestion, our *Tor+Coordinates* selection strategy is also able to improve upon *Tor* by 22%. Similar to the results for medium and high congestion networks, strategies that do not focus heavily on bandwidth do not perform particularly well.

SB-9 and *Tor+Coordinates* have the lowest time-to-first-byte, both with median times under one second. At their medians, they respectively perform 14% and 17% better than default *Tor*.

We see that even under low congestion, *Unweighted Tor* and *LASTor* do not appear to be effective at selecting paths through Tor, and experience the worst throughput and time-to-first-byte. It should be noted that *LASTor* again performs reasonably when performance is measured in terms of P-RTT. *Congestion-aware* continues to perform well but with less distinction, likely due to low levels of congestion.

We also explore the effects of relay selection when some clients elect to run a different relay selection strategy. This scenario is likely in incremental deployments, and additionally models *application-tunable anonymity* [63] in which clients select a relay selection policy to meet their underlying applications' communication requirements. When evaluating heterogeneous selection, 20% of clients use a specific selection method, while the remaining 80% use *Tor*. We present the median, 10th-, and 90th-percentiles of throughput, time-to-first-byte, and P-RTT for this heterogeneous selection under medium congestion in Table 4.2. We also show (1) the performance of the 80% of the clients that use the vanilla Tor client, and (2) the percentage improvement of the non-Tor strategy over *Tor*.

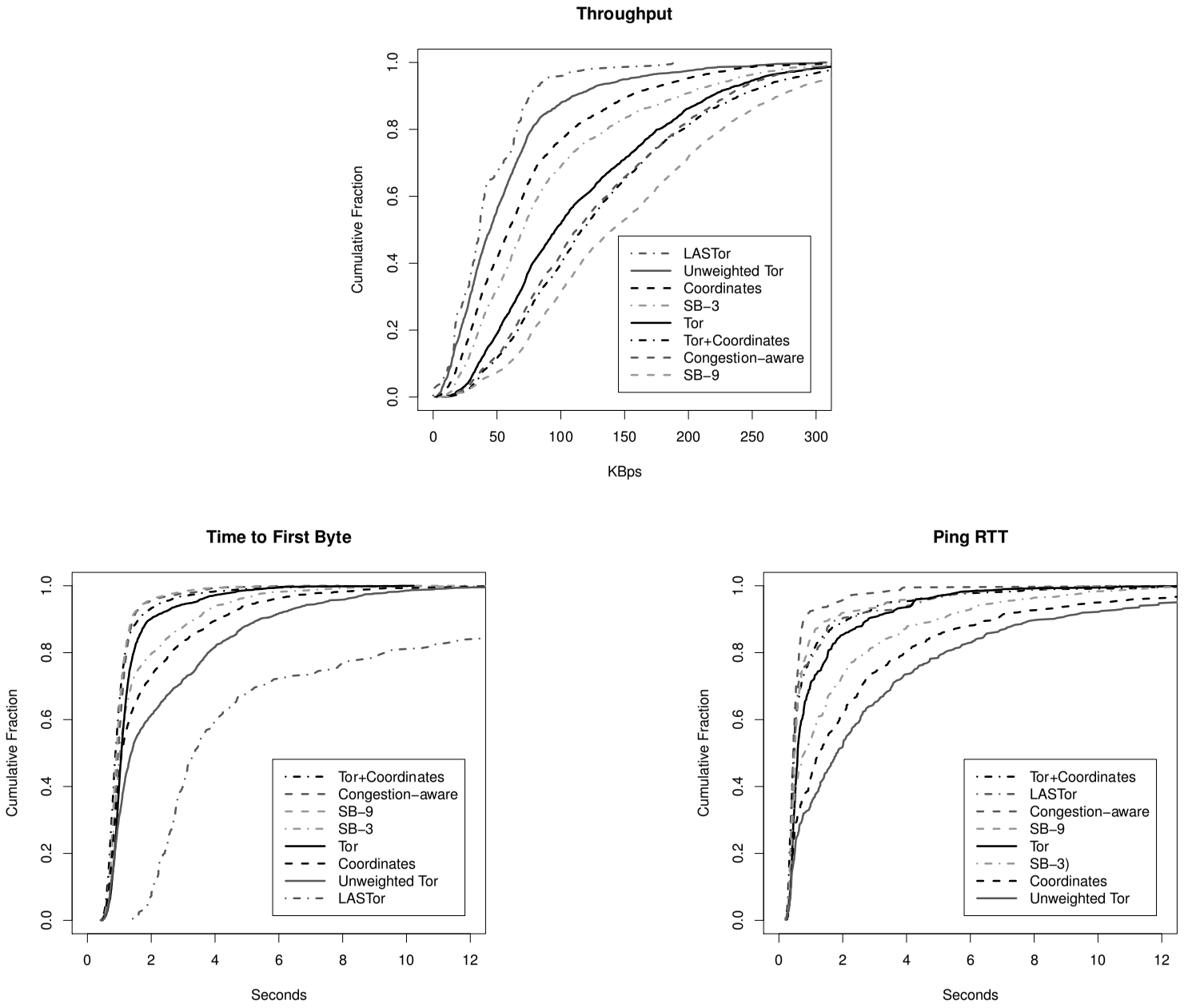

Figure 4.13: Cumulative distribution of measured client bandwidth (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with low congestion.

		throughput (KBps)		time-to-first-byte (seconds)		P-RTT (milliseconds)	
SB-9	*SB-9*	83.96 [30.19, 191.05]	**11.0%**	1.27 [0.83, 3.70]	**12.4%**	894 [401, 4152]	**9.7%**
	Default *Tor*	75.7 [29.77, 176.05]		1.45 [0.88, 3.95]		981 [427, 4423]	
SB-3	*SB-3*	46.66 [16.02, 133.93]	**-36.4%**	1.80 [0.87, 5.38]	**-33.3%**	1728 [436, 7821]	**-55.7%**
	Default *Tor*	73.34 [24.75, 177.20]		1.35 [0.86, 4.06]		1110 [422, 6063]	
Unweighted Tor	*Unweighted Tor*	29.21 [11.69, 75.43]	**-66.1%**	2.80 [0.95, 7.13]	**-101.4%**	2963 [536, 10235]	**-235.8%**
	Default *Tor*	86.13 [27.69, 197.43]		1.39 [0.89, 3.88]		885 [441, 4730]	
Tor+Coordinates	*Tor+Coordinates*	83.92 [32.84, 188.63]	**10.6%**	1.32 [0.87, 4.06]	**12.6%**	797 [396, 3655]	**17.4%**
	Default *Tor*	75.81 [29.80, 173.31]		1.51 [0.91, 3.84]		965 [436, 3973]	
Coordinates	*Coordinates*	61.53 [19.34, 148.37]	**-28.6%**	1.41 [0.85, 5.99]	**-2.9%**	1723 [422, 8051]	**-88.7%**
	Default *Tor*	86.15 [31.68, 191.12]		1.37 [0.89, 3.62]		913 [437, 4365]	
LASTor	*LASTor*	28.26 [8.11, 56.38]	**-70.0%**	6.78 [2.36, 63.91]	**-425.6%**	1409 [435, 6440]	**-82.3%**
	Default *Tor*	94.43 [35.20, 205.96]		1.29 [0.86, 3.03]		773 [418, 3840]	
Congestion-aware	*Congestion-aware*	86.38 [33.66, 186.10]	**9.5%**	1.31 [0.86, 3.11]	**10.3%**	648 [395, 1884]	**32.1%**
	Default *Tor*	78.86 [30.37, 176.99]		1.46 [0.90, 3.88]		954 [439, 6410]	
Homogeneous *Tor*	*Tor*	81.17 [31.51, 180.55]		1.44 [0.90, 3.64]		901 [447, 4009]	

Table 4.2: Performance metrics at the "median [10th percentile, 90th percentile]" for various relay selection strategies applied heterogeneously under medium congestion. Also shown in **bold** is the percentage improvement relative to default *Tor* at the median for each strategy and metric. Note that improvement means higher numbers for throughput, and lower numbers for time-to-first-byte and P-RTT.

SB-9, *Tor+Coordinates*, and *Congestion-aware* respectively provide performance improvements between 9% and 12% in throughput and time-to-first-byte, while other selection strategies generally under-perform compared to *Tor*.

Our results indicate that even in a heterogeneous environment, a selection strategy that does not use bandwidth weighting (e.g., *Coordinates*) performs poorly relative to the majority of clients who use *Tor*. Even small numbers of clients using specialized strategies must weight for bandwidth to obtain reasonable performance. Among those that consider bandwidth, some benefits can be observed. Congestion aware routing offers an improvement in anonymity and performance, especially under heavily congested conditions. *Tor+Coordinates* also shows potential for improvement over Tor's standard relay selection, posting modest benefits in throughput over Tor's default bandwidth-weighted strategy.

4.7 Discussion

One clear result of our performance evaluation is the critical importance of bandwidth to any effective relay selection strategy. The live Tor network is heavily oversubscribed and most network performance characteristics become irrelevant when bandwidth is the constraining factor. Our results show that strategies that weight heavily for bandwidth perform better than those that weight only lightly, and much better than those that do not do so at all. In particular, under our medium congestion level, the median throughput achieved by *Tor*, *Congestion-aware*, *SB-9* and *Tor+Coordinates* were all at least 70 KBps, while *Unweighted Tor* and *LASTor* — the two strategies that ignore bandwidth — both produced median throughput of less than 25 KBps.

One outlier is the *Coordinates* strategy, which often achieved throughput similar to *SB-3* without weighting on bandwidth. A likely explanation is that there is an inherent

correlation between bandwidth and latency *when empirically measured*. While the cost of pings in the coordinate system is not large, a low-bandwidth relay will be slower to respond than a high-bandwidth one. Additionally, there is an indirect relationship between latency and bandwidth when using TCP (as Tor does). Thus *Coordinates* actually incorporates an (admittedly loose) proxy for bandwidth in its selection.

Our results also indicate that the *LASTor* relay selection strategy is unlikely to be effective on the live Tor network. It is likely that the evaluation method used by Akhoondi *et al.* of making HTTP HEAD requests over *LASTor* paths did not realistically exercise the performance of those paths: a HEAD request is generally less than 1 KB, while typical web pages are two orders of magnitude larger [66]. Since *LASTor* does not take the bandwidth of relays into account, its performance degrades drastically when a realistic traffic load is applied. We note that our clean room implementation of *LASTor* does not include the AS awareness portion of the algorithm, although Akhoondi *et al.* showed that AS awareness increased the latency of instantiated paths so we believe it unlikely that its inclusion would have *improved* performance.

We did evaluate a variant of the bandwidth enhancement proposed by Akhoondi *et al.* [2]. They refine their basic *LASTor* strategy by restricting the relays chosen within each gridsquare to only those relays that have a reported bandwidth greater than 100 KBps. We implemented this refinement with one modification due to the potentially small number of relays per gridsquare in our emulation environment: we limit the set of paths to those in which a relay meeting the 100 KBps bandwidth restriction is guaranteed at each step.

Figure 4.14 shows the throughput and P-RTT of the modified version of *LASTor*, under medium congestion with homogeneous clients. The proposed refinement significantly increases the performance of *LASTor* (relative to the version without the bandwidth constraint), but still results in much worse performance than *Tor* and *Tor+Coordinates*.

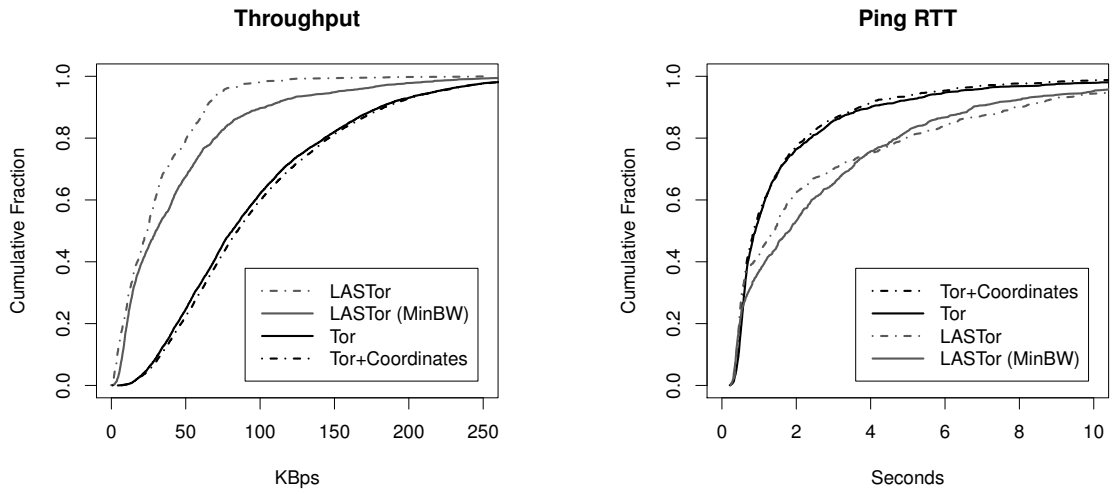

Figure 4.14: The throughput (top) and P-RTT (bottom) of *LASTor* with the proposed refinement shown alongside *Tor+Coordinates*, *Tor*, and the standard *LASTor*.

Finally, our results suggest that the addition of a coordinate-based system to Tor may provide some incremental advantage to relay selection. *Tor+Coordinates* obtained a slight performance improvement over *Tor* under most conditions, and rarely performed worse. Similarly the performance of the *Congestion-aware* strategy suggests that similar techniques provide a window of opportunity for incrementally improving Tor. While bandwidth remains the most significant indicator of performance, a layered approach which seeks to optimize latency and congestion as well as throughput might be a beneficial addition to Tor.

This sort of evaluation is not possible in the live Tor network – convincing each of the clients using the anonymity system to run separate relay selection algorithms while measurements were gathered is unrealistic. By using a performance-focused network map, we are able to shed light on which of the proposed ideas for relay selection in Tor are both feasible and beneficial.

CHAPTER 5

EVALUATING THREATS POSED BY NETWORK ADVERSARIES IN TOR

Tor is a volunteer-operated anonymity network that is estimated to protect the privacy of hundreds of thousands of daily users [18, 30]. However, Tor is known to be insecure against an adversary that can observe a user's traffic entering and exiting the anonymity network. Quite simple and efficient techniques can correlate traffic at these separate locations by taking advantage of identifying traffic patterns [49]. As a result, the user and his destination may be identified, completely subverting the protocol's security goals.

The traffic correlation problem in Tor has seen much attention in the literature. Prior Tor security analyses often consider entropy or similar statistical measures as metrics of the security provided by a system at a *static point in time*. In addition, while prior metrics of security may provide useful information about *overall* usage, they typically do not tell users how secure a *type of behavior* is. Further, similar previous work has thus far only considered adversaries that control either a subset of the members of the Tor network, a single AS, or a single IXP. These analyses have missed important characteristics of the network, such as that a single organization often controls several geographically diverse ASs or IXPs. That organization may have malicious intent or undergo coercion, threatening users of all network components under its control.

Given the severity of the traffic correlation problem and its security implications, we develop an analysis framework for evaluating the security of various user behaviors on the live Tor network and show how to concretely apply this framework by performing a comprehensive evaluation of the security of the Tor network [72] against the threat of

complete deanonymization. To enable such an analysis, we leverage a security focused network map as described in Section 3.2, resulting in (*i*) the largest and most accurate system for AS path inference yet applied to Tor and (*ii*) a thorough analysis of the threat of Internet exchange points and IXP coalitions. We also develop realistic metrics that inform this analysis, considering the network topology as it *evolves over time*, for example, as new relays are introduced and others go offline.

After describing background and related work, we next set out our adversary model and security metrics. We then describe our user models and the use of Monte Carlo simulation to sample how user traffic flows over the network, using the Tor Path Simulator (*TorPS*) [37] to generate paths. We describe the result of using our routing security network map to evaluate the security of circuits created via TorPS against a network adversary.

5.1 Vulnerabilities in Onion Routing

Onion routing is vulnerable to an adversary who can monitor a user's traffic as it enters and leaves the anonymity network; correlating that traffic using traffic analysis reveals the sender and receiver of the communication. Øverlier and Syverson first demonstrated the practicality of the attack in the context of discovering Tor Hidden Servers [55]. Later work by Murdoch and Danezis show that traffic correlation attacks can be done quite efficiently against Tor [49].

Given the potential severity of traffic correlation attacks, this paper explores in depth users' vulnerability to such attacks in the live Tor network. To quantify the anonymity offered by Tor, we examine path compromise rates and propose new time-based metrics for quantifying how *quickly* extended use of the anonymity network results in compromised paths.

Feamster and Dingledine first investigate the ability of AS-level adversaries to observe both sides of anonymous paths [23]. They argue that geographically diverse paths may adversely affect anonymity since paths that traverse many ASs are more likely than shorter paths to have the same AS on both sides of the path. Edman and Syverson also explore AS path diversity on Tor and introduce an AS-aware path selection algorithm that uses "snapshots" of Tor's AS graph to avoid AS-level traffic correlation attacks [21]. More recently, Akhoondi *et al.* propose a geographic-based relay selection method called LASTor [2] that ensures AS diversity in selected paths by relying on concise Internet atlases. The previous case study indicates that the same AS may appear in both sides of as many as 18% of anonymous circuits [75]. We seek to provide a more rigorous and useful interpretation of how this impacts Tor user security.

Murdoch and Zieliński argue that ensuring AS diversity in anonymous circuits is insufficient to safeguard against traffic correlation attacks by network adversaries, since traffic is routed between ASs at IXPs (and hence a single IXP may observe traffic traversing multiple ASes) [51]. They apply a Bayesian approach to show that an adversary positioned at an IXP could sample traffic from multiple ASs and correlate flows. Juen proposes a refined relay selection algorithm that provides both AS and IXP diversity [56]. We remark that Tor does not currently implement any protection against adversaries who operate ASs or IXP.

By considering how often *any* AS appears on both sides of circuits, these works implicitly assume that *all* ASs are malicious but are non-colluding. We also examine Tor's vulnerability to network adversaries, but improve upon existing work by modeling a more realistic adversary who monitors a fixed set of ASs or IXP.

5.2 MODELING

Unlike the traditional adversary considered by Tor - someone who runs a relay - a network adversary does not run relays in the hope that a client will choose one of those malicious relays at the guard and exit positions in its path. Instead, a network adversary leverages their position as a carrier of network traffic to correlate Tor traffic streams that cross their network *at some point* between the client and guard and exit and destination pairs.

We construct a network map focused on routing security to enable evaluation of the threat posed by network adversaries using the methodology described in Section 3.2. In constructing the map we draw the basic structure from CAIDA AS Links data from December 202 and BGP routes advertised to seven distributed RouteViews routers in March 2013. To construct the relationship data, we combine CAIDA's AS relationship dataset, the canonical relationship inference algorithm proposed by Gao [26], and the RIPE WHOIS dataset as described in Section 3.2.

We begin with a description of the path simulator we use to produce Tor path results and the types of client behavior simualted. We then discuss how Tor clients are exposed to network adversaries, starting with by considering the placement of clients within the network. We then consider the threat posed to those clients from three varieties of network adversaries: ASs, IXPs, and organizations which administer multiple IXP.

5.2.1 PATH SIMULATOR

We use the TorPS path selection simulator, which uses historical network data to recreate the conditions under which clients operated in the past and then executes path selection algorithms over those conditions given user actions. TorPS includes a model of the Tor relays and their past states, a model of user behavior, and a model of the Tor client[1]. For

[1] TorPS is based on the code in Tor version 0.2.3.25.

each sample simulation, it takes streams produced by the user model and network states from the network model and uses them as input to the client model, which chooses circuits and assign streams to them.

TorPS uses data from Tor Metrics [71] to model the past states of the Tor network. Tor Metrics provides archives of network consensuses and server descriptors, which TorPS uses to determine relay status over time, including flags, exit policies, hibernation state, and other parameters. Relays that do not appear in a consensus or do not have a descriptor are taken to be inactive.

5.2.2 Client Behavior and Location

We consider three types of clients in our analysis of a network-level adversary: Typical, BitTorrent, and IRC. We develop these user models in a manner designed to maximize their similarity to real user behavior. Each model consists of a sequence of Tor streams and the times at which they occur. Streams here include DNS resolution requests in addition to TCP connections to specific destinations. We construct the models by using client applications on the live Tor network and tracing the behavior of our local Tor client. Each trace consists of 20 minutes of a prescribed activity. The three user models we evaluate are as follows:

- *Typical*. This model is designed to represent average Tor use. It uses four traces consisting of (*i*) Gmail / Google Chat, (*ii*) Google Calendar / Docs, (*iii*) Facebook, and (*iv*) web search activity. These traces are played every day during the desired period, with one session of (*i*) at 9 a.m., one session of (*ii*) at 12 p.m., one session of (*iii*) at 3 p.m., and two sequential sessions of (*iv*) starting at 6 p.m.

- *IRC*. This model represents the use of Tor for the repeated but exclusive purpose of IRC chat. It uses the trace of a single IRC session and plays the trace sequentially from 8 a.m. to 5 p.m., Monday through Friday, a total of 27 times each day.

- *BitTorrent.* This model represents using BitTorrent over Tor. It consists of activity during the download of a single file. The model replays the trace sequentially from 12 a.m. to 6 a.m. on Saturday and Sunday, totaling 18 replays each day.

For each user behavior, we use TorPS to conduct 50000 Monte Carlo simulations of three months of client activity spanning the period from January 2013 to March 2013. We use the output of these simulations to model multiple clients.

TorPS simulated output paths are client agnostic; Tor currently makes no changes to path selection behavior based on client attributes (doing so could unintentionally decrease the safety of its users). However, since the path between client and guard is required to analyze exposure to network-level adversaries, we must place the clients somewhere within the network map. We assign clients to the five most popular[2] client ASs (AS3320, AS3209, AS3269, AS13184, and AS6805) as identified by Edman and Syverson in 2009 [21], noting that similar techniques have been used recently by papers investigating Tor network security [75]. The five ASs include four from Germany and one from Italy. We then analyze the client-to-guard path five times for each sample stream from our Monte Carlo simulations, once for each of the client origins.

5.2.3 Network Adversaries

We consider three types of network adversaries: autonomous systems, Internet exchange points, and Internet exchange point organizations. A network connection often transits multiple ASs as it moves from source to destination; a network operator interested in deanonymizing Tor traffic need only have the traffic transit through its domain of control once on each side of the path.

For each simulated client stream we compute the ordered sequence of ASs which occur on the client-guard connection, $path_{guard}$ and the exit-destination connection, $path_{exit}$. The

[2]We exclude Chinese ASs since Tor has subsequently been blocked in China.

ASs which could potentially deanonymize traffic for a given stream are the intersection of $path_{guard}$ and $path_{exit}$.

IXPs represent points where autonomous systems interconnect; traffic between multiple ASs may flow through a single IXP. In this position, IXPs may have significant ability to deanonymize Tor users.

We build the set of IXPs that observe the entry side of a client stream, denoted I_{entry}, by checking to see if any pair of consecutive ASs in $path_{guard}$ is contained in the IXP peering dataset described in Section 2. We repeat the procedure with I_{exit} and $path_{exit}$ to identify IXPs which exist on the exit to destination connection. Those IXPs which could potentially deanonymize traffic for a given stream are the intersection of the sets I_{entry} and I_{exit}.

As an extension of our IXP analysis, we also consider the situation in which a single organization may control multiple IXPs. Manually comparing IXPs descriptions from the IXP Mapping Project and company websites for IXPs, we identify 19 IXP organizations which collectively administer 90 distinct IXPs. To identify the organizations which are able to compromise client streams we perform the same procedure as for individual IXPs, replacing IXP identifiers with organization identifiers where possible. We include IXPs for which we have no identified organization as standalone organizations.

5.3 Evaluation: Security against Network Adversaries

We begin our analysis by identifying a set of specific *adversarial entities* for each combination of client behavior and client origin. Previous work has often considered the ability of network adversaries to compromise Tor circuits independently, reporting that a large percentage of circuits can be deanonymized by *some* AS. While this is a useful metric for system operators who are concerned with the security of Tor in the aggregate, it is not credible to consider the set of all independent ASs as potential adversaries from the user

Adv. Type	ID	Description	Comp. %
AS	3356	Level 3 Communications	0.5%
AS	1299	TeliaNet Global	0.5%
AS	6939	Hurricane Electric	0.4%
IXP	286	DE-CIX Frankfurt	0.1%
IXP Org.	DE-CIX	DE-CIX	0.1%

Table 5.1: Identified Adversarial Entities for clients originating in AS3320 using BitTorrent. Comp. % gives the probability that that entity will compromise any given stream.

perspective. We identify distinct *adversarial entities* specific to each simulated user origin and behavior. By focusing our analysis on the ability of these entities to compromise user streams, we are able to produce security metrics which are more relevant to an end user of Tor.

To identify candidate entities, we aggregate all streams over all client samples originating from a given client location. We then compute the client-side and destination-side paths, and count the number of streams in which a given adversarial entity (AS, IXP or IXP organization) exists on both sides. We then select the entity which compromises the largest number of streams to understand the extent to which a strong adversary affects user security. Table 5.1 shows a sampling of the identified adversarial entities for BitTorrent users originating from AS3320.

Our simulation results show that there is significant variation in the ability of network adversaries to compromise Tor users depending on where the user is located, but that on the whole network adversaries present a significant potential threat. While we run experiments for all selected client origins (as described in Section 5.2), we display only the best and worst cases in our results for readability. We measure *best* and *worst* as the client origin with the smallest and largest area under the curve, respectively, in their CDF of time to compromise.

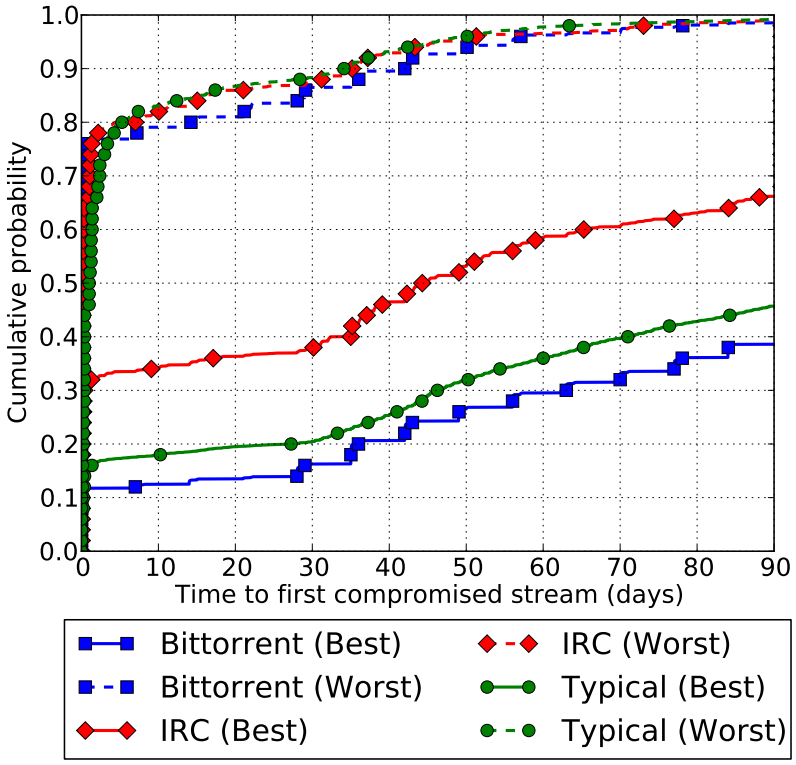

Figure 5.1: Time to first stream compromised by AS adversary. "best" and "worst" indicate the client origin from the top five ASs from [21] with the smallest and largest area under the curve, respectively. "*N* Adversaries" indicates an adversary that controls the top *N* AS entities.

Against an AS-level adversary (Figure 5.1), our results show compromise is highly likely in the worst case scenario regardless of user behavior. 45.9%, 64.9%, and 76.4% of Typical, IRC, and BitTorrent samples use a compromised stream within one day. At least one stream is compromised within the three month period for over 98% of samples. The best case client origins fare significantly better, but retain significant exposure to AS adversaries: IRC users are exposed within 44 days at the median. Although more than 50% of BitTorrent and Typical users evade compromise for the entire 90 day period, a significant proportion of them, 38% and 44% respectively, still use compromised streams.

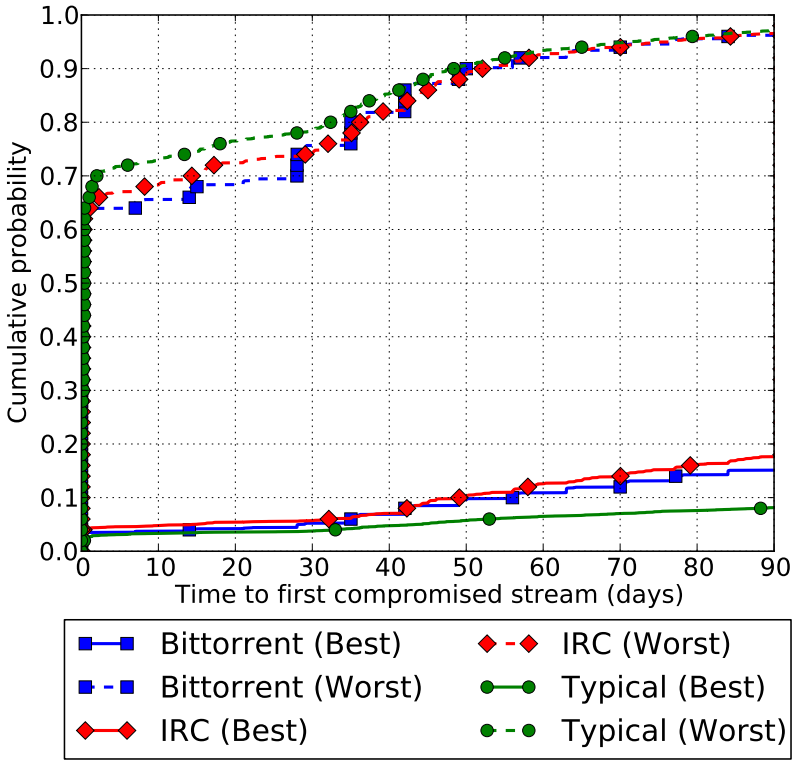

Figure 5.2: Time to first stream compromised by IXP adversary. "best" and "worst" indicate the client origin from the top five ASs from [21] with the smallest and largest area under the curve, respectively. "*N* Adversaries" indicates an adversary that controls the top *N* AS entities.

IXPs (Figure 5.2) and IXP organizations (Figure 5.3) appear similar to the AS adversary in the worst case, but significantly less of a threat in the best case. Fewer than 20% of clients use a stream that could be compromised within three months. This difference is not terribly surprising: while IXPs represent high-degree connection points through the network, 80% of the network links do not traverse IXPs. Thus, the worst case likely indicates a situation in which the client's outbound path to a guard transits through an IXP while the best case traverses non-IXP links.

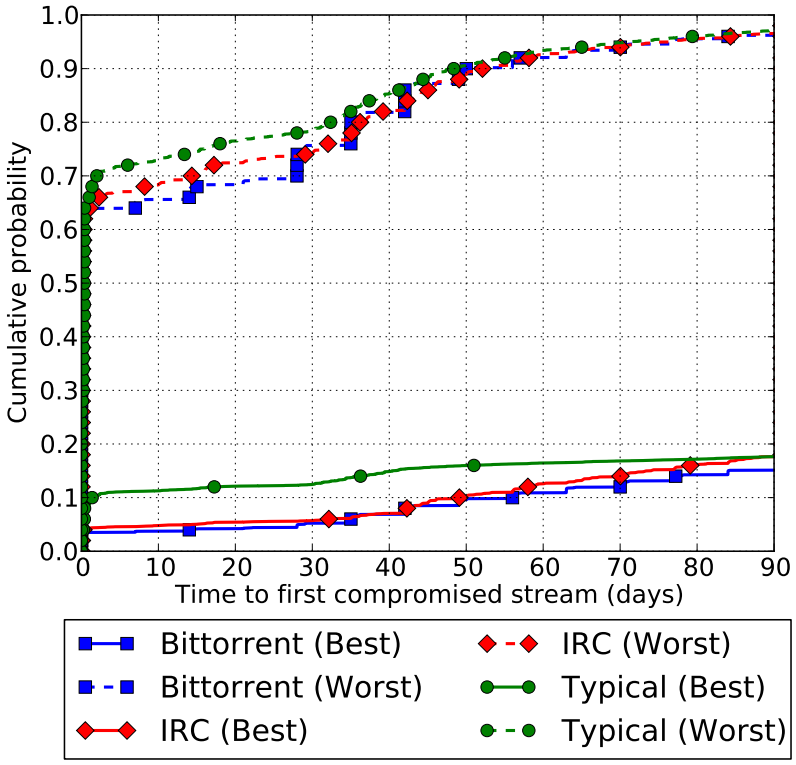

Figure 5.3: Time to first stream compromised by IXP Org. adversary. "best" and "worst" indicate the client origin from the top five ASs from [21] with the smallest and largest area under the curve, respectively. "*N* Adversaries" indicates an adversary that controls the top *N* AS entities.

While IXP and IXP organizations are generally similar, it is clear from the Typical user model that those concerned about the ability of IXPs to compromise Tor streams should consider organizations rather than individual IXP locations: in the best case standalone IXPs are able to compromise just 3.7% of samples within 30 days, while organizations compromise 12.4% in the same period.

We additionally consider how adversary strength affects the likelihood of stream compromise. We consider adversaries of varying strength by adjusting the number of *entities* they control from one to three. Thus, our weakest adversary controls the top *adversarial*

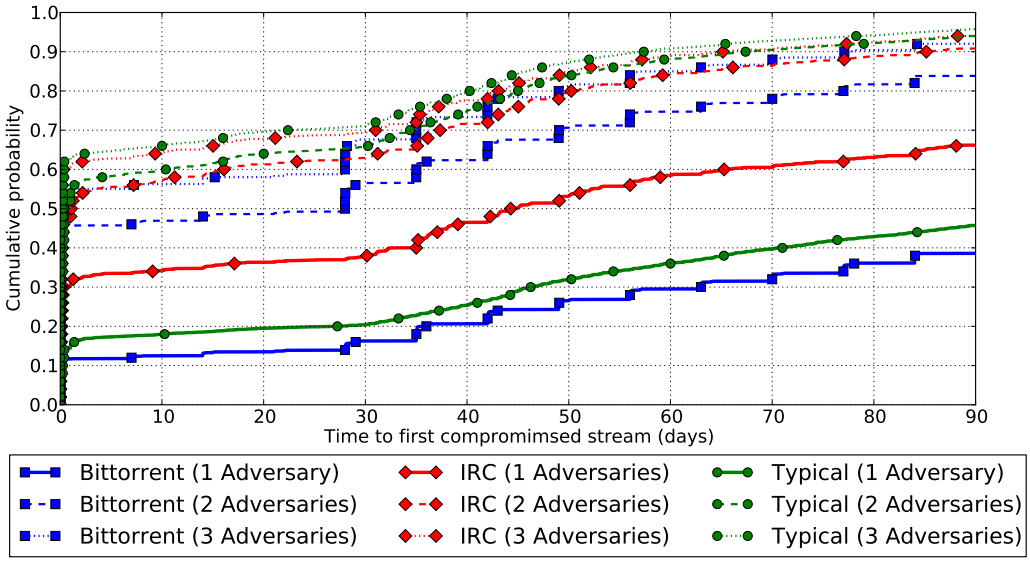

Figure 5.4: Varying time to first compromise as an AS adversary controls more entities. "best" and "worst" indicate the client origin from the top five ASs from [21] with the smallest and largest area under the curve, respectively. "*N* Adversaries" indicates an adversary that controls the top *N* AS entities.

entity of that type and the strongest controls the top three *adversarial entities*. Figure 5.4 shows how the time to first compromised stream drops as an adversary controls more of the top adversarial AS entities for each behavior model. Here we consider *only the best case* since just one AS entity is already able to compromise a significant fraction of samples in the worst case. The addition of even one more AS entity causes the number of samples compromised within 30 days to jump 156%, 65.8%, and 122% for BitTorrent, IRC, and Typical users. The amount of "ground" that two ASs can cover is significantly higher than the amount that one can cover.

In addition to the speed of compromise, we are equally interested in the probability that the adversary compromises any given stream. Figure 5.5 shows the fraction of streams that an AS adversary controlling the top entity compromises given a particular user activity. The

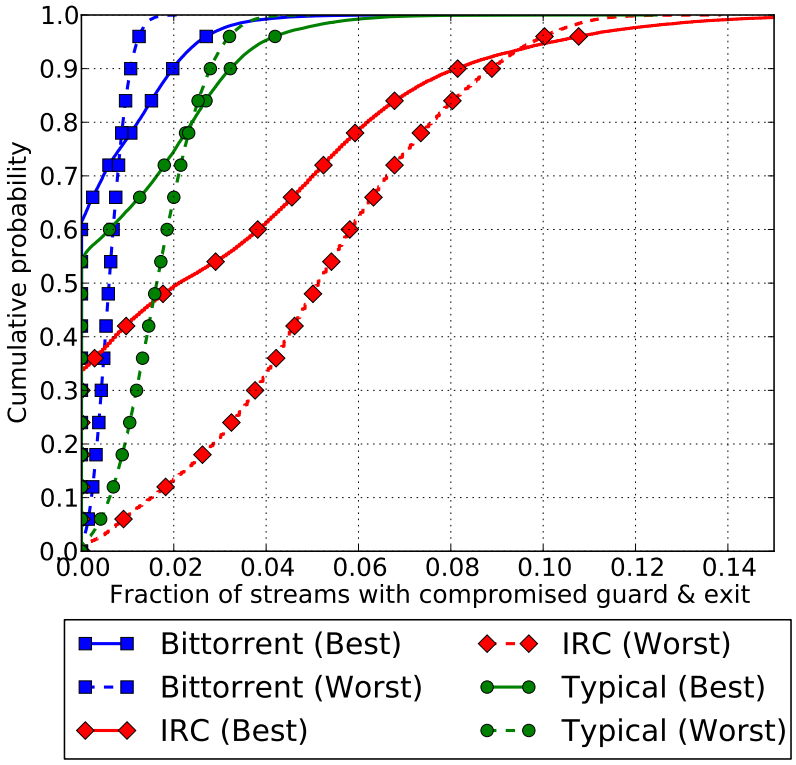

Figure 5.5: Fraction of streams compromised by AS adversary. "best" and "worst" indicate the client origin from the top five ASs from [21] with the smallest and largest area under the curve, respectively. "*N* Adversaries" indicates an adversary that controls the top *N* AS entities.

probability of compromising each stream is quite low even in the worst case: 0.6%, 5.1% and 1.6% at the median for BitTorrent, IRC and Typical users respectively.

5.4 Discussion

At a high level, the network adversary analysis shows that client behavior which results in low diversity of client destinations is most likely to result in a compromise. For example, a single AS adversary can compromise 50% of clients using IRC within 44 days, even under the most optimistic client placement. By contrast, fewer than half of Typical clients are

compromised within the entire period, and BitTorrent users have even lower compromise rates. This effect is attributable to simple probability: as the set of destinations which the adversary must cover narrows it becomes more likely that the user picked at the same time a destination and guard that the adversary is near.

It is also notable that a large number of clients encounter compromised streams very quickly followed by a steep decline in the rate. Figure 5.5 shows that the overall per-stream compromise rate is relatively low, so this phenomenon is somewhat surprising. Just as in the case of a relay adversary, this result is partly attributable to how guard selection interacts with relay selection. Given the initial set of guards, the path between the client and entry guard is relatively fixed. If the adversarial entity exists on the guard side of the path, then it need only wait until it appears on the exit side. However, if the entity does not exist on any of the paths from a client to it chosen guards, it will not compromise any streams until new guards are selected. This results in a relative plateau after initial compromises (disturbed only by minor guard as churn nodes leave the network or hibernate) until 30 days have passed and new guards begin to be selected.

Finally, while IXPs have a distinctly lower likelihood of compromising client traffic, it should be noted that the complexity of performing traffic correlation at an IXPs is likely to be significantly lower than at an AS. ASs may span large regions and traffic may not pass through the same routers on the forward and return path, while IXPs by their very nature are geographically concentrated. This may make it easier for a single rogue agent at an IXP to perform traffic analysis than it is to organize a concerted AS-wide effort. Tor users evaluating the ability of network adversaries to compromise their communications should consider this factor; IXPs may represent a lower overall threat profile, but have fewer obstacles to effecting a coordinated traffic analysis attack.

Some of our results against an adversary controlling multiple ASs or IXPs are similarly alarming. Some users experience over 95% chance of compromise within three months

against a single AS or IXP. We see that users' security varies significantly with their location. However, an adversary with additional ASs or IXPs has much higher compromise speed, notably against even those users in "safer" locations. Such an adversary is highly relevant in a world in which many large organizations control multiple ASs or IXPs. Surprisingly, we observe that high diversity in destinations may actually result in improved security against a specific network adversary.

CHAPTER 6

CONCLUSION

Evaluating the security and performance of applications which operate as a network overlay can be difficult for many reasons. Part of the difficulty is that while network overlays allow the application layer behavior to be more easily understood, the interaction with the underlying network may not be considered at all. That overlay networks are frequently distributed across the Internet and thus observation of nodes "in the wild" is hard only exacerbates the issue.

The difficult nature of distributed overlay system evaluation can manifest two effects – overlay network operators can be hesitant to modify their protocols for fear of adversely affecting users, or users simply end up bearing the brunt of unexpected interactions between deployed overlay networks and the network underlay.

This thesis suggests *network maps* as a method for alleviating this disconnect. Network maps combine information from many datasets into representations of the live Internet. These representations enable researchers and operators of overlay networks to evaluate systems in a "safe" simulation or emulation environment. This gives them the leeway to experiment and adapt their systems without assuming the risks of live experimentation.

Specifically, we contribute methodologies for building two types of network topologies; one focuses on performance aspects of the network while the other concentrates on representing accurate routing across the network. In two case studies focusing on Tor, we instantiate network topologies of each type, and present research about the performance and security of the Tor network.

Our first case study uses a performance-focused network map to show the comparative effectiveness and security of various relay selection strategies proposed in the literature. We highlight that Tor's embedded relay selection strategy is highly effective due to the presence of bandwidth as an overarching constraint on network performance. We also show that some orthogonal selection strategies may permit augmented performance when used *alongside* Tor's default strategy.

Our second case study investigates the threat posed to Tor users by network-level adversaries. These adversaries are represented by ISPs, IXPs, and other operators of network infrastructure. We use our routing topology, combined with route inference to suggest that Tor users may in fact be more vulnerable to network level adversaries than previously believed.

Both case studies show the advantage of using network maps of this type. By combining disparate data into representations of the Internet, we are able to holistically evaluate aspects of the network and determine where changes may be warranted. Importantly, the entire evaluation can be performed within the safe confines of an emulation or simulation environment without affecting the deployed network.

BIBLIOGRAPHY

[1] Jeff Ahrenholz, Claudiu Danilov, Thomas R Henderson, and Jae H Kim. Core: A real-time network emulator. In *IEEE Military Communications Conference (MILCOM)*, 2008.

[2] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE Symposium on Security and Privacy (Oakland)*, 2012.

[3] Mashael AlSabah, Kevin Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey Voelker. DefenestraTor: Throwing out Windows in Tor. In *Privacy Enhancing Technologies Symposium (PETS)*, 2011.

[4] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. IXPs: Mapped? In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2009.

[5] Kevin Bauer, Micah Sherr, Damon McCoy, and Dirk Grunwald. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2011.

[6] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[7] CAIDA. Archipelago Measurement Infrastructure. `http://www.caida.org/projects/ark`, 2012.

[8] CAIDA. IPv4 Routed /24 AS Links. `http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml`, 2012.

[9] CAIDA. IPv4 Routed /24 Topology Dataset. http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml, 2012.

[10] CAIDA. The CAIDA AS Relationships Dataset. http://www.caida.org/data/active/as-relationships/, 2012.

[11] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[12] Team Cymru. IP to ASN mapping. http://www.team-cymru.org/Services/ip-to-asn.html, 2008.

[13] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. *ACM SIGCOMM Computer Communication Review*, 34(4):15–26, 2004.

[14] Chris Davies. Skype supernode switch for stable scaling not Project Chess NSA spying. http://www.slashgear.com/skype-supernode-switch-for-stable-scaling-not-project-chess-nsa-spying-25287830/.

[15] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Privacy Enhancing Technologies Symposium (PETS)*, 2002.

[16] Roger Dingledine and Nick Mathewson. Tor Path Specification. http://www.torproject.org/svn/trunk/doc/spec/path-spec.txt, 2008.

[17] Roger Dingledine and Steven Murdoch. Performance Improvements on Tor, or, Why Tor is Slow and What We're Going to Do About It. https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf, 2009.

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, 2004.

[19] Matthew B Doar. A better model for generating test networks. In *Global Telecommunications Conference (GLOBECOM)*, 1996.

[20] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *22nd USENIX Security Symposium*, 2013.

[21] Matthew Edman and Paul Syverson. AS-Awareness in Tor Path Selection. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[22] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, October 2012.

[23] Nick Feamster and Roger Dingledine. Location Diversity in Anonymity Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2004.

[24] Sally Floyd and Vern Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(4):392–403, 2001.

[25] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking (TON)*, 9(5):525–540, 2001.

[26] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9:733–745, 2001.

[27] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)*, 9(6):681–692, 2001.

[28] Thomer Gil, Frans Kaashoek, Jinyang Li, Robert Morris, and Jeremy Stribling. p2psim, a simulator for peer-to-peer protocols, 2003. `http://pdos.csail.mit.edu/p2psim/`.

[29] David Goldschlag, Michael Reed, and Paul Syverson. Hiding Routing Information. In *Workshop on Information Hiding (IH)*, 1996.

[30] Sebastian Hahn and Karsten Loesing. Privacy-preserving Ways to Estimate the Number of Tor Users, November 2010. `https://metrics.torproject.org/papers/countingusers-2010-11-30.pdf`.

[31] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph B Kopena. Demonstration: Network simulations with the NS-3 simulator. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*.

[32] Rob Jansen. BRAIDS Tor Simulator. `http://code.google.com/p/braids-tor-simulator/`.

[33] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[34] Rob Jansen, Nicholas Hopper, and Yongdae Kim. Recruiting New Tor Relays with BRAIDS. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[35] Rob Jansen, Kevin Bauer, Nick Hopper, and Roger Dingledine. Methodically Modeling the Tor Network. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2012.

[36] Aaron Johnson, Paul Syverson, Roger Dingledine, and Nick Mathewson. Trust-based Anonymous Communication: Adversary Models and Routing Algorithms. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[37] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *ACM Conference*

on Computer and Communications Security (CCS), 2013.

[38] Srinivasan Keshav. Packet-pair flow control. *IEEE/ACM Transactions on Networking (TON)*, 1995.

[39] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *ACM Workshop on Hot Topics in Networks (HotNets)*, 2010.

[40] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. Timing Attacks in Low-latency Mix Systems. In *Financial Cryptography and Data Security (FC)*, 2004.

[41] MaxMind LLC. MaxMind GeoIP. `http://dev.maxmind.com/geoip/legacy/geolite`, 2008.

[42] Karsten Loesing. Measuring the Tor Network: Evaluation of Client Requests to the Directories. Technical report, Tor Project, 2009.

[43] Matthew Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 239–245. ACM, 2010.

[44] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2006.

[45] H.V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[46] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Peer-to-Peer Systems*, pages 53–65. Springer,

2002.

[47] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[48] Brad Moore, Chris Wacek, and Micah Sherr. Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race With Tortoise. In *Annual Computer Security Applications Conference (ACSAC)*, December 2011.

[49] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy (Oakland)*, 2005.

[50] Steven J. Murdoch and Robert N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[51] Steven J. Murdoch and Piotr Zieliński. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Privacy Enhancing Technologies Symposium (PETS)*, 2007.

[52] Tsuen-Wan "Johnny" Ngan, Roger Dingledine, and Dan Wallach. Building Incentives into Tor. In *Financial Cryptography and Data Security*, 2010.

[53] Gavin O'Gorman and Stephen Blott. Large Scale Simulation of Tor: Modelling a Global Passive Adversary. In *Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security (ASIAN)*, 2007.

[54] Ookla. Netindex. `http://www.netindex.com/source-data/`.

[55] Lasse Øverlier and Paul Syverson. Locating Hidden Servers. In *IEEE Symposium on Security and Privacy (Oakland)*, 2006.

[56] Joshua Paul and Joseph Juen. Protecting Anonymity in the Presence of Autonomous System and Internet Exchange Level Adversaries. Master's thesis, University of Illinois, 2012.

[57] Mike Perry. Torflow: Tor network analysis. Hot Topics in Privacy Enhancing Technologies (HotPETS), 2009.

[58] PlanetLab. PlanetLab. `http://www.planet-lab.org`, 2010.

[59] Jian Qiu and Lixin Gao. AS Path Inference by Exploiting Known AS Paths. In *Global Telecommunications Conference (GLOBECOM)*, 2006.

[60] Joel Reardon and Ian Goldberg. Improving Tor using a TCP-over-DTLS Tunnel. In *USENIX Security Symposium (USENIX)*, 2009.

[61] Andrei Serjantov and George Danezis. Towards an Information Theoretic Metric for Anonymity. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies Symposium (PETS)*, 2002.

[62] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable Link-Based Relay Selection for Anonymous Routing. In *Privacy Enhancing Technologies Symposium (PETS)*, August 2009.

[63] Micah Sherr, Andrew Mao, William R. Marczak, Wenchao Zhou, Boon Thau Loo, and Matt Blaze. A3: An Extensible Platform for Application-Aware Anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2010.

[64] Robin Snader and Nikita Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[65] Christopher Soghoian. Enforced Community Standards For Research on Users of the Tor Anonymity Network. In *Workshop on Ethics in Computer Security Research (WECSR)*, 2011.

[66] Sreeram Ramachandran. Web Metrics: Size and Number of Resources, May 2010. https://developers.google.com/speed/articles/web-metrics.

[67] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2001.

[68] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

[69] Can Tang and Ian Goldberg. An Improved Algorithm for Tor Circuit Scheduling. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[70] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topology generators: Degree-based vs. structural. In *SIGCOMM Computer Communication Review*, volume 32, pages 147–159, 2002.

[71] Tor Project, Inc. Tor Metrics Portal. https://metrics.torproject.org/, 2013.

[72] Tor Project, Inc. The Tor Project. https://www.torproject.org/, 2013.

[73] University of Oregon. RouteViews Project. http://www.routeviews.org/, 2013.

[74] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostić, Jeff Chase, and David Becker. Scalability and Accuracy in a Large-scale Network Emulator. *SIGOPS Operating Systems Review*, 36:271–284, December 2002.

[75] Chris Wacek, Henry Tan, Kevin Bauer, and Micah Sherr. An Empirical Evaluation of Relay Selection in Tor. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

[76] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. Congestion-aware Path Selection for Tor. In *Financial Cryptography and Data Security (FC)*, 2012.

[77] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004.

[78] Ellen W Zegura, Kenneth L Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *IEEE International Conference on Computer Communications (INFOCOM)*, 1996.

[79] Hubert Zimmermann. OSI reference model – the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communication*, 28(4):425–432, 1980.