Improving Security and Performance in Low Latency Anonymity Networks

by

Kevin Scott Bauer

B.S., University of Denver, 2005

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2011

This thesis entitled:
Improving Security and Performance in Low Latency Anonymity Networks
written by Kevin Scott Bauer
has been approved for the Department of Computer Science

Prof. Dirk Grunwald

Prof. Shivakant Mishra

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Bauer, Kevin Scott (Ph.D., Computer Science)

Improving Security and Performance in Low Latency Anonymity Networks

Thesis directed by Co-Chairs Prof. Dirk Grunwald and Prof. Douglas Sicker

Conventional wisdom dictates that the level of anonymity offered by low latency anonymity networks increases as the user base grows. However, the most significant obstacle to increased adoption of such systems is that their security and performance properties are perceived to be weak. In an effort to help foster adoption, this dissertation aims to better understand and improve security, anonymity, and performance in low latency anonymous communication systems.

To better understand the security and performance properties of a popular low latency anonymity network, we characterize Tor, focusing on its application protocol distribution, geopolitical client and router distributions, and performance. For instance, we observe that peer-to-peer file sharing protocols use an unfair portion of the network's scarce bandwidth. To reduce the congestion produced by bulk downloaders in networks such as Tor, we design, implement, and analyze an anonymizing network tailored specifically for the BitTorrent peer-to-peer file sharing protocol. We next analyze Tor's security and anonymity properties and empirically show that Tor is vulnerable to practical end-to-end traffic correlation attacks launched by relatively weak adversaries that inflate their bandwidth claims to attract traffic and thereby compromise key positions on clients' paths. We also explore the security and performance trade-offs that revolve around path length design decisions and we show that shorter paths offer performance benefits and provide increased resilience to certain attacks. Finally, we discover a source of performance degradation in Tor that results from poor congestion and flow control. To improve Tor's performance and grow its user base, we offer a fresh approach to congestion and flow control inspired by techniques from IP and ATM networks.

Dedication

To my parents.

Acknowledgements

This thesis is the culmination of over five years of work and I wish to thank the many people who have made invaluable contributions both to this thesis and to my professional development as a researcher. First, I thank my academic advisers, Dirk Grunwald and Doug Sicker, without whose encouragement and support this work would not have been possible. Second, I thank my thesis committee, Nikita Borisov, Shiv Mishra, and Stefan Savage, for their invaluable comments and suggestions.

I would also like to thank the many research collaborators and co-authors for their contributions to this thesis and to my professional development. I am particularly thankful to J. Trent Adams, Mark Allman, Mashael AlSabah, Eric Anderson, Aaron Beach, Markus Breitenbach, Nikita Borisov, Anders Drachen, Ian Goldberg, Harold Gonzales, Ben Greenstein, Greg Grudic, Dirk Grunwald, Asa Hardcastle, Joshua Juen, Yoshi Kohno, Hyunyoung Lee, Janne Lindqvist, Qin (Christine) Lv, Damon McCoy, Sears Merritt, Vern Paxson, Caleb Phillips, Stefan Savage, Micah Sherr, Doug Sicker, Robin Sommer, Parisa Tabriz, Rob Veitch, Geoff Voelker, and Gary Yee.

In addition, I am grateful to Nikita Borisov, Roger Dingledine, Paul Syverson, and countless anonymous reviewers for offering helpful and constructive comments and suggestions that improved the quality of various parts of this thesis. I also especially thank Roger Dingledine for his two visits to UCSD in August and December 2010, during which time he offered invaluable expert guidance through Tor's various layers of congestion control and flow control.

I would like to especially acknowledge J. Trent Adams at The Internet Society and Tom Lookabaugh formerly at PolyCipher for their generous financial support that, in part, made this research possible. In addition, I thank Stefan Savage, Geoff Voelker, and Damon McCoy for sponsoring my brief stint as research staff at UCSD from August–December 2010, during which time part of this thesis was completed. Lastly, I thank Vern Paxson for sponsoring my research visit to the International Computer Science Institute's Center for Internet Research (ICSI/ICIR) in Berkeley, CA during summer 2010, where I became fully immersed in the challenges of large-scale Internet measurement and network-based intrusion detection.

If I've learned anything in graduate school, I learned that research is a social activity. This thesis contains text, figures, tables, data, and ideas drawn from the following jointly authored papers: [49, 54–56, 60–63, 158, 159].

Finally, and most importantly, I thank my family for encouraging me to follow my dreams and for providing the emotional and financial support that enabled me to complete my education. Of course I would be remiss if I neglected to thank my furry friends Snoopy, Bridget, and Murphy (one canine, two felines) for not excessively peeing and pooping on the carpet or otherwise trashing our apartment while I worked many long nights to finish this thesis. Lastly, I thank Catherine, my friend and partner in life, for her love and support that makes everything I do worthwhile.

Contents

Tables

Figures

Figure

Chapter 1

Introduction

"If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place."

– Eric Schmidt, Chairman of Google

The TCP/IP protocol suite is among the fundamental building blocks of the Internet. However, it was designed without any regard for protecting the privacy of Internet users. Since the Internet's inception, protocols for protecting the confidentiality, integrity, and authenticity of communications have been developed and are now deployed ubiquitously to protect sensitive communications including financial transactions, personal e-mails, and social networking activities. While such protocols offer privacy by hiding the contents of a communication from unauthorized third-parties, they cannot alone conceal the fact that two parties are communicating.

During the Internet's early days, perhaps there was little need to communicate privately or anonymously. However, today's online world feels more like George Orwell's prophetic vision in *1984* of widespread surveillance and data mining than a technological utopia. Today, targeted behavioral advertising is commonplace [160], web search data is often aggregated and retained indefinitely [25], an arsenal of tracking technologies monitor and report on web browsing habits [38], fingerprinting techniques [30, 109] enable website operators or third-party advertising agencies to re-identify and track users even in the absence of explicit identifiers, large-scale domestic surveillance has been carried out by the U.S. National Security Agency (NSA) in collusion with telecommunications companies [29], and Internet users who reside in parts of the world controlled by autocratic governments are closely monitored and filtered by sophisticated deep packet inspection firewalls [165].

Commonly the “I have nothing to hide” argument justifies breaches of privacy in the name of national security or public safety; however, this justification is based on the faulty premise that privacy is about hiding amoral, illegal, or otherwise socially unacceptable behavior [214]. Rather, it has been argued that privacy is about protecting and controlling the flow of information about oneself to others [231], such as corporations that may wish to gather information to better target products at likely buyers or governments that surveil citizens in the name of public good.

1.1 Need for Privacy Enhancing Technologies

Given the absence of personal privacy protections built into the Internet, a variety of privacy enhancing technologies (PETs) have been designed and developed in an effort to give end-users effective tools to control and manage how their personal information is released and shared. For example, the Transport Layer Security (TLS) protocol [95] is the most widely used PET available today. TLS is a session layer security protocol that secures web logins, bank transactions, and personal communications from a third-party observer. While TLS ensures confidentiality, integrity, and authenticity, it cannot conceal the parties involved in a transaction or communication.

A variety of techniques have been designed and developed to hide the identities of parties engaged in communication. Single-hop anonymizing proxies such as Ipredator [22], BTGuard [11], and `anonymizer.com` [6] offer a form of anonymous communication by routing traffic from the sender through an intermediate proxy server and then on to the destination. While the identities of both communicating parties are hidden from network observers residing between the sender and the proxy or between the proxy and the destination, the proxy itself knows the identities of both parties and could betray them [124]. To overcome this limitation, de-centralized architectures consisting of multiple anonymizing proxies are employed, to reduce the amount of information that any single proxy knows about the sender or receiver. Anonymizing networks such as Tor [103], I2P [20], and AN.ON [134] attempt to conceal communicating parties’ identities from both network observers and the anonymization infrastructure itself. In combination with privacy enhancing proxy filters [31,33] that sanitize web page data, these PETs offer a strong form of privacy to protect users from invasive monitoring via deep packing inspection by network operators, profiling by website operators, and data mining of online behavior carried out by third-party advertising firms.

Anonymizing networks also offer a means to resist Internet censorship and filtering, promote freedom of speech and press online, and protect cyber-dissidents and activists. For example, Tor has been used to facilitate online protests following a highly contested Iranian presidential election in 2009 [44] and to help organize an Egyptian pro-democracy movement in 2011 [46]. PETs that enable anonymous and censorship resistant communications not only promote freedoms online, such tools can also be instrumental in affecting social or political change in the real world.

1.2 Better Performance Leads to Better Anonymity

It is generally accepted that a security system that is easy to use provides better security than one that may offer stronger security properties, but is harder to use. This is simply because there is a greater chance that the user may not configure the system properly or other may otherwise misuse the system in a manner that compromises security. In the realm of anonymity, this general principle also applies. Anonymity systems that are easy to use will attract a greater number of users, enlarging the system's anonymity set, increasing the difficulty of traffic analysis, and ultimately, offering better anonymity properties to all users [102].

We propose the following corollary to this principle: *anonymity systems that offer high security and performance will attract a greater number of users, enlarging the system's anonymity set, increasing the difficulty of traffic analysis, and, ultimately, offering better anonymity properties to all users.* Perhaps ironically, the greatest obstacle to the increased adoption of systems for low latency anonymous communications is their unacceptably high delays [92,105,184,187,230], which discourage delay-sensitive web users from participating. By reducing these delays, ideally to a level that makes such systems' performance indistinguishable from the native Internet, there will be fewer obstacles to participation. While any multi-hop anonymizing architecture may inherently necessitate an additional performance cost due to intentionally routing packets through multiple countries, autonomous systems (ASes), or Internet Service Providers (ISPs) to resist traffic analysis, ensuring that the performance cost associated with the anonymity system is as low as possible will attract users who otherwise may not tolerate the delays.

1.3 Problem Statement

While it is clear that the design of low latency anonymity networks involves a careful consideration of security and performance trade-offs, current systems offer weaker security and worse performance than may be expected by users. In this dissertation, we seek to improve the security and performance properties of low latency anonymity networks. We assert the following thesis:

> *Low latency anonymity networks can offer greater anonymity and better performance than provided by existing systems.*

We focus primarily on improving Tor, which has become the most popular low latency anonymity network with an estimated 200,000 daily users [151] and the defacto platform for research. We begin our analysis with a comprehensive characterization of real-world Tor usage to obtain an understanding of the strengths and weaknesses of a currently deployed low latency anonymity network. Armed with the results of this study, we proceed by offering a variety of solutions that collectively improve Tor's security, anonymity, and performance.

1.4 Fundamental Contributions

This thesis contributes the following to the field of anonymous communications.

- **Real-world anonymity network characterization.** (*Chapter 3*) To better understand the security and performance of currently deployed low latency anonymity networks, we characterize the Tor network in terms of application usage, geopolitical client and router distributions, performance for end-users measured over the course of four years, and the potential for abuse. Among other findings, we show that peer-to-peer file sharing protocols such as BitTorrent consume an unfair portion of the available network bandwidth. This degrades performance for users who wish to anonymize their web browsing or instant messaging traffic.

- **Traffic confirmation attacks and defenses.** (*Chapter 4*) We experimentally explore how Tor is vulnerable to end-to-end traffic confirmation attacks by weak adversaries who misrepresent their bandwidth to attract traffic. Also, we propose and evaluate a novel circuit linking technique that accurately correlates clients and destinations before any data traffic is sent. We develop a wide range of defenses, many of which have been adopted and deployed on the live Tor network.

- **Path length implications for security and performance.** (*Chapter 5*) We study the performance and security trade-offs that revolve around path length design decisions. While we empirically verify that performance improves with shorter paths, we also find that shorter paths are more resilient to certain attacks relative to longer paths.

- **An alternative anonymity system for file sharing.** (*Chapter 6*) To alleviate the high traffic load and congestion in networks like Tor that results from bulk transfer protocols such as BitTorrent, we design, implement, and evaluate an anonymizing network designed specifically for BitTorrent. With an alternative anonymity system for bulk traffic, low latency anonymity networks such as Tor may experience less congestion and offer faster service for delay-sensitive web users.

- **Improved congestion and flow control.** (*Chapter 7*) We seek to improve Tor's performance by diagnosing the shortcomings of its window-based congestion and flow control and offering the design, implementation, and performance analysis of a per-link congestion and flow control strategy from ATM networks. We argue that by improving performance, Tor becomes more attractive to users, and enlarging the user base ultimately enhances the system's anonymity properties.

1.5 Dissertation Outline

The remainder of this dissertation is organized as follows. Chapter 2 offers a survey of the relevant background and related work from the anonymity and privacy literature. Chapter 3 analyzes how Tor has been used in practice, who uses Tor, how Tor performs from the end-user's perspective, and how Tor's anonymity properties may be mis-used. Chapter 4 studies Tor's security and anonymity properties, focusing on how a adversary with very modest computing resources can compromise the system's fundamental anonymity properties. Chapter 5 examines the anonymity and performance trade-offs that are related to path length design decisions in Tor. Chapter 6 presents a case study of information leaks in BitTorrent to motivate the need for an anonymizing protocol tailored to enhance privacy for peer-to-peer file sharers. The design, implementation, and analysis of a Crowds-style anonymity network built into the BitTorrent protocol is also presented. Chapter 7 aims to improve Tor's performance by critically evaluating its end-to-end window-based congestion and flow control mechanics and proposing alternative approaches that leverage techniques from IP and ATM networks. Finally, Chapter 8 summarizes the fundamental contributions of this thesis and highlights a variety of avenues for future work.

Chapter 2

Background and Related Work

The Internet's fundamental architecture built around the TCP/IP protocol suite was ostensibly designed without any regard for preserving Internet users' privacy or facilitating anonymous communications. For example, while globally unique identifiers such as IP addresses have made routing packets at the network layer a relatively easy task, they can also be used to monitor and track a user's online activities. Enabling users to communicate anonymously is inherently difficult, since a packet's source and destination fields are explicitly provided in the IP header. In an effort to enable anonymity without redesigning the fundamental architecture of the Internet, a variety of overlay solutions have been proposed to effectively re-write the packet's source and destination in such a manner that hides the true sender and receiver.

This chapter provides an overview of the fundamental terminology, techniques, protocols, and systems from the anonymity literature. In particular, we provide a survey of the fundamental techniques for achieving anonymous communications including mix networks, dining cryptographers networks, onion routing, and information slicing. These methods are analyzed in terms of their security and performance properties. Next, an overview of a sample of the systems that implement these techniques is provided. Finally, this literature survey concludes with a discussion of the common metrics that help to understand and quantify the degree of anonymity that a system provides.

2.1 Preliminaries and Definitions

Before discussing the fundamental techniques that enable anonymous communications and the various systems that implement them, we first define the key terminology that will be used throughout the remainder of this document. All defined terms are based on Pfitzmann and Hansen's proposal for a common vocabulary [179].

Definition 1 *Anonymity* of a subject means that the subject is not identifiable within a set of subjects, the *anonymity set.*

The notion of an anonymity set was proposed by Chaum [76] and further discussed by Sweeney [218]. In this model, users are indistinguishable within a set of k subjects, the anonymity set. These subjects could be the users of a particular system or servers that make up the necessary infrastructure of an anonymous communication system. In the best case, an adversary should have no better than a $1/k$ chance of identifying a subject from within this set. This corresponds to an adversary without any prior knowledge about the anonymity set guessing at random about a user's identity with the anonymity set. In information theoretic terms, the entropy over the probability distribution within the anonymity set is maximized (Chapter 2.4 introduces information theoretic anonymity metrics). Defining anonymity in terms of a set of indistinguishable subjects provides an easily quantifiable metric that can be applied to help reason about anonymity.

This anonymity property may hold for certain subjects and not for others. For example, some systems are said to provide sender anonymity, where a message initiator's identity is hidden. Some systems may protect the responder's identities, providing receiver anonymity. Systems may also provide both sender and receiver anonymity.

Definition 2 *Identifiability* of a subject from an attacker's perspective means that the attacker can sufficiently identify the subject within a set of subjects, the *identifiability set.*

Being identifiable within the anonymity set with a probability significantly greater than $1/k$ makes the subject identifiable. The precise probability that implies identifiability is subjective and can be defined on a case by case basis.

Definition 3 *Unlinkability* of two or more items of interest from an attacker's perspective means that within the system, the attacker cannot sufficiently distinguish whether these items of interest are related or not.

In addition to quantifying the degree of anonymity in terms of the anonymity set, it is often necessary to describe relationships between subjects within the anonymity set. Unlinkability captures the degree to which subjects' relationships are hidden from an adversary. For example, an anonymous communication service may wish to guarantee that messages being sent to the anonymity service are unlinkable from messages leaving the service. This ensures that the senders are unlinkable to their respective receivers. In addition, unlinkability can also be defined in terms of sender unlinkability or receiver unlinkability.

Definition 4 *Linkability* of two or more items of interest from an attacker's perspective means that within the system, the attacker can sufficiently distinguish whether these items of interest are related or not.

Linkability is the inverse of unlinkability. This captures the degree to which an adversary can discern a relationship between subjects of interest. For example, suppose that an adversary can infer patterns in the messages entering and leaving an anonymous communication service. The adversary is said to be able to link the communications, revealing the sender's and/or receiver's identities.

Definition 5 *Undetectability* of an item of interest from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not.

Denying the existence of an item of interest, such as a message in a communication channel, implies that third parties who are not involved in the message (as a sender or receiver) should not be able to infer the existence of the message. Suppose that a system sends constant rate cover traffic (or dummy traffic) in a manner that is indistinguishable from real messages that are processed by the system. If an external adversary cannot distinguish this cover traffic from real traffic, then the system is said to provide the property of undetectability.

Definition 6 *Unobservability* of an item of interest implies the following: (1) Undetectability of all items of interest against all subjects uninvolved in it, and (2) Anonymity of the subject(s) involved in the items of interest even against the other subjects involved in that item of interest.

A service that satisfies the properties of undetectability with regard to external observers and anonymity of all subjects involved in the system is said to be unobservable. Adding cover traffic to an anonymous communication service in such a manner that (1) legitimate items of interest cannot be distinguished from the cover traffic and (2) all subjects are anonymous themselves. Only one of the subjects can identify its own item of interest within such a system.

Definition 7 A *pseudonym* is an identifier of a subject other than one of the subject's real names.

Definition 8 A subject is *pseudonymous* if a pseudonym is used as an identifier instead of one of its real names.

A pseudonym is simply an alternate identifier. While the pseudonym and the real identifier refer to the same identity, the pseudonym may be semantically different. For example, a real identifier such as an IP address can be made pseudonymous by applying a one-way keyed cryptographic hash to produce a pseudonym. This pseudonym enables the real identifier's actions to be linked, but it may conceal additional semantic information such as the identifier's Internet service provider, autonomous system, *etc.* By studying the subject's behavior or learning its attributes, it may be possible to infer the mapping between pseudonyms and real identities.[1]

[1] An example of how easy it often is to identify real identities from their pseudonyms is the AOL search data release of 2006 [14]. AOL released an "anonymized" data set consisting of the contents and times of it's customer's search queries. Data from over 650,000 users over the course of three months was released to the general public. Each user was assigned a pseudonym to protect their real identity. However, it quickly became clear that this data could be de-anonoymized by analyzing the contents of the user's search queries and correlating their searches with external data sources [41].

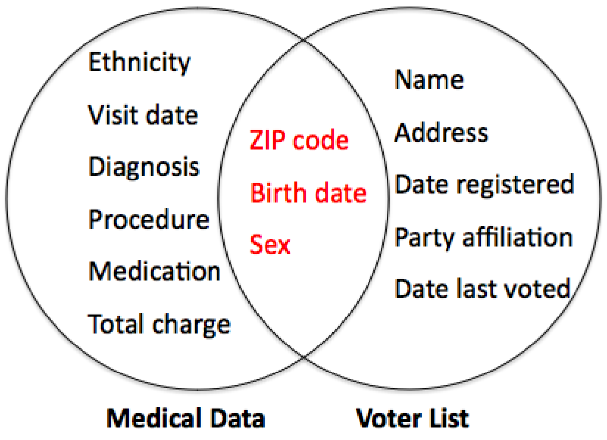


Figure 2.1: An example of how a subject's attribute information can be used to reconstruct identities

Definition 9 An *identity* is any subset of attributes of an individual person which sufficiently identifies this individual person within a set of persons.

An identity could be an explicit identifier such as a name, social security number, or a network address. Identity can also be defined in more subtle terms. A set of attributes, or information about a subject, could themselves be used to construct an identity. These attributes could be unique or they could form an anonymity set with a very large k. For example, the attribute "U.S. citizen" creates an anonymity set with $k \approx 227,000,000$. However, a particular set of attributes may — in some cases — establish an identity that only one subject possesses.

Consider the following example: In Massachusetts, the Group Insurance Commission (GIC) released the medical records of state employees in an "anonymized" form consisting of only the following set of attributes {ethnicity, visit date, diagnosis, procedure, medication, total charge, **ZIP code**, **date of birth**, **sex**}. No explicit identifiers such as names or social security numbers were released. However, a clever graduate student named Latanya Sweeney believed that this data could be used to re-identify patients. To prove her point, she obtained a voter registration list consisting of {name, address, date registered, party affiliation, date late voted, **ZIP code**, **date of birth**, **sex**} and cross-referenced the two data sources as shown in Figure 2.1. Through this process, Sweeney was able to uniquely identify William Weld, the then-Governor of Massachusetts, and send his office a copy of his medical records [218]. Anonymizing data is a challenging problem, since a subset of a subject's attribute information can be used to reconstruct an explicit identifier such as an individual's name. Similarly, anonymizing network traffic shares many of the same challenges. In the next section, we survey the fundamental techniques that have historically been applied to overcome these challenges and enable anonymity of network traffic.

2.2 Anonymity Techniques

Unlike the traditional properties that are required to establish a secure communication channel, anonymous communication cannot be provided using cryptographic primitives alone. For example, encryption can be applied to ensure confidentiality, digital signatures or message authentication codes (MACs) can prove the authenticity of an identity, and hash functions verify a message's integrity to ensure that no data has been unknowingly modified. However, anonymous communication necessitates resistance to *traffic analysis*, which involves hiding the identities of the communicating parties.

One trivial solution to the traffic analysis problem is to relay all communications through a centralized and trusted authority. However, this ostensibly places enormous trust upon this centralized authority, since the authority knows the identities of the two communicating parties and could arbitrarily disclose this information to another third-party such as a law enforcement agency or a political regime. Thus, it is desirable to design an anonymous communication infrastructure that is decentralized and spans several authoritative jurisdictions [114]. In such a decentralized architecture, it should be infeasible for the identities of the two communicating parties to be discovered unless the entire infrastructure colludes.

In this section, we present an overview of the most fundamental techniques for realizing anonymous communication. These approaches fall into two general categories: *high latency* (or message-based) and *low latency* (or stream-based).

2.2.1 High Latency Anonymity

High latency techniques attempt to hide timing information that could be used as a side channel to facilitate traffic analysis. While high latency anonymity techniques provide the most optimal security properties, they are often impractical due to their requirements that traffic be delayed, reordered, or otherwise manipulated in a manner that hides timing information.

2.2.1.1 Single Mix

The *mix* is the fundamental building block of most high latency anonymous communication systems [75]. A mix is a server that maintains a asymmetric key pair (K_{pub}, K_{priv}). The mix publishes K_{pub} with a trusted public key infrastructure. Users wishing to send messages anonymously simply encrypt their message with K_{pub} using a well-known public key cryptosystem such as RSA [191] and send the encrypted message to the mix.

A single mix can hide the correspondences between senders and receivers with the following procedure: First, suppose that all users have access to a public key infrastructure that distributes public keys. Also, suppose that a user wants to send a message M anonymously to another user at address A. The user prepares M by first padding it with a random string R_0 to prevent chosen-ciphertext attacks [185] and then encrypting M with the recipient's public key K_{pub_A}. This encrypted message and the recipient's address are added with another random string R_1 and encrypted with the mix's public key K_{pub}:

$$K_{pub}[R_1, K_{pub_A}[R_0, M], A] \rightarrow_{mix_1} K_{pub_A}[R_0, M], A$$

The message on the left-hand side of the arrow is the input to mix_1. On receipt of this encrypted message, mix_1 uses its private key K_{priv} to decrypt the message. Now, mix_1 can see the recipient's address A, so it can deliver the message. Finally, the recipient A decrypts the message with its own private key K_{priv_A}.

The mix stores each message for some time and forwards messages in batch to their intended destinations. Batching prevents an eavesdropper who is monitoring the links between (1) the sender and the mix and (2) the mix and the destination from inferring the source and destination pairs by observing the timing properties of the messages entering and leaving the mix. Thus, mixes attempt to frustrate traffic analysis by perturbing temporal information. While this increases the latency of the communications by artificially manipulating message timings, it provides strong anonymity guarantees even in the presence of a passive and global adversary. The precise manner in which the mix processes the batches is referred to as the *flushing algorithm.* Attacks on the flushing strategies are collectively referred to as *blending attacks.*

Anonymous return addresses. Suppose that the recipient A wants to reply to the initial sender, but A doesn't know the identity of the sender. To facilitate anonymous replies, the initial sender creates an untraceable return address of the form:

$$K_{pub}[R_1, A], K_{pub_x}[.]$$

where A is it's real address, R_1 is a random bit string chosen by the sender, and K_{pub_x} is a one-time use public key chosen by the sender. The sender sends this untraceable return address to the recipient using the technique described above. The recipient replies with the following through the mix:

$$K_{pub}[R_1, A], K_{pub_x}[R_0, M] \rightarrow_{mix_1} A, R_1[K_{pub_x}[R_0, M]]$$

The mix decrypts with its private key K_{priv_1} and re-encrypts the message with the random bit string R_1. Only the original sender A knows both the private key K_{priv_x} and the random string R_1, since it chose R_1, so only the initial sender can decrypt the reply message. These anonymous return addresses can be generalized to a mix network or mix cascade using layered public key encryptions in a similar fashion as above.

This technique is useful to issue a receipt, as a form of certified e-mail, in a privacy-preserving online voting system. However, in such an online voting system, it would be essential to ensure that only eligible and registered voters cast ballots and that votes are cast only once per voter. To ensure these properties, it is necessary to provide digital pseudonyms, which are unique public keys that can identify voters.

We now discuss a sample of high latency mix schemes and their security properties (the proceeding summary is based primarily on [196]).

Threshold mix. Chaum's mix uses a threshold scheme in which the mix flushes all messages after every n-th message is received [75]. However, this flushing approach is vulnerable to an active attack called the flooding attack. In this attack, the adversary first causes the mix to flush by flooding the mix with dummy message. After it flushes, the adversary waits for a target message to enter the mix, and again floods the mix with dummy traffic until the mix flushes with precisely one legitimate message and $n-1$ dummy messages. Since the adversary can identify and ignore their dummy messages, the only remaining message (that is not theirs) is a legitimate message. The adversary can identify the target message and its destination. Cover traffic has been proposed to defend against this $n-1$ attack [89].

Timed mix. The timed mix flushes all messages every t seconds. The timed mix can be defeated by the *trickle* attack, in which an adversary delays all messages entering a mix for precisely t seconds. The adversary then allows only a single target message to enter the mix. Since there is only a single message in the mix, it is easy for the adversary to observe the message's destination.

Threshold or timed mix. The threshold or timed mix flushes every t seconds or after n messages have been received. This flushing algorithm is vulnerable to both the trickle attack, the flooding attack, or a combination of both. This design provides no additional benefit over each individual flushing method.

Threshold and timed mix. The threshold and timed mix flushes every t only when at least n messages have entered the mix. This flushing strategy is vulnerable to a combination of the trickle and flooding attacks.

Threshold pool mix. The threshold pool mix flushes after $n+f$ messages are received. A pool of precisely f messages (f is a random number) are kept in the mix at each flush while the other n are forwarded. Which messages are kept and which are forwarded are also chosen at random. To defeat this flushing, the adversary first floods the mix to flush all messages. Next, the adversary forwards the target message to the mix. The adversary again floods the mix and counts how many of the adversary's messages leave the mix. If $j < f$ of their messages leave, then $f - j$ messages are still in the mix. The adversary delays all other messages entering the mix and flushes the remaining messages. Finally, the target message is flushed.

Timed pool mix. The timed pool mix flushes every t seconds and keeps a pool of f message in the mix. f is chosen at random. The mix flushes only if there are greater than f messages in the mix. An adversary may flood the mix in the hope that all f of the messages are their messages.

Timed dynamic-pool mix. The timed dynamic-pool mix flushes every t seconds only if there are $n + f_{min}$ messages in the mix. Rather than always sending n messages, the mix sends $max(1, \lfloor m \times frac \rfloor)$ and keeps the remaining messages in the mix. $m + f_{min}$ is the number of messages in the mix, subject to $m \geq n$. $frac$ is a parameter that selects what fraction of messages should be kept in the mix. The timed pool mix attacks are still possible, but now the probability of all of the attacker's messages being kept in the mix is a function of $frac$.

2.2.1.2 Mix Networks and Mix Cascades

While the single mix provides strong anonymity properties against a powerful adversary, it is often vulnerable to a variety of active attacks, depending on the flushing algorithm used. Furthermore, the mix server itself knows the message's sender and the destination. To prevent any single entity from linking messages, Chaum proposed the first decentralized protocol for providing anonymous communications [75].

A *mix network* is a store-and-forward network in which a series of single mixes are used to process messages after they are sent and before they are received. The procedure to send a message through a single mix generalizes to a *network* of n mixes as follows. The sender encrypts the message with the public keys of each of the n mixes in a layered fashion and delivers it to the first mix in the path, mix_n:

$$K_{pub_n}[R_n, K_{pub_{n-1}}[R_{n-1}, ..., K_{pub_2}[R_2, K_{pub_1}[R_1, K_{pub_A}[R_0, M], A]]...]] \rightarrow_{mix_n}$$

When a message is received by mix_n, it removes a layer of encryption by decrypting with its private key K_{priv_n} and forwarding the message to the next mix mix_{n-1} in the network. mix_{n-1} receives the following:

$$K_{pub_{n-1}}[R_{n-1}, ..., K_{pub_2}[R_2, K_{pub_1}[R_1, K_{pub_A}[R_0, M], A]]...] \rightarrow_{mix_{n-1}}$$

In order to determine the sender and recipient pair, it is necessary for all n mixes to collude. The message leaving the mix network has the form $K_{pub_A}[R_0, M], A$ which is the same as the single mix. Any of the flushing algorithms discussed in Chapter 2.2.1.1 can be applied to the network of mixes.

Decentralized anonymity systems built with a series of mixes need to have a methodical way to choose which mixes to use. A user can choose mixes in a free-route manner, where the user source-routes their traffic through a path of mixes of a particular length. Alternatively, mix cascades require that users always route their traffic through the same fixed route of mixes [106]. Mix cascades ostensibly place a tremendous amount of trust on the mix cascade while free-route mixes need some way to distribute information about the available mixes (such as their IP addresses and public keys) in a secure manner.

2.2.2 Low Latency Anonymity

In contrast to high latency techniques that attempt to mask timing information, a variety of approaches have been proposed to enable low latency anonymous communications by processing messages in real-time. These systems make a compromise. They provide a lower security guarantee, but provide performance sufficient to transport interactive traffic. This inherent trade-off between performance and security is explored further in this section.

2.2.2.1 Single Hop Proxy

A naive method for achieving a very weak form of anonymity is to proxy all traffic between a user and a destination server through a single intermediate server, such as a virtual private network (VPN) server. Examples of this centralized approach in practice include `anonymizer.com` [6], BTGuard [11] and Ipredator [22]. From the destination server's perspective, the traffic appears to originate at the VPN server. Also, an eavesdropper monitoring the user's link would not be able to determine the true destination of the user's traffic. While the single hop proxy is simple and elegant, the proxy itself trivially knows the identities of both the user and the destination. Consequently, a decentralized approach where no single entity knows both endpoints of the communications is desirable.

2.2.2.2 Onion Routing

Onion routing attempts to limit a network's vulnerability to traffic analysis by providing a bi-directional, real-time virtual circuit of layered encryption between two communicating parties [122]. In contrast to mix networks that employ batching and re-ordering to alter the temporal properties of a communication session, onion routing attempts to provide sufficient performance to support interactive applications such as HTTP. Consequently, onion routing is vulnerable to statistical correlation attacks at the end-points of the network that mix networks inherently resist. The first implementation of onion routing used proxies to transparently connect the user's application to the onion routing infrastructure. However, in the initial implementation, a distinct proxy must be written for each application to be used. In later generations of onion routing, a more generic proxy interface such as SOCKS is used.

Onion routing architecture. An onion routing network consists of a set of *routing nodes* that proxy traffic using a layered encryption scheme. To obscure the correspondence between the initiator and the responder, the traffic is source-routed through a subset of the onion routers. The onion routing technique's ability to obscure the sender's and receiver's identities comes from the assumption that any single onion router can only know the identities of the it's previous and next node in the onion router chain. For instance, if a particular onion router is the first hop along the path from the initiator, then this onion router trivially knows the identity of the initiator. However, since this router forwards the traffic to another router, it cannot know the final destination's identity. In the original design, the number of routers that should be used for a communication session is not strictly specified — it could be fixed or some random number $i > 1$. In the model, it is assumed that if an adversary controls the first node in the onion chain and the final onion router before the destination, then that adversary can learn the identities of both communicating parties.

To mitigate this threat, it is suggested that the number of onion routers used on a path from the initiator to the responder be sufficiently high to ensure that the probability of a single entity controlling the routers at both endpoints is sufficiently small. However, as more routers are used for these paths, the performance degrades. Therefore, it is important to find a suitable balance between performance and security when choosing the number of routers through which to forward. For instance, in Tor [103] — perhaps the most popular and successful anonymous communication system based on the onion routing design — the number of routers is fixed at three.

Virtual circuits of layered encryption. Onion routing allows users to establish a virtual circuit through multiple onion routers using a data structure called an onion that is encrypted in layers. The basic structure of the onion is based on the route to the responder that is chosen by the initiator. For instance, the initiator chooses to route its message M through onion routers OR_1, OR_2, and OR_3 in that order. Using the routers' public keys, the initiator prepares an onion with M at the center:

$$K_{OR_1}(K_{OR_2}(K_{OR_3}(M)))$$

Having prepared the message as an onion, the initiator forwards the message to OR_1, which has the private key S_{OR_1}. OR_1 uses its private key to remove the outer-most layer of encryption from the onion and forwards $K_{OR_2}(K_{OR_3}(M))$ to the next router specified. This continues until the final layer of encryption is removed, revealing the final destination address and payload. This process can establish a virtual circuit, which enables the initiator to send messages and the respondent to reply via the same circuit (*i.e.*, set of onion routers). Reply messages are forwarded in a similar fashion, except that each onion router adds a layer of encryption. In most implementations, circuits are identified with a unique circuit identifier. Once a virtual circuit is created, the initiator can send data along the circuit and the respondent can reply similarly. To destroy a circuit when the communication session is coming to an end, the initiator can send a special destroy message along the circuit to remove that circuit.

2.2.2.3 A Network Coding Approach

Mix networks and onion routing both require a public-key infrastructure (PKI). To eliminate the need for a PKI, information slicing chops a message into multiple components and then sends those components over multiple disjoint paths. Recent work [138] shows how messages can be broken into multiple parts using pseudo-randomized decompositions, preventing information leakage. Here, portions of a message from Alice to Bob are injected in the network at two points (Alice and Alice'). The parts are delivered via disjoint paths to Bob, who can then reassemble them. However, these approaches require the simultaneous use of multiple links (Alice and Alice'). For most network users, it is impractical to have multiple physical network connections. In the case of physical surveillance, this form of anonymity provides little confidence because both lines could be monitored at the sender's location.

2.2.2.4 DC-Nets

Another anonymous communication protocol was proposed by Chaum which can mask the identities of the parties that send messages [76], providing *unobservability* properties. A protocol based upon a solution to the dining cryptographers problem is presented that is either unconditionally secure — if a one-time pad is used — or is cryptographically secure if public key cryptography is used. Unconditional security is the "gold standard," however realizing one-time use keys is not practical. We next discuss the DC-Nets protocol to highlight its appealing theoretical properties.

The dining cryptographers problem. The dining cryptographers problem is stated as follows: Suppose that three cryptographers are eating dinner and their waiter informs them that a third-party has generously picked up their bill. The cryptographers wonder if one of them is paying the bill, or if the NSA is paying. They resolve this question using a clever protocol: Each cryptographer secretly flips an unbiased coin and shares the outcome with the cryptographer to her right — so only two of the cryptographers know the outcome. Next, each cryptographer states aloud whether the two coins she sees fell on the same or different sides. This is logically equivalent to performing the exclusive or operation on the two outcomes. If one of the cryptographers, in fact, paid the bill, then she should say the opposite of what she sees. If there is an odd number of differences stated at the table, then one of the cryptographers is paying. Otherwise, if there is an even number of differences, then someone else is paying the bill. Note that if one of the cryptographers did pay the bill, their identity remains anonymous.

DC-nets as a theoretical construct. To make the protocol more concrete, consider the following example: Suppose that the three cryptographers flip coins in secret and the outcomes are A, B, and C. The outcomes are shared according to the protocol described above, so $AB = A \oplus B$, $BC = B \oplus C$, and $CA = C \oplus A$. These three values are shared and $AB \oplus BC \oplus CA$ reveals whether one of the cryptographers paid the bill. Put another way, suppose that one of the cryptographer wants to broadcast a message m anonymously. That cryptographer simply performs the exclusive or operation with their message, for instance, $CA = C \oplus A \oplus m$. Now, when the three cryptographers share their outcomes with a third-party mediator, the message is revealed: $AB \oplus BC \oplus CA = m$. The cryptographer who pays is an analogy for the one sending m. The message is revealed, but the identity of sender is secret. Note that this scheme is secure to a globally passive eavesdropper who can see every message sent, since m is composed by all parties. Similarly, no subset of the cryptographers can learn the identity of the sender.

This simple protocol achieves unconditional security since each cryptographer cannot know for certain if the coin flip is the same or different from the cryptographer's coin that she cannot see. Thus, it is not possible to determine who is honest and who has lied (*i.e.*, who actually paid or sent m).

This solution to the dining cryptographers can be generalized to n participants. Suppose that each participant has a secret key, of which one bit is shared in common with every other participant. Each participant shares a single bit with another participant. After the sharing, each participant takes the sum modulo two of all shared key bits. The protocol proceeds in rounds. If

Table 2.1: A taxonomy of anonymous communications systems

	Security			*Performance*	
	Anonymity	**Undetectability**	**Unobservability**	**High Lat.**	**Low Lat.**
Anonymizer.com	✓				✓
Tor	✓				✓
Crowds	S				✓
Tarzan	✓	✓	✓		✓
Mixminion	✓	✓	✓	✓	
Mixmaster	✓	✓	✓	✓	
Morphmix	✓	✓	✓	✓	
Herbivore	✓	✓	✓		✓
P^5	✓/S/R				✓
JAP/AN.ON	✓				✓
Nonesuch	✓	S	S	✓	
AP3	✓				✓
Cashmere	✓				✓
Freenet	✓				✓
Salsa	✓				✓
Freedom	✓				✓
UDP-OR	✓				✓
IPpriv	✓				✓

no participant transmits a message during this round, then the sum modulo two is zero, since every bit appears precisely twice. Otherwise, one of the participants transmitted a message. However, it is possible that if an even number of participants transmit during a single round, then the sum is zero. Similarly, if an odd number of participants transmit during a single round, then the sum is one. This type of message collision is mitigated if each participant has a jth bit of their key in common with every other participant, and the ith bit (for $i \leq j$) is used for the ith round. Chaum proposes a public-key distribution technique that can be used to construct a computationally secure channel with sender anonymity.

Despite its strong anonymity properties, DC-Nets presents significant practical obstacles to deployment. Since participants are modeled as nodes in a fully connected graph where the edges are shared keys, it is necessary to share $\frac{n(n-1)}{2} = O(n^2)$ keys. This becomes impractical as n grows.

2.3 Anonymity Systems

Having presented the fundamental techniques for achieving anonymity in a network setting, in this section, we provide a brief overview of a variety of systems that have been proposed and implemented to achieve anonymous communications. These systems provide anonymity properties with high latency or low latency performance properties. In addition, systems built on peer-to-peer (P2P) and client/server architectures have been developed. Table 2.1 provides a taxonomy of these systems in terms of the security properties that they provide and their performance (✓indicates that the property holds for both senders and receivers, S indicates that the property holds for only senders, and R indicates that the property holds for only receivers). These systems are classified according to the security properties of anonymity, undetectability, and unobservability and according to whether they provide high latency or low latency service. In the remainder of this section, we discuss each system in turn, highlighting their distinguishing features.

2.3.1 Crowds

Crowds is unique in that it derives its anonymity properties from the concept of "blending into a crowd." Crowds' architecture uses a set of special *Jondo* nodes that simply proxy HTTP traffic on behalf of the nodes. When a Jondo receives a message, it either forwards it to another Jondo node or delivers it to its final destination with a certain probability p which is decided before-hand. This probability defines the degree of anonymity or the level of certainty that a particular Jondo has that the previous node was the initiator. To prevent local eavesdroppers from inspecting the traffic, link encryption is used between Jondos (such as TLS), however, no cryptography is required to satisfy the system's anonymity properties. Crowds achieves a certain level of *source* anonymity, but since any Jondo can see the payload, the destination's identity is not hidden. Crowds provides sender anonymity and sender/receiver unlinkability.

2.3.1.1 When is Crowds-style Anonymity Appropriate?

Crowds was initially proposed for anonymizing HTTP traffic. However, due to the dynamic content of HTTP, it is likely that a user may inadvertently identify themselves in an HTTP request, perhaps by visiting a website with an non-TLS-protected login process. Thus, we assert that HTTP is not ideal for the Crowds-style of anonymity. In Chapter 6, we argue that peer-to-peer file sharing protocols (such as BitTorrent) is more well-suited to this form of anonymity since the content is publicly available and there is no dynamic content that could reveal identifying information.

2.3.2 Tarzan

Tarzan is an anonymity-preserving network layer based on a peer-to-peer overlay model that is transparent to higher-layer applications [116]. Tarzan uses layered encryptions much like mix networks or onion routing and it also provides cover traffic to frustrate an adversary's ability to conduct traffic analysis and infer the initiator of a message. Of particular interest is Tarzan's ability to select peers in such a way that's difficult for an adversary to influence. A network address translator (NAT) is used to bridge hosts using Tarzan and those using standard IP.

2.3.2.1 Design Goals

Most importantly, Tarzan attempts to provide an anonymity-preserving network layer that, like IP, is transparent to existing applications and services. Both sender and receiver anonymity should be ensured even in the presence of colluding nodes. Fault-tolerance and availability should also be guaranteed in the presence of colluding nodes that attempt to disrupt service. In addition, since Tarzan has been proposed as an anonymity-capable replacement for IP, it should provide the highest level of performance possible. Finally, perhaps its most ambitious design goal is to maintain sender and receiver anonymity even against a global eavesdropper, *i.e.*, an eavesdropper who can monitor the entire network. We next describe how Tarzan is able to achieve these goals.

2.3.2.2 System Architecture

Tarzan uses a distributed peer-to-peer architecture of relays that use layered encryption in the style of the Chaumian mix, where each relay either adds or removes a layer of encryption (depending upon the packet's direction). Suppose that a client desires anonymity for a particular communication session. The client chooses a set of relay peers through which to route its traffic. The client establishes session keys with these peers and establishes a tunnel using these nodes. The client sends its data through the tunnel, and the final relay peer forwards the data to the intended destination. The peers use a NAT to bridge their private address space with the Internet.

For peer discovery, Tarzan uses a simple gossip protocol and peers are selected by clients with a diverse set of IP prefixes. For instance, the assumption is that an adversary may control a set of peers within the same /16 subnet (according to CIDR notation). This mitigates an adversary's ability to compromise an entire path.

One particularly interesting aspect of Tarzan's design is its use of *cover traffic* to defend against traffic analysis. By forwarding cover traffic that consists of real (replayed) traffic in a constant manner, it becomes difficult for an eavesdropper to analyze the usage patterns and link a message to its initiator. While this approach offers additional security, it has a performance cost.

2.3.2.3 Security Analysis of Tarzan

Tarzan is among the only anonymous communication systems that is based on a peer-to-peer model. This approach offers desirable scalability properties and is difficult to explicitly block by an ISP or a government. Similar to onion routing networks, if the last relay peer is compromised, then it is possible to determine both the content of the message and the destination's identity. However, if the first relay is compromised, the payload is still secure (since it is encrypted), but it is not clear whether the identity of initiator is revealed. Due to the peer-to-peer architecture, it is unclear whether the initiator sent the message, or is merely a relay. Additional traffic analysis is required to obtain more information. Thus, Tarzan's peer-to-peer architecture offers more security against corrupt nodes at the beginning of the path than onion routing offers. In addition, since Tarzan uses constant cover traffic that is hard to distinguish from real traffic, it is difficult for an eavesdropper to determine if certain peers are forwarding real data or cover traffic.

2.3.3 Tor

Tor exemplifies the greatest success to date of research and development in practical anonymous communications systems. Based on an onion routing architecture, Tor attempts to provide

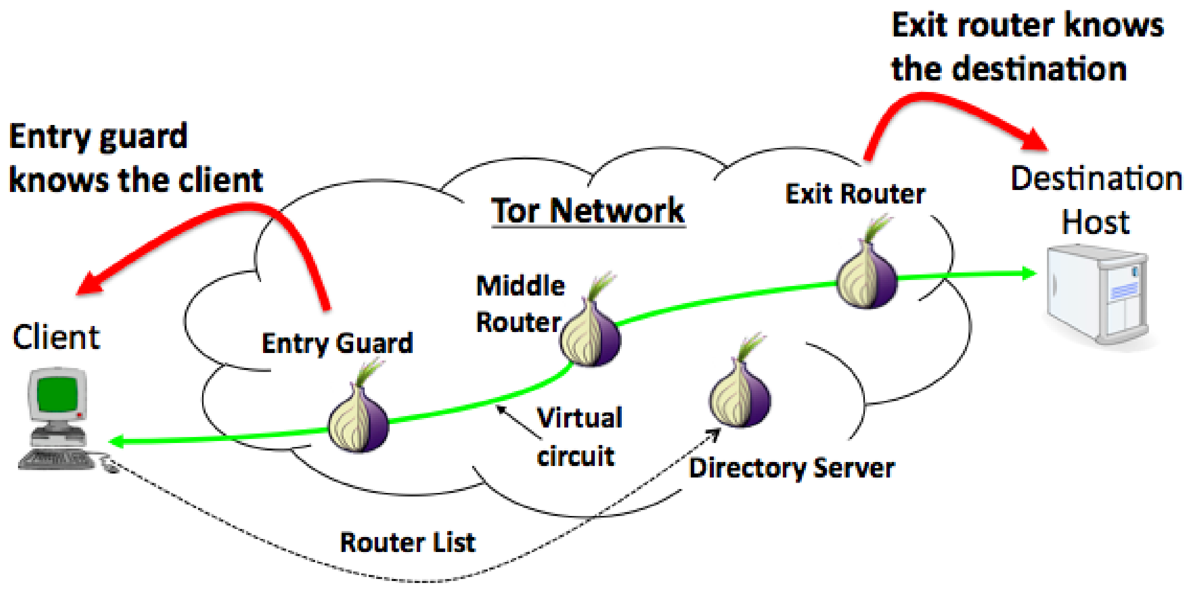


Figure 2.2: Tor's system architecture

low-latency anonymity service for TCP with perfect forward secrecy, a standard proxy interface, no mixing, TCP stream multiplexing over circuits, variable exit policies, and support for location hidden services (see [147, 153, 164, 172] for an overview of Tor's hidden services, as our work does not address hidden services). In Chapter 3, we show that Tor is used widely across 126 different countries, particularly those where the policies of local governments restrict Internet freedoms. As of October 2009, there are over 1,400 active Tor routers operated by volunteers around the world. Tor clearly exists as the most successful and widely adopted anonymity solution. As such, it is also the defacto platform for research in anonymous communications. Perhaps the greatest explanation for Tor's success is that it is easy to use with standard proxies such as SOCKS and browser plug-ins and more importantly, it has achieved a sufficient balance between security and performance to enable applications that require a low-latency transport such as HTTP while still resisting a variety of attacks.

2.3.3.1 Tor's Design

Tor is the second generation of the onion routing design, built as a circuit-switching overlay. Tor's system architecture (illustrated in Figure 2.2) consists of Tor routers and a set of directory server replicas to advertise the Tor routers' addresses, public keys, and other information to Tor clients. To establish a virtual circuit, a client chooses precisely three Tor routers and constructs an onion structure around messages in a similar fashion as described in Chapter 2.2.2.2, but with a few important differences. Since public key decryptions are computationally slow, Tor creates circuits in such a way that establishes shared symmetric keys between the client and each router in the circuit in a telescoping fashion. Once the circuit is established, the client shares a symmetric key with each router along the circuit, so it can build the onion structure using layered symmetric key encryption such as AES. Similar to the original onion routing design, each router along the circuit removes its respective layer of encryption and forwards the packet to the next hop or the final destination when the final layer of encryption has been removed.

To mitigate traffic analysis attacks that try to correlate packets entering and leaving the network based on their sizes, Tor pads all packets, or *cells* to a fixed size of 512 bytes. Fairness is ensured at each Tor router by employing round-robin circuit queuing. Routers may also be rate limited to obey desired bandwidth constraints.

2.3.3.2 The Evolution of Tor's Design

Tor has changed significantly since its initial design. Through Tor's active and open volunteer development process, changes are constantly being made to its protocol. Here, we focus upon a significant change that brings to light a fundamental challenge in designing and implementing *practical* anonymous communications systems. Initially, Tor clients chose the routers for their circuits uniformly at random. This provides a relatively strong resistance to the traffic correlation attack, since the probability of an adversary controlling both the first and last Tor routers is $(c/n)^2$, which is negligible as n is large. However, as Tor has grown in popularity, it has been necessary to ensure that the traffic load is balanced across the available bandwidth in the network. Thus, Tor clients now choose their routers by weighting their selection by the amount of perceived bandwidth that a Tor router has available. However, verifying a router's available bandwidth is fundamentally difficult, since available bandwidth is a shared resource that is variable as a function of time or other factors. In Chapter 4, we show that this load balancing has increased Tor's vulnerability to traffic correlation attacks from low resource adversaries who only control a few nodes and little bandwidth.

In addition to load balancing, Tor has added *entry guard* nodes [172] to mitigate the impact of the predecessor attack. Recall that the predecessor attack allows an adversary the ability to profile a large number of users by simply accepting connections and examining the previous node in the circuit to see if it's a client. To mitigate this, Tor restricts routers that can be used as the first hop in a circuit. More specifically, only nodes that have uptime and bandwidth capabilities that are at or above the median of all routers in the network are entry guards. Each client chooses a set of precisely three entry guards to be used exclusively for that client until these guard nodes become unavailable. In doing so, this makes the predecessor attack more difficult, since a malicious router must be an entry guard and then can only profile the clients for whom it is an entry guard – presumably a small subset of the entire client population. In the low resource routing attacks presented in Chapter 4, we also show that an adversary can report a very high uptime values to not only become an entry guard, but also cause all non-malicious guard nodes to lose their entry guard flags. This guarantees that the first hop is always compromised for new clients.

In response to our attack on entry guards, the entry guard selection mechanism was modified to keep track of the mean time between failures (MTBF) for each router, rather than self-reported (and therefore untrustworthy) uptime. Only those routers with a MTBF in the top half of all routers can be entry guards. The rationale is that MTBF is harder to fake than self-reported uptime statistics. In addition, to enforce greater location diversity within routers on the same path, Tor does not build paths with more than one router from the same /16 network according to CIDR notation.

2.3.3.3 Tor's Path Selection Algorithm

The manner in which Tor clients select their routers has serious implications for the network's security properties. For example, if a client chooses malicious routers, then they may experience lost anonymity. At Tor's conception, it was composed of only a few high-bandwidth routers and had few users, so it was sufficient to select routers uniformly at random. As the network grew in popularity and router bandwidth diversity, it became necessary to balance the traffic load over the available bandwidth resources, which can be achieved by selecting routers according to their bandwidth capacities. However, Tor routers self-advertise their bandwidth capacities. In Chapter 4, we demonstrate that an adversary can falsely advertise high bandwidth claims to attract traffic and increase their ability to compromise circuits.

Recent work has proposed methods to securely verify these self-reported bandwidth claims [212]. In addition, active measurements have been integrated into the Tor network's directory servers to verify routers' bandwidth claims [176]. However, the security of these active measurements has yet to be evaluated.

Tor's router selection algorithm [101] chooses routers according to the following constraints:

- A router may only be chosen once per circuit.
- To prevent an adversary who controls a small network from deploying a large number of routers, each router on a circuit must be from a distinct /16 subnet. Tor also allows an operator of many relays to set an advisory `Family` flag that will ensure that their nodes are not chosen twice per circuit.
- Each router must be marked as `Valid` and `Running` by the authoritative directory servers.
- For non-hidden service circuits, each router must be marked as `Fast`, indicating that the router has at least 100 KB/s of bandwidth or is within the top 7/8 of all routers ranked by bandwidth.

- The first router on the circuit must be marked as an `Entry Guard` by the authoritative directory servers. Clients select precisely three entry guards to use on all of their circuits.
- The last router (called an "exit router") on the circuit must allow connections to the client's destination host and port.

For general purpose circuits, Tor's path selection algorithm weighs router selection by each router's perceived bandwidth capacity. In order to ensure that there is sufficient exit bandwidth available, the bandwidth of `Exit` routers is weighted differently depending on the fraction of bandwidth that is available from non-`Exit` routers. Suppose that the total exit bandwidth is E and the total bandwidth available is T. If $E < T/3$, then `Exit` routers are not considered for non-exit positions. Otherwise, their bandwidth is weighted by $(E-(T/3))/E$ [101].

Entry guards were introduced to Tor's design to mitigate the threat of profiling and the predecessor attack [172]. Entry guard nodes have special uptime and bandwidth properties. A router is marked as a `Guard` by the authoritative directory servers only if its mean time between failures is above the median of all "familiar" routers and its bandwidth is greater than or equal to 250 KB/s [100]. A router is "familiar" if one-eighth of all active routers have appeared more recently than it [100]. By default, clients choose precisely three entry guards to use for their circuits. To ensure that there is sufficient guard bandwidth available, guard node selection is weighted by $(G-(T/3))/G$, where G is the amount of available guard bandwidth. If $G < T/3$, then guard nodes are not considered for non-guard positions [101].

Persistent applications such as FTP and SSH that establish sessions that are long-lived require more stable circuits than applications with short-lived sessions like HTTP. For such long-lived applications, Tor builds circuits solely with routers that are marked as `stable` by the trusted directory servers. A router is stable if it has been observed by the directory servers for 30 days or if it is above the median of all routers in terms of mean time between failures [157].

Finally, since routers self-advertise their bandwidth capabilities, we show in Chapter 4 that an adversary can attract traffic and increase their probability of controlling the endpoints of circuits by falsely reporting high bandwidth values. To mitigate the effectiveness of this attack, all bandwidth advertisements are restricted with by an upper limit. More details about Tor's path selection algorithm may be found in the path specification [101].

2.3.3.4 Censorship and Blocking Resistance

In addition providing source-destination unlinkability for TCP traffic, Tor also attempts to resist censorship and blocking. For example, in early 2010, China explicitly blocked access to Tor from behind the so-called Great Firewall of China. Consequently, Chinese users were unable to use Tor. In response to this and other blocking attempts by governments or Internet Service Providers, Tor introduced "bridges." A bridge is effectively a normal Tor client that helps other clients who reside within networks that block Tor's directory and router infrastructure to reach Tor. These bridges are advertised in an ad-hoc manner, often via e-mail or social networking channels, and they should be hard to identify and enumerate, which is a difficult practical problem [161].

2.3.3.5 Tor Attacks

Due to Tor's status as the current platform for research in anonymous communications and its extensive use in practice, a wide variety of attacks have been proposed to degrade Tor's anonymity. These include the following: determining the circuit of routers that are used by increasing congestion [166], locating hidden services within the Tor network [172], inferring the source of a request through Tor using cell counts [149], locating hidden services with recognizable clock skew [164,237], using network coordinate systems to infer Tor routers on a circuit using latency [127], examining how attacks on reliability (*i.e.*, DoS attacks) can reduce anonymity [61,67], injecting a covert signaling mechanism with AES counter mode decryption error propagation [183], injecting a watermark using variations in packet sizes [150], congesting routers with long paths [113], and understanding the extent of Tor's vulnerability to traffic correlation attacks [61, 167].

Tor certainly does not provide a perfectly secure anonymity service. However, to achieve performance suitable to support delay-sensitive applications, Tor's designers made explicit compromises in the system's anonymity properties to achieve low latency performance. Understanding the trade-offs between anonymity and performance is a topic of considerable interest both research perspective and practical perspectives. We discuss these issues in Chapter 2.5.5.1.

2.3.3.6 Tor's Transport Design

Tor uses pairwise TCP transport between Tor routers, and thus, it can leverage TLS link security to further resist traffic analysis by hiding virtual circuit identifiers from a network eavesdropper. In addition, multiple circuits may be multiplexed over the same TCP connection, which has been shown to result in an unfair application of TCP's congestion control mechanisms when large flows compete with small flows for service [187,188]. In Chapter 7.1, we overview and analyze Tor's TCP-based transport design, particularly as it effects congestion and flow control. In the remainder of Chapter 7, we diagnose a source of performance degradation due to poor congestion and flow control and we offer solutions.

2.3.4 IPpriv

Motivated by the desire to remove all congestion controlled links within the anonymizing overlay, Kiraly *et al.* [142, 143] address the performance anomalies of contemporary low latency anonymity networks by designing an anonymizing technique that operates at the network layer, rather than the transport layer. Their solution employs an overlay of IPsec [139] onion routers and they design a telescoping session key establishment procedure that leverages standards such as ESP [140] and IKEv2 [70]. Experiments show that this network layer approach to anonymity offers improved performance in terms of faster web download times relative to Tor. By removing all congestion control loops within the overlay, this solution relies only on the single end-to-end congestion control loop provided by the TCP connection between the client and destination. Ignoring the cryptography necessary to implement the onion routing, this solution functions in a manner similar to traditional IP.

By tunneling end-to-end TCP connections over multiple IPsec routers using layered encryption, this approach achieves the same protection for the source and destination's addresses in the IP header as Tor. While anonymity is provided to the network layer, this solution preserves the endpoints' transport headers, revealing information including source and destination ports, sequence numbers, and flags (for TCP). Furthermore, information revealed by the transport layer can be used to profile the client, by employing passive operating system fingerprinting [173], TCP/IP stack fingerprinting [32], and clock-skew fingerprinting [144, 164]. Such information could aid in conducting traffic analysis and deanonymizing clients. Also, unreliable transport enables a new and inexpensive traffic confirmation attack that works as follows. A malicious exit router drops cells and – if it can watch a set of clients, either by controlling several entry guards nodes or by watching the network traffic between a set of clients and the Tor network – can correlate exit traffic with clients by identifying which client re-transmits the dropped cells. While unreliable transport may offer desirable performance properties, it also enables new attacks and side channels for information leakage.

2.3.5 UDP-OR

UDP-OR also endeavors to improve Tor's performance by eliminating Tor's pairwise TCP transport. Viecco [227] proposes an alternate architecture for low latency anonymity networks where all overlay routers communicate over connection-less, best effort UDP transport; reliability, in-order packet delivery, and congestion control are handled by the endpoints on a per-circuit basis. In this design, the client's TCP packets are encapsulated over UDP and transported through a chosen circuit. Clients establish a TCP connection with the chosen exit router, which is also a SOCKS proxy, tunneled over the unreliable UDP entry and middle routers. By removing TCP from the overlay core, inter-circuit interference during congestion control is eliminated and the client's native TCP implementation provides congestion and flow control, as it would over an ordinary IP network.

TCP tunneled over an unreliable, best effort UDP transport offers the potential to remove undesirable transport layer interactions. However, this approach may have limitations. First, UDP link security is provided by an un-verified and non-standard security protocol designed by the author. Consequently, its implementation may be insecure. Furthermore, it is unclear if removing reliability guarantees from the overlay while still providing end-to-end reliability improves performance. Circuits with high latency (due to dropped packets in the unreliable overlay) may require superfluous re-transmissions at the end-points. Further experiments are necessary to understand the full performance implications of this design. Also, like the IPsec proposal, this design is vulnerable to client OS, TCP/IP stack, and clock-skew fingerprinting. However in this design, the client's TCP stack is exposed to the exit router that may be able to learn information about the client.

2.3.6 Freedom

The Freedom Network [51, 68] was a commercial anonymizing overlay developed by Zero Knowledge Systems, Inc. In addition to employing a decentralized architecture of anonymizing proxy servers, Freedom enabled users to create pseudonyms for each of their distinct types of online activity, to reduce the possibility for linkability across these activities. Freedom's proxy servers were operated by Zero Knowledge Systems and their partner organizations. Freedom clients select routes through the network of proxies, implement the cryptography needed to secure the routers, and manage pseudonyms. Freedom clients interface with the proxies by first intercepting the client's networking system calls (*e.g.*, `send()`, `sendto()`, `recv()`, `recvfrom()`, etc.) and redirecting them to filters that sanitized application-layer data in addition to identifying features of the IP, TCP, or UDP headers. Also, the anonymizing proxies transport encrypted IP packets using a custom kernel module.

2.3.7 HerbivoreFS

Starting with Chaum's DC-Nets design, an anonymity-preserving file sharing system based upon a peer-to-peer model is proposed called HerbivoreFS [210]. To improve the practicality of

Chaum's original design, they make the following modifications. First, to eliminate the coin tosses, they use a cryptographically secure pseudo-random number generator to produce the bit stream (or key). Each participant now simply shares their seed value and uses these seeds to produce each other participant's bit stream. Next, HerbivoreFS assumes a fully connected graph and eliminates the third-party mediator using a broadcast network. Finally, they assume that participants can only transmit at an assigned time slot – this eliminates the confusion caused when multiple participants transmit at once.

HerbivoreFS is able to scale to many participants by partitioning the entire network into smaller anonymizing cliques. Each clique can communicate efficiently within itself, and is connected to the other cliques using a Pastry distributed hash table (DHT) [192]. A prototype is implemented and evaluated on the PlanetLab testbed. The results indicate that the expected throughput is dependent upon the size of the cliques. To highlight their results, for small cliques of 10 nodes, HerbivorFS achieves throughput of over 200 Kb/s when there is a single sender, and nearly 150 Kb/s when there are four senders. As the clique grows to 40 nodes, a single sender network achieves just under 100 Kb/s and a four sender network achieves over 50 Kb/s. These results are encouraging and represent the first and only real anonymous communications system based upon the DC-Net design. Clearly there is an inherent trade-off between the strength of the anonymity that can be provided and the level of performance that can be maintained.

2.3.8 Anonymous Remailers

High latency anonymous communication techniques have bean popular for asynchronous communications such as e-mail. The Cypherpunks remailer is a Type I anonymous remailer [83]. To send an anonymous message with the remailer, the user retrieves the remailer's public key and imports the public into a PGP [71] software implementation such as GPG [19]. The message is encrypted using the PGP algorithm and sent to the Cypherpunk remailer. The remailer, upon receipt of a message, decrypts the message with its private key and then forwards the message to the intended recipient named in the `anon-to:` field of the SMTP body. A special STMP field called `latent time` tells the remailer how long to hold the message before delivering it.

Type I remailers are built on a centralized architecture where the remailer knows the message's sender and receiver. A decentralized architecture is more desirable so that no single entity can de-anonoymize the message. Mixmaster is a Type II anonymous remailer that sends messages in fixed-size packets through a Chaumian mix network [163]. Mixmaster does not allow anonymous replies nor does it offer any mechanism to try to mitigate abuse. Mixminion, a Type III anonymous remailer, offers anonymous reply blocks, forward security by using TLS-protected SMTP with ephemeral keys for each message, replay protection and key rotation, exit policies to allow mixminion node operators to curb abusive or hateful anonymous messages, integrated directory servers to organize mixminion nodes, and dummy traffic to further mitigate traffic analysis vulnerabilities [87].

2.3.9 P^5

The Peer-to-Peer Personal Privacy Protocol, or P^5, provides sender, receiver, and sender-receiver anonymity [204]. P^5 is based on a broadcast channel to which participants can send messages to all other participants, such as a peer-to-peer ring topology. Nodes wishing to send a message broadcast their message and nodes that have no message to send broadcast a dummy message. Sender anonymity is provided since all messages to a particular receiver come from previous hop node, so the receiver does not know the true identity of the sender. Receiver anonymity is provided, too, since the sender does not know the receiver is located in the broadcast channel. Broadcasting messages around a ring-like topology is very inefficient. P^5 instead creates a hierarchy of progressively smaller broadcast channels and allows each individual participant to choose their position in the hierarchical broadcast channel, thereby choosing to trade-off between anonymity and communication efficiency.

2.3.10 Nonesuch

Nonesuch is a high latency mix network that allows senders to submit messages obliviously by embedding steganographically encoded messages within images that are posted to newsgroups or social networking pages [125]. The system provides the same degree of anonymity and sender-receiver unlinkability as mix cascades, but it also provides a stronger degree of sender anonymity. The protocol uses the Minx packet format [88]. To submit a message to Nonesuch, the user chooses a route through the mix network and creates a layered onion using a key for each mix in the path. The sender next steganographically encodes the completed packet within an image file and posts the file to a new group or social network site. All non-steganographically encoded image files are regarded as cover traffic.

2.3.11 AP3

AP3 [162] is an anonymizing overlay network that provides anonymous message delivery, anonymous communication channels, and secure pseudonyms in a light-weight manner similar to Crowds [189]. The notion of anonymity provided by AP3 is "probable innocence," which means that an adversary cannot identify the sender of a message through the overlay with a probability greater than 0.5. Anonymous message delivery is achieved with the help of a set of overlay nodes that forward an initiator's message to other overlay nodes with a certain probability p_f before delivering the message to the destination. Anonymous communication channels can be constructed in this manner with each node in the forwarding chain remembering its previous and next hop nodes so that the channel is consistent for all of the initiator's messages. Participants can establish secure pseudonyms by generating their own public/private key pair for each session and signing messages with their private key.

2.3.12 Cashmere

Cashmere [239] is motivated by the observation that the failure of a single mix or onion router can result in data loss or high latency as the anonymous path is rebuilt. Built on a structured peer-to-peer overlay, Cashmere provides resilient anonymous routing by selecting regions of the overlay name space to act as a mix, rather than single mix nodes.

2.3.13 Salsa

Many anonymity systems require centralized points of trust and full knowledge of the routing information, which limits scalability. Salsa [169] attempts to distribute trust and improve scalability by giving each user a partial view of the network by using a distributed hash table routing mechanism. Using a virtual tree structure, nodes do not need global knowledge to route look ups.

2.3.14 Java Anonymous Proxy (JAP)

JAP is an anonymizing network built on a low latency mix cascade design [134]. Users route their traffic through a fixed set of mix nodes which remove layers of encryption and finally forward the data to its intended destination. Unlike traditional mixes, these mixes do not attempt to perturb timing information to frustrate traffic analysis. As a result of the mix cascade design, users must trust each mix operator not to keep logs or disclose information to third-parties such as law enforcement or governments. Today, there are two JAP variants. JonDo [23] is a commercial mix cascade and AN.ON is freely available.

2.3.15 Freenet

Freenet [80] provides an anonymous publishing, replication, and retrieval service using a peer-to-peer storage model. To obtain data, a user computes a hash of a descriptive string to obtain the file's look up key and sends the look up key to the nearest node in its routing table. Once the requested data is found, it is sent back through each of the requesting intermediate nodes until it reaches the original requester. Anonymity for both data requesters and providers is provided because no single node knows whether the request came from the previous hop or another node several hops away. Similarly, it is unclear which node fulfilled the data request, since the data could have been provided by the previous hop or a node several hops aways.

2.3.16 Privacy-preserving File Sharing Protocols

Peer-to-peer file sharing protocols such as BitTorrent provide fast and efficient content dissemination, since the available bandwidth to share files scales with the number of peers. However, to enable efficient peer discovery, these protocols typically operate by publicly advertising all participating peers' network addresses. This has led to a climate of large-scale surveillance of peer-to-peer file sharing networks by copyright investigators [180, 182, 208].

To improve privacy in peer-to-peer file sharing networks, Friend-to-Friend (F2F) networks have been proposed, with systems such as Turtle [181] and OneSwarm [130]. F2F networks leverage existing trust relationships among peers to reduce the possibility of a peer participating for the sole purpose of monitoring the other peers' activity. In Chapter 6, we offer an alternate technique to improve privacy for peer-to-peer file sharers: we propose an anonymizing ad-hoc relay network similar to Crowds, but specifically designed for BitTorrent, called BitBlender. BitBlender offers a degree of plausible deniability for the set of peers returned through any of BitTorrent's peer discovery mechanisms. Choffnes *et al.* offer a similar approach, where peers participate in random file sharing swarms in an effort to hide their real file sharing behaviors [78].

2.4 Anonymity Metrics

Chaum introduces the concept of an *anonymity set* in his security analysis of DC-Nets [76]. Even though individual participants cannot be directly identified as having sent a message in this framework, the size of the anonymity set gives the number of other participants with whom the sender may be confused, depending on the attacker's knowledge of a subset of the participants' keys. In an attempt to formalize this concept, Pfitzmann and Hansen propose the following definition of anonymity: *"Anonymity is the state of being not identifiable within a set of subjects, the anonymity set"* [179]. They further qualify this definition by arguing that anonymity is *stronger* when the anonymity set is larger *and* the sending and receiving of messages is evenly distributed across the subjects within the set. In general, this implies that the more *uniform* the messages are across

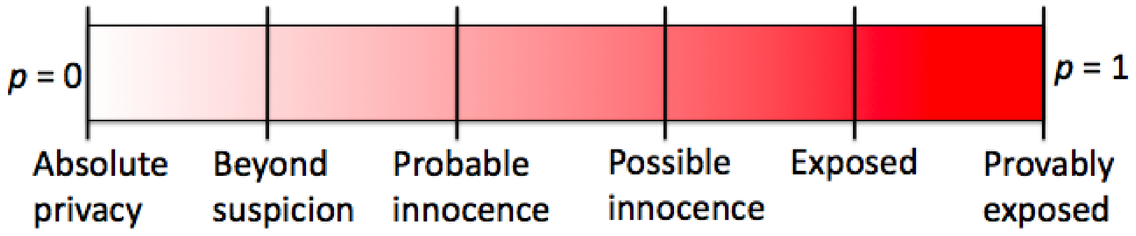


Figure 2.3: Degrees of anonymity expressed as a spectrum, given the adversary's probability p of knowing that a subject had a role in a message

the participants of the system, the stronger the level of anonymity provided. Thus, understanding the different probabilities of various participants in the anonymity set of having sent or received a message gives an adversary additional useful information that can be used to profile users beyond the anonymity set alone.

2.4.1 Degrees of Anonymity

Reiter and Rubin present a notion of anonymity where an attacker knows that a user sent a message with a certain probability p [189]. The degree of anonymity is defined as $1 - p$. The possible p values are regarded as a spectrum ranging from absolutely privacy ($p = 0$) to provably exposed ($p = 1$) as shown in Figure 2.3. The endpoints of the spectrum correspond to the case where an adversary has no information about a subject's role in a particular message and where an adversary has absolute confidence that they have identified a subject, respectively. The remaining points along the spectrum are more subjectively defined. "Beyond suspicion" means that it is very improbable that the subject had a role in a message. "Probable innocence" means that the subject most likely had no role in a message. The space between "probable innocence" and "possible innocence" corresponds to the case of a subject having an equal chance of having a role or not in a message (*i.e.*, $p = 0.5$). The "exposed" state implies that the subject has a high likelihood of having had a role in a message. In Chapter 6, we present the design, implementation, and analysis of an anonymizing system designed specifically for BitTorrent that derives its anonymity from Reiter and Rubin's notion of degrees of anonymity.

While expressing degrees of anonymity in this manner provides a convenient mapping between an adversary's probabilistic confidence p in having identified a subject and a qualitative description, it does not provide useful information about *how indistinguishable* users are from one another within the anonymity set. We next discuss an alternative metrics that provides a quantifiable anonymity measure.

2.4.2 An Information-theoretic Approach

In high-latency anonymity systems, a global attacker endeavors to reduce the size of the anonymity set of participants to a probability distribution that has low entropy. First, we first describe the metric that considers the state of the anonymous communication system from the perspective of a single message or single user. Next, we explain how anonymity can be quantified from a whole-network perspective.

2.4.2.1 Entropy

Suppose that each participant in the anonymity set has a probability distribution associated with having sent or received a particular message. More formally, let Ψ be the set of users in the anonymity set and let $r \in R$ be a role for the user, where $R = \{\text{sender}, \text{receiver}\}$. Let U be the attacker's probability distribution of the users $u \in \Psi$ having a role r with regard to a message m such that $\sum_{u \in \Psi} U(u, r) = 1$. U may assign a probability of 0 to certain users, if there was no chance that they had a role in m, or U may assign a non-zero probability to certain users if there is evidence that they had a role in m. Serjantov and Danezis define the effective size of the probability distribution S as the classical entropy of the distribution [195]:

$$S = -\sum_{u \in \Psi} p_u \log_2(p_u) \tag{2.1}$$

for $p_u = U(u, r)$.

The common interpretation of this metric is the number of additional bits of information that the attacker needs in order to identify the user. If $S = 0$, then the system provides no anonymity.

This corresponds to the case where some user $u \in U$ has a probability of 1 of having a role in the message. Also, note that S is bounded by the logarithm of the set size such that $0 \leq S \leq \log_2 |\Psi|$. Finally, the ideal entropy which corresponds to the probability distribution being uniform over the set of users is $S = \log_2 |\Psi|$. This knowledge provides a quantifiable scale to express the degree of anonymity provided by a system in a particular configuration of users and messages.

Diaz *et al.* propose a similar metric for quantifying anonymity that is normalized by the maximum entropy possible for a system configuration:

$$S_{norm} = -\frac{\sum_{u \in \Psi} p_u \log_2(p_u)}{\log_2 |\Psi|} \tag{2.2}$$

This provides an anonymity measure relative to the maximum possible anonymity, which corresponds to the case where all users have an equal chance of having a role in m. While entropy metrics are not typically applied to low-latency systems, in Chapter 4 we propose an information-theoretic metric that describes the degree of non-uniformity in router selection in Tor-like networks.

2.4.2.2 The Limitations of Entropy

Entropy and normalized entropy capture a system's anonymity with regard to a single message or user, but not to the anonymity provided system as a whole. Edman *et al.* propose a metric that quantifies the degree of anonymity offered to all users of an anonymous communications system based on the permanent of a matrix [110]. This measures the amount of information needed by an observer to reveal the global communication patterns between both senders and receivers in an anonymity network.

2.4.3 Metrics for Low-latency Systems

In low-latency systems, entropy is generally not applied because attacks against such systems either succeed (reducing the entropy to zero) or fail (with a non-zero entropy). Instead, low-latency systems typically measure their security as the probability that an adversary occupies the right positions on a path to compromise security, *e.g.*, the path's endpoints. In the original onion routing design that assumed uniform router selection, the probability is approximated as $(c/n)^2$, where there are c corrupt onion routers in a network of n nodes [122]. In Chapter 4, we show that for modern onion routing systems like Tor that select routers with non-uniform probabilities, this analytical security model under-estimates the attack's probability of success. We adopt an empirical approach to measuring Tor's security by implementing attacks and observing the fraction of paths that are compromised. Subsequently, similar empirical approaches have been adopted by Murdoch and Watson [167] and Snader and Borisov [211].

2.4.3.1 Measures of Bias in Router Selection

The manner in which routers are selected in low-latency systems can dramatically impact the system's security properties. Modern systems such as Tor have a large and diverse set of volunteer routers, making it necessary to balance the traffic load over the heterogeneous distribution of routers and bandwidth that exists in the network. Routers are chosen with a bias toward those that appear to offer higher bandwidth capacities to provide traffic load balancing.

Since there exists bias in the router selection process, it is desirable to express some notion of system's security as a function of the bias in router selection. In Chapter 4, we propose an information theoretic measure that applies Shannon's entropy over the router selection probability distribution and normalizes this value by the maximal (optimal) entropy for the system in a manner similar to Diaz *et al.* [94] (defined in Equation 2.2). This metric captures the degree to which the router selection probability distribution is uniform or skewed. A value of $S_{norm} = 1$ implies that routers are selected uniformly at random and values $S_{norm} < 1$ imply a bias. There is an inverse relationship between S_{norm} and the degree of selection bias.

To measure inequality in router selection, Snader and Borisov [211] adopt the Gini coefficient [119], an equality metric commonly used in the field of economics to express the distribution of wealth. The Gini coefficient is defined as the ratio between the line of perfect equality and an empirical cumulative distribution function of router selection. Mathematically, this is expressed as:

$$G = \frac{1}{\mu} \int_0^\infty CDF(x)(1 - CDF(x))\, dx \tag{2.3}$$

where μ is the mean of the cumulative distribution function and $CDF(x)$ is the observed cumulative distribution function of router selection. A Gini coefficient $G = 0$ implies perfectly equal router selection, *i.e.*, router selection with a uniformly random probability. $G = 1$ implies perfect *in*equality, or the same single router is deterministically chosen always.

2.5 Anonymity Attacks

We now turn our attention toward understanding how anonymous communication systems can be attacked and caused to fail.

2.5.1 Traffic Analysis with Packet Sizes and Timing

Despite even the best security practices, it is often possible to use side channel information such as packet sizes, counts, and timing information to identify the underlying application-layer protocol, and even the content, in some cases. Statistical and machine learning techniques are commonly applied to identify applications and infer users' behavior. Such behaviors can leak information about who the user is and what they are doing. Examples of the types of information that may be inferred by a third-party eavesdropper from observed secure connections include videos watched [193], passwords typed [215], web pages viewed [126, 149, 217], languages and phrases spoken [232, 233], and applications run [234].

2.5.2 Packet Counting and Timing Analysis Attacks

It is well understood that there exists a fundamental trade-off between security and performance. In Back *et al.* [52], several threats to low-latency anonymous communications systems are enumerated. Suppose that an anonymous communication system is modeled as a black-box. An adversary can see packets entering the network and packets leaving the network. To link a sender with a receiver, an adversary notes that it saw j packets from client x and a short time-period later, saw precisely j packets leaving the network being delivered to y. This is called a *packet counting* attack. The adversary can conclude with certain confidence that x and y are communicating. Suppose that the entrance and exit nodes of an onion routing network, for instance, are used infrequently, then the adversary can be nearly certain that x and y are communicating, since there are little or no other possible initiator/destination pairs during that time period. For instance, a similar technique has been used to locate hidden services within Tor [172]. Traffic shaping methods to ensure that all nodes have a constant traffic rate help to mitigate this vulnerability.

Another fundamental traffic analysis technique described by Back *et al.* is the latency attack. Suppose that an adversary measures the latency through every possible path in an anonymity network. Then, if an initiator contacts a destination server that is controlled by the adversary, then the adversary can measure the observed latency for this session and make a reasonable guess about the full path of routers that is being used. With the full path, it may be possible to find the initiator. One could envision an adversary with access to latency measurements between the nodes (similar to the King data set [118]) in the anonymity network to estimate the end-to-end latency of the anonymous paths. Hopper *et al.* used a network coordinate system to determine how much information is leaked by latency knowledge in Tor [127]. However, as the size of the anonymity network grows, the number of false positives, *i.e.*, the number of paths that have indistinguishable end-to-end latencies, may increase.

The following variant of the latency attack is also proposed. Suppose that the adversary has latency information for paths in the network. Next, if the adversary induces load on a particular path – perhaps by issuing a flood of requests through a path – the effect of this flood of requests would be detectable at the destination by a spike in latency, if this is the client's path. This is called the *clogging* attack. This attack has been successfully implemented on the early Tor network [166] and more recently on the mature Tor network [113].

These clever attacks all exploit the low-latency requirement of certain anonymous communication systems. In general, all low latency anonymity systems are vulnerable to some extent to the attacks outlined above. For a comprehensive discussion and analysis of additional attacks against specific systems, see Wright *et al.* [235].

2.5.3 Predecessor Attack

This attack occurs in a multi-hop network when an adversary can infer that the previous node is the source of an anonymous communication [236]. For example, Crowds is vulnerable to a set of timing attacks where a Jondo node can determine if the previous Jondo on the path is the initiator of a request based upon an analysis of the time that elapses until the request is fulfilled. If the time is sufficiently small, then the intermediate node can conclude with certain confidence that the preceding node is the initiator. Over time, an adversary's confidence increases and the initiator is identified.

Tor is trivially vulnerable to the predecessor attack. A malicious router can attempt to enumerate all clients by simply observing their connection's previous hop and comparing it to the list of all known Tor routers obtained by the trusted directory servers. Tor mitigates the risk of predecessor attacks by using entry guards (see Chapter 2.3.3.2). Entry guards ensure that if a client chooses a malicious entry guard, their first-hop is always compromised. However, for clients that do not have this misfortune, their first-hop is never compromised.

2.5.4 Disclosure, Intersection, and Statistical Disclosure Attacks

Suppose that an adversary observes the messages coming in and out of a mix and wishes to identify a sender and the corresponding receiver for a message. Suppose that the initiator sends messages to precisely n distinct receivers. The adversary can observe precisely n mutually disjoint sets of recipients (one for each of the receivers). Each set contains exactly one of the sender's communication endpoints. This is referred to as the *disclosure attack* [141].

The adversary can isolate the receiver in each set by observing new sets by finding the one receiver that is in the intersection of these sets. The attack may proceed in multiple rounds, where each round reduces the size of this intersection until only one receiver remains – the true receiver of the message. This is known as the *intersection attack* [64, 90].

The disclosure attack requires the adversary to find n mutually disjoint sets. Kesdogan *et al.* show that this problem can be reduced to the binary Constraint Satisfaction Problem, which is NP-complete [141]. Danezis relaxes the requirements of the disclosure attack by proposing the *statistical disclosure attack*, where an adversary's goal is to infer the sender's most likely recipients [85].

2.5.5 Onion Routing Attacks

Onion routing networks have the ability to provide better performance in terms of higher throughput and lower latency than its predecessor the mix network since onion routing does not re-order or delay messages. Consequently, onion routing networks are more vulnerable to attacks by individual onion router operators. In particular, there are three cases of compromise to consider [219]. Suppose that there are r total routers in the network and $c < r$ of the routers are compromised. For the remainder of this discussion, assume that routers are chosen uniformly at random. First, suppose that the first router on a client's circuit is malicious. In this case, the adversary can profile the client's behavior, but cannot examine the contents of its messages, since the client pre-encrypts its outgoing traffic. However, despite message confidentiality, it is possible for a malicious entry router to conduct any one of a variety of traffic analysis attacks using the number of packets sent and their timings to infer information about the user's application-layer behavior [149, 193, 215, 217, 232, 234]. The probability of an adversary compromising the first router on a circuit is given by c/r.

Next, suppose that the adversary compromises the final onion router on a circuit. In this case, the adversary removes the last layer of encryption and forwards the payload to the indented destination. The adversary now knows the identity of the destination as well as any potentially identifying information about the initiator that may be contained within the payload. For instance, in Chapter 3 we note that Tor clients are at risk of inadvertently revealing identifying information based on our observations of the relatively high volume of insecure (*i.e.*, non-TLS) traffic that flows through exit routers [158]. For example, instant messaging protocols – such as the popular AIM – send user names in plain-text and popular websites such as myspace do not use TLS for their login process, so user names and passwords are revealed. The probability of this case – again assuming uniform router selection – is c/r.

Finally, assume the worst-case, where the adversary has compromised both the beginning and the end of a circuit. In this case, the adversary knows the identities of both the initiator and the destination, in addition to the contents of the communication session. The probability of this type of compromise is negligible – $(c/r)^2$ – assuming uniform router selection. However, if there exists a bias in the router selection method, then it's possible to exploit this bias and compromise significantly more circuits, as we demonstrate in Chapter 3. Further analysis of onion routing security can be found in Syverson *et al.* [219].

2.5.5.1 Security of Tor's Path Selection

Murdoch and Watson [167] extend the analysis of our low resource routing attack on Tor presented in Chapter 4 to analyze how the attack performs when the routing algorithm changes. They consider the routing strategy proposed by Snader and Borisov where the degree of bias in the selection is a parameter that can be tuned by the user [211] in comparison to Tor's default bandwidth-weighted algorithm along security and performance dimensions. Snader and Borisov's algorithm works as follows: if a user requires the strongest anonymity, they would want to choose routers uniformly at random. However, if anonymity is only a minor concern, they could skew the selection process even more toward routers that appear to have higher bandwidth – though, at the risk of a greater possibility of circuit compromise. More formally, the algorithm chooses routers using a family of functions defined as follows:

$$f(s,x) = \begin{cases} \frac{1-2^{sx}}{1-2^{s}}, & \text{if } s \neq 0 \\ x, & \text{otherwise} \end{cases} \tag{2.4}$$

where s is the tunable selection parameter and $x \in [0,1)$ is a random number drawn from a uniform distribution. Supposing that there are n routers stored in a list ordered by reported bandwidth, the index of a router to select is given by $\lfloor n \times f(s,x) \rfloor$. For instance, when $s = 0$, nodes are selected uniformly at random and as s increases, nodes with higher reported bandwidths will be selected more often. But even at the higher end, there still exists non-determinism in the selection process; at $s = 10$, the highest bandwidth router is chosen only 6% of the time.

Murdoch and Watson simulate Tor's current router selection algorithm and the new algorithm of Snader and Borisov at various s values to analyze the routing algorithm's impact on anonymity. They consider the fraction of circuits compromised across several simulations as their anonymity metric (*i.e.*, the fraction of circuits where a malicious router appears at both the beginning and the end of the circuit). They also present an analysis of the routing algorithm's effect on expected performance using models derived from queuing theory. A surprising result is presented: in addition to providing better performance over uniform router selection, Tor's current bandwidth-weighted selection offers improved anonymity in the presence of a botnet adversary (*i.e.*, one that has access to a large number of low bandwidth nodes). This apparent vulnerability of the uniform path selection algorithm challenges the conventional wisdom that uniform selection is always the most secure path selection strategy.

2.6 Summary

We have presented a detailed overview of the significant research within the field of anonymous communications. In particular, we provided an overview of the three most common and fundamental techniques for enabling two parties to communicate without the threat of traffic analysis. These include mix networks, DC-nets, and onion routing. We next discussed their implementations in but a few of the significant anonymous systems that have been developed. Finally, we explored how anonymity is commonly understood and measured. Further background on the fundamental techniques, representative systems, and analytical tools from the field of anonymous communications can be found in Edman and Yener [112] and Danezis *et al.* [86].

Chapter 3

Characterizing a Popular Low Latency Anonymous Network

Tor is a popular privacy enhancing system that is designed to protect the privacy of Internet users from traffic analysis attacks launched by a non-global adversary [103]. Because Tor provides an anonymity service on top of TCP while maintaining relatively low latency and high throughput, it is ideal for interactive applications such as web browsing, file sharing, and instant messaging. Since its initial development, researchers have analyzed the system's performance [105,153,159,184,188,230] and security properties [55,61,67,111,113,121,127,150,161,164,166–168,172,183,211,237]. In this chapter, we characterize Tor, utilizing observations made by running a Tor router to answer the following questions:

How is Tor being used? We analyze application layer header data relayed through our router to determine the protocol distribution in the anonymity network. Our results show the types of applications currently used over Tor, a substantial amount of which is non-interactive traffic. We discover that web traffic makes up the vast majority of the connections through Tor, but BitTorrent traffic consumes a disproportionately large amount of the network's bandwidth. Perhaps surprisingly, protocols that transmit passwords in plain-text are fairly common, and we propose simple techniques that attempt to protect users from unknowingly disclosing such sensitive information over Tor.

How is Tor being *mis*-used? To explore how Tor is currently being misused, we examine both malicious router and client behaviors. Since insecure protocols are common in Tor, there is a potential for a malicious router to gather passwords by logging exit traffic. To understand this threat, we develop a method to detect when exit routers are logging traffic, under certain conditions. Using this method, we did, in fact, catch an exit router capturing POP3 traffic (a popular plain-text e-mail protocol) for the purpose of compromising accounts.

Running a router with the default exit policy provides insight into the variety of malicious activities that are tunneled trough Tor. For instance, hacking attempts, allegations of copyright infringement, and bot network control channels are fairly common forms of malicious traffic that can be observed through Tor.

Who is using Tor? In order to understand who uses Tor, we present the geopolitical distribution of the clients that were observed. Germany, China, and the United States appear to use Tor the most, but clients from 126 different countries were observed, which demonstrates Tor's global appeal. In addition, we provide a geopolitical breakdown of who participates in Tor as a router. Most Tor routers are from Germany and the United States, but Germany alone contributes nearly half of the network's total bandwidth. This indicates that implementing location diversity in Tor's routing mechanism is not possible with the current distribution of router resources.

How does Tor perform? We lastly present a circuit-level performance analysis based on measurements collected in 2007 and 2010. Our observations in 2007 indicate that Tor users experience high and variable delays and generally receive low throughput service. However, by 2010, these delays are reduced the throughput is increased. We hypothesize about the potential reasons for this apparent improvement in performance.

3.1 Data Collection Methodology

To better understand real world Tor usage, we set up a Tor router on a 1 Gb/s network link.[1] This router joined the live deployed Tor network during December 2007 and January 2008. This configuration allowed us to record a large amount of Tor traffic in short periods of time. While running, our node was consistently among the top 5% of routers in terms of bandwidth of the roughly 1,500 routers flagged as `Running` by the directory servers at any single point in time.

[1] Our router used `Tor` software version 0.1.2.18.

We understand that there are serious privacy concerns that must be addressed when collecting statistics from an anonymity network [207]. Tor is designed to resist traffic analysis from any single Tor router [103]; thus, the information we log — which includes *at most* 96 bytes of application-level data — cannot be used to link a sender with a receiver, in most cases. We considered the privacy implications carefully when choosing what information to log and what was too sensitive to store. In the end, we chose to log information from two sources: First, we altered the Tor router to log information about circuits that were established though our node and cells routed through our node. Second, we logged only enough data to capture up to the application-level protocol headers from the exit traffic that was relayed through our node.

In order to maximize the number of entry and exit connections that our router observed, it was necessary to run the router *twice*, with two distinct exit policies:[2] (1) Running with an *open exit policy* (the default exit policy[3]) enabled our router to observe numerous exit connections, and (2) *Prohibiting all exit traffic* allowed the router to observe a large number of clients.

Entrance/middle traffic logging. To collect data regarding Tor clients, we ran our router with a completely restricted exit policy (all exit traffic was blocked). We ran our Tor router in this configuration for 15 days from January 15–30, 2008. The router was compiled with minor modifications to support additional logging. Specifically, for every cell routed through our node, the time that it was received, the previous hop's IP address and TCP port number, the next hop's IP address and TCP port number, and the circuit identifier associated with the cell is logged.

Exit traffic logging. To collect data regarding traffic exiting the Tor network, we ran the Tor router for four days from December 15–19, 2007 with the default exit policy. For routers that allow exit traffic, the default policy is the most common. During this time, our router relayed approximately 709 GB of TCP traffic exiting the Tor network.

[2] Due to the relatively limited exit bandwidth that exists within Tor, when we ran the default exit policy, our node was chosen as the exit router most frequently on established circuits. As a result, in order to observe a large number of clients, it became necessary to collect data a second time with a completely restricted exit policy so that we would not be an exit router.

[3] The default exit policy blocks ports commonly associated with SMTP, peer-to-peer file sharing protocols, and ports with a high security risk.

Table 3.1: Exit traffic protocol distribution by number of TCP connections, size, and number of unique destination hosts

Protocol	**Connections**	**Bytes**	**Destinations**
HTTP	12,160,437 (92.45%)	411 GB (57.97%)	173,701 (46.01%)
SSL	534,666 (4.06%)	11 GB (1.55%)	7,247 (1.91%)
BitTorrent	438,395 (3.33%)	285 GB (40.20%)	194,675 (51.58%)
Instant Messaging	10,506 (0.08%)	735 MB (0.10%)	880 (0.23%)
E-Mail	7,611 (0.06%)	291 MB (0.04%)	389 (0.10%)
FTP	1,338 (0.01%)	792 MB (0.11%)	395 (0.10%)
Telnet	1,045 (0.01%)	110 MB (0.02%)	162 (0.04%)
Total	13,154,115	709 GB	377,449

In order to gather statistics about traffic leaving the network, we ran `tcpdump` on the same physical machine as our Tor router. Tcpdump was configured to capture only the first 150 bytes of a packet using the "snap length" option (`-s`). This limit was selected so that we could capture up to the application-level headers for protocol identification purposes. At most, we captured 96 bytes of application header data, since an Ethernet frame is 14 bytes long, an IP header is 20 bytes long, and a TCP header with no options is 20 bytes long. We used `ethereal` [16], another tool for protocol analysis and stateful packet inspection, in order to identify application-layer protocols. As a post-processing step, we filtered out packets with a source or destination IP address of any active router published during our collection period. This left only exit traffic.

3.2 Protocol Distribution

As part of this study, we observe and analyze the application-level protocols that exit our Tor node. We show in Table 3.1 that interactive protocols like HTTP make up the majority of the traffic, but non-interactive traffic consumes a disproportionate amount of the network's bandwidth. Finally, the data indicates that insecure protocols, such as those that transmit login credentials in plain-text, are used over Tor.

3.2.1 Interactive vs. Non-interactive Web Traffic

While HTTP traffic comprises an overwhelming majority of the connections observed, it is unclear whether this traffic is interactive web browsing or non-interactive downloading. In order to determine how much of the web traffic is non-interactive, we counted the number of HTTP connections that transferred over 1 MB of data. Only 3.5% of the connections observed were bulk transfers. The vast majority of web traffic is interactive.

3.2.2 Is Non-interactive Traffic Hurting Performance?

The designers of the Tor network have placed a great deal of emphasis on achieving low latency and reasonable throughput in order to allow interactive applications, such as web browsing, to take place within the network [103]. However, the most significant difference between viewing the protocol breakdown measured by the number of bytes in contrast to the number of TCP connections is that while HTTP accounted for an overwhelming majority of TCP connections, the BitTorrent protocol uses a disproportionately high amount of bandwidth.[4] This is not shocking, since BitTorrent is a peer-to-peer (P2P) protocol used to download large files.

Since the number of TCP connections shows that the majority of connections are HTTP requests, one might be led to believe that most clients are using the network as an anonymous HTTP proxy. However, the few clients that do use the network for P2P applications such as BitTorrent consume a significant amount of bandwidth. The designers of the network consider P2P traffic harmful, not for ethical or legal reasons, but simply because it makes the network less useful to those for whom it was designed. In an attempt to prevent the use of P2P programs within the network, the default exit policy blocks the standard file sharing TCP ports. But clearly, our observations show that port-based blocking strategies are easy to evade, as these protocols can be run on non-standard ports.

[4] Recall that our router's default exit policy does not favor any particular type of traffic. So the likelihood of observing any particular protocol is proportional to the usage of that protocol within the network and the number of other nodes supporting the default or a similar exit policy.

3.2.3 Insecure Protocols

Another surprising observation from the protocol statistics is that insecure protocols, or those that transmit login credentials in plain-text, are fairly common. While comprising a relatively low percentage of the total exit traffic observed, protocols such as POP, IMAP, Telnet, and FTP are particularly dangerous due to the ease at which an eavesdropping exit router can capture identifying information (i.e., user names and passwords). For example, during our observations, we saw 389 unique e-mail servers, which indicates that there were at least 389 clients using insecure e-mail protocols. In fact, only 7,247 total destination servers providing SSL/TLS were observed.

The ability to observe a significant number of user names and passwords is potentially devastating, but it gets worse: Tor multiplexes several TCP connections over the same circuit. Having observed identifying information, a malicious exit router can trace all traffic on the same circuit back to the client whose identifying information had been observed on that circuit. For instance, suppose that a client initiates both an SSL connection and an AIM connection at the same time. Since both connections use the same circuit (and consequently exit at the same router), the SSL connection can be easily associated with the client's identity leaked by the AIM protocol. Thus, tunneling insecure protocols over Tor presents a significant risk to the initiating client's anonymity.

To address this threat, a reasonable countermeasure is for Tor to explicitly block protocols such as POP, IMAP, Telnet, and FTP[5] using a simple port-based blocking strategy at the client's local socks proxy.[6] In response to these observations, Tor now supports two configuration options to (1) warn the user about the dangers of using Telnet, POP2/3, and IMAP over Tor, and (2) block these insecure protocols using a port-based strategy [59].

However, this same type of information leakage is certainly possible over HTTP, for instance, so additional effort must also be focused on enhancing Tor's HTTP proxy to mitigate the amount of

[5] Anonymous FTP may account for a significant portion of FTP exit traffic and does not reveal any information about the initiating client. Therefore, blocking FTP may be unnecessary.

[6] Port-based blocking is easy to evade, but it would protect naive users from mistakenly disclosing their sensitive information.

sensitive information that can be exchanged over insecure HTTP. For instance, a rule-based system could be designed to filter common websites with insecure logins.

Finally, protocols that commonly leak identifying information should not be multiplexed over the same circuit with other non-identifying traffic. For example, HTTP and instant messaging protocols should use separate and dedicated circuits so that any identifying information disclosed through these protocols is not linked with other circuits transporting more secure protocols.

3.3 Malicious Router Behavior

Given the relatively large amount of insecure traffic that can be observed through Tor, there is great incentive for malicious parties to attempt to log sensitive information as it exits the network. In fact, others have used Tor to collect a large number of user names and passwords, some of which provided access to the computer systems of embassies and large corporations [42].

In addition to capturing sensitive exit traffic, a Tor router can modify the decrypted contents of a message entering or leaving the network. Indeed, in the past, routers have been caught modifying traffic (i.e., injecting advertisements or performing man-in-the-middle attacks) in transit, and techniques have been developed to detect this behavior [176].

We present a simple method for detecting exit router logging under certain conditions. We suspect — and confirm this suspicion using our logging detection technique — that insecure protocols are targeted for the specific purpose of capturing user names and passwords.

3.3.1 Detection Methodology

At a high level, the malicious exit router logging detection technique relies upon the assumption that the exit router is running a packet sniffer on its local network. Since packet sniffers such as `tcpdump` are often configured to perform reverse DNS queries on the IP addresses that they observe, if one controls the authoritative DNS server for a specific set of IP addresses, it is possible to trace reverse DNS queries back to the exit node that issued the query.

More specifically, the detection method works as follows:

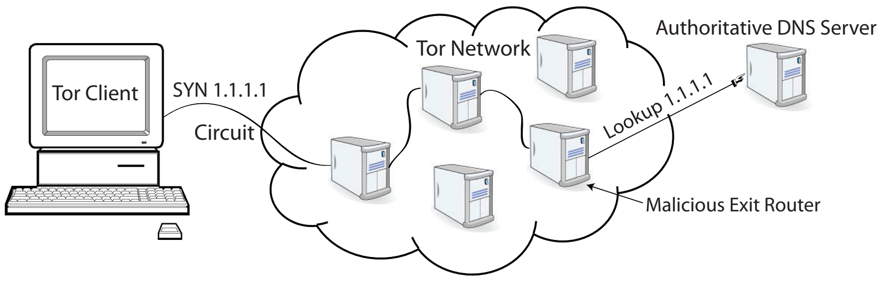


Figure 3.1: Malicious exit router logging detection technique

(1) We run an authoritative domain name server (DNS) that maps domain names to a vacant block of IP addresses that we control.

(2) Using a Tor client, a circuit is established using each individual exit router.

(3) Having established a circuit, a `SYN` ping is sent to one of the IP addresses for which we provide domain name resolution.

This procedure (shown in Figure 3.1) is repeated for each exit router. Since the IP address does not actually exist, then it is very unlikely that there will be any transient reverse DNS queries. However, if one of the exit routers we used is logging this traffic, they may perform a reverse DNS look-up of the IP address that was contacted. In particular, we made an effort to direct the `SYN` ping at ports where insecure protocols typically run (ports 21, 23, 110, and 143).

3.3.2 Results

Using the procedure described above, over the course of only one day, we found one exit router that issued a reverse DNS query immediately after transporting our client's traffic. Upon further inspection, by `SYN` ping scanning all low ports (1-1024), we found that only port 110 triggered the reverse DNS query. Thus, this router only logged traffic on this port, which is the default port for POP3, a plain-text e-mail protocol. We suspect that this port was targeted for the specific purpose of capturing user names and passwords.

Further improvements on this logging detection could be made by using a honeypot approach and sending unique user name and password pairs through each exit router. The honeypot could detect any login attempts that may occur. This method would find the most malicious variety of exit router logging. In fact, upon detecting the logging exit router (using the method described above), we also used this honeypot technique and observed failed login attempts from the malicious IP address shortly after observing the logging.

These results reinforce the need to mitigate the use of protocols that provide login credentials in plain-text over Tor. Given the ease at which insecure protocols can be captured and the relative ease at which they could be blocked, it is a reasonable solution to block their default ports.

3.3.3 Discussion

This approach to detecting exit router logging has limitations. First, it can only trace the reverse DNS query back to the exit router's DNS server, not to the router itself. To complicate matters more, there exist free domain name resolution services (such as OpenDNS [171]) that provide somewhat anonymous name resolution for any host on the Internet. If one assumes that the exit router is logging and performing reverse DNS queries in real-time, then it is easy to correlate reverse DNS queries with exit routers using timing information.

If reverse DNS is *not* performed in real-time, then more sophisticated techniques for finding the malicious exit router are required. For instance, if one controls the domain name resolution for several IP addresses, then it is possible to embed a unique pattern in the order of the `SYN` pings to different IPs through each exit router. This order will be preserved in the exit router's queries and can be used to determine the exit router that logged the traffic. Here we can leverage many of the same principles as explored in [65, 205].

The detection method presented makes the key assumption that the logging process will trigger reverse-DNS queries. However, this is not always the case. For example, exit routers that transport traffic at high bandwidth cannot feasibly perform reverse DNS queries in real-time. Also, this technique can be evaded simply by not performing reverse DNS when logging.

3.4 Misbehaving Clients

While Tor provides an invaluable service to protecting online privacy, over the course of operating a Tor router with the default exit policy, we learned about a wide variety of malicious client behavior. Since we are forwarding traffic on behalf of Tor users, our router's IP address appears to be the source of sometimes malicious traffic. The large amount of exit bandwidth that we provided caused us to receive a large number of complaints ranging from DMCA §512 notices related to allegations of copyright infringement, reported hacking attempts, IRC bot network controls, and web page defacement. However, an enormous amount of malicious client activity was likely unreported.

As a consequence of this malicious client behavior, it becomes more difficult to operate exit routers. For instance, our institution's administration requested that we stop running our node shortly after the data for this chapter was collected. Similar accounts of administrative and law enforcement attempts to prevent Tor use are becoming more common as Tor becomes more popular to the masses [73]. The Electronic Frontier Foundation (EFF), a group that works to protect online rights, has provided template letters [34] and offered to provide assistance [96] to Tor router operators that have received DMCA take-down notices.

One solution to our problems could have been to change our router's exit policy to reject all exit traffic, or specific ports (such as port 80) that generate a large portion of the complaints. However, this is not practical, since Tor requires a certain amount of exit bandwidth to function correctly. Another solution is to provide a mechanism for anonymous IP address blocking, such as Nymble [135]. Our first-hand observations with misbehaving clients reinforces the need to further study anonymous IP address blocking mechanisms.

3.5 Geopolitical Client and Router Distributions

As part of this study, we investigate where Tor clients and routers are located geo-politically. Recall that a client's IP address is visible to a router when that router is used as the entrance

Table 3.2: Geopolitical client distributions, router distributions, and the ratio of Tor users relative to Internet users

Client Distribution		**Router Distribution**	
Country	*Total*	*Country*	*Total*
Germany	2,304	Germany	374
China	988	United States	326
United States	864	France	69
Italy	254	China	40
Turkey	221	Italy	36
United Kingdom	170	Netherlands	35
Japan	155	Sweden	35
France	150	Finland	25
Russia	146	Austria	24
Brazil	134	United Kingdom	24

Relative Tor Usage	
Country	*Ratio*
Germany	7.73
Turkey	2.47
Italy	1.37
Russia	0.89
China	0.84
France	0.77
United Kingdom	0.75
United States	0.62
Brazil	0.56
Japan	0.32

node on the client's circuit through the Tor network. In the current Tor implementation, only particular routers, called *entry guards*, may be used for the first hop of a client's circuit. A router is labeled as an entry guard by the authoritative directory servers. All Tor router IP addresses are maintained by the directory servers, and we keep track of the router IP addresses by simply polling the directory servers periodically.

In order to map an IP address to its corresponding country of origin, we query the authorities responsible for assigning IP blocks to individual countries [3, 5, 7, 24, 35]. In order to determine the geopolitical distribution of Tor usage throughout the world, we aggregate IP addresses by country, and present the client and router location distributions observed during the January 2008 data collection period.

3.5.1 Observations

In this section, we present our observations regarding the client and router location distributions.

Client geo-political distribution. During a one day period when our Tor router was marked as an entry guard by the authoritative directory servers, it observed 7,571 unique clients.[7] As depicted in Table 3.2, the vast majority of clients originated in Germany, with China and the United States providing the next largest number of clients. Perhaps the most interesting observation about the client distribution is that Tor has a global user base. While most of the clients are from three countries, during the course of the entire 15 day observation period, clients were observed from 126 countries around the world, many of which have well-known policies of Internet censorship.

To put these raw geopolitical client distributions into perspective, Table 3.2 includes a ratio of the percentage of Tor users to the percentage of Internet users by country, using data on the distribution of broadband Internet users by country [128]. These percentages were computed by dividing the total number of Tor clients located in each country by the total number of Tor clients we observed, which provides the percentage of Tor users located in each country. For example, the relative Tor usage for Germany is computed as follows: The percentage of the total Internet users who are from Germany is 3.9% and according to our client observations, Germany makes up 2,304 of the 7,571 total Tor clients, which is 30.4%. Thus, the ratio of Tor users to Internet users in Germany is 7.73.

These ratios show that Tor is disproportionately popular in Germany, Turkey, and Italy with respect the the number of broadband Internet users located in these countries. It is unclear why there is such a large scale adoption of Tor in these specific countries, relative to Tor usage in other countries. An investigation of the possible technological, sociological, and political factors in these countries that are causing this might be an enlightening area of research.

Examining the number of clients that utilized our router as their entry router when *it was not marked as an entry guard* provides insight into the approximate number of clients that are using a significantly old version of the Tor client software. Specifically, this indicates that these clients are using a version *before* entry guards were introduced in `Tor` version 0.1.1.20 (May 2006). Over four

[7] We assume that each unique IP address is a unique client. However, dynamic IP addresses or network address translators (NATs) may be used in some places.

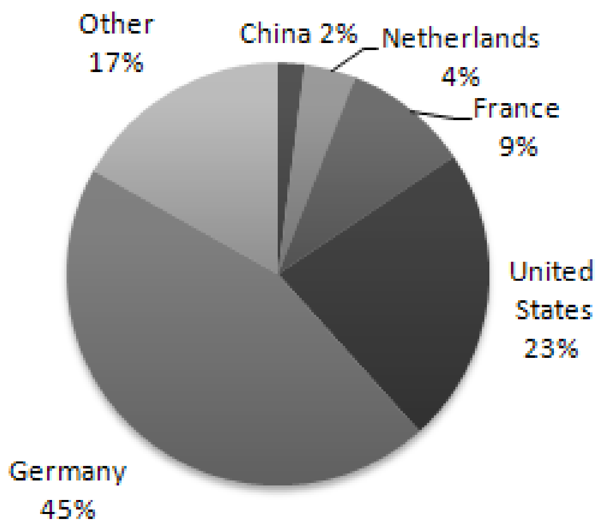
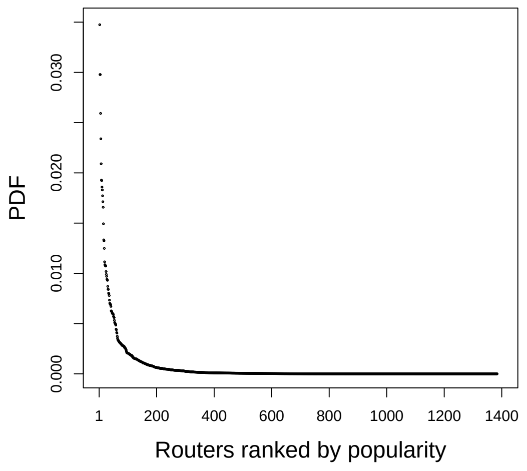


Figure 3.2: Distribution of Tor router bandwidth around the world

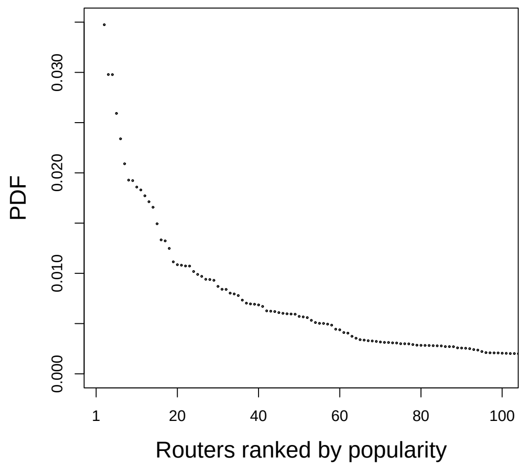
days, only 206 clients were observed to be using Tor software that is older than this version.

Incidentally, entry guards were added to prevent routers from profiling clients, and indeed the reliance on entry guards prevented us from profiling a large number of clients beyond what we describe above. Before entry guards were widely adopted, a strong diurnal usage pattern had been observed [159]. Since entry guards are now widely adopted, utilizing multiple entry guard perspectives gives a larger snapshot of the clients' locations and usage patterns. We informally compared our geopolitical client distribution to that which was observed from other high bandwidth entry guard routers. The distribution was consistent across each entry guard. However, we attempted to observe the current client usage patterns, but this required a more global perspective than we were able to obtain.

Tor router geo-political distribution. During our data collection, we monitored the authoritative directory servers to determine the total number and geopolitical distribution of Tor routers. Over the course of 7 days, we took hourly snapshots of the authoritative directory servers, noting each router's IP address and bandwidth advertisements. During this time, on average 1,188 Tor routers were observed in each snapshot. As shown in Table 3.2, Germany and the United States together contribute nearly 59% of the running routers. However, in terms of total bandwidth,



(a) PDF of all routers.



(b) PDF of the top 100 routers.

Figure 3.3: PDFs of Tor's traffic distribution over its routers during a one hour snapshot

as depicted in Figure 3.2, Germany provides 45% of the bandwidth and the United States only provides 23% of the bandwidth.

It has been suggested that location diversity is a desirable characteristic of a privacy enhancing system [114]. However, given the current bandwidth distribution, location diversity while maintaining adequate load balancing of traffic is difficult to guarantee. It is currently possible to build circuits with at least one router from Germany and the remaining routers from other countries. However, if a location-aware routing mechanism mandated that a user's traffic should exit in a specific country — such as the Netherlands, for example — then it is necessary to ensure that there is sufficient exit bandwidth in that country. Incentive programs to encourage volunteers to run routers in under-represented countries should be investigated. In addition, mitigating malicious client behavior (as noted in Section 3.4) can consequently attract more Tor routers.

3.5.2 Modeling Router Utilization

Understanding the distribution with which different routers are utilized on circuits can provide valuable insights regarding the system's vulnerability to traffic analysis. In addition, a probability distribution can be used to build more realistic analytical models and simulations.

By counting the number of times that each router appears on a circuit with our router, we provide probability density functions (PDFs) to model the probability of each router forwarding a particular packet (shown in Figure 3.3). In a one hour snapshot during the January data collection period, the top 2% of all routers transported about 50% of traffic from the perspective of our router. Within this top 2%, 14 routers are hosted in Germany, 6 are hosted in the United States, 4 are in France, and Switzerland, the Netherlands, and Finland each host a single router. These numbers are consistent with the bandwidth distributions given in Figure 3.2, and further highlight the difficulty of providing strict location diversity in Tor's routing mechanism. The PDF curve drops sharply; the bottom 75% of the routers together transported about 2% of the total traffic. The most traffic that any single router transported was 4.1% of the total traffic. This indicates that the vast majority of Tor traffic is handled by a very small set of routers. Consequently, if an adversary is able to control a set of the highest performing routers, then its ability to conduct traffic analysis increases dramatically. Finally, the PDFs calculated from our router's observations are very similar to the router distribution based on routers' bandwidth advertisements, as reported by Tor's directory servers.

3.6 Circuit-level Performance Measurements

One of Tor's most important design goals is to provide a low latency, high throughput transport service that is suitable for supporting interactive applications. However, a common reason why people do not use Tor is because it is too slow. Tor certainly incurs greater latency when compared to direct connections, since Tor routes cells through a circuit of three hops (by default), potentially circling the world multiple times. Some routers are also highly congested since their

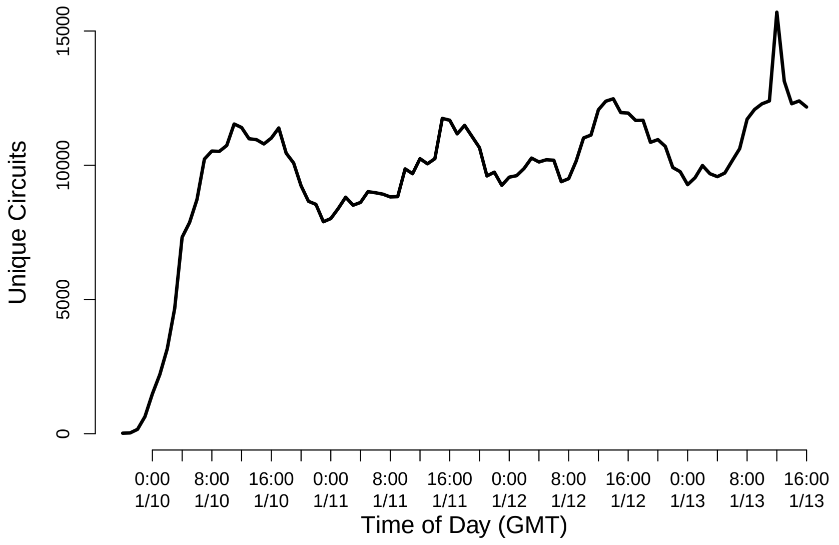


Figure 3.4: The observed circuit connections are plotted over the course of one data collection period

limited bandwidth must be shared among several circuits simultaneously, some of which may be transferring large amounts of data.

We next measure Tor's performance at the circuit-level. In particular, we examine how Tor use varies with the time of day, we measure end-to-end circuit latency across a large number of circuits, and measure circuits' throughput. We finally look at circuit duration and the amount of data transported by circuits routed through our Tor router. In the following, we analyze measurements collected in December 2006, January 2007, and September 2010. Repeating the measurements over the course of several years enables us to observe and analyze changes in Tor's performance. Note that unless otherwise stated, the measurements were collected in December 2006 and January 2007.

3.6.1 Diurnal Patterns in Traffic Load

We did not observe any cyclical patterns in client usage during the December 2007 and January 2008 observation periods. However, over the course of previous data collection periods

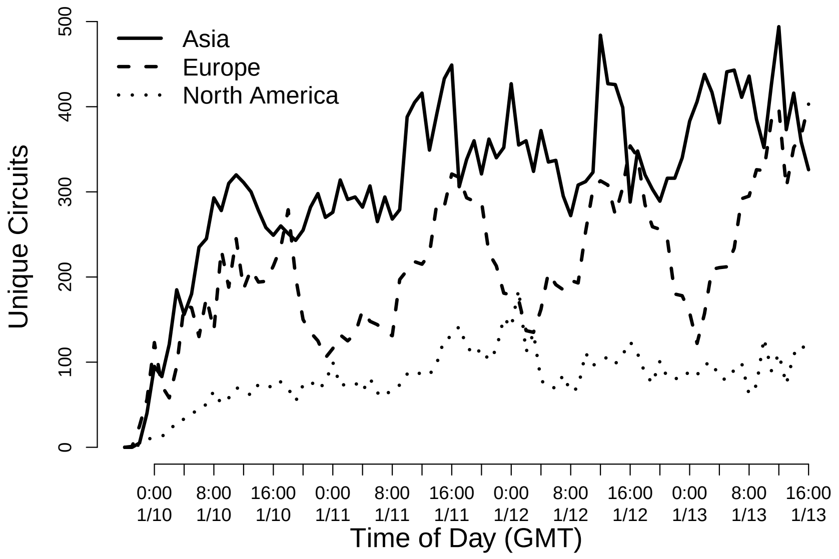


Figure 3.5: Asian, European, and North American circuit connections as a function of the time of day

from December 2006 and January 2007, we observed more prominent daily patterns in traffic load. We present these observations here.

In order to understand Tor use as a function of time of day, we examine the number of circuits observed through our router over the course of the data collection periods. We plot the number of unique connections observed versus time of day in Figure 3.4.

When our Tor router first joined the network, it took several hours to integrate into the network, and the number of connections slowly increases over this warm-up time. Once integrated, the graph indicates that Tor use is cyclical, with a period of approximately one complete day. In addition, peak hours of Tor use occur at 14:00-16:00 GMT, when over 12,000 unique circuits per hour were observed. Tor use is lowest at 0:00 (midnight) GMT, when less than 9,000 circuits per hour were observed. The greatest difference between the high and low times was a 37% decrease from the peak. While the network remains used at all times of day, this shows that a significant correlation between Tor use and time of day exists.

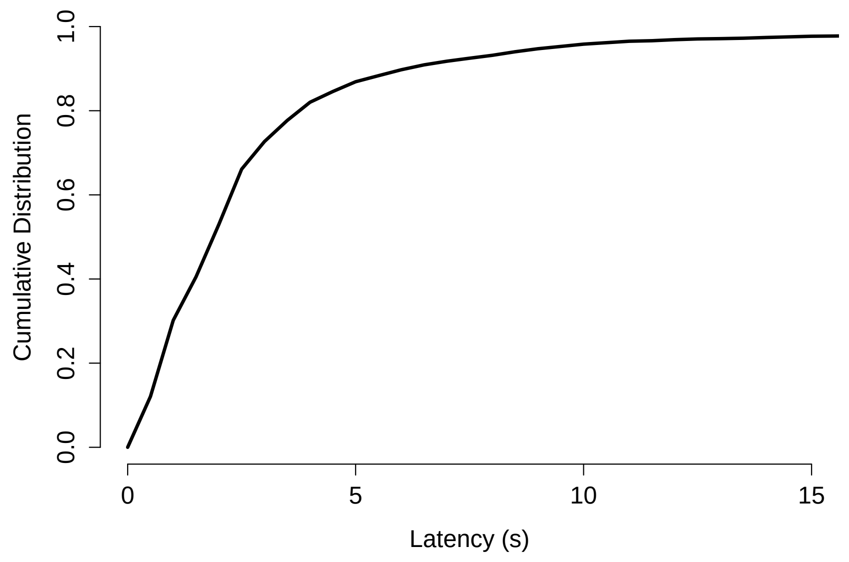


Figure 3.6: CDF of end-to-end latency through Tor circuits

We examine the nature of Tor usage within Asia, Europe, and North America separately in Figure 3.5. To obtain the locations of Tor users over time, it is only possible to ascertain the client's location when our Tor router is used as the entrance router in a circuit. Using this data, users from Asia comprise the most circuit connections, following by Europe and North America. Usage over time in Asia and the United States does not conform to a clear cyclical pattern; however, European users are most frequent during European daytime hours and are least abundant during the night. European users decrease by as much as 62.5% during their off-peak hours. This pattern contributes highly to the global Tor usage pattern shown in Figure 3.4.

3.6.2 End-to-end Latency

To measure latency of circuits, we used `echoping` [13], to measure the end-to-end circuit round-trip time (RTT) over Tor circuits to an echo server running on a machine with a high bandwidth link. We report end-to-end circuit latencies as $RTT/2$.

Figure 3.6 shows a CDF of Tor's end-to-end circuit latency, as experienced by Tor clients. The

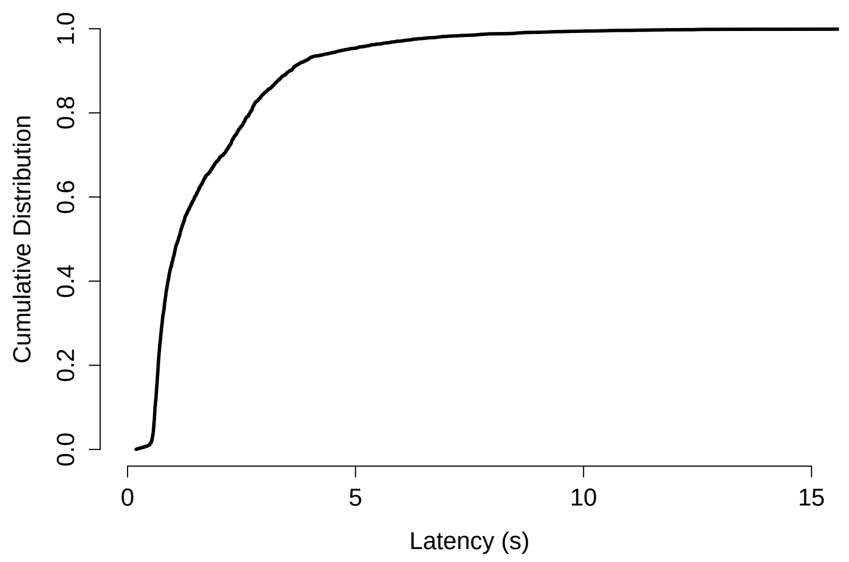


Figure 3.7: CDF of end-to-end latency through Tor circuits in September 2010

graph reveals that the median end-to-end latency of a circuit is 2 seconds with a high variance and a maximum observed latency of 120 seconds. At the 25th percentile, a circuit experienced under 0.5 second of latency, at the 75th percentile, a circuit incurred about 3.5 seconds of latency, and at the 90th percentile, 6.5 seconds of latency was observed. The mean was 3.1 seconds with a standard deviation of 5.1 seconds. In the case of higher latency circuits (above the 90th percentile), it is probable that the circuit timed out and was reconstructed. While these observations show the cost of Tor's anonymity in the form of increased latency, these end-to-end latencies are unacceptably high for delay-sensitive interactive web users, who must incur precisely two end-to-end circuit round-trip times before the client receives the first byte of data (one RTT to connect to the destination server, and a second RTT to issue an HTTP `GET` and receive the first byte of data in return). These delays translate to a median web page response delay of 8 seconds before the client receives the first byte of data, which is unacceptably high.

Figure 3.7 shows a CDF of end-to-end circuit latency as measured in September 2010. The median latency is 0.6 seconds and 75% of circuits have a latency of 1.6 seconds or less. Interestingly,

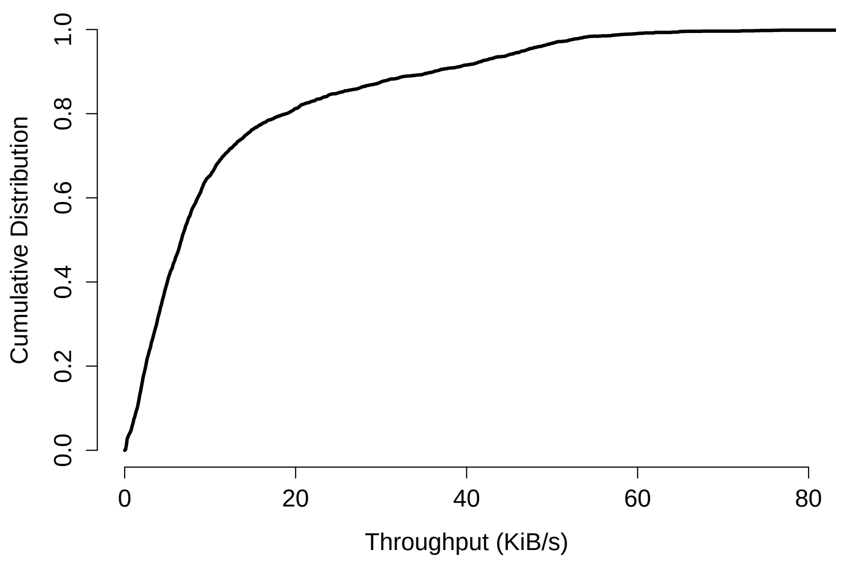


Figure 3.8: CDF of end-to-end throughput through Tor circuits

these delays are significantly less than observed in 2006 and 2007, a reduction in delay of nearly 200% at the median. This means that web page response times are significantly faster than they were a few years ago. Possible explanations are that Tor's bandwidth capacity has increased or that the traffic load has significantly decreased. The later explanation is very likely, as the Great Firewall of China blocked Tor in March 2010 [45], which consequently reduced Tor's traffic load significantly. However, despite this large improvement, latency still remains relatively high, which translates to significant delays for web fetches, which must incur precisely two end-to-end circuit round-trip times before the client receives the first byte of data; this is a median delay in web page response time of 2.4 seconds.

3.6.3 End-to-end Throughput

To collect data about throughput over Tor circuits, we transferred a 128 KiB file from a web server through the Tor network and measured the download time. A base-line was produced by measuring the throughput while downloading the file directly. The base-line throughput was 270

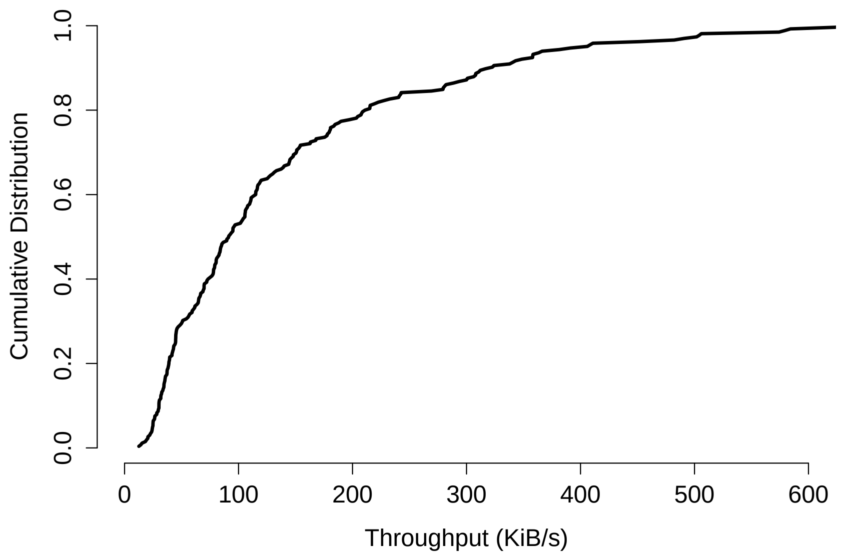


Figure 3.9: CDF of end-to-end throughput through Tor circuits in September 2010

KiB/s with negligible variance. Figure 3.8 shows the CDF for throughput measurements through Tor circuits. The median throughput was 6.8 KiB/s and the mean was 12.6 KiB/s with a standard deviation of 15.2 KiB/s. At the 25th percentile, circuit throughput was 3.2 KiB/s, at the 75th percentile, a circuit provided 14.6 KiB/s, and at the 90th percentile, 35.8 KiB/s throughput was maintained. The maximum observed throughput during the observation was 180.8 KiB/s. This demonstrates that most Tor circuits provide low throughput.

Similar to the improvement in end-to-end latency, Tor's throughput has improved significantly between 2007 and 2010. Figure 3.9 shows that the median end-to-end circuit throughput is 91 KiB/s and 75% of streams achieve less than 179 KiB. This is an improvement of an order of magnitude.

3.6.4 Circuit Duration

Circuit duration was measured during our data collection while operating a Tor router. As depicted in Figure 3.10, the median circuit duration was 30 seconds. At the 25th percentile, a circuit lasted for under 10 seconds, at the 75th percentile, a circuit is used for 210 seconds, and

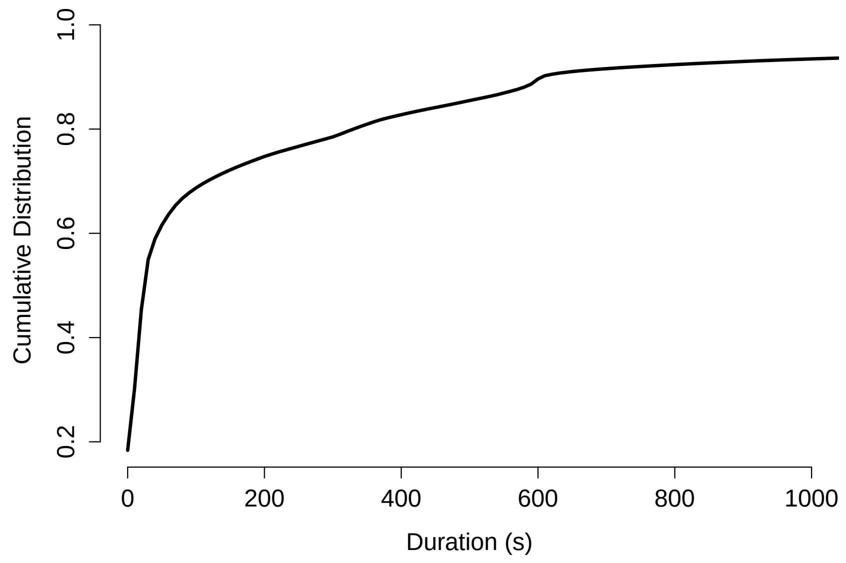


Figure 3.10: CDF of Tor circuits' durations

at the 90th percentile, a circuit is used for 610 seconds. The mean circuit duration is 814 seconds with a standard deviation of 7 900.5 seconds. The longest living circuit was observed for 242 380 seconds, or 67.3 hours. This shows that the vast majority of circuits are short-lived. The median circuit duration is sufficient for transferring small amounts of data over HTTP, for example.

To demonstrate that the two data collection periods are consistent, we show a quantile-quantile (QQ) plot in Figure 3.11. A linear relationship between the observed quantiles from the two data collection periods indicates that the two data sets are taken from the same distribution [131]. This demonstrates that the data collection was consistent between observations.

Figure 3.12 shows that the median circuit lasted for 44 seconds and 75% of circuits lasted for no longer than 365 seconds. Thus, circuits appear to last longer in 2010 than they did in the earlier measurements.

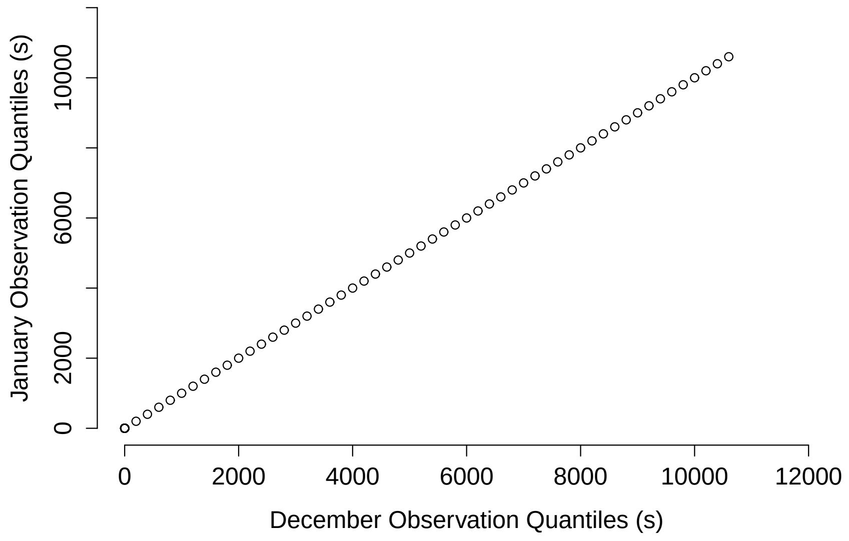


Figure 3.11: Quantile-Quantile plot of the duration observations from the January and December data collection periods

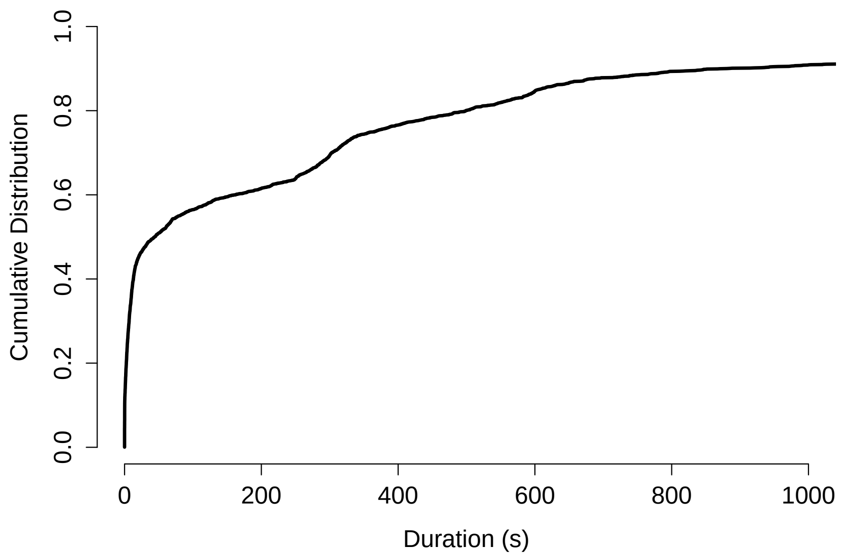


Figure 3.12: CDF of Tor circuits' durations in September 2010

3.6.5 Circuit Capacity

In order to measure capacity of Tor circuits, we observed how many bytes transversed circuits during our router's participation in the Tor network. A CDF of the bytes transferred per circuit

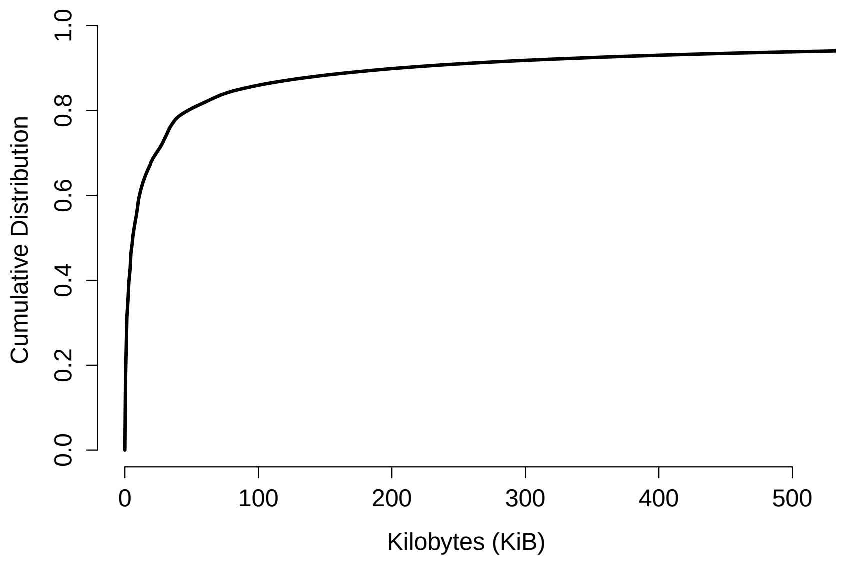


Figure 3.13: CDF of data transferred over Tor circuits

are given in Figure 3.13. At the median, 6.1 KiB traversed a circuit. At the 25th percentile, about 1.0 KiB flowed through a circuit, at the 75th percentile, 31.2 KiB was sent, and at the 90th percentile, 201.7 KiB was transferred. However, the mean circuit transported 730.8 KiB with a standard deviation of 10 312.6 KiB.

The maximum amount of data observed to be transferred over a circuit was 1.5 GiB. This demonstrates that while most circuits transport very little data, there exist outliers that are able to sustain the circuit for a sufficient amount of time to transfer several orders of magnitude more data, although this is quite rare. The median circuit capacity would be sufficient to transfer a relatively small web page. These circuit-level measurements are consistent with the observed protocol distribution. We provide a QQ plot in Figure 3.14 that shows the amount of data transported is consistent between the December and January data collection periods.

We repeated the circuit capacity measurement in September 2010 and found that circuits tend to transport far more data than they did in the past. Figure 3.15 shows that the median circuit transports 640.5 KiB, which is roughly an order of magnitude more than observed in 2007. One

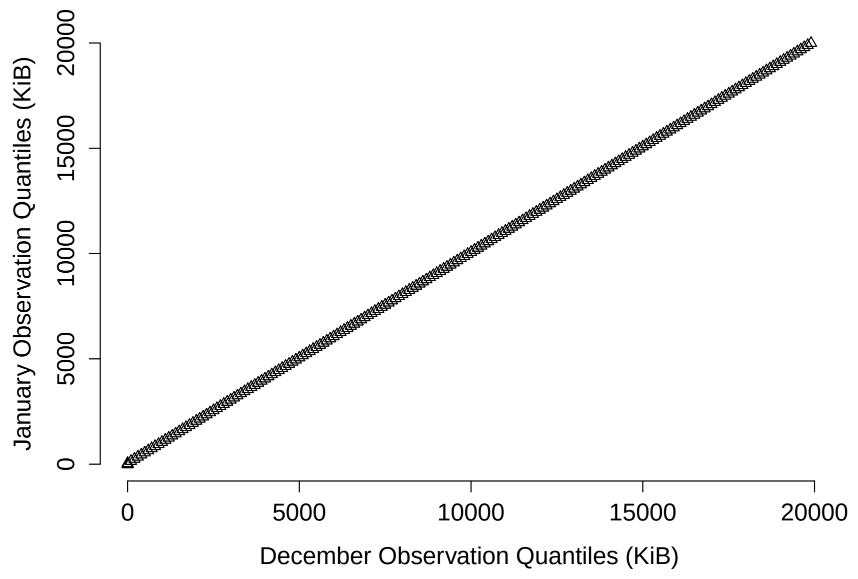


Figure 3.14: Quantile-Quantile plot of the kilobytes transferred from the January and December data collection periods

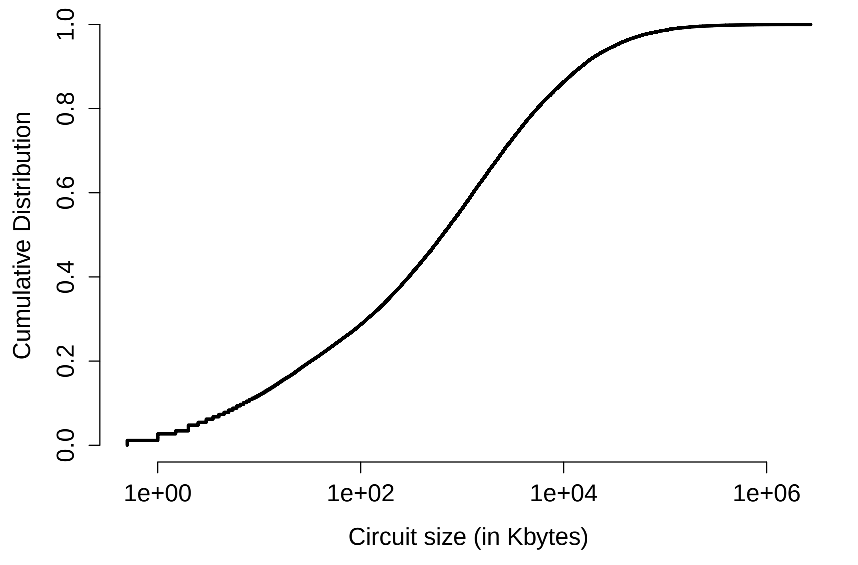


Figure 3.15: CDF of data transferred over Tor circuits in September 2010

possible explanation is that Tor circuits have become more reliable, potentially due to optimizations in the circuit building process such as adaptive circuit building timeouts [77], improvements in Tor's load balancing [176], an increase in Tor's available bandwidth, or a reduction in the traffic load.

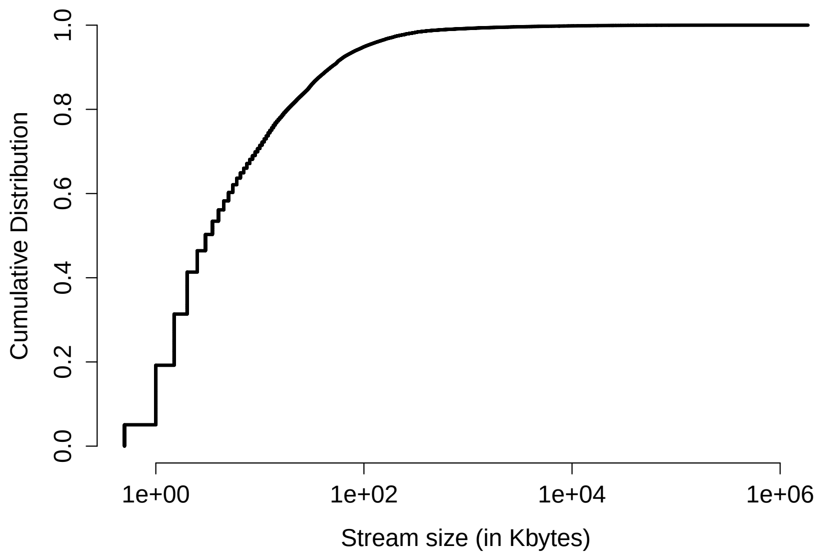


Figure 3.16: CDF of stream size within Tor circuits in September 2010

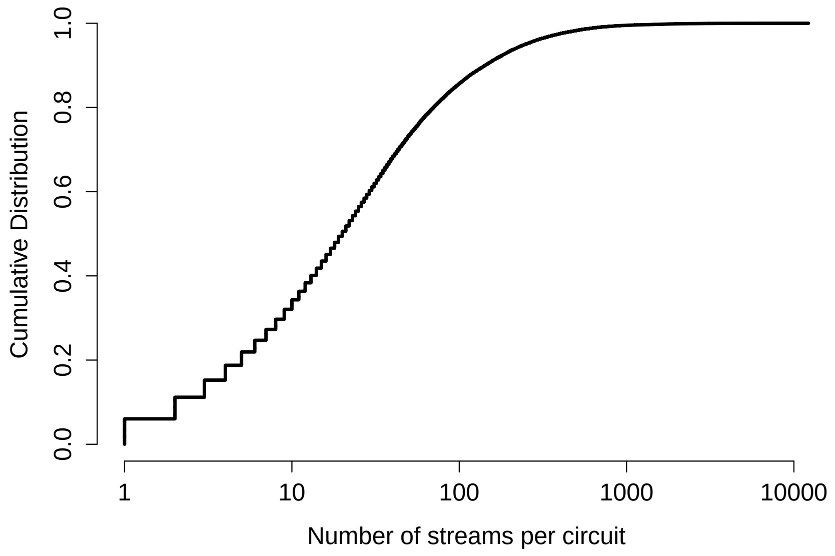


Figure 3.17: CDF of number of streams within Tor circuits in September 2010

To understand the nature of TCP streams within circuits, we measure the size and number of distinct TCP streams per circuit in September 2010. Figure 3.16 shows that the median stream tunneled over a circuit Tor is 3.0 KiB, and 75% of all streams are smaller than 12.5 KiB. The largest stream observed transported 1.7 GiB. As shown in Figure 3.17, the median circuit transported 20 streams and 75% of circuits transported less than 54 streams. These observations are consistent with the hypothesis that circuit reliability has increased over the years between measurements.

3.6.6 Discussion

In our analysis of Tor performance at the circuit level, we have shown that the volume of Tor traffic was correlated with the time of day in December 2006 and January 2007, but as Tor became more popular across different regions of the world, these patterns became less pronounced. We have also found that in 2006 and 2007, end-to-end circuit latency was higher than is typically expected by delay-sensitive web clients and that circuit throughput was low, but in 2010 latency had decreased and throughput had increased significantly. The improvement in performance over nearly four years is very encouraging. However, perhaps the largest contributing factor to this improvement comes from the Chinese government's decision to explicitly block its citizens' ability to access Tor in March 2010 [45]. Recall that in Chapter 3.5, we found that China was the second most common country of origin among Tor clients. This indicates that the departure of most Chinese users likely reduced Tor's traffic load significantly. Finally, given the highly dynamic nature of Tor as a system, some of the observations presented in this chapter could change as user behavior, government or Internet Service Provider blocking agendas, and other factors change. To understand how Tor usage has changed since our study, we direct the reader to a contemporary Tor study by Chaabane *et al.* based on data collected in 2009 and 2010 [74].

3.7 Ethics and Community Standards

Any large-scale study that involves live users and real traffic must be conducted carefully and with the highest ethical standards. An important and an ongoing debate has erupted regarding how to balance the potential benefits of live measurement and analysis in cyber-security and privacy research with the ethical questions, legal constraints, and conformity to community standards that revolve around such research. To help understand the fundamental issues at stake, Dittrich *et al.* conduct a survey of prior cyber-security and privacy research that potentially raises ethical questions, and offer an initial framework for reasoning about such questions [107]. Also, Sicker *et al.* provide a perspective on Internet measurement studies with regard to United States Federal Law and suggest guiding principles to promote safe measurement including obtaining user consent, analyzing synthetic data instead of real data, and obeying policies of data anonymization, reduction, and minimization [207]. While obtaining user consent is not always possible and using synthetic or simulated data may not enable the researchers to adequately analyze the problem or phenomena being studied, their final suggestion of anonymization, reduction, and minimization is a reasonable best practice.

In conducting the Tor measurement study presented in this chapter, we exercised extreme caution and care to protect any personally identifiable information collected in our traces. Furthermore, we practiced data reduction and minimization by collecting only enough payload data to enable accurate protocol identification. While we would have liked to anonymize the traces during the capture process, the amount of data and speed at which it was collected made anonymization computationally infeasible. In addition, obtaining user consent is inherently impossible, as the users are, in fact, anonymous. Ultimately, we believe that the potential risk to Tor users from this study was very low, and the potential to better understand and improve Tor's design was very high.

Collecting useful data safely from anonymous networks such as Tor remains a challenging research problem. One approach is to collect and release aggregated statistical data [152]. While such an approach is promising, it is unclear whether statistical data is as useful as raw data for analytical purposes. Also, there may be cases where even aggregated statistical data exposes sensitive information.

3.8 Broader Impact

Our primary objective in this chapter has been to better understand how Tor is used, who uses Tor, how Tor's anonymity may shroud malicious activities, and how Tor's performance has improved over the course of several years. From these observations and measurements, we have offered a variety of suggestions to improve systems like Tor.

First, given our observation that non-TLS protected protocols, including those used for e-mail (POP3 and IMAP) and even archaic remote terminal protocols such as telnet are present in Tor exit traffic. To mitigate the risks associated with not only leaking information about the initiator's identity (by observing user names, for instance), but potentially leaking login credentials which could lead to account compromise, we proposed that Tor explicitly warn the user or block such protocols at the client's local proxy [59]. Also, a set of scripts have been developed and deployed to actively scan Tor exit nodes for malicious behaviors and the `BadExit` flag has been added to the trusted directory servers' consensus information to inform clients not to select these routers [175].

Second, we observed that router bandwidth is centralized within only a few countries. This has significant implications to Tor's anonymity properties, as it is well-known that location diversity reduces the likelihood that a single Internet Service Provider (ISP), autonomous system (AS), or country-level adversary can observe the network links at circuits' endpoints [111, 114]. This observation has motivated additional work that aims to promote router diversity and participation by employing incentive schemes [50, 133, 170].

Third, our analysis of malicious activities undertaken by both Tor clients and routers highlights the need for effective mechanisms to enforce better accountability, but without sacrificing anonymity. A large body of work has been developed to improve anonymity networks' abilities to block malicious users while not sacrificing anonymity for non-malicious users [82, 123, 220, 224, 225].

Fourth, we observed that BitTorrent traffic, while small in terms of the number of TCP connections, consumes a vastly disproportionate amount of Tor's scarce bandwidth. Bulk data transfer is a significant source of congestion that degrades performance for delay-sensitive web users.

The observed demand for anonymous file sharing has motivated the development of anonymizing techniques tailored specifically to preserve privacy within bulk file sharing traffic [62, 78, 130].

Lastly, this work has, in part, initiated an important discussion on community standards, norms, and best practices for cyber-security and privacy research studies that involve live users. The publication of part of this chapter's work [158] has spurred an ongoing conversation and debate regarding best practices for such studies [107, 117], leading to the formation of a series of academic workshops to discuss and debate these issues [2, 18] and the development of a community-driven repository for aggregated statistical Tor data [222].

3.9 Summary

This chapter is focused on understanding Tor usage. In particular, we provided observations that help understand how Tor is being used, how Tor is being mis-used, and who participates in the network as clients and routers. Through our observations, we have made several suggestions to improve Tor's current design and implementation. First, in response to the fairly large amount of insecure protocol traffic, we proposed that Tor provide a mechanism to block the ports associated with protocols such as POP3, IMAP, and Telnet. Given the ease at which an eavesdropping exit router can log sensitive user information (such as user names and passwords), we developed a method for detecting malicious logging exit routers, and provided evidence that there are such routers that specifically log insecure protocol exit traffic. Also, we show the disparity in geopolitical diversity between Tor clients and routers, and argue that location diversity is currently impossible to guarantee unless steps are taken to attract a more diverse set of routers. As a final avenue of study, we characterize Tor's performance as perceived by end-users.

Due to its popularity, Tor provides insight into the challenges of deploying a real anonymity service, and our hope is that this work will encourage additional research aimed at (1) providing tools to enforce accountability while preserving strong anonymity properties, (2) protecting users from unknowingly disclosing sensitive/identifying information, and (3) fostering participation from a highly diverse set of routers.

Chapter 4

Practical Attacks against Low Latency Anonymous Networks

We present methods for compromising the security of the Tor overlay network [103]. This work focuses on the following two scientific questions: (1) how can we *minimize* the requirements necessary for any adversary to compromise the anonymity of a flow; and (2) how can we harden Tor against our attacks?

Central to our attacks is the fact that a *lying* adversary — by exaggerating its resource claims — can compromise an unfair percentage of Tor entry and exit nodes. Further, we show how an adversary can compromise the anonymity of a Tor path *before* any data is transmitted, which enables us to further reduce the resource requirements on the attacker. We experimentally evaluate the efficacy of our attacks via experiments with an isolated Tor deployment on PlanetLab. We also explore methods for mitigating the severity of our attacks.

Historical balance between anonymity and performance. Conventional wisdom suggests that it is impossible for practical privacy-enhancing systems to provide perfect anonymity. Therefore, the designers of such systems must consider restricted threat models. Consider, for example, an anonymous communications system that routes traffic through multiple intermediate nodes. While it is generally possible to perform a traffic analysis attack against a connection if both of the endpoints are compromised, theoretical analyses of anonymity networks show that the likelihood of successfully launching such a traffic analysis attack becomes negligible as the network size increases [103, 189, 219].

When the Tor network was launched, it consisted of few routers transporting little traffic. Consequently, in its initial design, Tor provided no traffic load balancing capability, and it was with respect to this initial design that the above-mentioned theoretical analyses were performed [103]. As the network grew to include nodes with a wide variety of bandwidth capabilities, it became necessary to ensure that the traffic is efficiently balanced over the available resources in order to achieve low latency service. Tor's routing mechanism was modified to prefer high-bandwidth, high-uptime routers that have the resources to accept new connections and transport traffic. Dingledine, Mathewson, and Syverson suggested that a non-uniform router selection mechanism may increase an attacker's ability to compromise the system's anonymity [104], though the full security implications of this load balancing was left to further research.

Our approach. Within Tor's routing model, an adversary could deploy a few nodes that have — or *appear* to have — high-bandwidth connections and high-uptimes. In the latter case, the adversary is said to *lie* about its resources. With high probability, such an adversary would be able to successfully compromise the two endpoints — the *entry node* and the *exit node* — of a new Tor client's connections. Compromising the entry and exit nodes with non-uniform probability is the first step in our attack.

As noted above, previous works showed that, upon compromising the entry and exit nodes, it is possible to compromise the anonymity of a connection via traffic analysis. However, in the spirit of *minimizing* the resource requirements for the adversary, we develope an end-to-end method for associating a client's request to its corresponding destination *before* any payload data is sent. This is important since low-resource malicious nodes may lack the bandwidth to forward significantly many data packets.

Experimental evaluation. We experimentally show, using an isolated Tor deployment on PlanetLab, that adversaries with sparse resources — such as adversaries with a few nodes behind residential cable modems — can compromise the anonymity of many paths for new clients. In a Tor deployment of 60 honest and 6 malicious Tor routers, our attack compromised over 46% of the path-building requests from new clients through the network. This illustrates the inherent difficulty of simultaneously attempting to provide optimal anonymity and efficient use of the network's resources.

Prior attacks. Other attacks against Tor have focused on traffic analysis and locating hidden services. Murdoch and Danezis presented a low cost traffic analysis technique that allowed an outside observer to infer which nodes are being used to relay a circuit's traffic [166], but could not trace the connection to the initiating client. Øverlier and Syverson demonstrated a technique for locating hidden services that used false resource claims to attract traffic [172]. Murdoch and Zieliński [168] analyze the route selection problem in Tor in the context of Internet exchanges (IXes), and give a new traffic analysis technique for processing data from IXes. We extend the attack on hidden services to effectively compromise the anonymity of general-purpose paths using resource-constrained nodes.

Attack variants and improvements. We consider additional attack variants and improvements in the body of this chapter. For example, we show how to adapt our attack to compromise the flows of pre-existing Tor clients; recall that our attack as described above is (generally) only successful at compromising new clients, who have not already picked their preferred entry nodes. We also consider further resource reductions, such as using watermarking techniques to, in some cases, eliminate the need for a compromised exit node. An important extension is an attack on the entry guard selection process, where we show that it is possible to displace all legitimate entry guards with malicious nodes. Additionally, we consider methods to improve the effectiveness of our attack, such as a variant of the Sybil attack [108].

Countermeasures. Next we explore counter-measures to routing attacks in Tor. High-resource adversaries, even if only in possession of a few malicious nodes, seem to pose a fundamental security challenge to any high-performance, multi-hop privacy enhancing system. We focus on designing solutions to mitigate the low-resource attacker's ability to compromise anonymity. These solutions include verifying information used in routing decisions, allowing clients to make routing decisions based on observed performance, and implementing location diversity in routers to mitigate Sybil attacks.

Context. Following the initial disclosure of the primary vulnerability behind these attacks [60], development began by the Tor community on measures to mitigate the effectiveness of these attacks. While an adversary's ability to launch a large number of malicious nodes from the same physical machine or network has been partially addressed [57], the more challenging problem of verifying bandwidth claims remains open.

4.1 Background

In order to present the methodology used in our experiments, we first provide a brief overview of the Tor system architecture, an in depth analysis of Tor's router selection algorithms, and a description of Tor's attack model.

4.1.1 Tor's Router Selection Algorithms

We next describe Tor's router selection process as specified and implemented in 2007. Note that, as a result of the attacks presented in this chapter, some aspects of the router selection algorithm have changed. We describe the broader impact of these attacks and proposed defenses in Chapter 4.8.

As of `Tor` version 0.1.1.23, there are two parts to the algorithm that Tor uses to select which routers to include in a circuit. The first part is used to select the entry router, and the second part is used to select subsequent routers in the circuit. We will show methods to exploit both of these algorithms, as implemented in Tor in 2007, in Section 6.1.2.

Entry router selection algorithm. The default algorithm used to select entry routers was modified in May 2006 with the release of Tor version 0.1.1.20. Entry guards were introduced to protect circuits from selective disruption attacks, thereby reducing the likelihood of an attacker intentionally breaking circuits until they are on a target victim's circuit [172]. The entry guard selection algorithm works by automatically selecting a set of Tor routers that are marked by the trusted directory servers as being "fast" and "stable." The directory server's definition of a fast router is one that reports bandwidth above the median of all bandwidth advertisements. A stable router is defined as one that advertises an uptime that is greater than the median uptime of all other routers.

The client will only choose new entry guards when one is unreachable. Currently the default number of entry guards selected is three, and old entry guards that have failed are stored and retried periodically. There is also an option added to use only the entry guards that are hard-coded into the configuration file, but this option is disabled by default. This algorithm was implemented to protect the first hop of a circuit by using a limited pool of nodes.

Non-entry router selection algorithm. The second algorithm to select non-entry nodes is intended to optimize router selection for bandwidth and uptime, while not always choosing the very best nodes every time. This is meant to ensure that all nodes in the system are used to some extent, but nodes with more bandwidth and higher stability are used most often. Tor has a set of TCP ports that are designated as "long-lived." If the traffic transiting a path uses one of these long-lived ports, Tor will optimize the path for stability by pruning the list of available routers to only those that are marked as stable. This causes Tor's routing algorithm to have a preference towards routers marked as stable nodes. For more details on this part of the algorithm, see the Tor Path Specification [101].

The next part of the algorithm optimizes the path for bandwidth. Briefly, this algorithm works as follows: Let b_i be the bandwidth advertised by the i-th router, and assume that there are N routers. The probability that the i-th router is chosen is proportional to $b_i \; / \left(\sum_{j=1}^{N} b_j \right)$. We assume that $\sum_{j=1}^{N} b_j > 0$, since a zero value would imply that the system has no available bandwidth. We provide pseudocode for the bandwidth optimization part in Algorithm 1.

The most significant feature of this algorithm is that the more bandwidth a particular router advertises, the greater the probability that the router is chosen. The routing algorithm's tendency to favor stable and high bandwidth nodes is fundamentally important to the implementation of our attack.

Algorithm 1: Non-Entry Router Selection

```
Input: A list of all known Tor routers, $router\_list$
Output: A pseudo-randomly chosen router, weighted toward the routers advertising the highest bandwidth
$B \leftarrow 0$, $T \leftarrow 0$, $C \leftarrow 0$, $i \leftarrow 0$, $router\_bw \leftarrow 0$
$bw\_list \leftarrow \emptyset$
foreach router $r \in router\_list$ do
    $router\_bw \leftarrow \text{get\_router\_adv\_bw}(r)$
    $B \leftarrow B + router\_bw$
    $bw\_list \leftarrow bw\_list \cup router\_bw$
end
$C \leftarrow \text{random\_int}(1, B)$
while $T < C$ do
    $T \leftarrow T + bw\_list_i$
    $i \leftarrow i + 1$
end
return $router\_list_i$
```

4.1.2 Tor's Threat Model

Tor's design document [103] lays out an attack model that includes a non-global attacker that can control or monitor a subset of the network. The attacker can also inject, delay, alter, or drop the traffic along some of the links. This attack model is similar to the models that other low latency anonymity systems such as Freenet [80], MorphMix [190], and Tarzan [116] are designed to protect against.

As a component of Tor's attack model, the designers acknowledge that an adversary can potentially compromise a portion of the network. To predict the expected percentage of flows compromised by such an adversary, a simplified theoretical analysis of a privacy enhancing system is provided in Tor's design document [103]. This analysis is based on a combinatorial model that assumes nodes are chosen at random from a uniform distribution.

4.2 Compromising Anonymity

We now consider how an adversary might compromise anonymity within the Tor threat model by gaining access to a non-global set of malicious nodes. In our basic attack, we assume that these malicious nodes are fast and stable, as characterized by high bandwidths and high uptimes. While even the basic attack squarely compromises anonymity under Tor's target threat model [103], we also show how to remove these performance restrictions for an even lower-resource attack.

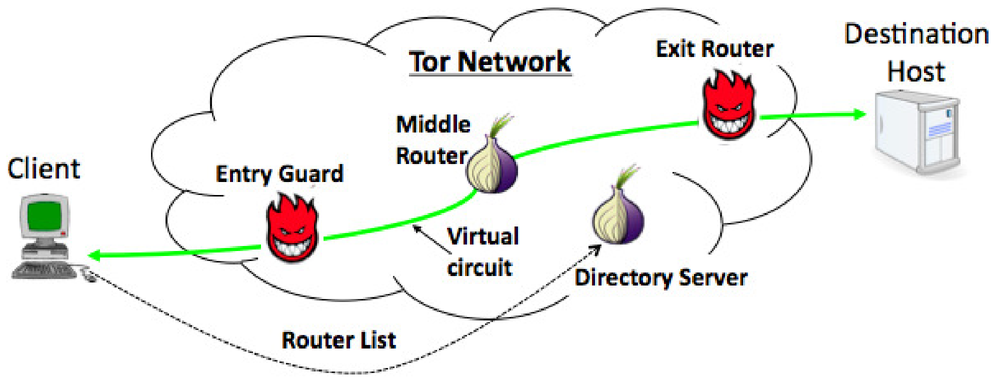


Figure 4.1: *Attack Model:* Malicious Tor routers are positioned at both the entry and exit positions for a given client's circuit to the requested destination server through the Tor network

We focus on attacking the anonymity of clients that run in their default configurations; in particular, we assume that clients function only as Tor proxies within the network. We also focus on attacking clients that join the network after the adversary mounts the first phase of our attack (Section 4.2.1); we shall remove this restriction in Section 6.8.

4.2.1 Phase One: Setting Up

To mount our attacks, an adversary must control a subset of $m > 1$ nodes in the pool of active Tor routers. The adversary might obtain such nodes by introducing them directly into the Tor network, or by compromising existing, initially honest nodes. The adversary may coordinate these compromised machines in order to better orchestrate the attack.

The basic attack. In our basic attack, the adversary's setup procedure is merely to enroll or compromise a number of high-bandwidth, high-uptime Tor routers. If possible, the adversary should ensure that all of these nodes advertise unrestricted exit policies, meaning that they can forward any type of traffic.

Resource reduction. We can significantly decrease the resource requirements for malicious nodes, thereby allowing them to be behind low-bandwidth connections, like residential broadband Internet connections. This extension exploits the fact that a malicious node can report incorrect (and large) uptime and bandwidth advertisements to the trusted directory servers [172]. These false advertisements are not verified by the trusted directory servers, nor by other clients who will base their routing decisions on this information, so these false advertisements will remain undetected. Thus, from the perspective of the rest of the network, the adversary's low-resource routers actually appear to have very high bandwidths and uptimes. It is important that the malicious nodes have just enough bandwidth to accept new connections. This is achieved by focusing the nodes' limited resources toward accepting new client connections.

Selective path disruption. If malicious nodes do not exist at both the entry and exit positions of a circuit, but at only one position (either entry, middle, or exit), it can cause the circuit to break simply by dropping all traffic along the circuit. This causes the circuit to be rebuilt with a chance that the rebuilding process will create a path configuration in which both the entry and exit nodes are malicious.

What happens next. Since one of Tor's goals is to provide a *low latency* service, when a new client joins the network and initiates a flow, the corresponding Tor proxy attempts to optimize its path by choosing fast and stable Tor routers. By deploying nodes with high bandwidths and high uptimes, or by deploying nodes that give the *impression* of having high bandwidths and high uptimes, the adversary can increase the probability that its nodes are chosen as both entry guards and exit nodes for a new client's circuit. Compromising the entry and exit position of a path is a necessary condition in order for the second phase of our attack (Section 4.2.2) to successfully correlate traffic.

As a brief aside, on the real Tor network, roughly half of the Tor routers have restricted exit policies that do not allow them to be selected as exit nodes for all flows. This situation further increases the probability that one of the adversary's nodes will be chosen as a flow's exit node.

4.2.2 Phase Two: Linking Circuits

We have shown a method that increases the likelihood of a malicious router existing on a particular proxy's path through Tor. In the improbable case when the full path has been populated with malicious nodes, it is trivial to compromise the anonymity of the path. However, in the more likely case, if only the entry and exit nodes are malicious, we have developed a technique that allows paths to be compromised with a high probability of success (see Figure 4.1). Our approach here is independent of whether the adversary is implementing the basic or the resource-reduced attack described in Section 4.2.1.

While others have postulated the possibility that an adversary could compromise the anonymity of a Tor route if the adversary controlled both the route's entry and exit nodes [103, 172], to the best of our knowledge, our approach is the first that is capable of doing so *before* the client starts to transmit any payload data. This ability is important, because a resource-starved adversary should desire to minimize the cost of the attack in order to maximize the number of circuits that may be compromised. Furthermore, we experimentally verify the effectiveness of our approach in Section 6.1.3.

Overview. In order for the attack to reveal enough information to correlate client requests to server responses through Tor, each malicious router logs the following information for each cell received: (1) its location on the current circuit's path (whether it is an entry, middle, or exit node); (2) local timestamp; (3) previous circuit ID; (4) previous IP address; (5) previous connection's port; (6) next hop's IP address; (7) next hop's port; and (8) next hop's circuit ID. All of this information is easy to retrieve from each malicious Tor router. Once this attack has been carried out, it is possible to determine which paths containing a malicious router at the entry and exit positions correspond to a particular Tor proxy's circuit building requests. With this information, an attacker can associate the sender with the receiver, thus compromising the anonymity of the system. In order to execute this algorithm, the malicious nodes must be coordinated. The simplest approach is to use a centralized authority to which all malicious nodes report their logs. This centralized authority can then execute the circuit-linking algorithm in real-time.

Tor's circuit building algorithm. Tor's telescoping circuit building algorithm sends a deterministic number of packets in an easily recognizable pattern. Figure 4.2 shows the steps and the timing associated with a typical execution of the circuit building algorithm. A Tor proxy creates a new circuit through Tor as follows: First, the proxy issues a circuit building request to its chosen entry

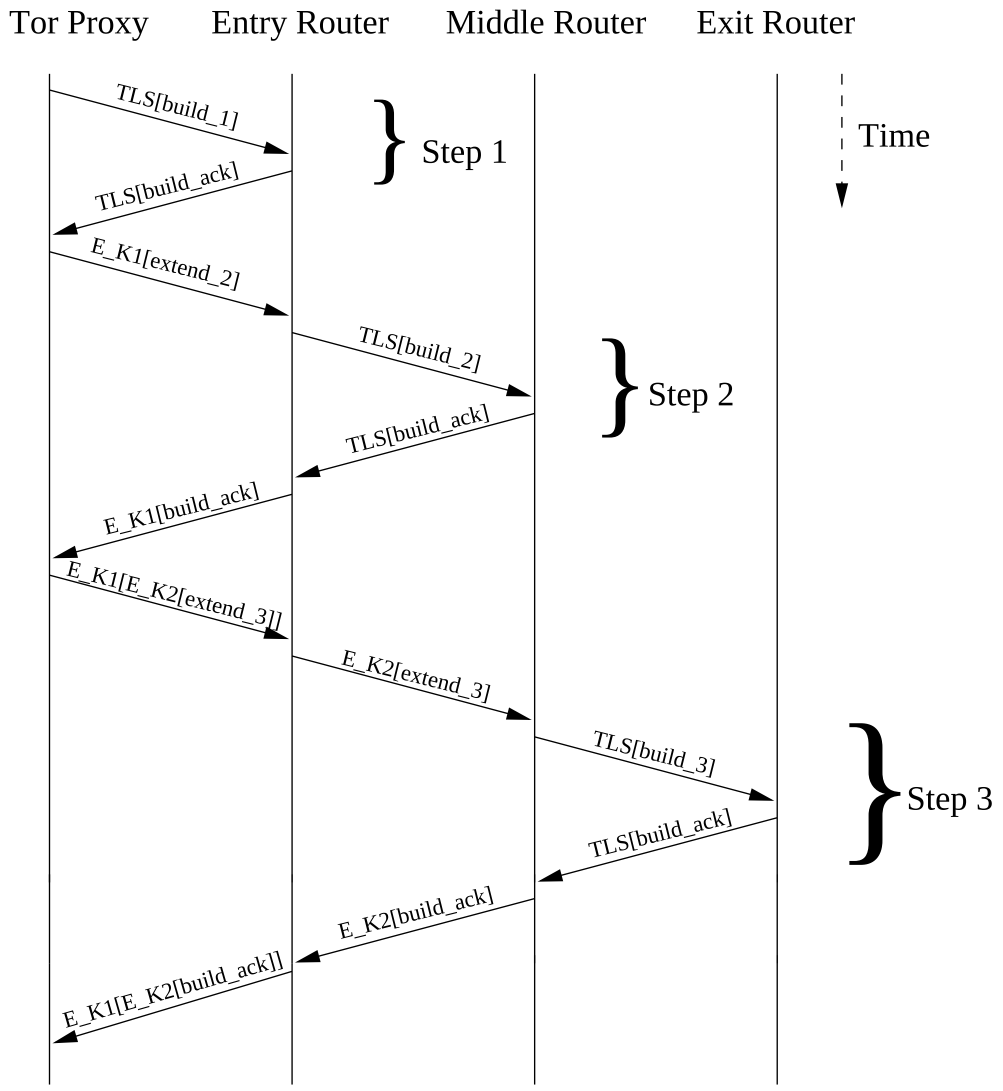


Figure 4.2: A sequential packet diagram of Tor's circuit building process

guard and the entry guard sends an acknowledgment (Step 1). This establishes a shared symmetric key K_1 between the client and the entry guard. Next, the proxy sends another circuit building request to the entry router to *extend* the circuit through a chosen middle router. This message is encrypted with K_1 and establishes another shared symmetric key between the entry guard and the middle router. The middle router acknowledges the new circuit by sending an acknowledgment back to the client via the entry node (Step 2). Finally, the proxy sends a request to extend the circuit to the chosen exit node, which is forwarded through the entry and middle routers to the chosen exit router. This extend message is encrypted with K_1 and K_2 in a layered fashion. This also establishes a shared secret key K_3 between the middle router and the exit router. Once the exit router's acknowledgment has been received through the middle and entry nodes, the circuit has been successfully established (Step 3).

In order to exploit the determinism of the circuit building algorithm, it is necessary to associate the timing of each step and analyze the patterns in the number and direction of the cells recorded. A packet counting approach as used to locate hidden services [172] would not be sufficient, since not all cells sent from the Tor proxy are fully forwarded through the circuit; thus, the number of cells received at each Tor router along the circuit is different. This pattern is highly distinctive and provides a tight time bound, which we utilize in our circuit linking algorithm.

Circuit linking algorithm. The circuit linking algorithm works as follows:

(1) The entry node verifies that the circuit request is originating from a Tor proxy, not a router. This is easily determined since there will be no routing advertisements for this node at the trusted directory servers.

(2) Next, the algorithm ensures that Steps 1, 2, and 3 (from Figure 4.2) occur in increasing chronological order. Also, it is necessary to verify that the next hop for an entry node is the same as the previous hop of the exit node.

(3) Finally, in Step 3, it is verified that the cell headed towards the exit node from the entry node is received before the reply from the exit node.

If every step in the algorithm is satisfied, then the circuit has been compromised.

4.3 Experiments

In this section, we describe the experimental process we used to demonstrate and evaluate our resource-reduced attack. By experimentally evaluating our resource-reduced attack, our experimental results also immediately extend to the basic attack scenario in Section 6.1.2.

Table 4.1: Bandwidth distributions

Tier	**Tor Networks**		
	Real Tor	40 Node	60 Node
996 KB/s	38	4	6
621 KB/s	43	4	6
362 KB/s	55	6	9
111 KB/s	140	13	20
29 KB/s	123	11	16
20 KB/s	21	2	3
Total	103.9 MB/s	10.4 MB/s	15.7 MB/s

4.3.1 Experimental Setup

In order to evaluate this attack in a realistic environment, we set up an isolated Tor deployment on the PlanetLab overlay testbed [177]. We were advised not to validate our attack on the real Tor network because of its potentially destructive effect [97]; however, we did verify that our technique for publishing false router advertisements did, in fact, propagate for a single test router on the real Tor deployment.

To ensure that the experimental Tor networks are as realistic as possible, we surveyed the real Tor network in August 2006 to determine the router bandwidth distribution. This data is given in Table 4.1. According to the real trusted Tor directory servers, there are roughly 420 Tor routers in the wild that forward at least 5 KiB per second. However, due to limitations on the number of PlanetLab nodes that were available over the course of the experiments, we created smaller Tor networks according to our analysis of the router quality distribution in the real deployment. We created two isolated Tor networks on PlanetLab, consisting of 40 and 60 nodes, each running exactly one router per node. Each experimental deployment has precisely three directory servers, which are also nodes from PlanetLab.

When choosing nodes from PlanetLab for the experimental deployments, each node was evaluated using `iperf`, a common bandwidth measurement tool [129], to ensure that it had sufficient bandwidth resources to sustain traffic at its assigned bandwidth class for the course of each experiment. Also, as is consistent with the real Tor network, we chose PlanetLab nodes that are geographically distributed throughout the world.

All Tor routers (both malicious and benign) advertise the same, unrestricted exit policy. The exit policies of routers in the real Tor network are difficult to accurately model due to the reduced size of our network. The global use of unrestricted exit policies in our experimental Tor testbed actually decreases the potential effectiveness of our attack. With the potential for more restrictive exit policies in a real Tor network, we expect the attack's performance to improve since the malicious routers would have a higher probability of compromising the exit position.

To demonstrate the effect of the attack, we introduced a small number of malicious Tor routers into each private Tor network. In the 40 node network, experiments were conducted by adding two (2/42) and four (4/44) malicious nodes. In the 60 node network, three (3/63) and six (6/66) malicious nodes are added. The fraction of each network's bandwidth that is malicious is given in Section 4.3.1.2. All experiments were conducted in October 2006 with `Tor` version 0.1.1.23.

The experiments were conducted as follows: The three trusted directory servers and each benign Tor router in the network are started first, then the client pool begins generating traffic through the network. The network is given two hours for the routing to settle and converge to a stable state,[1] at which point the clients are promptly stopped and all previous routing information is purged so that the clients behave exactly like new Tor proxies joining the network. The malicious nodes are then added to the network and the clients once again generate traffic for precisely two hours. This procedure is repeated for the 2/42, 4/44, 3/63, and 6/66 experiments. The results of these experiments are given in Section 4.3.3.

4.3.1.1 Traffic Generation

To make the experimental Tor deployments as realistic as possible, it is necessary to generate traffic. For these experiments, we adopt the same traffic generation strategy as Murdoch [164]. To

[1] Our attempts to start the network with both honest and malicious nodes at once failed, due to the inability of the honest nodes to integrate into the hostile network. The two hour time period allowed the honest nodes time to fully integrate into the routing infrastructure before adding the malicious nodes.

generate a sufficient amount of traffic, we used six dual Xeon class machines running GNU/Linux with 2GB of RAM on a 10 Gbit/s link running a total of 60 clients in the 40 node network, and a total of 90 clients in the 60 node Tor deployment. These clients made requests for various web pages and files of relatively small size (less than 10 MiB) using the HTTP protocol. The interface between the HTTP client and the Tor proxy is made possible by the `tsocks` transparent SOCKS proxy library [39]. The clients also sleep for a random period of time between 0 and 60 seconds and restart (retaining all of their cached state including routing information and entry guards) after completing a random number of web requests so that they do not flood the network.

4.3.1.2 Malicious Node Configuration

To maximize the amount of the anonymized traffic that an attacker can correlate, each malicious router advertises a read and write bandwidth capability of 1.5 MiB/s and a high uptime. Furthermore, each malicious node is rate limited to a mere 20 KiB/s for both data and control packets to make the node a low resource attacker. In terms of the total network bandwidth in each deployment, the addition of malicious nodes contribute a negligible amount of additional bandwidth. In the 2/42 and 3/63 experiments, the malicious nodes comprise 0.38% of each network's bandwidth while actually advertising approximately 22% of the total bandwidth. In the 4/44 and 6/66 experiments, malicious nodes make up 0.76% of the bandwidth while advertising about 36% of the total network's bandwidth.

In addition, each malicious node logs the necessary information for the path linking algorithm, as described in Section 4.2.2. The malicious routers' behavior is aimed at maximizing the probability that it will be included on a circuit.

4.3.2 Measuring Bias in Router Selection

This attack is a direct consequence of Tor's necessity to balance the traffic load over the available bandwidth in the network, thereby introducing bias into the router selection process. To express the degree of bias, or non-uniformity, in Tor's router selection algorithms, we apply a

Table 4.2: The raw number of compromised circuits

	Number of Circuits	
	Compromised	Total
2/42	425	4,774
4/44	3,422	10,199
3/63	535	4,839
6/66	6,291	13,568

measure based on the classical entropy metric from Shannon's information theory [198] to quantify the bias in router selection. Entropy was originally applied to express the degree of certainty that an attacker has regarding whether a particular user is the sender or receiver of a message through the anonymizing network. We use entropy as a metric of bias over the router selection probability distribution.

We apply normalized entropy from Diaz *et al.* [94] which is defined as follows:

$$S_{norm} = -\frac{\sum_{u \in \Psi} p_u \log_2(p_u)}{\log_2 |\Psi|} \tag{4.1}$$

where p_u is the probability of router $u \in \Psi$ being included on a circuit, Ψ is the set of all available routers, and $\log_2 |\Psi|$ is the ideal entropy which corresponds to a system in which all routers are chosen uniformly at random. Normalizing the entropy metric relative to the ideal (or maximal) entropy ensures that $S_{norm} \in [0, 1]$. We apply this metric in our subsequent analysis of router selection bias. More details on information-theoretic anonymity metrics can be found in Chapter 2.4.2.

4.3.3 Experimental Results

In this section, we present the results of the experiments on our isolated Tor deployments. To demonstrate the ability of the attack to successfully link paths through the Tor network, we measured the percentage of the Tor circuits that our path linking algorithm (Section 4.2.2) can correctly correlate.

Using the data logged by malicious routers, our path linking algorithm was able to link a relatively high percentage of paths through Tor to the initiating client. In the 40 Tor router

Table 4.3: The number of predicted and actual circuits compromised in the 40 node PlanetLab network

	Experiments	
	2/42	4/44
Random Selection	0.12%	0.63%
Experimental	8.90%	33.55%
Improvement	$75.65x$	$51.90x$

deployment, we conducted experiments by adding two (2/42) and four (4/44) malicious nodes. The malicious routers composed roughly 4.8% and 9.1% of each network, or 0.38% and 0.76% of each network's bandwidth. In the 2/42 experiment, the malicious nodes were able to compromise approximately 9% of the 4,774 paths established through the network. We then performed the 4/44 experiment, and were able to correlate approximately 34% of the 10,199 paths through the network. Thus, the attack is able to compromise the anonymity of over one-third of the circuit-building requests transported through the experimental network.

These experiments are repeated for a network of 60 Tor routers by adding three (3/63) and six (6/66) malicious nodes. The malicious routers composed about 4.8% and 9.1% of each network, or 0.38% and 0.76% of each network's bandwidth. With only three (3/63) malicious routers, the attack compromises about 11% of the 4,839 paths and in an experiment with six (6/66) malicious Tor routers, the attack compromised over 46% of the 13,568 paths. The results as percentages of compromised paths are given in Tables 4.3 and 4.4. The raw number of compromised circuits in each experiment is given in Table 4.2.

In addition to the correctly correlated paths, there were only 12 incorrectly correlated paths over all the experiments (one false positive in the 3/63 experiment, three in the 4/44 experiment, and eight in the 6/66 experiments). The negligible number of false positives shows that our path linking algorithm is highly accurate; however, the low false-positive rate may also be a result of the relatively light and uniform traffic load that was generated.

In Tables 4.3 and 4.4, the experimental results are compared to an analytical expectation of

Table 4.4: The number of predicted and actual circuits compromised in the 60 node PlanetLab network

	Experiments	
	3/63	6/66
Random Selection	0.15%	0.70%
Experimental	11.06%	46.36%
Improvement	70.97x	65.30x

Table 4.5: The empirical normalized entropy S_{norm} for each experimental configuration

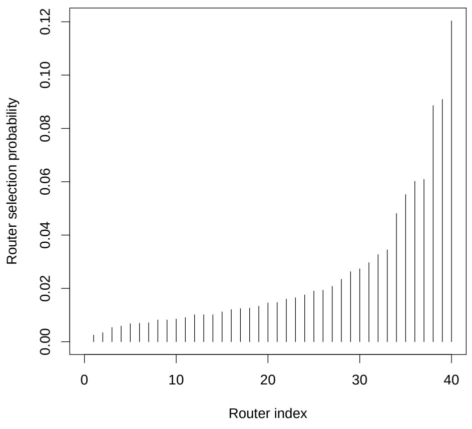
	Experiment Configuration					
	0/40	2/42	4/44	0/60	3/63	6/66
S_{norm}	0.885	0.755	0.579	0.892	0.726	0.504

the percentage of paths that can be compromised by controlling the entry and exit nodes if routers are selected uniformly at random. The analytical expectation is based on a simple combinatorial model originally defined in Tor's design document [103] as $(\frac{c}{N})^2$, where $c > 1$ is the number of malicious nodes and N is the network size.[2] This analytical model does not take into account the fact that a Tor router may be used only once per circuit. Thus, a more precise expectation can be described by $(\frac{c}{N})(\frac{c-1}{N-1})$, $c > 1$. The predicted fraction of compromised circuits if routers were chosen at random is given in Tables 4.3 and 4.4.

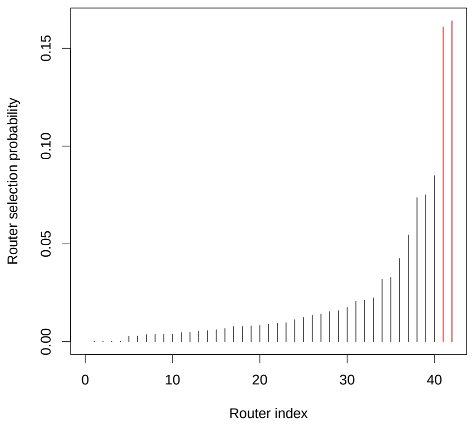
In summary, the distinction between the uniform selection expectations and the experimental results is clear; the experiments demonstrate that Tor's addition of load balancing has caused the number of circuits compromised to increase by 52 and 76 times over uniformly random router selection.

Analyzing router selection bias. We next analyze the extent of the router selection bias by calculating the normalized entropy over the router selection process. We model router selection as an empirically-derived probability distribution P. Let p_i be the observed probability of the i-th router being chosen for a circuit in our experiments, subject to $\sum_{p_i \in P} p_i = 1$.

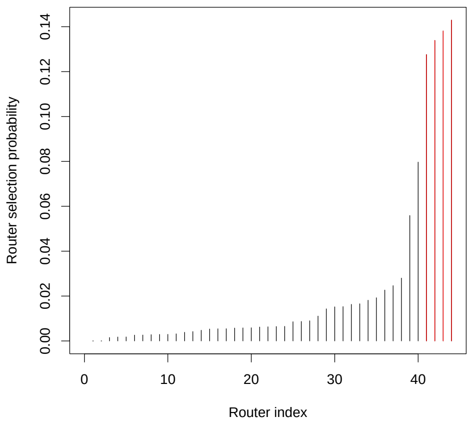
[2] At its inception, Tor did not provide load balancing, *i.e.*, routers were selected uniformly at random.



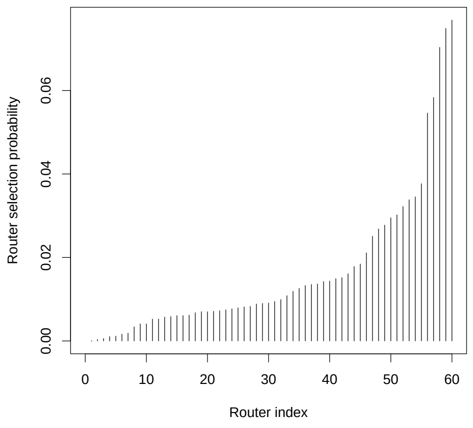
(a) 0/40



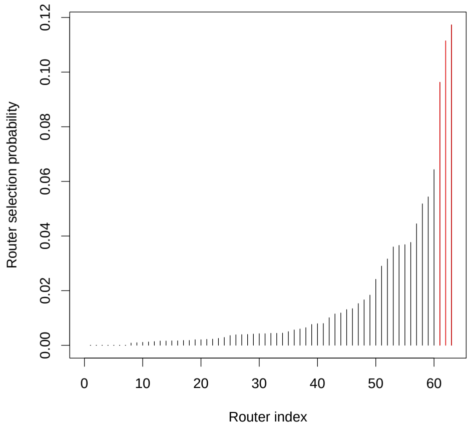
(b) 2/42



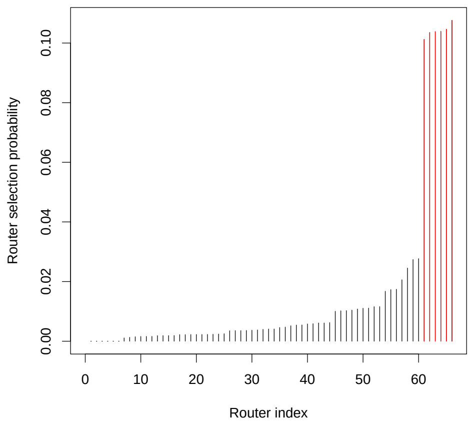
(c) 4/44



(d) 0/60



(e) 3/63



(f) 6/66

Figure 4.3: Empirical router selection probabilities for all routers in each experimental network (malicious nodes are shown in red)

Table 4.5 shows the normalized entropy for each experimental configuration. When no malicious nodes are present, routing bias is minimized: In the 40 node deployment, the normalized entropy is 0.885 and for the 60 node deployment, the entropy is 0.892. These values are less than one, which shows that the router selection is not uniformly random. This is consistent with Tor's by-design tendency to choose routers with a probability that is proportional to their bandwidth claims. As malicious nodes are added to the networks, the normalized entropy decreases to 0.579 in the 40 node network with four malicious routers and 0.504 in the 60 node network with six malicious routers. This measurement demonstrates that the insertion of malicious nodes introduces additional bias in the router selection process. This is a surprising result, since the routing bias should be independent of configuration of the nodes in the network. However, since the attack attempts to maximize the number of circuits compromised by selectively disrupting circuits for which the adversary's nodes do not control both of the circuit's endpoints, this causes circuits to be rebuild until the adversary's nodes *do* control the endpoints. As a consequence, the malicious nodes are chosen far more frequently than they should be chosen without the selective disruption behavior.

Figure 4.3(a)-(e) shows the empirically derived router selection probability density functions (PDFs) for each of the experimental deployments. All experiments show a clear bias toward particular routers. However, in the networks with malicious routers (Figures 4.3(b)(c)(e)(f)), the probability of selecting a malicious router is significantly higher than the probability of selecting a non-malicious router. Across these experiments, the probability of choosing a malicious router is at least twice the probability of selection a non-malicious router. In fact, the 6/66 experimental configuration shows the most extreme bias in router selection. Not surprisingly, this configuration yielded the lowest normalized entropy (0.504) and the highest fraction of compromised circuits (46.46%).

Evaluating the realism of the experiments. In order to argue that the results obtained through our experiments are directly applicable to the larger and more complex real Tor network, it is necessary to show that the dynamics of the experimental deployments are characteristic of

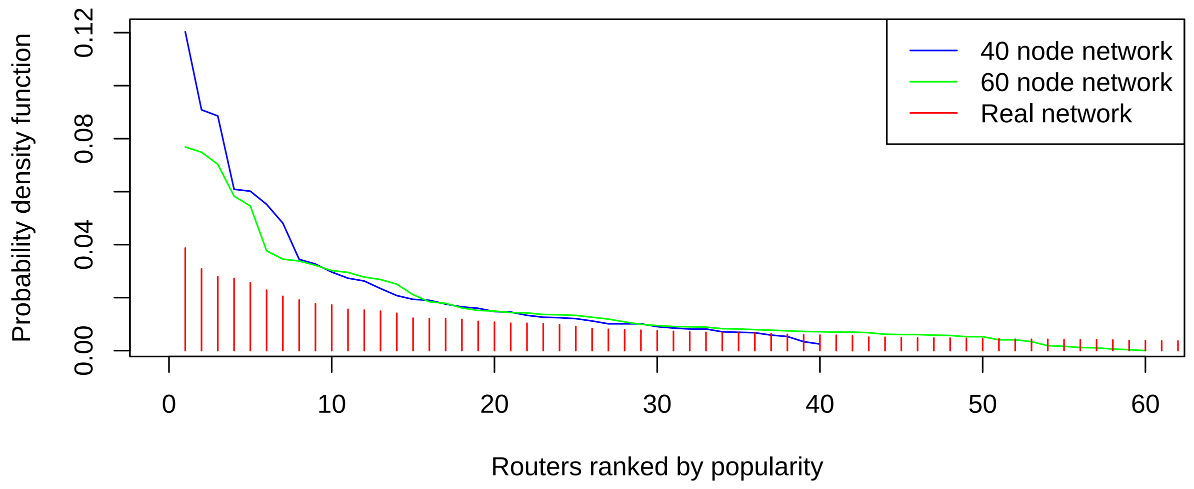


Figure 4.4: Router selection probability distributions for the non-adversarial 40 node, 60 node, and real Tor networks

the real Tor network. In particular, we would like to show that the router selection probability distribution on the smaller experimental networks is similar to the router selection probability distribution on the real Tor network.

In Chapter 3, we present observations and measurements obtained by monitoring a Tor router on the real Tor network. Of particular interest here is the router selection probability distribution based on observations of the frequency with which other routers appear on circuits adjacent to our router (see Figure 3.3). Using this information, we can construct a router selection probability distribution for the real Tor network. Figure 4.4 shows the empirical router selection probabilities measured in the non-adversarial 40 and 60 node networks. For comparison, the router selection probability distribution observed on the real Tor network is plotted. This graph shows that the smaller experimental networks have a greater bias toward the highest bandwidth routers in comparison to the real Tor network. The observed selection distribution from the real network shows a slower, more gradual decline in selection probabilities. This is simply a result of the necessity to scale down the experiments, due to the node constraints on PlanetLab. However, we believe that our expectation of the attack's success is a best estimate of the attack's performance on a larger network.

Scaling the attack to the real Tor network. To better understand the scaling properties of the attack, we present an analysis of the expected number of malicious nodes needed to achieve results similar to our experiments on the real network. The 2/42 network configuration compromised almost 9% of all circuits with about 0.32 probability of choosing a malicious router for a circuit. The 4/44 network achieved nearly 34% compromise with a 0.54 probability of choosing a malicious router. The 3/63 network exhibited about 11% compromise with a 0.32 probability of choosing a malicious router. Finally, the 6/66 network achieved a 46% compromise rate with a 0.60 probability of choosing a malicious router.

By extrapolation, we can estimate the number of malicious nodes that would be necessary to achieve similar attack performance on the real network consisting of roughly 1,400 active routers (as of October 2009). The probability of malicious nodes appearing on a circuit given a total number of malicious nodes c can be estimated by evaluating the definite integral of the router selection probability distribution over a fixed interval. The calculation is as follows:

$$P_{agg} = \operatorname*{argmin}_{c} \int_0^c f_P(x)\,dx, \text{ such that } P_{agg} \geq X \tag{4.2}$$

where $f_P(x)$ is the empirical router selection probability density function and $c > 0$ is the number of highest performing nodes satisfying the constraint $P_{agg} \geq X$, where $0 \leq X \leq 1$ is the target aggregate probability of inclusion on a circuit.

From the data presented in Chapter 3, to obtain a $P_{agg} = 0.32$ probability of being on a circuit on the real network, it is necessary to control only the 15 highest performing routers. This probability corresponds to about 10% circuit compromise rate from our experiments. 15/1400 routers is roughly the top 1% of the network. Similarly, to obtain $P_{agg} = 0.55$ probability of appearing on a circuit, it is necessary to control the top 45/1400 routers, or the top 3% of the network. The scaling properties show a super-linear resource requirement in comparison to our smaller experiments. This is because the router selection probability distribution for the real Tor network is, in fact, less skewed than the distribution for our experiments.

Another, perhaps more meaningful, way to estimate the adversary's resource requirement and expectation of attack success is in terms of bandwidth. An adversary contributing between 0.38% and 0.78% of the network's aggregate bandwidth is able to compromise up to 46% of the circuits for new Tor proxies. Suppose that the Tor network has a bandwidth distribution as observed on the real network in August 2006 (see Table 4.1) with 104 MiB/s of total available bandwidth. If an adversary contributes an upper bound of an additional 1% of bandwidth — only 1.04 MiB/s — then the attack should compromise a similar fraction of circuits as in our experiments.

4.3.4 Attack Discussion

It is worth asking why the earlier analytical model based upon uniform router selection that predicted a strong resistance to this type of attack does not match our experimental results. Besides enabling malicious nodes to lie about their resources, the primary issue is that the analytical model assumes resource homogeneity across the set of Tor nodes, when in fact the real Tor network has a heterogeneous resource distribution (see Table 4.1). As a result, routers are not chosen with an equal probability; those with higher bandwidth claims are chosen more frequently. Also, our attack used selective path disruption to cause circuits to fail, which is not considered in the analytical model.

Furthermore, since routers have finite resources, they must reject new connections once their resources are consumed. To make matters worse, most of the available bandwidth from low resource nodes may be exhausted by protocol control overhead, which decreases the effective size of the network while increasing the probability of a malicious node being selected. This means that if the Tor network becomes heavily congested, it would magnify the effectiveness of our attack.

4.4 Effects of Exit Bandwidth Distribution

The experiments from PlanetLab show that within a realistic network setting, Tor is vulnerable to end-to-end traffic correlation attacks by an adversary who falsely advertises high bandwidth and uptime values. While these experiments offer value in their realism (*i.e.*, they run real Tor code on an overlay testbed with real network conditions), there are several interesting aspects to the attack that we simply cannot evaluate due to the amount of time necessary to construct and deploy real, live experiments on PlanetLab. The PlanetLab experiments made simplifying assumptions about the distribution of traffic on the network – we assumed only HTTP with small and large files to simulate interactive traffic and bulk file transfers. However, from the data presented in Chapter 3, we found that protocols other than HTTP are common in Tor also. Thus, we wish to determine if all protocols exhibit the same vulnerability to this attack, or if some are more robust while others are at higher risk.

In this section, we wish to answer the following lines of questions regarding the attack: How does the distribution of traffic on Tor effect the attack's performance? Are certain protocols more robust to attack? Also, are certain protocols more vulnerable?

4.4.1 Experimental Setup

We evaluate how vulnerability to end-to-end circuit compromise varies between applications by simulating Tor's router selection algorithm as described in Chapter 2.3.3.3. We obtained a snapshot of the active routers from Tor's trusted directory servers on May 31, 2009. This provides information such as each router's bandwidth claim, exit policy, entry guard status, and uptime. Using this data in our path selection simulation ensures that our results are indicative of what a client would experience while participating in the real Tor network. This snapshot consists of 1,444 total routers with 403.3 MiB/s of total bandwidth marked as `active` and `valid` by the directory servers. Of these routers, 770 routers with 326.9 MiB/s of total bandwidth are marked as `stable`.

The path selection simulator generates 10,000 circuits for each of the following protocols (and default port numbers): FTP (21), SSH (22), Telnet (23), SMTP (25), HTTP (80), POP3 (110), HTTPS (443), Kazaa P2P (1214), BitTorrent tracker (6969), Gnutella P2P (6346), and eDonkey P2P (4661). This list represents a diverse set of popular applications. The path selection simulator ensures that an entry guard is chosen for the first router of the circuit. Also, an exit router is chosen that allows connections to the default port for the application being transported. Each malicious router advertises an exit policy that allows the client's application to exit. Finally, the malicious routers are *passive*; they do not selectively disrupt circuits for which the endpoints are not compromised.

4.4.2 Experimental Results

For all applications considered, the path compromise rate increases with additional malicious routers. Figure 4.5 shows that the compromise rates for web browsing (HTTP), outgoing e-mail (SMTP), and peer-to-peer file sharing with BitTorrent vary directly with the increase in malicious bandwidth. Interestingly, the compromise rate is higher and increases faster for certain applications. For example, an adversary with 60 MiB/s of malicious bandwidth can compromise 7.0% of all circuits that transport HTTP traffic. However, this same adversary can compromise between 18.5–21.8% of all circuits transporting SMTP or BitTorrent traffic. The complete results are given in Table A.1 in Appendix A. The protocols that exhibit a significantly higher path compromise rate are marked with a † in the table. For each application, the compromise rate shows diminishing returns as the adversary has more than 760 MiB/s of bandwidth.

We identify two factors that explain this difference in path compromise performance: (1) exit bandwidth is not uniformly distributed among all application-layer protocols, and (2) long-lived circuits require stable routers, which reduces the number of candidate routers when choosing a path.

Exit bandwidth is not uniformly distributed. Since Tor allows router operators to specify exit policies, operators may choose to block certain ports to curtail complaints of abuse that would

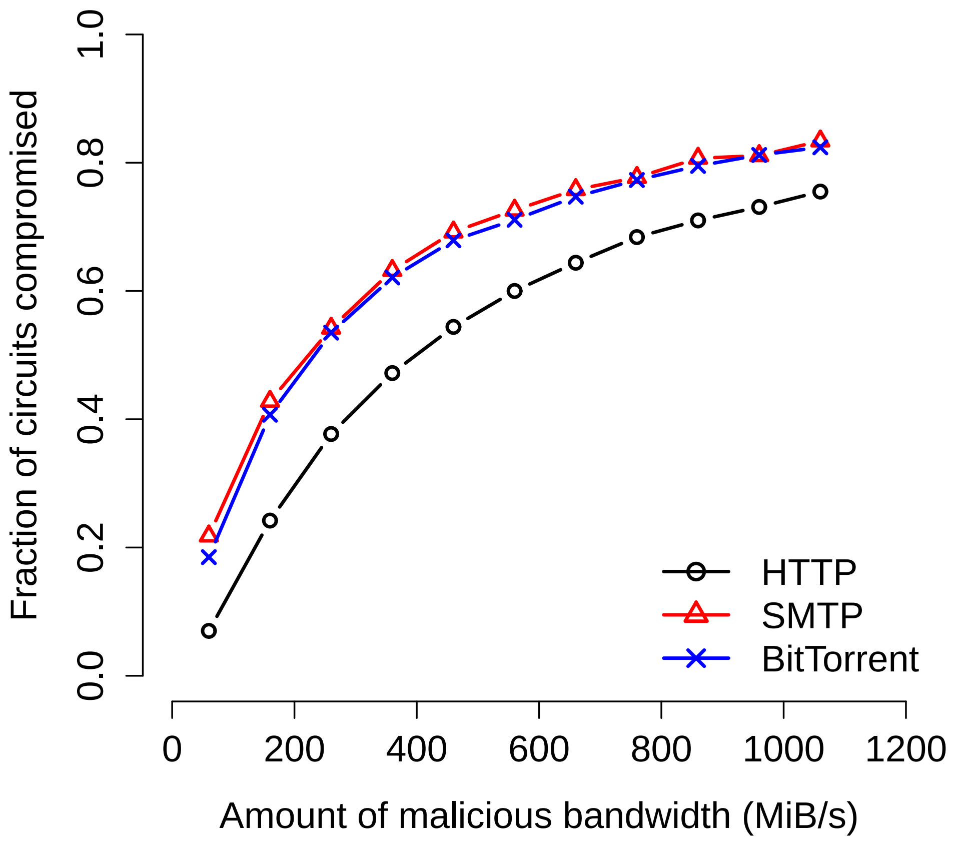


Figure 4.5: Fraction of circuits compromised for web browsing, outgoing e-mail, and peer-to-peer file sharing traffic

be directed at the Tor router. For example, outgoing e-mail (SMTP) ports may be blocked to prevent spam and popular peer-to-peer file sharing ports may be blocked to eliminate DMCA take-down notices that are often distributed in response to file sharing of copyright-protected content. In an attempt to protect Tor router operators, Tor's recommended (default) exit policy blocks the ports commonly associated with SMTP and peer-to-peer file sharing protocols.

Table 4.6 shows the distribution of exit bandwidth for web browsing, outgoing e-mail, and BitTorrent (exit bandwidth for all applications is given in Table A.2). BitTorrent and SMTP have the fewest routers and least amount of exit bandwidth among all applications considered.

Table 4.6: Tor's distribution of exit bandwidth for web browsing, outgoing e-mail, and peer-to-peer file sharing

	SMTP	*HTTP*	*BitTorrent*
Number of routers	13	625	23
Exit bandwidth (in MiB/s)	1.4	116.9	9.1

Since routers are selected in proportion to their bandwidth claims, the malicious routers constitute a significant fraction of the available exit bandwidth for these applications. Consequently, the malicious exit routers appear frequently at the exit position, increasing the probability that the circuit will be compromised. Even if the adversary only controls the exit router, they may be able to observe unencrypted traffic leaving the Tor network. For HTTP traffic, an adversary with six routers appears as the exit 33.6% of the time and an adversary with 16 routers (160 MiB/s) controls the exit 56.5% of the time. For FTP, an adversary with six routers (60 MiB/s) controls the exit router 46.7% of the time and an adversary with 16 routers (160 MiB/s) controls the exit 70.7% of the time. This is significant because an attacker can observe identifying information and even login credentials in plaintext leaving their exit router.

It may be tempting to conclude that the additional path compromise threat exhibited by SMTP and peer-to-peer file sharing protocols is not a significant concern because these protocols are not popular with Tor in practice. However, in Chapter 3 we found that BitTorrent is the most popular application after HTTP and HTTPS in terms of number of connections. Thus, an adversary can expect to compromise a significant number of BitTorrent tracker circuits by deploying only 6-16 (60-160 MiB/s) routers.

Long-lived circuits require stable routers. Recall that persistent applications with long-lived sessions build circuits only with stable routers. In these experiments, only 770/1,444 routers are marked as stable by the directory servers. FTP and SSH are regarded as long-lived and there are only 184 and 197 routers, respectively, that are suitable to exit these protocols (with 65.4 MiB/s and 73.7 MiB/s, respectively). Consequently in Figure 4.6, FTP and SSH exhibit a higher path compromise rate than short-lived applications such as HTTP, POP3, and HTTPS. However, the

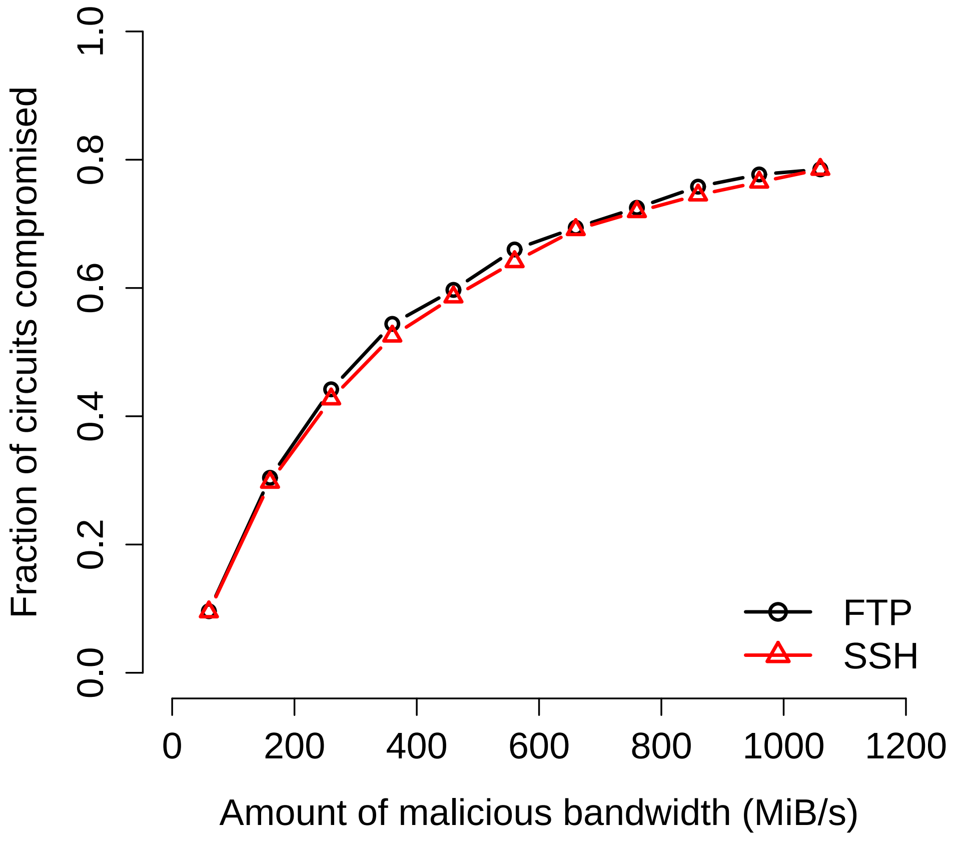


Figure 4.6: Fraction of circuits compromised for FTP and SSH traffic

path compromise is not as high as SMTP or the peer-to-peer file sharing protocols because the long-lived circuits have significantly more available exit bandwidth.

Discussion. These results indicate that certain applications may be more vulnerable to endpoint compromise than others, due to the non-uniform distribution of exit bandwidth among applications. We regard the results presented here as a lower bound on the attack's attainable performance, since an active adversary could selectively disrupt circuits that are not compromised to increase the chances of compromising future circuits. We consider the effects of selective disruption, in addition to the potential role that entry guards play in mitigating these attacks, in Chapter 5.

4.4.3 Mitigating Circuit Compromise for Limited Bandwidth Protocols

In this section we have shown that some applications are inherently more vulnerable to path compromise than others due to the non-uniform distribution of Tor's exit bandwidth. One solution to mitigate the threat is to reduce or eliminate the bias toward high bandwidth routers for these protocols. Using the Snader-Borisov router selection algorithm described in Chapter 2.5.5.1, Tor clients can choose routers with less bias toward high bandwidth routers than Tor's current router selection algorithm offers. For maximum protection, the clients could choose routers uniformly at random.

However, since the number of routers available to exit the peer-to-peer and SMTP traffic is so low, even uniform router selection may not significantly decrease the adversary's ability compromise the exit position. However, it does significantly reduce the adversary's ability to control both endpoints. Suppose there are $c > 1$ malicious routers, N total routers, and E available exit routers for a particular port. Uniform router selection gives the adversary an expected path compromise rate determined by $(\frac{c-1}{N-1})(\frac{c}{E})$. For BitTorrent, an adversary with six malicious routers can compromise only 0.09% of circuits, which is a significant improvement over the 18.5% circuit compromise rate observed using Tor's default router selection algorithm. However, this increase in security comes at a performance cost.

4.5 Attack Extensions

Having presented the basic ideas behind our attacks, we consider further attack variants and improvements, such as attacking existing Tor clients instead of only new Tor clients, router advertisement flooding, and watermarking attacks.

Compromising existing clients. Clients that exist within the network before the malicious nodes join will have already chosen a set of entry guard nodes. We present two methods to compromise the anonymity of existing clients. First, if an attacker can observe the client (*i.e.*, by sniffing the client's 802.11 wireless link), he/she can easily deduce the entry guards used by a particular client. The adversary can then make those existing entry guards unreachable or perform a denial-of-service (DoS) attack on these entry guards, making these nodes unusable. This forces the client to select a new list of entry guards, potentially selecting malicious Tor routers. Another method to attack clients that have a preexisting list of entry guard nodes would be to DoS a few key stable nodes that serve as entry guards for a large number of clients. This would cause existing clients to replace unusable entry guards with at least one new and potentially malicious entry guard node.

Improving performance under the resource reduced attack. One concern with the resource-reduced attack that we describe in Section 6.1.2 is that, by itself, the attack can seriously degrade the performance of new Tor clients. The degradation in performance could then call attention to the malicious Tor nodes. Naturally, the basic attack in Section 6.1.2 would be completely indistinguishable from a performance perspective since the basic adversary does not lie about its resources.

The first question to ask is whether poor performance under an adversarial situation is a sufficient protection mechanism. We believe that the answer to this question is "no" — it is a poor design choice for users of a system to have to detect an attack based on poor performance. A better approach is to have an automated mechanism in place to detect and prevent our low-resource attack. Furthermore, a resource-reduced adversary could still learn a significant amount of private information about Tor clients between the time when the adversary initiates the attack and time when the attack is discovered.

The second, more technical question is to ask what a resource-reduced adversary might do to improve the perceived performance of Tor clients. One possible improvement arises when the attacker wishes to target a particular client. In such a situation, the adversary could overtly deny service to anyone but the target client. Specifically, an adversary's Tor nodes could deny (or passively ignore) all circuit-initiation requests except for those requests that the target client initiates. This behavior would cause the non-target clients to simply exclude the adversary's nodes from their lists of preferred entry guards, and would also prevent non-target clients from constructing circuits with the adversary's nodes as the middle or exit routers. Since circuit-formation failures are common in Tor [174], we suspect that this attack would largely go unnoticed.

Improving performance. It is possible to target a specific client to monitor, thereby improving the throughput performance of the malicious nodes that exist on its circuit. This would cause the victim client to keep sending traffic along the compromised circuit while the malicious nodes can determine the sender, receiver, and perhaps the contents (if the payload is not encrypted at the application layer) of the communication session. Forcing other clients to use non-adversarial Tor servers would also help keep from drawing attention to the adversarial nodes.

Displacing honest entry guards. Recall that Tor uses special entry guard nodes to protect the entry of a circuit from selective disruption. In order to be marked by the directory servers as a possible entry guard, a Tor router must advertise an uptime and bandwidth greater than the median advertisements (in `Tor` version 0.1.1.23). Another attack, which is a variant of the Sybil attack [108], can be conducted by flooding the network with enough malicious routers advertising high uptime and bandwidth. On our isolated Tor network, we successfully registered 20 routers all on a single IP address and different TCP port numbers.[3] Flooding the network with false router advertisements allows a non-global adversary to effectively have a "global" impact on Tor's routing structure. Namely, this attack increases the median threshold for choosing entry guards, thereby, preventing benign nodes from being marked as potential entry guards. This attack could help guarantee that only malicious nodes can be entry guards.

Compromising only the entry guard. As another extension to our attack, suppose that an adversary is less interested in breaking anonymity in general, but is instead particularly interested in correlating Tor client requests to a specific target website (such as a website containing controlled or controversial content). Suppose further that the adversary has the ability to monitor the target website's network connection; here the adversary might have established the target website to lure potential clients, or might have obtained legal permission to monitor this link. Under this scenario,

[3] The trusted directory servers currently (as of `Tor` version 0.1.1.23) have no limits as to the number of routers that can be hosted on a single IP address. In theory, an attacker can register up to $2^{16} - 1$ Tor routers to the same IP address.

an adversary only needs to compromise an entry node in order to correlate client requests to this target website. The critical idea is for the entry router to watermark a client's packets using a time-based watermarking technique, such as the technique used in [228]. The adversary's malicious entry routers could embed a unique watermark for each client-middle router pair. A potential complication might arise, however, if the client is using Tor to conceal simultaneous connections to multiple websites, and if the circuits for two of those connections have the same middle router.

4.6 Traffic Analysis on the Live Tor Network

In order to understand whether our low resource traffic analysis techniques are effective in de-anonymizing clients on the live Tor network, we deploy our own routers and clients on the live network and launch our attacks. Of most interest, we wish to confirm that only circuit construction messages are sufficient to link clients and destinations, even with a realistic traffic load.

Experimental setup. We deploy two Tor routers running version `Tor 0.2.1.20`. hosted on a 100 Mb/s network link onto the live Tor network in February 2010. Each router has a distinct configuration:

(1) One Tor router is configured as a non-exit and after roughly ten days of uninterrupted operation, it obtained the `Guard` flag from the authoritative directory servers.

(2) A second Tor router is configured with the default exit policy.[4]

During their operation, each router sustained roughly 24 Mb/s of traffic.

To evaluate the expected success of traffic analysis tasks, we operate our own Tor clients and attempt to link their circuits to their destinations. Upon building a circuit, each client downloads `www.google.com`, tears down the circuit, and repeats this procedure. To preserve users' privacy,

[4] Ports often associated with outgoing e-mail, peer-to-peer file sharing applications, and high security risk services are blocked.

we ignore traffic at the entry guard that is not produced by one of our clients.[5] Note that we do not retain any linkable data nor do we attempt to de-anonymize any other clients but our own.

Traffic analysis methodology. We apply our traffic analysis technique described in Chapter 4.2 in which circuits are linked by their circuit building messages before the clients send any data cells. This approach leverages the fact that Tor's circuit establishment procedure sends a fixed number of circuit building messages in an identifiable pattern.

Results. On the live Tor network, our clients build a total of 1 696 circuits that always use our entry guard. Of these 821 circuits use our exit router and 875 circuits use a different exit router. This setup allows us to count the number of false positives that occur during circuit linking. The middle routers are chosen according to Tor's default router selection algorithm. Using our low resource circuit linking algorithm, we correlate clients and destinations with 97% accuracy, 0.6% false negatives (6 false negatives in total), and 6% false positives (52 false positives in total). We regard these results as a lower bound on attainable traffic analysis success, as it should be possible to increase the accuracy by also using data cells to link circuits. Also, we observe that circuits that use a popular (*i.e.*, high bandwidth) middle router tend to be more prone to false positives. Thus, an attacker who sees a positive result with a low bandwidth middle router can be more confident in the result. To conclude, these results re-enforce the danger that these attacks pose to real Tor users.

4.7 Defenses

Non-global, but high-resource (uptime, bandwidth), adversaries seem to pose a fundamental security challenge to any high-performance, multi-hop privacy enhancing system that attempts to efficiently balance its traffic load, and we welcome future work directed toward addressing this challenge. We consider, however, methods for detecting non-global *low-resource* adversaries.

[5] This experiment was deemed not to involve human subjects by the University of Colorado's Institutional Review Board. See Certificate of Determination: Not Human Subject Research. PI: Kevin Bauer, Protocol: 1209.44, Protocol Title: "Research Study to Better Understand the Tor Anonymizing Network," Date: 01/14/2010.

In order to mitigate the negative effects of false routing information in the network, it is necessary to devise a methodology for verifying a router's uptime and bandwidth claims. Here, we provide a brief overview of some potential solutions and alternative routing schemes.

4.7.1 Resource Verification

Verifying uptime. A server's uptime could be checked by periodically sending a small heartbeat message from a directory server. The additional load on the directory server would be minimal and it could effectively keep track of how long each server has been available.

Centralized bandwidth verification. Since Tor relies upon a centralized routing infrastructure, it is intuitive to suggest that the trusted centralized directory servers, in addition to providing routing advertisements on behalf of Tor routers, also periodically verify the incoming bandwidth advertisements that are received from Tor routers. The directory server could measure a router's bandwidth before publishing the routing advertisement and correct the value if it found that the router does not have its claimed bandwidth. The difficulty with this approach is that it cannot detect selectively malicious nodes. Therefore, it is necessary for the bandwidth verification mechanism to continuously monitor each node's bandwidth. Due to the significant traffic load that would be placed upon the few centralized directory servers, this centralized bandwidth auditing approach would create a significant performance bottleneck.

Distributed bandwidth verification. In order to detect false bandwidth advertisements, it may be tempting to augment the routing protocol to allow Tor routers to proactively monitor each other. Anonymous auditing [209], where nodes anonymously attempt to verify other nodes' connectivity in order to detect collusion, has been proposed as a defense against routing attacks in structured overlays. A similar technique could be designed to identify false resource advertisements. However, this technique is also insufficient at detecting selectively malicious nodes. In addition, this approach introduces additional load in the network and could result in significant performance degradation. Borisov and Snader propose that routers opportunistically probe other routers' bandwidth rather than rely on self-reported bandwidth claims [211].

Distributed reputation system. Reputation systems have been proposed for anonymity systems within the context of reliable MIX cascades [106]. One could envision a reputation system similar to TorFlow [176], that actively verifies the directory server reported bandwidth claims for each Tor router and dynamically updates each bandwidth claim. Selectively malicious nodes are still difficult to detect with such a reputation system. Snader and Borisov propose EigenSpeed [212], a system that allows routers to opportunistically estimate other routers' bandwidth capacities in a distributed fashion and share this reputation information in a manner similar to the EigenTrust reputation algorithm [137].

4.7.2 Mitigating Sybil Attacks

In order for any reputation system to be effective, it is necessary to address the Sybil attack [108]. Recall that the directory servers place no constraints upon the number of Tor routers that may exist at a single IP address (as of `Tor` version 0.1.1.23). This can be exploited to effectively replace all entry guards with malicious nodes (see Section 4.5). To help mitigate this kind of attack, the directory servers should limit the number of routers introduced at any single IP address. Furthermore, enforcing location diversity increases the resources required to perform this attack [114]. Following the initial disclosure our results in a technical report [60], Tor adopted countermeasures that (1) allow only three Tor routers to be hosted at any single IP address and (2) dictate that circuits may not include more than one router from a particular class B address space [57].

4.7.3 Alternative Routing Strategies

Since Tor's desire to efficiently balance its traffic over the available resources in the network has left it vulnerable to traffic correlation attacks, it is prudent to consider alternate routing strategies that may provide adequate load balancing while preserving the network's anonymity.

Proximity awareness. Secure routing based on *proximity awareness* has been proposed in peer-to-peer networks [72]. In such a routing strategy, the next hop is computed by minimizing a distance metric, such as round trip time (RTT). Proximity-based routing may over-optimize and cause circuits to be built deterministically. Also since paths are multi-hop and source routed, the client would need distance metrics for the first hop to the second hop and the second hop to the third hop. In addition, these metrics must be verified. Finally, proximity routing seems to be incompatible with enforcing location diversity. Sherr *et al.* proposed that clients use network coordinate systems to build circuits based on link characteristics such as latency, jitter, loss rate, and autonomous system traversals rather than node characteristics such as bandwidth [200]. Such a router selection mechanism could be more challenging for an adversary to influence.

Loose routing. Loose routing in anonymity systems has been proposed in Crowds [189], where path lengths are non-deterministic since each hop chooses to forward to another intermediate hop probabilistically. This strategy places a great amount of trust on the entry nodes. A malicious entry node could simply route all traffic immediately to the exit server. In fact, loose source routing was also proposed in the original onion routing paper [122].

Local reputation-based routing. Another scheme could be to initially choose paths with a uniform probability and over time, maintain local reputation information for all nodes used in a path. At the start, the performance would be expectedly poor, but over time, as clients begin to choose high quality Tor routers, they can begin to optimize for performance. This approach is not vulnerable to false advertisements.

4.7.4 Mitigating Selective Disruption DoS Attacks

Danner *et al.* propose an algorithm to detect selective disruption DoS attacks in Tor based on probing circuits [91]. This approach could be employed to identify malicious routers who use DoS attacks to increase their ability to compromise paths.

4.8 Broader Impact

Our primary objective in this chapter is to empirically evaluate how contemporary low latency anonymity networks such as Tor are vulnerable to de-anonymization attacks via end-to-end traffic correlation. While is it well-known that low latency anonymity networks are vulnerable to traffic correlation attacks if the endpoints are watched or controlled by an adversary or a pair of colluding parties [197, 206], we show that load balancing optimizations and an insecure entry guard design make such attacks practical for adversaries with few nodes and little bandwidth. Since the initial disclosure of the vulnerabilities and exploits presented in this chapter, there has been a significant amount of effort aimed at improving Tor's resilience to such attacks.

To secure entry guards against our honest entry guard displacement attack, we suggested that Tor's directory servers limit the believable uptime values that are reported [58]. Later, uptime was replaced by mean-time-between-failures, a more traditional reliability metric, for determining entry guard assignment [157]. In addition, Tor has adopted a router bandwidth monitoring strategy using periodic active measurements and subsequent bandwidth re-weighing [176]. Lastly, to reduce the number of routers that are controlled by a single (potentially malicious entity), Tor adopted our recommendations that the number of routers per IP address and the number of routers per /16 network be limited [57].

Subsequent work has sought to further improve Tor's resilience to our attacks. To mitigate the low resource attacker's ability to inflate their bandwidth claims, Snader and Borisov proposed a tunable router selection strategy, where users who want greater protection against these attacks should choose routers more uniformly at random and users who want the performance benefits of bandwidth-weighted routing should skew their selections toward higher bandwidth routers [211, 213]. Also, to reduce an adversary's ability to influence the router selection process, alternate selection metrics have been proposed that rely on link-based attributes (such as latency, jitter, loss rate, or AS-traversals) instead of bandwidth-weighting [199–202]. Also, progress has been made toward detecting the selective disruption attacks that we describe [91]. While collectively this body of work has improved Tor's resilience to the low resource variant of the attacks, Tor still remains vulnerable to attack by an adversary who controls high bandwidth nodes distributed among several networks.

4.9 Summary

We present a low-resource end-to-end traffic analysis attack against Tor that can compromise anonymity before any payload data is sent. The attack stems from Tor's tendency to favor routers that claim to be high-resource with high-uptime in its routing process in an attempt to optimally balance the traffic load. However, since there is no mechanism to verify resource claims, we experimentally show that it is possible for even a low-resource adversary to compromise an unfairly large fraction of the circuit-building requests through the network.

In addition, we illustrate the feasibility of displacing all entry guard nodes with malicious nodes, thereby having a global effect upon the Tor's routing mechanism. Attack extensions are presented that further reduce the cost of launching this attack.

To mitigate the low-resource variety of these attacks, we propose solutions aimed at verifying all bandwidth and uptime claims. However, these attacks highlight the inherent challenge in designing an anonymity-preserving reputation system that is robust to a selectively malicious adversary.

Since this chapter shows how anonymity and efficient bandwidth use appear to be diametrically opposed, our hope is that these attacks motivate further research in the area of designing and implementing optimal routing algorithms in Tor-like overlay networks that deliver a high level of performance without compromising the security of any aspect of the system.

Chapter 5

Improving Performance (and Security) with Two-Hop Paths

Decisions made in the design of low-latency anonymizing networks frequently involve achieving a proper balance between security and performance. For example, Tor currently does not employ padding or add intentional delays in an attempt to provide performance sufficient to support interactive applications such as web browsing. However, as we experimentally demonstrated in Chapter 4, this decision has made Tor vulnerable to end-to-end traffic correlation attacks [148,206].

Another key design decision is path length. Tor employs a decentralized architecture of precisely three routers in order to mitigate any single router's ability to link a source and its respective destination. While three-hop paths may offer security benefits relative to shorter paths, they have a performance cost in terms of high latencies and slow downloads, which negatively impacts Tor's usability.

Through critical analysis, simulation fueled by data obtained from the live Tor network, and experiments conducted on the live Tor network, we evaluate the advantages and disadvantages of two-hop and three-hop paths from security and performance perspectives. In addition, we identify and discuss a variety of open issues related to different path length choices.

Path length and security. We first consider an adversary who employs selective disruption tactics (as in [61, 67, 91, 223]) to force clients onto adversary-controlled circuits. Through simulation of Tor's router selection algorithms fueled by real router data obtained from Tor's directory servers, we show that three-hop paths are up to 7% more vulnerable to path compromise than two-hop paths under the same attack. In addition, irrespective of path length, we show that entry guards, originally developed to mitigate the threat of the predecessor attack [236], offer significant protection from end-to-end correlation attacks by increasing the resources necessary for an attacker to compromise the first router on circuits.

However, one potential disadvantage of a two-hop design is that exit routers can trivially learn clients' entry guards (since they communicate directly). Next, we empirically demonstrate that even with three-hop paths, malicious exit routers can still identify clients' entry guards by deploying malicious middle-only routers. For the current Tor network, our results indicate that an adversary with only ten malicious exit routers and 50 middle-only routers can learn the entry guards for nearly 80% of all circuits constructed. We lastly analyze entry guard usage data and quantify the potential to identify clients through knowledge of their entry guards. To mitigate some of the security risks associated with shorter paths, we propose *path length blending* and describe a technique that makes two-hop and three-hop paths appear indistinguishable from a network observer's perspective.

Path length and performance. In addition to a security analysis of path length, we show that shorter paths offer substantially better performance as perceived by end-users. We perform an analysis of typical web browsing and show that most clients experience a 35% or better improvement in download time with two-hop paths. Our findings indicate that users who wish to experience improved performance while maintaining a reasonable degree of anonymity should switch to two-hop paths.

5.1 Security Analysis

In this section, we first evaluate how an adversary's ability to compromise circuits varies between two-hop and three-hop paths. Next, we explore how two-hop paths reveal circuits' entry guards and discuss the potential for adaptive surveillance attacks. We also describe an attack where an adversary with few exit routers and comparatively many middle-only routers can identify the entry guards on a large fraction of circuits. Lastly, the amount of information about clients that is revealed by entry guard knowledge is analyzed.

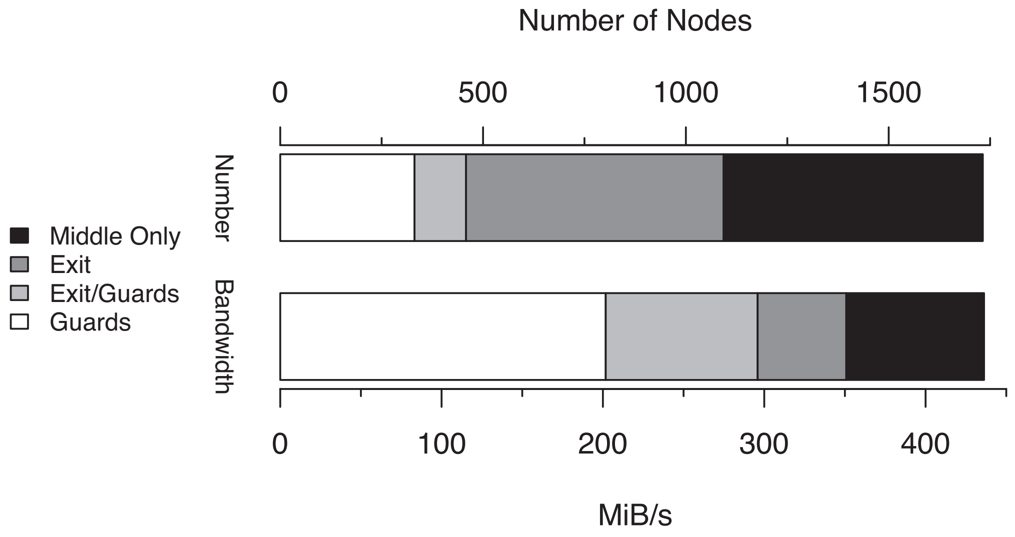


Figure 5.1: Observed router counts and bandwidth distribution used for simulations

5.1.1 Path Compromise

Prior work showed that selective disruption attacks can degrade anonymity in Tor-like networks [67]. However, it is unclear how path length effects an adversary's ability to disrupt circuits. To better understand this relationship, we conduct experiments through simulation of Tor's router selection algorithm (described in [101]).

Simulation setup. We adopt a simulation methodology similar to Murdoch and Watson [167] in which an increasing number of malicious routers are injected into the network to compromise circuits. We simulate 1 000 clients as follows. First, each client chooses precisely three entry guards to use on all circuits. Next, each builds 100 circuits of both length two and three that are suitable for transporting HTTP traffic.[1] We run experiments in which a number of malicious routers between 5 and 50 are added to the network. Each malicious router has 10 MiB/s of bandwidth,[2] offers stability and performance to be an entry guard,[3] advertises an exit policy that allows port 80 to exit, and is operated on a distinct /16 subnet from every other malicious router.

[1] We simulate HTTP exit traffic (port 80) because prior work found it to be the most common type of traffic on the real Tor network by connection [152, 158].

[2] Currently the largest believable bandwidth value.

[3] Obtaining the `Guard` flag only requires that the router demonstrate stability for a relatively short period of time. We anecdotally found that a new router on a high bandwidth link can obtain the `Guard` flag after running for roughly seven days.

To fuel the simulations, a typical snapshot of all routers was obtained from a directory server on January 6, 2010. Summarized in Figure 5.1, this snapshot consists of 1 735 total routers marked as `Valid` and `Running`. Note that the snapshot has sufficient entry guard and exit bandwidth such that both entry guards and exit routers may by used for any position of the circuit, provided that they have the appropriate flags.[4]

Results. We first characterize the effect of path length on an adversary's ability to compromise circuits through selective disruption tactics. Figure 5.2 shows the fraction of circuits that are compromised as the number of malicious routers and amount of adversary-controlled bandwidth increases. Note that for attackers that do not apply selective disruption, the circuit compromise rate is the same regardless of whether three- or two-hop paths are used. For an adversary who employs selective disruption, if the client has the misfortune of choosing three malicious entry guards, their circuits are always compromised. Conversely, if a client chooses no malicious entry guards, their circuits are never compromised.

For example, when there are 10 malicious routers and clients use three-hop paths, 38% of clients choose no malicious guards, 47% choose one malicious guard, 13% choose two malicious guards, and only 1% choose three malicious guards. Of the clients that choose one or two malicious guards, their circuits are compromised 63% and 85% of the time, respectively. By design [172], entry guards offer significant protection against circuit compromise, since clients that choose no malicious entry guards are safe and the threat increases with the selection of additional malicious entry guards.[5]

Across all malicious router configurations, the fraction of circuits compromised is up to 7% higher for three hops relative to two-hop paths. With three-hop paths, when the client selects one or two malicious guards, circuits are disrupted when they use a non-malicious guard and **either** a malicious middle or exit (or both). However with two-hop paths, if the client does not use

[4] In order to ensure that there is enough available entry guard and exit bandwidth, Tor's router selection algorithm requires that there is at least $T/3$ bandwidth available for both guards and exits (where T is the total bandwidth). If the entry guard bandwidth is less than $T/3$, then entry guards may only be used for the guard position. Similarly, if the available exit bandwidth is less than $T/3$, then exit routers may only be used for the exit position.

[5] Note that if Tor didn't use entry guards, the attack could succeed for any client.

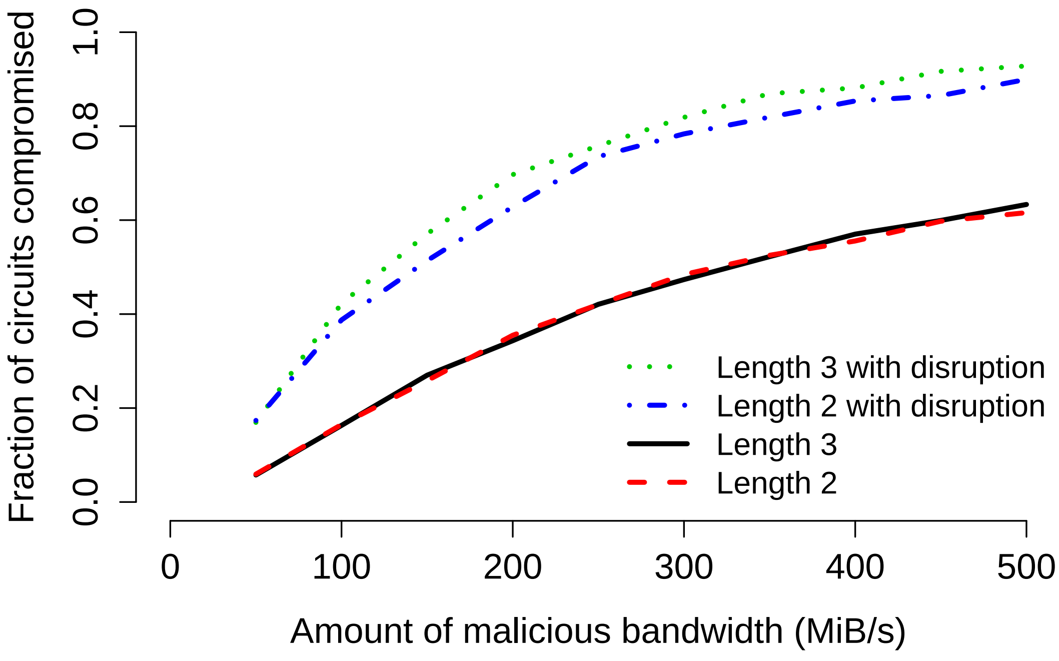


Figure 5.2: Fraction of HTTP circuits compromised

a malicious guard, only the exit position can disrupt non-compromised circuits. Because three-hop paths have one extra position from which to disrupt circuits, they exhibit a slightly higher compromise rate relative to two-hops.

5.1.2 Adaptive Surveillance

In addition to the threat posed by compromised routers, Tor is vulnerable to attacks where a powerful ISP or government adversary can monitor a targeted circuit's endpoints' networks to identify communication pairs. This attack is believed to be difficult because it relies on a circuit having the misfortune of choosing an entry and exit router that reside within monitored networks. Since Tor achieves network diversity in its route selection [111], this attack may require collusion by many network operators.

However with two-hop paths, exit routers can directly observe the entry guards. If an exit router learns an entry guard for a client of interest (identified perhaps by a distinguishing feature in the exit traffic), they could adaptively demand logs from the entry guard's network to reveal the client.[6]

While two-hop paths enable adaptive surveillance attacks by leaking entry guards to the exit router, adaptive surveillance is possible even with Tor's current three-hop design. If an adversary deploys malicious exit routers and malicious middle-only routers, they can collude to identify the entry guards used for every circuit on which they are used for the middle and exit positions. Through simulation, we next show that an adversary who controls few exit routers and comparatively many malicious middle-only routers can identify the entry guard used for a large fraction of circuits.

Simulation setup. Experiments are conducted where an adversary injects ten exit routers configured to exit HTTP (port 80) traffic to the Tor network described in Figure 5.1. The adversary also injects middle-only routers. All malicious routers have 10 MiB/s of bandwidth and disrupt circuits when they do not control both the middle **and** exit positions. We simulate 1 000 clients who each build 100 circuits.

This attack strategy has a low cost for the adversary, since they do not need to demonstrate router stability (as is necessary to obtain the guard flag). In addition, all malicious middle-only nodes could be deployed on the same /16 network and all malicious exit routers could be deployed on a second /16 network. Thus, the resources required to launch this attack are modest.

Results. For an attacker with ten exit routers and 50 middle-only routers, the adversary can identify the entry guard for 79% of all circuits constructed. When the attacker deploys 75 middle-only routers, they discover the client's entry guard for 85% of all circuits. For these circuits, the adversary could apply pressure and potentially coerce the entry guard (or its network operator) into revealing the identity of the client.

Perhaps the most compelling argument in favor of three-hop paths for Tor is that the middle

[6] Note that, while well-configured Tor relays do not maintain logs themselves, many ISPs maintain logs of connections and traffic statistics as part of their standard operating procedures.

Table 5.1: Daily statistics for clients per entry guard and information estimates

	Minimum	**Maximum**	**Median**	**95% CI**
No. of clients	680	164 000	8416	(24 104, 27 176)
Bits of information	8.20	0.29	4.57	(3.05, 2.88)

router hides the entry guards from exit routers. By using a middle router, an exit router typically knows only information about the client that is leaked by their applications. However, if malicious exits collude with middle routers who can observe the entire circuit, it becomes feasible for the exit to learn a large fraction of the total client population's entry guards.

To make matters worse, deploying a relatively large number of middle-only routers causes a global change in Tor's router selection process. In these experiments, when 50 middle-only routers are introduced, the total guard bandwidth G and total exit bandwidth E no longer satisfies $G \geq T/3$ and $E \geq T/3$, respectively, where T is the total router bandwidth. In this network configuration, exit routers may only be used for the exit position and entry guards may only be used for the guard position. This enables the adversary to focus their few exit routers toward occupying the exit position and maximize their ability to conduct adaptive surveillance.

5.1.3 Entry Guard Linkability

With two-hop paths, exit routers can trivially discover clients' entry guards. It is also possible that clients' entry guards may be uniquely identifying or place clients into dangerously small anonymity sets. To understand how knowledge of clients' entry guards may be identifying, we analyze publicly available data on entry guard usage from the Tor Metrics Project [222]. From this data set, eleven entry guards provide information about the number of clients that they observe over 737 total days.[7]

We measure the amount of information that is revealed about a user by their entry guard selections. Information is measured in bits according to $I(X) = -\log_2 \Pr(X = x)$, where $\Pr(X = x)$

[7] To preserve users' privacy, this data is aggregated by country of origin, quantized by multiples of eight, and compiled daily.

is the probability that a client uses guard x. The total number of unique Tor users per day is currently estimated to be around 200 000 [151]. Thus, without any additional knowledge, 17.61 bits of information are necessary to uniquely identify a Tor user. Now suppose that a malicious exit router knows one of a particular client's chosen entry guards. On average, roughly 25 000 clients use the same entry guard, so this knowledge leaks only 2.96 bits of information about a user's identity. Even in the worst case when a client shares a guard with as few as 680 other clients, only 8.20 bits are revealed. The full results are shown in Table 5.1.

However, if an attacker knows all three of a client's entry guards, the client may be far more identifiable.[8] To gain intuition about this threat, we simulate 200 000 clients choosing their three entry guards and measure the amount of information revealed through their three guard selections.[9] We find that 85% of simulated clients are uniquely identifiable and over 99% of clients share their guards with fewer than six other clients. In the best case, 26 clients share the same set of guards, which still reveals 12.9 bits of information. Tor clients do, however, expire and rotate their entry guard selections periodically, which may help to protect users from this type of profiling.

5.2 Performance Analysis

We next examine the performance implications of path length choices. Since the vast majority of Tor traffic is interactive web browsing [152, 158], we investigate the performance benefits of a two-hop design in terms of download times from a typical web browsing end-user's perspective.

Measurement setup. We measure download times for Firefox clients that fetch the fifteen most popular websites on the Internet[10] using Tor 0.2.1.24 with Polipo 1.0.4. Experiments were conducted over the course of six days from June 16–22, 2010 and measurements were collected as follows.[11] First, a three-hop circuit is built according to Tor's default router selection algorithm and a randomly chosen website from the fifteen is downloaded. Next, a two-hop circuit is built using

[8] While it is usually difficult to link a client across multiple entry guards, if a client inadvertently identifies herself — perhaps by logging-in to a website or using an application that does not support SSL/TLS — over time her full set of entry guards could be leaked to a malicious exit router.

[9] This experiment simplifies some details, such as the fact that older guards may be used by more clients than newer guards.

[10] According to `http://www.alexa.com`. These websites vary in size between 14.4–619.8 KiB.

[11] This setup takes into account variations in traffic load that may occur at certain times of day or certain days of the week.

the same entry guard and exit router and the client downloads the same website. This procedure allows us to directly measure the performance cost incurred by adding the middle router. In total, 2 726 measurements with both two-hop and three-hop circuits are collected (5 452 combined).

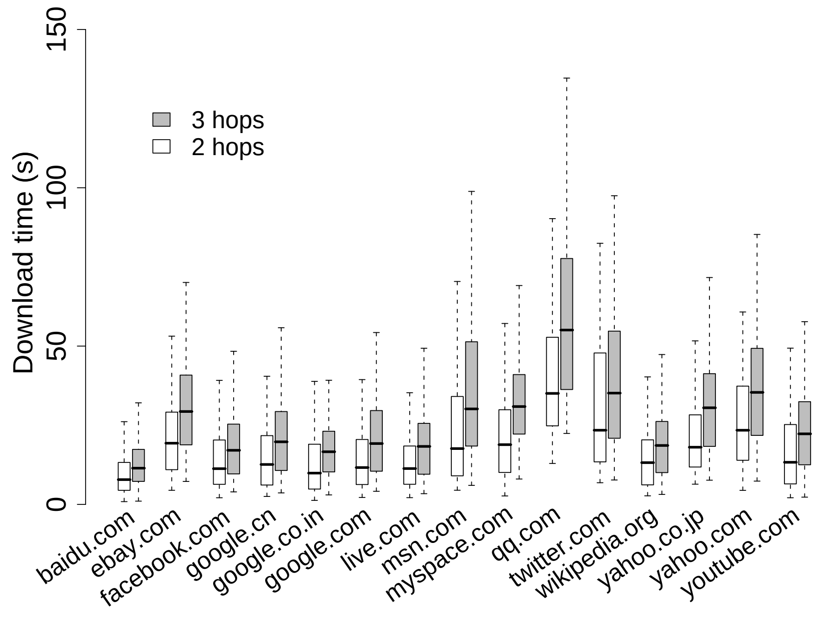


Figure 5.3: Download time comparisons between two- and three-hop paths for 15 popular websites

Results. Figure 5.3 compares download times for clients who use two- and three-hop circuits. For each website, the inter-quartile ranges show longer download times for three-hop circuits. Figure 5.4 shows CDFs of download times aggregated across all websites. Two-hop paths offer faster download times relative to three-hop paths, by up to seven seconds at the median.

Figure 5.5 shows the performance impact of adding a middle router to each two-hop circuit. This allows for a direct comparison between two-hop and three-hop circuits that use the same entry guard and exit router. For 81% of circuits, a middle router adds additional delay beyond the delay incurred by the entry guard and exit router.[12] Additionally, half of all circuits experience at least a 35% increase in download time and one-quarter of all circuits have more than **twice** the delay

[12] We observe that 19% of 3-hop circuits performed no worse with a middle router than without it. This shows that there exists some variability in Tor's performance.

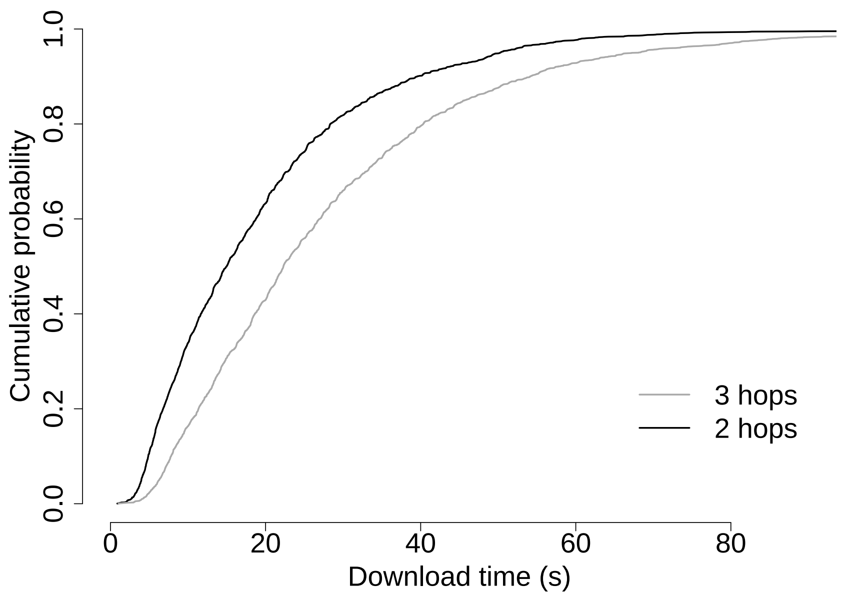


Figure 5.4: Cumulative distribution of download time across all 15 websites

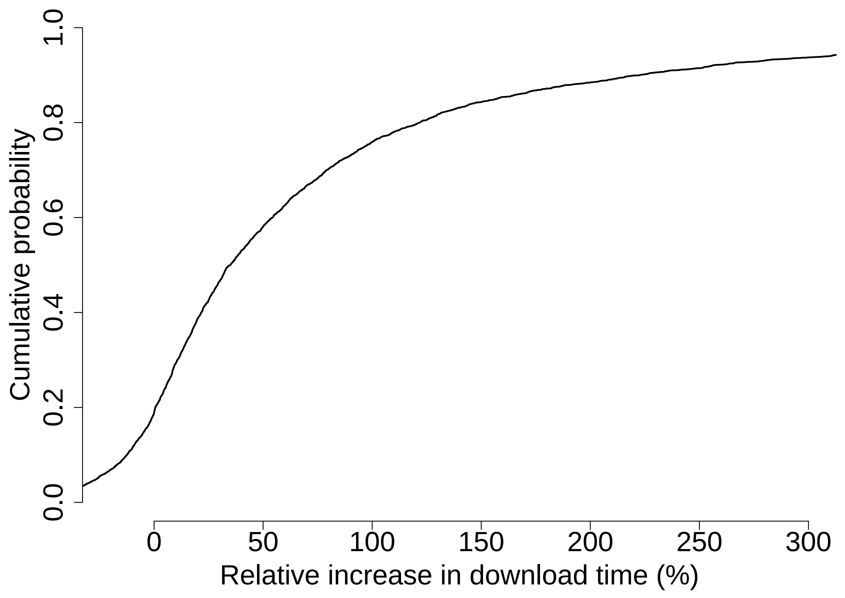


Figure 5.5: Percent increase in download time for three-hop circuits compared to two hops

with a middle router than without it. This shows that middle routers often contribute significantly to circuits' overall delays.

5.3 Blending Different Paths Lengths

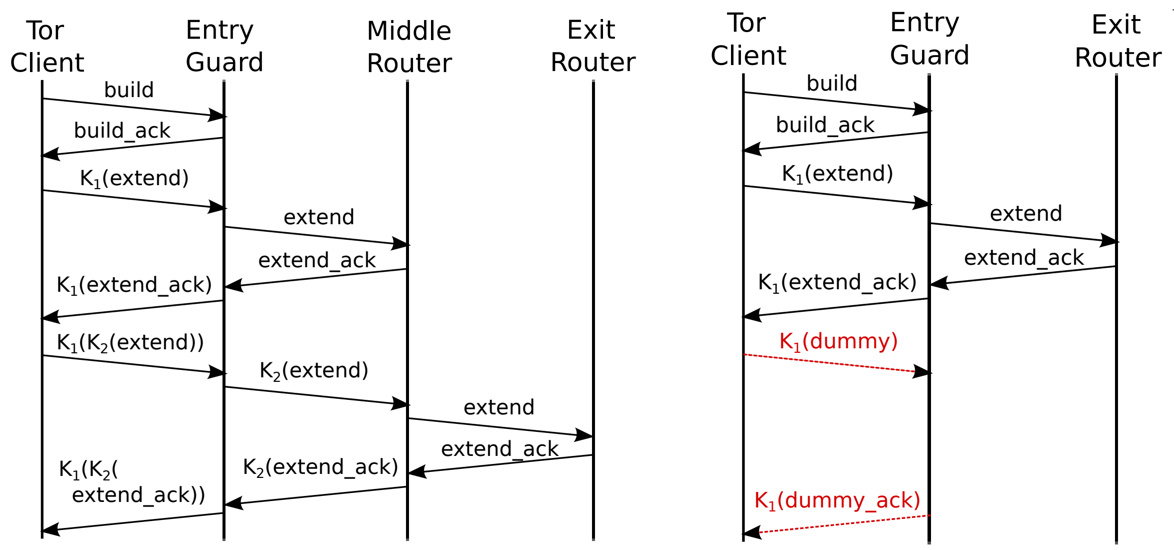
One potential obstacle to allowing users to choose their own circuit length is that if an adversary can discover a client's circuit length, then they may be able to craft attacks specifically tailored to the path length and more effectively attack Tor's source-destination unlinkability. Furthermore, leaking information about a client's desired path length may also reveal whether the client is concerned more about performance or security, and clients who are known to desire security over performance may draw additional attention from adversaries simply because they appear to have something to hide. Thus, in order for different path lengths to safely co-exist, it is necessary to mitigate the amount of information that is leaked regarding each client's path length choice.

In this section, we first demonstrate how an adversary observing the network traffic between a Tor client and one of their chosen entry guards can infer the length of the client's circuit by analyzing the circuit construction traffic. Next, we offer techniques to obfuscate these traffic signatures, and thereby *blend* different path lengths together.

5.3.1 Circuit Length Discovery through Traffic Analysis

Tor's telescoping circuit construction protocol uses a deterministic sequence of messages to establish shared keys between the client and each router on the circuit as depicted in Figure 5.6(a). For an adversary who can observe the network traffic between a client and their entry guards, the construction of a standard three-hop circuit requires precisely three request-reply message pairs in increasing chronological order. These messages are trivial to identify, because they are always sent before the client sends or receives any data. Upon observing this unique traffic signature, the adversary can conclude that the client is using three routers.

Similarly, if the adversary observes only two pairs of request-response messages at the start of the client's transaction, as shown in Figure 5.6(b) (non-red lines), then they can conclude that the client is using only two routers for their circuit. Since it is possible to differentiate two- and three-hop circuits, an adversary could tailor their attacks for the path length of a client of interest. For



(a) Three-hop circuit construction

(b) Two-hop circuit construction with length blending cells in red

Figure 5.6: Circuit building messages. K_1, K_2, and K_3 are the symmetric keys derived from Tor's telescoping Diffie-Hellman key establishment shared between the client and the entry guard, middle router, and exit router, respectively. Also, note that all messages are further protected by TLS.

example, if an adversary detects a two-hop path, then they could focus on breaking into the entry guard or applying legal subpoena pressure to obtain traffic logs from the entry guard's operator or Internet Service Provider. Also, even if an adversary detects a standard three-hop path, implying that the client is particularly concerned about the security of their traffic, they may wish to exert the effort of applying any one of a variety of traffic analysis techniques [47, 150, 197, 206] because they know the client's traffic may be particularly sensitive. Thus, we wish to mitigate a network observer's ability to infer circuits' path lengths.

5.3.2 Blending Techniques

In order to effectively blend two- and three-hop circuits from an observer on the network link between the client and one of their entry guards, we introduce dummy traffic into the circuit building process for two-hop paths. In particular, after the client establishes shared keys with their entry guard and second (exit) router, the client should send a third dummy cell (encrypted with the key shared between the client and the entry router), to mimic the final circuit building cell that would establish a shared key between the client and the exit router in a standard three-hop circuit. Upon receiving the dummy cell, the entry guard should also reply with another dummy cell, of course, adding an intentional delay to simulate the time required for the third circuit construction cell to be sent to and return from the third (three-hop exit) router. The precise sequence of messages is depicted in Figure 5.6(b), with the dummy messages illustrated in red.

While a network observer on the client's link cannot distinguish this dummy traffic from an ordinary three-hop circuit construction, the entry guard knows that the circuit uses only two routers and can therefore trivially learn the circuit's exit router. However, the exit router observes the exact same sequence of messages for circuit construction in both two- and three-hop circuits; thus, it does not explicitly learn any information about path length via circuit construction. We believe that protecting path length information from a network observer watching the client and the exit router is essential, since otherwise they could collude to trivially de-anonymize clients, or the exit router could learn a client's entry guards by linking a pseudonym observed in the exit traffic to the observed entry guards. Furthermore, clients already have trust relationships established with their chosen entry guards, since they use the same set of guards over a fairly long period of time.

It is also possible that a network observer on the client's link may infer path length by analyzing the round-trip times of the client's traffic and observing that two-hop paths tend to have less network latency than longer paths. However, in Chapter 6.7, we showed that 81% of the time two-hop paths offer faster download times than standard Tor circuits. Interestingly, for the remaining 19%, there was either no difference in performance, or two-hop paths performed *worse* than standard Tor paths. This apparent variability in performance likely would introduce inherent noise into any attempts to infer path length using performance characteristics.

5.4 Discussion

Having analyzed Tor's path length from security and performance perspectives, we next discuss a variety of open issues related to path length.

5.4.1 User-configurable Path Lengths

Since two-hop paths offer better performance, it may be tempting to allow users who value performance over security to use two-hop paths while users who need stronger security may use three-hop paths. Suppose that most users value performance and consequently, Tor chose a default path length of two hops. Security-conscious users could optionally use three hops to take advantage of the additional security that three-hop paths offer against adaptive surveillance. However, clients who choose to use longer paths may be identified as desiring additional security, which alone could draw an adversary's attention. To mitigate this risk, we suggest a blending strategy we present in Chapter 5.3 to reduce the amount of information leaked about a client's desire for stronger security or better performance. Furthermore, it has been argued that most users tend to keep default options, even when the defaults may not be optimally suited to their needs [102]. Allowing users to configure their own path lengths assumes that users understand the full security implications of their choice, which may be unlikely, particularly for novice users. Thus, all users should be encouraged to use the same path length by selecting a default path length value wisely, and given our findings, we recommend that two hops be used.

5.4.2 Potential Liabilities for Exit Routers

Two-hop paths could be a legal liability for exit router operators in some jurisdictions. With three-hop paths, exit routers know nothing about clients other than what may be revealed by their traffic. However, with two-hop paths, exit routers are exposed to clients' entry guards; thus, they are no longer agnostic with regard to the clients whose traffic they transport. Exit routers could be presented with subpoenas to reveal entry guard information to governments or law enforcement agents, which increases the risks associated with operating an exit router. Since Tor's exit bandwidth is relatively scarce yet essential to the network's ability to offer satisfactory performance, liabilities for exit router operators should be minimized to attract additional exit routers.

5.4.3 Secure Bandwidth Estimation

The attacks that we describe in Chapters 5.1.1 and 5.1.2 are particularly dangerous in the absence of secure bandwidth verification, since malicious routers could otherwise inflate their perceived bandwidth to attract traffic. With secure bandwidth estimates in place, it will no longer be possible to carry out these attacks with few resources. However, it is important to remember that such attacks are still within reach of medium-to-large organizations, or even determined individuals: at current hosting rates, running a 10 MiB/s node for one week (long enough for a node to be declared a guard) can cost less than $1 000;[16] thus, the financial resources required to attack the network successfully are moderate at best. Additionally, attackers may be able to insert their own high-bandwidth nodes into the Tor network by compromising computers at well-provisioned institutions.

5.4.4 Does a Two-hop Design Discard Too Many Routers?

Many Tor routers are not configured to allow exit traffic and are not fast and/or stable enough to be an entry guard. These routers are only used for the middle position. We next consider whether a two-hop design would discard a significant number of middle-only routers and their collective bandwidth.

From the directory server snapshot analyzed in Chapter 5.1, we find that 639 routers may only be used for the middle position. These routers collectively contribute about 85 MiB/s of bandwidth. To understand how bandwidth is distributed among non-exit and non-guard routers, Figure 5.7 shows a CDF of these routers' bandwidth contributions. Half contribute less than 50.3 KiB/s each and only 11% offer the 250 KiB/s necessary to meet the bandwidth criterion for the guard flag. These higher bandwidth routers collectively contribute 54.3 of the 85 MiB/s of middle-only bandwidth. If stable enough, they could eventually obtain the guard flag and be used for the entry position.

[16] See, for example, `http://aws.amazon.com/s3/`.

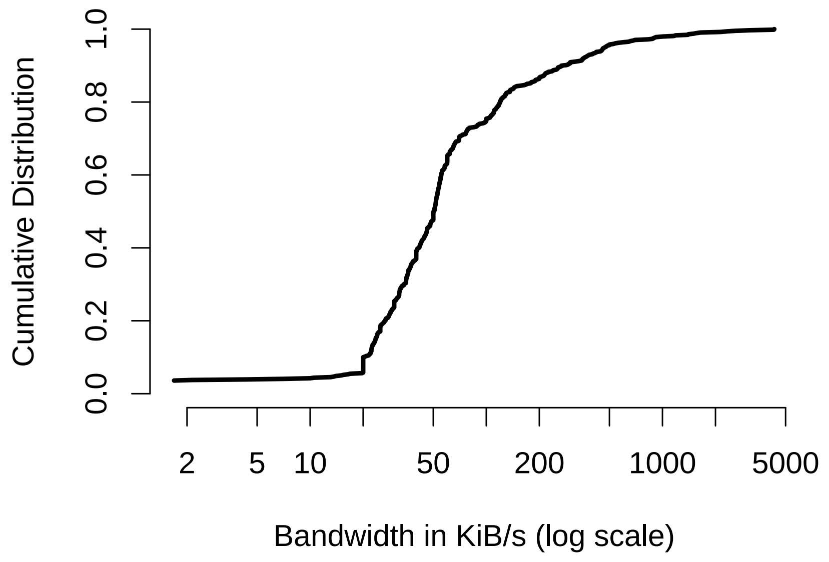


Figure 5.7: Bandwidth contributions from middle-only routers

5.5 Summary

In summary, we have shown that two-hop paths generally perform better than three-hop paths and are less vulnerable to endpoint compromise in the presence of an adversary who employs selective disruption tactics. However, two-hop paths potentially allow an attacker to enumerate a client's set of entry guards and increase the liability risk for exit node operators. By outlining a method to effectively blend two and three hops together from the perspective of a network eavesdropper, we explore how to get the best of both worlds.

In the end, these are but a few of many potential attacks and defenses associated with path length choices. There are deep-rooted tussles surrounding the security and performance trade-offs when choosing path-length and it is not yet clear what the "correct" length is or even if there is a "correct" length that fits every user's security and performance threshold and volunteer node operator's level of liability risk. We must look at Tor as having many stake-holders (individual user's privacy and performance, node operator's liability, and the overall security and performance offered via Tor to all users) and explore how each is effected by path length variations. Viewed in this light, our work serves as a piece in the puzzle by identifying the key issues, analyzing potential directions for solutions, and offering empirical measurements to guide future exploration of this problem. Our hope is that this work encourages further dialogue about a path length choice that most appropriately balances security and performance.

Chapter 6

Crowds-style Anonymity for BitTorrent

Since low latency anonymous networks do not artificially perturb the timing characteristics of their traffic, they are well-suited to anonymize interactive applications such as web browsing (HTTP) and instant messaging. However, in an analysis of Tor usage by protocol presented Chapter 3, we found that HTTP traffic is the most popular application by the number of connections observed, but bulk transfer protocols like BitTorrent (a popular peer-to-peer file sharing application) consume an unfair amount of the network's scarce bandwidth. While this analysis shows a clear demand for an anonymizing solution for bulk transfer protocols, it also implies that bulk transfers are diminishing the network's utility for interactive traffic – the targeted audience for networks such as Tor.

Many users may wish to anonymize bulk transfer protocols not to exercise free speech or circumvent censorship, but to transfer copyright-protected media files without detection by copyright enforcement agents. It is well-known that entities representing the film and recording industries have engaged in widespread monitoring of peer-to-peer file sharing networks to identify individuals engaged in the illegal transfer of copyright-protected media [180]. Furthermore, a recent study has found that a single computer's perspective is sufficient to identify **all** BitTorrent peers and profile their uploading and downloading behaviors [66].

In the event of positive identification of such file sharing, copyright-holders have, in some cases, initiated lawsuits to collect monetary damages from infringers. Such a climate of widespread surveillance and prosecution may encourage file sharers to endeavor to protect themselves by using strong anonymizing networks like Tor [54], however, at the potential expense of the quality of service for other users including cyber dissidents, corporate whistle blowers, and those who wish to exercise free speech.

In this chapter, we seek to further understand why file sharers resort to using anonymizing networks to hide their file sharing behaviors. We first present a case study, focusing on BitTorrent, that illuminates a few sources of information leakage in the BitTorrent protocol. We also develop a series of techniques to collect forensic evidence of file sharing and hypothesize that next-generation file sharing investigations may incorporate such techniques to improve the accuracy of the current methodology. Lastly, to mitigate bulk transfer traffic on networks like Tor, we design, implement, and evaluate an anonymizing network based on the well-studied Crowds protocol [189] that offers a user-tunable degree anonymity within the BitTorrent protocol itself without any changes to BitTorrent clients.

6.1 Case Study: Information Leaks in BitTorrent

While BitTorrent provides the ability to transfer files among many users quickly and efficiently, experience has shown that its decentralized architecture also makes it appealing for sharing copyright protected files illegally. With a peer-to-peer network like BitTorrent, content is distributed and replicated among a potentially large set of peers, making the process of finding and contacting each peer hosting the content in question a difficult task. Despite the challenge, entities acting on behalf of copyright holders have begun to monitor BitTorrent file transfers on a massive scale to identify and contact users who violate copyright laws.

In fact, a recent study [180] shows how the entities representing copyright holders use naïve techniques such as querying the BitTorrent tracker servers to identify individual users participating in an illegal file transfer. After being identified, these entities often distribute DMCA take-down notices or even pursue more formal legal sanctions against individuals who appear in the tracker's peer list. However, this simple approach is prone to a wide variety of errors. For instance, it is trivial to introduce erroneous information into the tracker lists by explicitly registering fake hosts to the tracker. The authors of the recent study demonstrate this type of false positive identification by registering networked devices such as printers and wireless access points to tracker lists and subsequently receiving DMCA take-down notices for their suspected participation in illegal file transfers.

This strategy of polluting tracker lists with fake peers could be used to frustrate anti-piracy investigations. The Pirate Bay, a popular tracker hosting site, has allegedly begun to inject arbitrary, but valid IP addresses into their tracker lists [43]. This counter-strategy may further increase the potential for false positive identification, which could have serious consequences as this evidence can be used to initiate legal action against suspected file sharers.

Given the inaccurate nature of the current techniques for monitoring BitTorrent file transfers and the clear need for effective anti-piracy tactics, we consider this question: Is it feasible to develop and deploy an efficient technique for identifying and monitoring peers engaged in file sharing that is more accurate than querying the trackers?

To answer this question, we propose a technique that is active, yet efficient. Starting with the tracker's peer lists, each peer listed by the tracker server is actively probed to confirm their participation in the file sharing and to collect concrete forensic evidence. Our tool, called BitStalker, issues a series of lightweight probes that provide increasingly conclusive evidence for the peers' active participation in the file sharing.

To evaluate the feasibility of this active approach in practice, we conduct a measurement study with real, large torrents. In particular, we quantify the number of peers that can be identified, the potential for falsely identifying peers, the potential for missing peers, and the cost associated with this technique in terms of bandwidth. Our results indicate that active probing can identify a sufficiently large portion of the active peers while requiring only 14.4–50.8 KiB/s and about five minutes to monitor over 20,000 peers (using a commodity desktop machine). We also show that the active probing can be parallelized and scale to monitor millions of peers inexpensively using cloud computing resources such as Amazon's Elastic Compute Cloud (EC2) [4]. Using EC2, we estimate that our method can monitor the entire Pirate Bay (about 20 million peers) for only $12.40 (USD).

6.1.1 Background

Before we describe our method for monitoring large BitTorrent swarms, we first provide a description of the BitTorrent protocol and an overview of the techniques currently being applied to identify peers who are sharing a file with BitTorrent.

6.1.1.1 The BitTorrent Protocol

To share a file, BitTorrent first breaks the file into several fixed size *pieces* and computes a SHA1 hash of each piece to verify integrity. Pieces are sub-divided into smaller data units called *blocks*, typically 16 KiB in size. A metadata file containing the SHA1 hashes for each piece along with other information necessary to download the file including a URI to the *tracker server* is distributed to interested users via an out-of-band mechanism. Once a user has obtained the metadata for a file of interest, they proceed by contacting the tracker server to obtain a randomly chosen subset of peers who are sharing the file. This is called the *peer list*. By obtaining a peer list from the tracker (or another distributed hash table-based or gossip-based mechanism), the peer also registers itself with the tracker. The peer then begins requesting blocks of the file. Peers that are downloading pieces of the file are called "leechers," while peers that possess all pieces and participate as uploaders are referred to as "seeders."

The precise sequence of messages involved in the request of pieces is shown in Figure 1. A leecher establishes communication with another peer by exchanging `handshake` messages. The handshake consists of a plain text protocol identifier string, a SHA1 hash that identifies the file(s) being shared, and a peer identification field. After the handshake exchange, the leecher transmits a `bitfield` message. This contains a bit-string data structure that compactly describes the pieces that the peer has already obtained. After exchanging bitfields, the leecher knows which pieces the other peer can offer, and proceeds to request specific blocks of the file. The leecher sends an `interested` message to notify the other peer that it would like to download pieces. The other

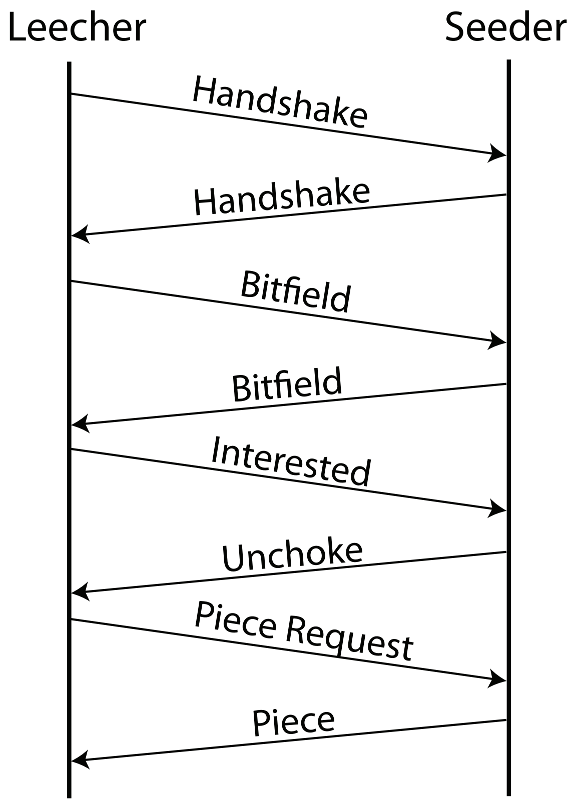


Figure 6.1: BitTorrent message exchange to start a piece transfer

peer responds with an `unchoke` message only if it is willing to share pieces with the leecher. Upon receiving an unchoke message, the leecher asks for specific blocks of the file.

6.1.1.2 BitTorrent Monitoring Practices

While BitTorrent provides an efficient way to distribute data to a large group of users, it is also an appealing technique to distribute copyright protected files illegally. Copyright enforcement is particularly challenging within the context of BitTorrent, since the file(s) in question are distributed among a set of arbitrarily many peers. The copyright holders must first *identify* every user who appears to be sharing the file and ask them to stop sharing.

Despite the significant amount of work required to monitor BitTorrent networks, a recent study has gathered evidence showing that investigative entities acting on behalf of various copyright holders are monitoring and tracking BitTorrent users who are suspected of sharing copyright protected files [180]. These investigators — including BayTSP [9], Media Defender [26], and Safenet [36] who are hired by organizations such as the Motion Picture Association of America (MPAA) and the Recording Industry Association of America (RIAA) — are using *passive* techniques, such as querying the trackers for the peer lists to identify users who are engaged in illegal file sharing. Once a list of peers has been obtained, an ICMP echo (ping) message is sent to each IP address to ensure that it is alive.

However, as the aforementioned study notes, these methods for monitoring large BitTorrent networks can be wildly inaccurate. For instance, it is possible to implicate arbitrary networked devices by simply registering their IP addresses with the tracker server. In addition, *false positive* identification is also possible as a result of naturally occurring (*i.e.,* non-intentional) activity. For instance, the tracker may provide stale peer information, which may result in a user who recently obtained a DHCP lease on an IP address being implicated in the file sharing. The very real potential for false positives could have serious implications, since the investigators who conduct this monitoring often issue DMCA take-down notices or even initiate legal actions against the suspected file sharers.

6.1.2 Accurate and Efficient Monitoring

In order to study the feasibility of collecting forensic evidence to concretely prove a peer's participation in file sharing, we present *BitStalker*. BitStalker is active, yet efficient, since it consists of small probe messages intended to identify whether a peer is actively engaged in a file transfer. First, to obtain the list of peers who are potentially sharing the file, the tracker is queried. For each IP address and port number returned, we conduct a series of light-weight probes to determine more conclusively whether the peer really exists and is participating in the file transfer.

TCP connection. The first probe consists of an attempt to open a TCP connection to the IP address on the port number advertised by the tracker. A successful TCP connection indicates that the suspected peer is listening for connections on the correct port.

Handshake. If a TCP connection is established, a valid BitTorrent handshake message is sent. If the handshake succeeds, then the investigator has obtained evidence that the suspected peer is responding to the BitTorrent protocol, and may even provide information about the BitTorrent client software being used.

Bitfield. If the handshake probe succeeds, then a BitTorrent bitfield message is sent. This message contains a concise representation of all pieces that have been downloaded by the peer. A random bitfield is generated so that the probe looks like a valid bitfield message. If a peer responds with a valid bitfield message, then the investigator has obtained evidence that the peer has downloaded the part of the file that is described by their bitfield. This also indicates whether the peer is a seeder or a leecher. This provides the strongest form of forensic evidence that the peer is actively sharing the file without exchanging file data.

Block request. If the bitfield probe succeeds, we finally attempt to request a 16 KiB block of the file from the peer. First, the peer's bitfield is examined to find a piece of the file that the peer has obtained. Next, this probe sends an interested message to indicate that we want to exchange pieces with this peer. The peer responds with an unchoke message, which implies that we are allowed to ask for pieces. We finally request a 16 KiB block. If the peer responds with the block requested, then this probe succeeds. A single block is the smallest amount of data necessary to confirm that another peer is sharing the file. If the investigator has the remaining blocks of that piece, then they can verify the hash to ensure that the block is valid.

We argue that each probe type provides increasingly conclusive evidence of a peer's active involvement in file sharing. A successful TCP probe indicates that the peer is listening on the correct port. However, an effective counter-strategy could be to register arbitrary IP addresses with ports that are opened (such as web servers). The subsequent handshake probe is more conclusive, as it indicates that the BitTorrent protocol is running on the correct port and also identifies the content being shared by a SHA1 hash. The bitfield probe provides stronger evidence still, since it describes all pieces that the peer has downloaded, which implies active sharing. Finally, requesting and subsequently receiving a block of the file provides the strongest form of concrete evidence for file sharing.

Practical considerations. The active probing framework can monitor peers who are actively participating in the file sharing. However, if a peer has just joined the torrent when they are probed, then they may not have any pieces of the file yet. Consequently, according to the BitTorrent protocol, if a peer has no pieces, then the bitfield probe is optional. Since the peer has not yet obtained any pieces of the file, the probing does not collect any evidence from this peer. If peers are probed repeatedly over time, then the likelihood of this case becomes negligible.

Additionally, “super-seeding” mode is enabled when a torrent is first established and there are few seeders. Super-seeding mode ensures that the original seeder is not overwhelmed by piece requests from other peers before it transfers data to another peer. When super-seeding is activated, the seeder may advertise an empty or modified bitfield, even though they possess every piece. Since we are interested in monitoring mature torrents consisting of at least tens of thousands of peers, we disregard new torrents in super-seeder mode.

Lastly, it is possible that peers may be able to detect the monitors and blacklist them. Siganos *et al.* show that the current passive BitTorrent monitors can be detected by observing that the frequency with which the monitor’s IP addresses occur across a large number of tracker lists is statistically higher than that of normal peers [208]. Our active monitoring may also be identifiable in the same manner. To address this, we recommend that the monitoring be distributed across a large number or dynamic set of IP addresses.

6.1.3 Experimental Evaluation

In this section, we present experiments to quantify both the effectiveness and the cost of monitoring large BitTorrent swarms using the active probing technique. In addition, we compare the accuracy, potential for false positives and false negatives, and the cost with the current strategy employed widely by anti-piracy investigators.

Table 6.1: Summary of data sources

Torrent ID	**Total Peers**	**Media Type**
1	20,354	TV Series
2	16,979	TV Series
3	11,346	TV Series
4	14,691	TV Series
5	23,346	Movie
6	20,777	TV Series
7	24,745	TV Series
8	13,560	TV Series
9	19,694	TV Series
10	20,611	Movie
Total:	186,103	

6.1.3.1 Data Sources and Methodology

To evaluate our light-weight probing technique, we selected ten large torrents each containing between 11,346 and 24,745 unique peers. In total, our experimental evaluation consists of over 186,000 peers. Peers participating in these torrents were sharing new theatrical releases and episodes of popular television shows (summarized in Table 1). These swarms represent the type of file sharing that may be monitored by copyright enforcement agencies.

To conduct the active probing, we wrote a tool called BitStalker that can perform the following tasks:

- Establish a TCP connection with another peer
- Exchange handshake messages with the correct SHA1 content hash and receive handshake responses
- Exchange bitfield messages and receive bitfield responses
- Request and receive a 16 KiB block of file data

In short, BitStalker efficiently probes for participation in the BitTorrent protocol by sending and receiving a minimal number of small control messages rather than downloading the entire file from other peers.

The experiments were conducted as follows: The tracker server is contacted to obtain a subset of the peers who are currently believed to be sharing the file. Since the trackers only return a randomly selected set of 100 peers, it is necessary to query the tracker several times to obtain a large portion of the hosts registered with the tracker. Once peers are obtained from the tracker, BitStalker attempts to establish a TCP connection with each peer on its advertised TCP port. If a connection is established, a handshake message exchange is attempted. If handshake messages are exchanged, BitStalker attempts to exchange bitfield messages. Finally, if bitfields are exchanged, the tool attempts to retrieve a single block of the file. This procedure is repeated for each torrent to be monitored.

We compare our active probing method with the current approach to peer identification described in Section 6.1.1.2. After obtaining the list of suspected peers from the tracker, our tool sends precisely five ICMP echo (ping) messages to each IP address in the peer list. If a host responds to at least one ping, then it is assumed (perhaps erroneously) to be alive and sharing the file.

6.1.3.2 Experimental Results

We evaluate the proposed peer probing technique with regard to the number of peers that can be identified, an estimate of the number of peers that are falsely identified as being a file sharer (false positives), an estimate of the number of peers that this technique fails to identify (false negatives), and the measured cost of performing this active probing. The probing mechanism is compared along each of these metrics to the passive identification process using ping messages to verify the tracker's peer list.

Fraction of peers that respond. We first consider how many peers can be identified by active probing. As shown in Table 2, the fraction of peers that can be positively identified by each probe type increases with additional repetitions. To determine if additional peers can be identified through multiple probing attempts, the experiments are repeated ten times. Even though the number of

Table 6.2: The average fraction of peers identified in one, five, and ten iterations of the monitoring across all ten torrents

Repetitions	**Connection**	**Handshake**	**Bitfield**	**Block Request**
1	30.8%	18.9%	17.7%	0.29%
5	35.9%	26.3%	25.3%	0.80%
10	36.9%	28.4%	27.6%	1.13%

peers probed remains constant for each repetition, we find that the fraction of peers that respond to probes increases, since some peers may be busy interacting with other peers when we probe.

The complete results for each torrent are given in Figure 2. Across the ten torrents, we could establish a TCP connection with between 26.7–44.6% of the peers listed by the tracker. While this percentage seems low, it is reasonable since many BitTorrent clients impose artificial limits on the number of open connections allowed, in order to reduce the amount of bandwidth consumed. A similar fraction of peers that establish connections is reported by Dhungel *et al.* [93].

The naïve ping method returns roughly the same fraction of peers as the active TCP connection probe. However, as we will show, the ping probes are susceptible to an intolerably high number of false positives, while active probing significantly reduces the potential for false positives.

Both the handshake and bitfield probes succeed for between 18.6–36.6% of the peers. While this is lower than the TCP connection probe, it provides significantly stronger evidence for file sharing. For this fraction of the peers, an investigator can tell that the peer is obeying the BitTorrent protocol, sharing the correct file identified in the handshake probe by a SHA1 hash, and advertising the pieces of the file that the peer already possesses as identified in the bitfield probe. We argue that this small reduction in the fraction of peers that respond to bitfield probes is a small price for greater confidence in the identification results.

Finally, we observe that block request probes succeed for a very small faction of the peers, only 0.6–2.4%. This may be partly a result of BitTorrent's tit-for-tat incentive mechanism [10], which attempts to mitigate selfish leechers by enforcing reciprocity in the piece request process.

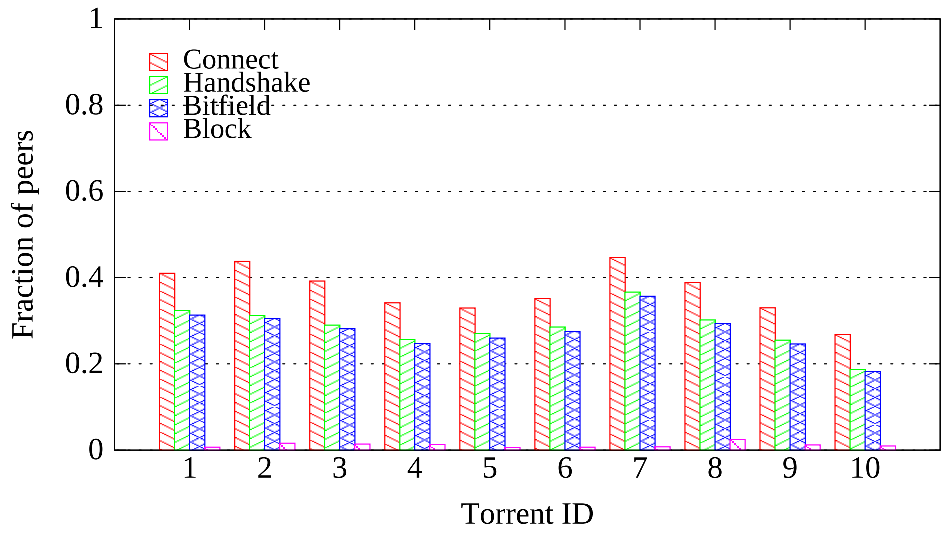


Figure 6.2: Over ten runs, the cumulative fraction of peers identified with connections, handshakes, bitfields, and block requests across all ten torrents

This is implemented by uploading to other leechers from whom you download. The leecher with the highest upload rate receives download priority. Since BitStalker has a zero upload rate, it does not receive priority for piece requests. However, BitTorrent does offer optimistic unchoking, which enables a leecher to download regardless of their upload rate. BitStalker only receives pieces from other peers who have chosen to optimistically unchoke.[1] Since only about 1% of the peers respond to our block requests on average, we argue that the minimal additional evidence obtained through this probe is not worth the extra time and bandwidth required to collect this evidence.

False positives. The most serious flaw with the past and present investigative tactics based on tracker list queries and ping probes is the real potential for a high number of false positives. Furthermore, active peer list pollution further increases the potential for false positives.

To establish a lower bound on false positives obtained by the naïve investigative strategy, we count the number of peers that respond to pings yet show no indication of running any network service on their advertised port. More technically, if a peer responds to a TCP SYN request with a TCP RST (reset) packet, this indicates that the remote machine exists, but it is not running any

[1] Additional blocks may be received if BitStalker offered blocks before asking for blocks.

service on the advertised TCP port. From our experiments, we observe that 11% of peers exhibit this behavior on average and are, therefore, definite false positives using this naïve investigative strategy.

In addition, we count the number of peers that *could* be false positives with the ping method. These are the peers that respond to ping probes, but ignore the TCP probe (*i.e.*, no connection or reset packet). From our experiments, we find that on average an additional 25.7% of the peers could potentially be false positives, but we cannot say this conclusively. It's possible that some of these peers could have reached a connection limit in their BitTorrent client or could be filtering incoming traffic.

In contrast to the naïve ping method, the active probing strategy offers more reliable peer identification with few avenues for false positives. For instance, a successful TCP probe indicates that the peer is listening for connections on its advertised port. However, one could envision a more intelligent pollution strategy where arbitrary IP addresses with open ports are inserted into trackers (*i.e.*, real HTTP or FTP servers). The subsequent handshake and bitfield probes would then eliminate this form of pollution by checking that the host is running the BitTorrent protocol.

However, the active probing approach is not entirely immune from the possibility of false positive identification. For example, peers using an anonymizing network such as Tor [103] may produce false positives, since the last Tor router on the client's path of Tor routers (called a Tor exit router) would be implicated in the file sharing. In Chapter 3, we found that BitTorrent is among the most common applications used with Tor, particularly when measured by traffic volume.

To determine how common this type of false positive is in practice, we compare the list of potential BitTorrent peers obtained through our experiments to the list of all known Tor exit routers provided by Tor's public directory servers. On average, we find that only approximately 1.8% of the peers are using Tor to hide their identities.[2] However, these are not false positives using active probing, since a peer using Tor (or another anonymizing network or proxy service) cannot bind to the advertised port on the exit host to accept incoming connections. Consequently,

[2] However, several peers could be using each of these Tor exit nodes.

active probing does not provide any evidence for these peers. Furthermore, peers using Tor are easily identifiable and can be filtered out of the results.

In addition to general-purpose anonymizing networks, solutions have been proposed specifically for anonymizing BitTorrent. For instance, SwarmScreen's goal is to obscure a peer's file sharing habits by participating in a set of random file sharing swarms [78, 79]. Also, BitBlender attempts to provide plausible deniability for peers listed by the trackers by introducing relay peers that do not actively share files, but rather act as proxies for other peers actively sharing the file [62]. The active methods we propose would identify peers utilizing SwarmScreen and BitBlender as file sharers. While these peers are not intently sharing content, an investigator may still be interested in pursuing these peers since they contribute pieces of the file to other peers who are actively sharing.

False negatives. False negative identification occurs when a peer who is actively sharing a file cannot be identified as a file sharer. Both the active probing technique and the naïve ping method suffer from the potential for false negatives. The ping method may miss peers who are behind a firewall that blocks incoming ICMP traffic. For example, this is the default configuration for Windows Vista's firewall settings. The active probing method may also suffer from false negatives when a peer's number of allowed connections is at the maximum. In this case, the initial TCP connection probe will fail to identify that the peer is listening on its advertised port. In general, we found that repeating the monitoring procedure decreases false negatives. Table 2 shows that the number of false negatives decreases as the experiment is repeated. Although there are diminishing returns, as the false negatives do not decrease significantly between 5 and 10 iterations of the monitoring.

We can, however, provide a lower bound on false negatives obtained with the naïve ping method. This is achieved by counting the number of peers that do not respond to pings, but do respond to the TCP connection probe. Our experiments show that the naïve ping method would fail to identify at least 22.3% of the peers on average.

Cost. In order for an active probing strategy to be a feasible technique to monitor large BitTorrent swarms in practice, it is necessary for the probing to be as efficient as possible. Table 3 shows that

Table 6.3: Size of each probe type (assuming no TCP options)

Probe Type	**Description**	**Size**
TCP connection	Three-way handshake	162 Bytes
Handshake	Handshake request/reply	244 Bytes
Bitfield	Bitfield request/reply	Variable
Block Request	Block request/reply	16.7 KiBytes
ICMP Ping	Ping request/reply	86 Bytes

the size of each probe is small and Figure 3 shows the amount of traffic that was required to monitor each torrent using the active probing technique. For comparison, the cost for the ping method is also plotted. While the ping approach requires less bandwidth, we have shown that it is not sufficiently accurate in identifying active file sharers. Using a modest Linux desktop machine, it took 304.5 seconds on average to monitor an entire torrent, which required only 14.4–50.8 KiB/s of bandwidth. The active probing overhead is dependent on the fraction of peers that respond to active probes. This is an intuitive result, implying a direct relationship between the number of peers identified and the amount of bandwidth required by the probing.

The active probing method is also highly scalable, particularly when inexpensive cloud computing resources such as Amazon's Elastic Compute Cloud (EC2) [4] are utilized. Machines from EC2 are available at a small cost dependent on the execution time and bandwidth usage of the jobs. From our experiments, on average we probed approximately 61 peers/second, uploaded 288.2 bytes/peer and downloaded 296.6 bytes/peer. Using EC2's pricing model, we estimate that it is possible to monitor peers at an expected cost of roughly 13.6 cents/hour (USD). In fact, it's possible to scale the active probing to monitor the entire Pirate Bay, which claims to track over 20 million peers [37]. We estimate that this method can monitor the Pirate Bay for $12.40 (USD).

6.1.4 Summary

We present *BitStalker*, a low-cost approach to monitoring large BitTorrent file sharing swarms. BitStalker collects concrete evidence of peers' participation in file sharing in a way that is robust

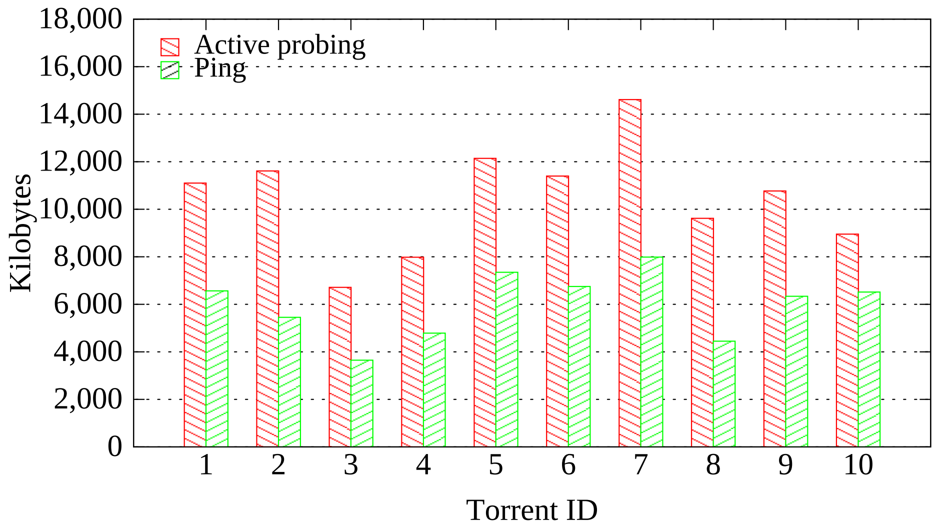


Figure 6.3: Total amount of traffic necessary to monitor each torrent using active probing and pings

to tracker pollution, highly accurate, and efficient. In contrast, the past and present investigative monitoring strategy consists of tracker server queries and ICMP ping probes. While this method is simple, it is also prone to a variety of significant errors, especially false positive identification, since this monitoring technique does not verify participation in the file sharing. We present an alternative monitoring strategy based on actively probing the list of suspected peers to obtain *more conclusive* evidence of participation in the file sharing.

There are several aspects of our approach that warrant additional attention. In particular, a specific definition of what constitutes "forensic evidence" in the context of file sharing across various legal systems should be explored. Also, the general legal issues that this type of monitoring exposes should also be investigated further.

6.2 BitBlender: Light-weight Anonymity for BitTorrent

General purpose, low latency anonymity networks are currently being used to provide private and anonymous communication services for a variety of applications. For instance, Tor [103] has become the standard tool for anonymizing TCP traffic. This is largely because of its ability to provide low latency anonymous transport to facilitate interactive applications such as web browsing and instant messaging. However, in Chapter 3, we offer evidence that Tor's ability to provide a low latency anonymous transport service is being potentially threatened by the excessive amount of peer-to-peer (P2P) file sharing traffic that it transports. Since there is a clear demand for anonymous file sharing and to alleviate the strain placed upon the Tor network by this P2P traffic, we present the design and implementation of an anonymity-preserving network protocol specifically tailored to P2P file sharing, and in particular, the BitTorrent protocol.

BitTorrent ostensibly provides no built-in support for anonymous file sharing. In fact, as part of the default peer discovery method, the protocol requires that the IP addresses of all peers sharing the file be published by a well-known and publicly accessible server called a tracker. By querying the tracker, it is currently trivial to determine who is actively sharing a particular file. In fact, a study found that information from these trackers is currently being used to identify file sharers, often with poor accuracy [180]. Hence, it is not surprising that many BitTorrent users resort to using Tor to remain anonymous.

Traditionally, *mix networks* have been the fundamental building block for many privacy enhancing systems. Mix networks construct a chain of intermediate hops between the source and destination of a message to conceal the message's true sender and receiver. Batching of messages and cover traffic are common techniques to further frustrate traffic analysis attempts. Most mix networks attempt to provide a high degree of anonymity, suitable for protecting cyber-dissidents in countries where Internet freedoms are not protected. In this case, the strongest practical anonymity available is required.

In contrast, it is often acceptable to provide a lower degree of anonymity. Reiter and Rubin [189] describe *degrees of anonymity* as a spectrum that expresses the confidence that an adversary has regarding the identity of the real initiator of a message. Their system, Crowds, achieves varying degrees of anonymity for web transactions by routing each message through a set of intermediate hops in a probabilistic fashion. When a message is received by an intermediate node, it is either forwarded to another intermediate hop or delivered to its final destination with a certain pre-defined probability. From the perspective of the destination server, it is unclear whether the node from which it received the message is the initiator or a proxy for another node. Thus, all nodes that participate in such a network enjoy a certain degree of plausible deniability with regard to requesting a file.

Inspired by Crowds, we propose a similar low-overhead anonymity layer for BitTorrent that offers sufficient anonymity properties to achieve the condition of plausible deniability. To this end, we present BitBlender, an anonymity layer for the BitTorrent protocol that has low overhead and provides varying degrees of anonymity. BitBlender achieves plausible deniability by introducing special "relay peers" that forward data on behalf of the peers that are actively sharing a file. When peers request pieces of a file, it is difficult to determine whether a piece is delivered by another peer engaged in the file transfer, or if it is delivered through one or more relay peers. Thus, the degree of anonymity provided is dependent upon the number of relay peers relative to the number of normal peers participating in a file transfer; however, the expected performance overhead is also dependent upon the number of relay peers participating, as the path length between the initiator and the responder is higher on average as more relay peers participate.

BitBlender is among the first to explore light-weight privacy enhancing system designs without the use of cryptography. Strict data confidentiality within BitTorrent is unnecessary, since files are typically shared publicly and anyone can participate without requiring any special access or authorization. In addition, this protocol has the ability to provide a level of anonymity that is tunable, so it can be adjusted for the sensitivity of the data transferred. Finally, since BitBlender requires no modifications to the existing BitTorrent protocol, it is easy to deploy within BitTorrent's current system architecture.

We provide an analysis of the protocol in terms of its expected path length as the ratio of normal peers to relay peers varies. In addition, we show that BitBlender has a lower hop count on average than Tor, even when the ratio of relay peers to normal peers is greater than 1/2. To analyze the anticipated performance overhead, we implement a prototype and perform experiments to quantify the expected additional download time that will be experienced by the end users as the number of participating relay peers varies. We also compare BitBlender's expected performance to that of BitTorrent tunneled over Tor.

Having presented the basic protocol, we present extensions aimed at strengthening the anonymity and increasing the performance. A confidentiality and access control mechanism is detailed that would provide confidential and authenticated file transfers. Also, we address traffic analysis attacks and present simple countermeasures. We lastly explore selective caching as a mechanism to simultaneously impede certain traffic analysis tactics and decrease the expected path length.

Finally, we discuss some of the legal questions that BitBlender and other general purpose anonymity networks present. In particular, the legality of operating an open relay is unclear, and arguably the continued success of anonymous communication systems relies on policy makers from around the world providing some form of legal protection for the operators of anonymity networks.

6.3 Degrees of Anonymity

In order to describe our anonymity layer for BitTorrent, it is necessary to define the notion of anonymity that the protocol provides. Reiter and Rubin describe anonymity as a spectrum, with degrees ranging from “absolute privacy” to “provably exposed” [189]. Between these extremes, the level of anonymity varies between states of “probable” and “possible” deniability. Probable deniability exists when an adversary can determine with a probability $0.5 \leq p < 1$ that a message in the system originated at a particular user. Possible deniability is the state at which there is a probability $0.5 > p > 0$ that a message originated at a specific user. We define *plausible deniability* as the state that encompasses both probable and possible deniability ($1 > p > 0$). The specific probability is precisely the ratio of relay peers to total peers (both relay and normal peers). With no additional information, an adversary has a probability p of correctly guessing whether an individual peer is a relay or a normal peer. Further details on anonymity metrics can be found in Chapter 2.4.

6.4 Design Principles

In order to describe the BitBlender protocol, it is necessary to first explain the design goals and the envisioned threat model.

6.4.1 Design Goals

BitBlender's design achieves the following:

- **Low overhead:** The protocol should be more light-weight in terms of computational resources, and should provide better throughput and lower latency in comparison to general-purpose anonymity networks. There is no overhead for cryptographic operations and protocol overhead associated with potentially routing messages through multiple relay peers is minimal.
- **Usability:** An important goal is usability, which means that the protocol should be easy to use, work seamlessly with the existing BitTorrent architecture, and offer performance that is comparable to – or better than – general-purpose anonymous networking protocols such as Tor. Performance is considered to be fundamental to the system's adoption and usability, since end users will be unlikely to use BitBlender if other systems such as Tor provide better performance.
- **Plausible deniability:** The protocol provides plausible deniability for peers that are listed by the tracker for a particular torrent. This is achieved by introducing relay peers that do not initiate file downloads or uploads, but simply proxy requests on behalf of other peers. By introducing relay peers, it is no longer the case that every peer in the tracker list is actively initiating uploads or downloads. An adversary must now engage in more sophisticated and potentially error-prone traffic analysis techniques to determine the true initiators. A detailed discussion of such traffic analysis attacks is provided in Section 6.6.3.

- **Tunable anonymity:** It is well-known that there is always a trade-off between the anonymity that a system can provide and its performance. BitBlender allows the trade-off between performance and anonymity to be made by tuning a system parameter, specifically the number of relay peers participating relative to the number of normal peers. This is an important feature, since some torrents may be more sensitive than others.

6.4.2 Threat Model

We assume a non-global adversary that can participate in the BitBlender protocol as a colluding fraction of the total peers (either relay or normal). This implies that the adversary can see the traffic flowing through the subset of the peers that it controls. In addition, the adversary can monitor the tracker list to see which other peers are participating in the torrent. This threat model is the same as that which is assumed in other low-latency anonymity networks [80,103,116,189,190]. We further assume that the adversary cannot passively monitor arbitrary links between peers in the network.

6.5 The BitBlender Protocol

Building upon the notion of anonymity provided by Crowds, we present BitBlender, an anonymity layer for BitTorrent. Before giving a high-level overview of the protocol, it is necessary to define each component. As in traditional BitTorrent, there are *peer* nodes that wish to share content. We introduce *relay peers* as peers that do not initiate downloads or uploads, but simply proxy traffic on behalf of normal BitTorrent peers. Relay peers and anonymous torrents are organized by an entity called a *blender*, which could be a single directory server, a set of directory server replicas, or a DHT.

The protocol proceeds as follows: In order to attract relay peers, the tracker for an anonymous torrent contacts the blender and requests that relay peers join the torrent with a certain probability. Given the degree of anonymity desired, the tracker asks each relay peer to probabilistically join its torrent.

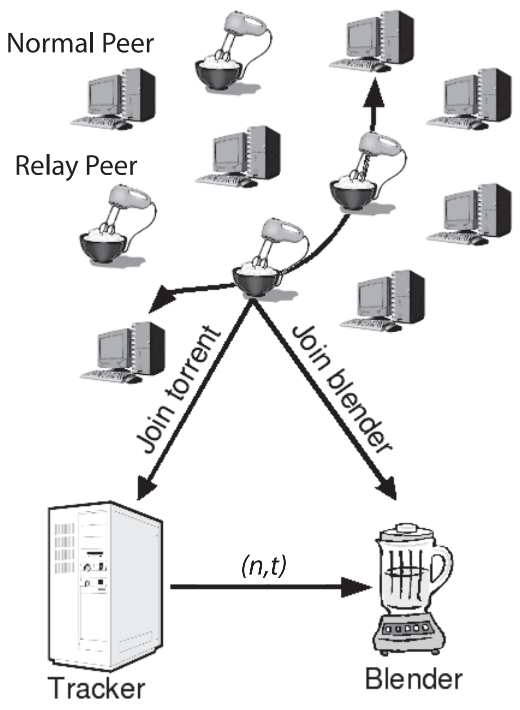


Figure 6.4: The BitBlender protocol system architecture. The protocol proceeds as follows: (1) A relay peer joins the blender; (2) The tracker requests relay peers; and (3) Relay peers probabilistically join the torrent. A piece request through two relay peers is shown (the path length is three hops).

Once the relay peers have joined the torrent, they proceed by transparently accepting piece requests and *forwarding* them to another member of the torrent. This peer may, in fact, be another relay peer, or it may be a real peer participating in the file transfer. Replies are also transparently forwarded in the same manner along the same relay path. Thus, an ad-hoc relay network is created, where the path lengths are somewhat non-deterministic. The relay peers could appear to be seeders, or they could advertise only a subset of the pieces for a particular file. The protocol's system architecture is described pictorially in Figure 6.4.

6.5.1 Relay Peer Joining

Let N be the set of peers participating in an anonymous torrent and M be the set of relay peers participating in the anonymous torrent such that $M \cap N = \emptyset$. The set of all relay peers listed by the blender is B, such that $M \subseteq B$. To establish an anonymous torrent, it is necessary that the tracker request a subset of the relay peers to join the anonymous torrent. The request sent by the tracker to the blender consists of the tuple (n, t), where n is the number of relay peers requested and t is a unique identifier for the tracker (such as a URI). Upon receipt of this message, the blender must calculate a join probability p, based upon the number of nodes requested and the size of the relay peer set, where $p = \frac{n}{|B|}$. This enables the blender to remain agnostic about which relay peers join the torrent.

Each $b_i \in B$ chooses a pseudorandom number $r \in \mathbb{R}$ subject to $0 \leq r \leq 1$ and joins the torrent identified by t iff $r \leq p$. On average, the requested number of relay peers n will join the anonymous torrent.

6.5.2 Anonymity Layer

In order to provide an anonymity layer, the normal peers simply make requests as in the traditional BitTorrent protocol; however, if a relay peer is requested for a piece, the original request is forwarded to another peer, potentially another relay peer.

An ad-hoc relay network is constructed in this manner, where the path lengths are probabilistically influenced by the concentration of relay peers to real peers in the torrent. As the piece request reaches a real peer, it fulfills the request by sending the requested piece back through the chain of relay peers to the original requesting peer. A certain degree of anonymity is achieved since it is difficult to prove which peers in the torrent are relay peers and which are real peers.

6.5.3 Discussion

BitBlender is a low-overhead, usable and inter-operable anonymity layer for BitTorrent that provides a dynamically tunable level of plausible deniability. Plausible deniability is achieved by adding relay peers, since it is no longer trivial to infer the set of peers participating in a particular torrent simply by inspecting the peer list maintained by the tracker.

It is important that relay peers not only appear in the tracker list, but also forward requests and replies. If an adversary participates in the protocol, it would be relatively easy to determine which peers are actively participating in the torrent and which do not issue piece requests or replies. Thus, in order to provide a higher degree of anonymity, it is essential that the relay peers appear to be actively participating.

By allowing the tracker to explicitly specify the number of relay peers that should join, this allows individual torrents to have a tunable anonymity parameter. The more relay peers that join a torrent, the more difficult it would be for an adversary to determine the true set of peers participating in the transfer of a torrent.

BitBlender is fully inter-operable with the existing BitTorrent protocol. Peers that wish to obtain a degree of anonymity may participate in BitBlender; however, those peers that do not desire anonymity may still participate in the torrent. In this case, they would be easily identified as normal peers, since they do not appear in the blender. Since inter-operability is a design goal, we do not provide any confidentiality or access control mechanisms. Such a layer would disallow non-BitBlender peers from participating in the torrent. We do, however, explore data confidentiality and access control as an extension in Section 6.8.

BitBlender's ability to provide an anonymity layer without the use of expensive cryptographic operations is unique when compared to previous mix and onion routing-based anonymity systems. BitTorrent is a protocol whose content does not typically leak personal information like HTTP or instant messaging protocols [48, 81]. Thus, it is not essential to provide strong data confidentiality, since the contents of the torrent are easily accessible to anyone.

Finally, there are several considerations that must be weighed when designing the blender. If the blender is a single centralized directory server, it becomes a single point of failure in the system and is open to denial of service (DoS) attacks. One solution may be to simply replicate the blender's database throughout the network and employ a consensus technique to issue queries. This is more fault-tolerant, but is susceptible to Byzantine faults [146]. Finally, the blender may exist as a service accessible via a distributed hash table (DHT). In this case, the blender is fully distributed; however, simple DHT schemes and other gossip protocols can be targeted with Eclipse attacks [209]. Designing a distributed and secure directory service is a challenging problem. For the sake of simplicity, we assume a blender based upon a single centralized directory server.

6.6 Protocol Analysis

In this section, we analyze BitBlender in terms of expected path lengths, a comparison to Tor, and the potential for traffic analysis attacks.

6.6.1 Expected Path Length

Since the protocol forwards requests and replies in a probabilistic fashion dependent upon the number of relay peers participating, we present an analysis of the expected path length. For simplicity, we assume that peers are chosen for piece requests uniformly at random from the set of all participating peers. Formally, let N be the set of peers (both relay and normal) associated with an anonymous torrent; the probability of choosing an arbitrary peer $p_i \in N$ is $1/|N|$.

Let $M \subseteq N$ be the set of relay peers participating in the torrent and $P \subseteq N$ be the set of normal peers subject to $M \cap P = \emptyset$ and $|M| + |P| = |N|$. The path length l for a piece request from peer $p_i \in P$ through relay peers $M_l \subseteq M$ to $p_j \in P$, the peer satisfying the piece request, is dependent upon the ratio of relay peers to total peers in a torrent. This ratio is defined as $r = |M|/|N|$. Thus, the expected path length $E[l]$ is defined as an infinite geometric series:

$$E[l] = \sum_{i=0}^{\infty} r^i = 1 + r + r^2 + r^3 + \cdots + r^\infty = \frac{1}{1-r} \tag{6.1}$$

subject to $0 \leq r < 1$.

The expected path length is plotted as a function of the ratio of relay peers to total peers in a particular anonymous torrent in Figure 6.5. When the ratio of relay peers to total peers is 0, the expected path length is 1.0. There is no relay overhead, since the peers are communicating directly (i.e., $M_l = \emptyset$). When the relay peers are 1/4 of the total peers, the expected path length is 1.33,

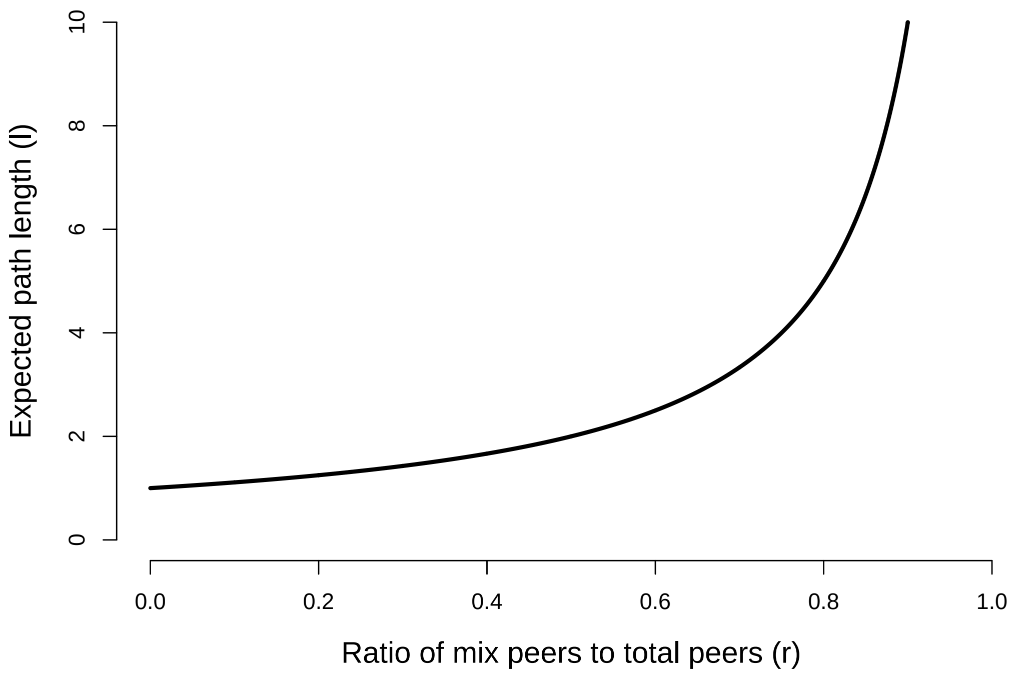


Figure 6.5: The expected path length (l) plotted as a function of the ratio of relay peers to total peers (r)

as the relay peers rise to 1/2 of the torrent, the expected path length is 2.0, and as the relay peers out number total peers as 3/4 of the torrent, the expected path length is 4.0 hops.

6.6.2 Comparison to Tor

As stated in Section 6.5, BitBlender relies on the formation of ad-hoc paths to relay requests and replies. On the other hand, Tor establishes source-routed circuits by choosing a set of precisely three Tor routers (by default) and transporting TCP traffic through these routers using a layered encrypted scheme before the traffic reaches its final destination. By building source-routed circuits, the protocol ensures a path length of precisely four hops from the initiating client to the destination server. This provides relatively strong anonymity properties at the cost of lower throughput and higher latency on average. BitBlender offers a lower expected path length for anonymous torrents in which relay peers constitute less than 3/4 of the total peers participating in the file transfer.

In addition to the relatively high path length, Tor incurs additional protocol overhead to establish these circuits. This consists of layered encryption applied to the circuit-building messages and data packets in a fashion based on onion routing [122]. These circuit building messages must be sent whenever the client chooses to build a new circuit.

Finally, since all traffic is routed through potentially malicious Tor routers, strong confidentiality must be ensured to protect the traffic from local eavesdroppers. However, strictly speaking, the final Tor router that forwards the traffic to the destination server removes the final layer of encryption and can examine a user's payload.

BitBlender is unique in its ability to provide an anonymity service with minimal protocol overhead. Since content is publicly available and specific to each torrent, it is not a strict requirement that BitBlender provide confidentiality and access control (although we do provide these mechanisms as an extension to the protocol in Section 6.8.1).

6.6.3 Security Analysis

Recall that the primary threat model that this protocol should protect against is that of a non-global adversary that participates in the protocol but cannot monitor arbitrary links. Within this model, there exist attacks through which an adversary may gain information about users by recording traffic that it observes during its participation. A naïve attacker may attempt to request pieces through a peer to determine if they are, in fact, a normal peer. This simple attack would be unsuccessful, since relay peers and normal peers both appear to issue and fulfill piece requests.

More intelligent strategies could potentially gain information about the set of real peers. Over time, if an adversary observes that a peer makes a request for the same pieces multiple times, they may be identified as a potential relay [235]. To mitigate this type of attack, normal peers can issue the same piece requests multiple times in a non-deterministic fashion to appear indistinguishable from the relay peers. This technique can be regarded as a form of cover traffic, which is a well-studied traffic analysis mitigation strategy within the context of mix networks.

Additionally, relay peers could cache previously requested pieces, and thereby exhibit more normal (and less distinguishable) behavior. Additional traffic analysis countermeasures are provided in Section 6.8.2.

In addition, Reiter and Rubin identify a set of timing attacks in which an intermediate node (i.e., a relay peer) can determine if the previous node on the path is the initiator of a request based upon an analysis of the time that elapses until the request is fulfilled [189]. If the time is sufficiently small, then the intermediate node can conclude with a certain level of confidence that the preceding node is the initiator. This is an instance of what Wright *et al.* call the predecessor attack [236]. BitBlender, like Crowds, is vulnerable to the predecessor attack. To address this threat, we propose that random delays and selective caching mechanisms be applied to perturb the timing of piece requests and responses (see Sections 6.8.2 and 6.8.3 for a discussion of these techniques). BitBlender's key accomplishment is that an adversary cannot determine which peers are sharing the file simply by examining the tracker; the adversary must now expend more resources and conduct traffic analysis.

6.7 Performance Analysis

In this section, we provide an analysis of the expected performance overhead for the BitBlender protocol in terms of download time as the number of relay peers varies. We also provide a performance comparison to BitTorrent tunneled over the Tor network.

6.7.1 Experimental Setup

In order to quantify the protocol's performance overhead, we implemented BitBlender's relay peers using the Enhanced `ctorrent` BitTorrent client [15]. To ensure that the performance evaluation is conducted in a realistic environment, we perform experiments using nodes from the PlanetLab testbed [177]. In these experiments, there are precisely three seeders, one centralized tracker hosting the torrent metafile for a 1 MiB file, and 20 normal peers actively sharing the file. The file is distributed in 1 KiB pieces. We emulate resource-constrained peers, such as those behind

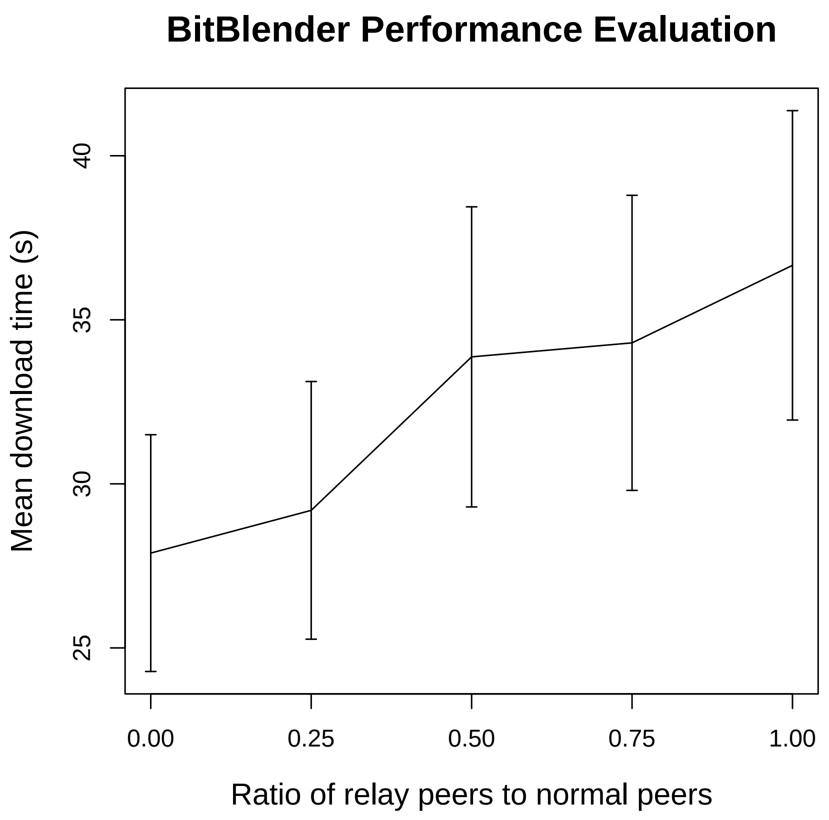


Figure 6.6: Mean download time with 95% confidence intervals as a function of the ratio of relay peers to normal peers

an asymmetric residential cable modem link. All peers are limited to 1 MiB/s for downloads and 256 KiB/s for uploads. To understand the effect of introducing relay peers into the network, we conduct experiments by adding 5, 10, 15, and 20 relay peers to the network. Each experiment is repeated three times to compute statistics.

In order to quantify the performance improvement that BitBlender offers, we provide a performance evaluation of a popular method for sharing files anonymously: BitTorrent run over the Tor network. In this experiment, there are 20 seeders and a single peer using the Azureus/Vuze BitTorrent client [8] tunneled over Tor version 0.2.0.30 (from August 2008). The peers share the same 1 MiB file and are rate-limited as described above. This experiment is repeated ten times.

6.7.2 Experimental Results

We first analyze BitBlender's performance in terms of the expected download time as the ratio of relay peers to normal peers varies. We next compare BitBlender's expected download times to that of BitTorrent over the Tor network.

6.7.2.1 Adding Relay Peers

As shown in Figure 6.6, the mean download time across all 20 peers steadily increases with the number of relay peers participating in the anonymous torrent. As a baseline, when no relay peers participate, peers download the file in approximately 27.9 seconds on average. In the worst case when the ratio of relay peers to normal peers is 1.0, the mean download time is about 36.7 seconds. Note that the download time increases only minimally as more relay peers are added. Since BitTorrent is by its nature a swarming protocol, the performance degradation introduced by having more relay peers participate is partially masked by BitTorrent's tendency to download and upload pieces from multiple peers simultaneously. Thus, the protocol offers reasonably high performance even as the ratio of relay peers to normal peer is relatively high.

6.7.2.2 Comparison to Tor

Over the course of the ten experiments using Tor, the mean download time is 215.1 seconds with a 95% confidence interval of 199.8–230.4 seconds. Not surprisingly, the expected download time using Tor for anonymity is significantly higher than BitBlender, even when the ratio of relay peers to normal peers is 1.0.

6.8 Protocol Extensions

Having presented the basic BitBlender protocol and analysis, we now focus on optional extensions to strengthen the anonymity and increase the performance.

6.8.1 Confidentiality and Access Control

While the attack model assumes that an adversary cannot monitor arbitrary links, it might be the case that an ISP or set of ISPs collude with the adversary. In this case, a confidentiality and access control mechanism would offer an increased level of privacy and anonymity. To this end, we present an additional confidentiality and access control layer. In addition to its role as a directory for relay peers, the blender should also sign public keys for both relay and normal peers as a trusted authority. This public key infrastructure (PKI) can be used by peers to authenticate each other and to restrict access to the content in the torrent. Once authenticated, peers can establish encrypted tunnels using a protocol such as Transport Layer Security (TLS) to protect the content of the torrent. This link encryption would transform messages as they enter one hop and are forwarded to the next hop along multiple hop chains such that it is more difficult to link them.

6.8.2 Traffic Analysis Countermeasures

As previously described, an adversary may attempt to gain information about the peers through traffic analysis techniques. A group of colluding peers could monitor the piece requests and look for anomalous requests, such as multiple piece requests from the same peer. Also, malicious relay peers may issue requests for pieces and determine if those pieces are subsequently requested from another colluding peer. Using a timing correlation attack, it could be possible to identify some of the relay peers by using information related to the timing of the requests. A possible solution to this attack would be introduce intentional random delays as piece requests are forwarded. However, this would have a negative impact upon the system's performance.

Another defense against traffic analysis is to note that these colluding peers will typically have a limited number of IP addresses and will exhibit behaviors that deviate from standard peers and relay peers. If peers could share information in a privacy preserving manner, then they may be able to detect peers performing traffic analysis attacks and blacklist them from the torrent. We encourage future work aimed at addressing traffic analysis attacks in BitBlender and other privacy enhancing systems.

6.8.3 Selective Caching

One technique that may mitigate an adversary's ability to conduct traffic analysis and simultaneously improve performance is the use of selective caching. As relay peers proxy requests, they could cache pieces in main memory as they are forwarded to the requester. As a consequence, the next time that a request is received for a piece that is cached, the relay peer could directly reply with the piece, rather than making another redundant request. This will reduce the expected path lengths for requests of pieces that reside in the relay's cache. Additionally, traffic analysis attempts may be frustrated, since the relay peers now behave as if they possess the requested piece. Using a selective caching policy, the expected path length presented in Section 6.6 becomes an upper bound. However, introducing a selective caching mechanism within an anonymity network exposes a variety of legal questions. In the next section, we provide a brief discussion of the potential legal liabilities that BitBlender (or *any* currently deployed anonymity network) presents.

6.9 Legal Issues

The success of BitBlender, or of any anonymity network, is dependent on the legality of operating an open relay. Anonymous networks can be used to enhance online privacy, enable free speech, and protect human rights; however, they can also be used to hide the identities of people engaged in illegal activities. Operators of relay nodes in networks such as Tor are sometimes accused of preforming illegal activities, since they appear to originate at the relay node [158]. Even though operators of these relay nodes are not directly causing harm, most Western countries have the legal notion of *indirect liability*, where one party can be held responsible for the actions of another party. There are two common categories of indirect liability – *vicarious liability* and *contributory liability.*

Vicarious liability arises when a third party has the ability, duty, or right to control the actions of another party. The main factors for a third party to be held vicariously liable are that they either enable or benefit from these actions. For example, if a bartender serves alcohol to a minor, the bar owner can be held liable for the actions of the bartender. Another common example is that parents or legal guardians can be liable for the wrong-doings of minors. However, phone companies are normally not liable for prank calls placed by their customers. It is unclear if the protection granted to phone companies also applies to Internet Service Providers (ISPs). It is equally unclear if operators of relay nodes in anonymizing multi-hop networks can be held vicariously liable for the actions of other users. However, United States law has a provision that makes caching and retransmission of unmodified cached files legal [1]. The law was upheld when Google's caching policy was challenged [17].

Contributory liability occurs when a third party has induced or has reasonable knowledge of wrong-doing and fails to act to prevent these actions. An example of when a third party can be found contributorily liable is if an ISP receives notice that there is copyright infringing material present on their network and does not remove the infringing material. It is unclear what responsibility operators of relay nodes have for removing copyright infringing material that is transmitted through their nodes. It is clear from the United States Supreme Court ruling against Grokster [27] that the operators of an anonymity network are more likely to be found liable for inducing illegal activity if they advertise their tools as mechanisms to commit infringement, although many other factors are also important. This means that it is essential that anonymizing multi-hop networks be advertised as tools to enhance privacy and enable anonymous speech. Further discussion of the legal and policy implications of P2P file sharing and anonymiziation can be found in Bauer *et al.* [54].

6.10 Summary

We present BitBlender, an efficient protocol that aims to offer a usable and inter-operable anonymity layer for BitTorrent. In contrast to several existing privacy enhancing systems that provide anonymity for general-purpose traffic, we explore the design of a protocol-specific service that does not rely upon cryptography to achieve its anonymity properties. We show that such a design offers increased performance and adequate anonymity properties for the purpose of file transfers.

BitBlender builds an ad-hoc relay network that offers plausible deniability for the initiators of piece requests. We argue that this degree of anonymity is sufficient to obscure the identities of peers participating in a file download through BitTorrent. In addition, the protocol has the ability to dynamically adjust the degree of anonymity provided for the torrent based upon the adjustment of system parameters, specifically the number of relay peers present in an anonymous torrent.

As future work, we propose studies aimed at exploring the feasibility of confidentiality and access control mechanisms within this framework. Also, additional work is necessary to adequately study traffic analysis attacks and provide practical solutions. Finally, performance improvements through the use of various caching policies should be explored further. Systems that offer a level of anonymity that is appropriate for the degree of anonymity required are an intriguing concept and deserve additional research.

Chapter 7

Improving Congestion and Flow Control in Tor

Tor [103] is a distributed circuit-switching overlay network consisting of over two-thousand volunteer-run Tor routers operating around the world. Tor clients achieve anonymity by source-routing their traffic through three Tor routers using onion routing [122].

Context. Conventional wisdom dictates that the level of anonymity provided by Tor increases as its user base grows [102]. Another important, but often overlooked, benefit of a larger user base is that it reduces suspicion placed on users simply because they use Tor. Today, there are an estimated 150 to 250 thousand daily Tor users [151]. However, this estimate has not increased significantly since 2008. One of the most significant road blocks to Tor adoption is its excessively high and variable delays, which inhibit interactive applications such as web browsing.

Many prior studies have diagnosed a variety of causes of this high latency (see Dingledine and Murdoch [105] for a concise summary). Most of these studies have noted that the queuing delays often dominate the network latencies of routing packets through the three routers. These high queuing delays are, in part, caused by bandwidth bottlenecks that exist along a client's chosen circuit. As high-bandwidth routers forward traffic to lower-bandwidth downstream routers, the high-bandwidth router may be able to read data faster than it can write it. Because Tor currently has no explicit signaling mechanism to notify senders of this congestion, packets must be queued along the circuit, introducing potentially long and unnecessary delays for clients. While recent proposals seek to re-engineer Tor's transport design, in part, to improve its ability to handle congestion [142, 187, 227], these proposals face significant deployment challenges.

Improving congestion and flow control. To reduce the delays introduced by uncontrolled congestion in Tor, we design, implement, and evaluate two classes of congestion and flow control. First, we leverage Tor's existing end-to-end window-based flow control framework and evaluate the performance benefits of using small fixed-size circuit windows, reducing the amount of data in flight that may contribute to congestion. We also design and implement a dynamic window resizing algorithm that uses increases in end-to-end circuit round-trip time as an implicit signal of incipient congestion. Similar solutions are being considered for adoption in Tor to help relieve congestion [98], and we offer a critical analysis to help inform the discussion. Window-based solutions are appealing, since they require modifications only to exit routers.

Second, we offer a fresh approach to congestion and flow control inspired by standard techniques from Asynchronous Transfer Mode (ATM) networks. We adapt an ATM-style per-link credit-based flow control algorithm called N23 [145] to Tor that allows Tor routers to explicitly bound their queues and signal congestion via back-pressure, reducing unnecessary delays and memory consumption. While N23 offers these benefits over the window-based approaches, its road to deployment may be slower, as it may require all routers along a circuit to upgrade.

Evaluation. We conduct a holistic experimental performance evaluation of the proposed algorithms using the ModelNet network emulation platform [226] with realistic traffic models. We show that the window-based approaches offer up to 65% faster web page response times relative to Tor's current design. However, they offer poor flow control, causing bandwidth under-utilization and ultimately resulting in poor download time. In contrast, our N23 experiments show that delay-sensitive web clients experience up to 65% faster web page responses and a 32% decrease in web page load times compared to Tor's current design.

7.1 Tor's Approach to Congestion and Flow Control

Since the Tor network consists of volunteer-run routers from across the world, these routers have varying and often limited amounts of bandwidth available to relay Tor traffic. Consequently, as clients choose their circuits, some routers have large amounts of bandwidth to offer, while

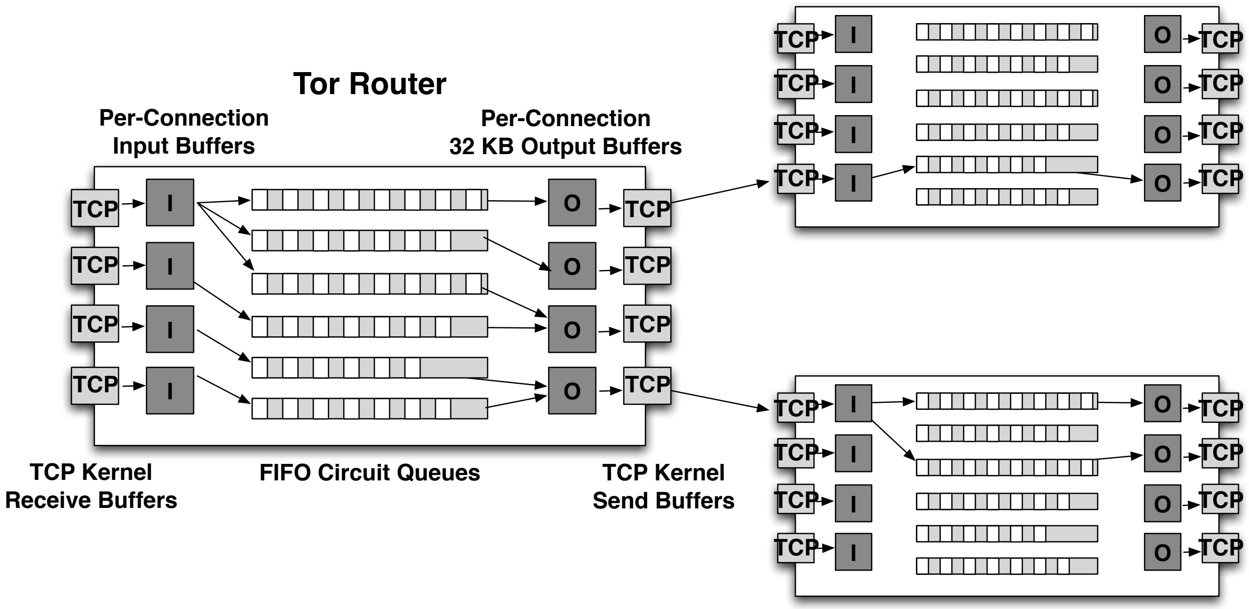

Tor Router
Per-Connection Input Buffers
Per-Connection 32 KB Output Buffers
TCP
I
O
TCP Kernel Receive Buffers
FIFO Circuit Queues
TCP Kernel Send Buffers

Figure 7.1: A Tor router's queuing architecture

others may be bandwidth bottlenecks. In order for Tor to offer the highest degree of performance possible, it is necessary to have effective mechanisms in place to ensure steady flow control, while also detecting and controlling congestion. In this section, we discuss the many features that directly or indirectly impact congestion and flow control in Tor.

7.1.1 Congestion and Flow Control Mechanisms

Pairwise TCP. All packets sent between Tor routers are guaranteed to be delivered reliably and in-order by using TCP transport. As a result of using TCP, communications between routers can be protected with TLS link encryption. However, several circuits may be multiplexed over the same TCP connections, which could result in an unfair application of TCP's congestion control [187].

Tiered output buffers. Each Tor router's internal queuing architecture is illustrated in Figure 7.1. When a Tor router receives a cell on one of its TCP connections, the cell is first copied from the connection's receive kernel buffer into an application-layer input buffer to be decrypted. Next, the cell is pushed onto a FIFO *circuit queue* for the cell's respective circuit. For each outgoing TCP connection, a FIFO *output buffer* is maintained. The output buffer has a fixed size of 32 KiB, while the circuit queue has no explicit bound, but the circuit window size restricts how many cells may be in flight (described below). Since multiple circuits are often multiplexed over the same TCP connection, when there is space available in the outgoing connection's respective output buffer, the router must choose which circuits' cells to copy onto the output buffer. Initially, cells were chosen by round-robin selection across circuits. Recently, circuit prioritization has been proposed to give shorter, burstier circuits that likely correspond to interactive traffic priority over long-lived, bulk circuits [221].

Circuit and stream windows. Tor uses two layers of end-to-end window-based flow control between the exit router and the client to ensure steady flow control. First, a *circuit window* restricts how many cells may be in flight per circuit. By default, Tor uses a fixed 500 KiB (1000 cell) circuit window. For every 50 KiB (100 cells) received, an acknowledgment cell called a `SENDME` is sent, informing the sender that they may forward another 100 cells to the receiver.[1]

Within each circuit window is a *stream window* of 250 KiB (500 cells) to provide flow control (or fairness) within a circuit. The receiver replies with a stream-level `SENDME` for every 25 KiB (50 cells) received. On receiving a stream-level `SENDME` the sender may forward another 50 cells.

Both the stream-level and circuit-level windows are relatively large and static. To illustrate how this can degrade performance, consider the following scenario. Suppose a client downloads a file through a circuit consisting of 10 MiB/s entry and exit routers and a 128 KiB/s middle router. Since the exit router can read data from the destination server faster than it can write it to its outgoing connection with the middle router, and the reliable TCP semantics preclude routers from dropping cells to signal congestion, the exit router must buffer up to one full circuit window (500 KiB) worth of cells. Furthermore, as shown in Figure 7.2, these cells often sit idly for several seconds while the buffer is slowly emptied as `SENDME` cells are received. Since cells may travel down a circuit in large groups of up to 500 KiB followed by periods of silence while the exit router waits for `SENDME` replies, Tor's window-based flow control does not always keep a steady flow of cells in flight.

[1] Due to a bug, clients running `Tor 0.0.0–0.2.1.19` erroneously reply with circuit-level `SENDME` cells after receiving 101 cells (rather than 100 cells).

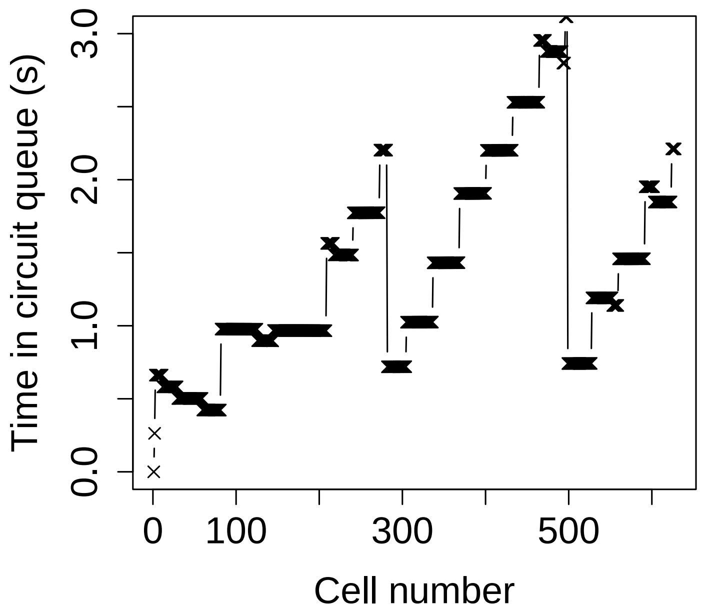


Figure 7.2: The exit router's circuit queue delays for a 300 KiB download

Token bucket rate limiting. In order to allow routers to set limits on the amount of bandwidth they wish to devote to transiting Tor traffic, Tor offers token bucket rate limiting. Briefly, a router starts with a fixed amount of tokens, and decrements their token count as cells are sent or received. When the router's token count reaches zero, the router must wait to send or receive until the tokens are refilled. To reduce Tor's CPU utilization, tokens are refilled only once per second. It has been previously observed that refilling the tokens so infrequently contributes in part to Tor's overall delays [92].

7.1.2 Alternate Proposals to Reduce Congestion

There have been several recent proposals aimed specifically at reducing Tor's congestion. First, Tor has incorporated adaptive circuit-building timeouts that measure the time it takes to build a circuit, and eliminate circuits that take an excessively long time to construct [77]. The intuition is that circuits that build slowly are highly congested, and would in turn offer the user poor performance. While this approach likely improves the users' quality of service in some cases, it does not help to relieve congestion that may occur at one or more of the routers on a circuit *after* the circuit has been constructed.

In addition, user-level rate limiting has been proposed to throttle over-active or bulk downloading users. Here, the idea is to reduce the overall bandwidth consumption by bulk downloaders by using per-connection token bucket rate limiting at the entry guard. Early experiments indicate faster downloads for small file downloaders (the majority of Tor users), while harming bulk downloaders [99].

7.2 Improving Tor's Congestion and Flow Control

Our primary goal is to improve Tor's performance, specifically by better understanding and improving Tor's congestion and flow control. We consider two broad classes of solutions. First, we wish to understand how much improvement is possible simply by adjusting Tor's existing end-to-end window-based flow control mechanisms to reduce the amount of data in flight, and thereby mitigate congestion. We also evaluate an end-to-end congestion control technique that enables exit Tor routers to infer incipient congestion by regarding increases in end-to-end round-trip time as a congestion signal. Second, we consider a fresh approach to congestion and flow control in Tor, eliminating Tor's end-to-end window-based flow control entirely, and replacing it with ATM-style, per-link flow control that caps routers' queue lengths and applies back-pressure to upstream routers to signal congestion.

7.2.1 Improving Tor's Existing End-to-end Flow Control

We first consider whether adjusting Tor's current window-based flow control can offer significant performance improvements. Keeping Tor's window-based mechanisms is appealing, as solutions based on Tor's existing flow control framework may be deployed immediately, requiring modifications only to the exit routers, not clients or non-exit routers.

Small fixed-size circuit windows. The smallest circuit window size possible without requiring both senders and receivers to upgrade is 50 KiB (100 cells, or one circuit-level `SENDME` interval). We evaluate how fixed 50 KiB circuit windows impact clients' performance.[2]

Dynamic circuit windows. We next consider an algorithm that initially starts with a small, fixed circuit-window and dynamically increases the window size (*e.g.*, amount of unacknowledged data allowed to be in flight) in response to positive end-to-end latency feedback. Inspired by latency-informed congestion control techniques for IP networks [69, 229], we propose an algorithm that uses increases in perceived end-to-end circuit round-trip time (RTT) as a signal of incipient congestion.

The algorithm works as follows. Initially, each circuit's window size starts at 100 cells. First, the sender calculates the circuit's end-to-end RTT using the circuit-level `SENDME` cells, maintaining the minimum RTT (rtt_{min}) and maximum RTT (rtt_{max}) observed for each circuit. We note that rtt_{min} is an approximation of the base RTT, where there is little or no congestion on the circuit. Next, since RTT feedback is available for every 100 cells,[3] the circuit window size is adjusted quickly using an additive increase, multiplicative decrease (AIMD) window scaling mechanism based on whether the current RTT measurement (rtt) is less than the threshold T, defined in Equation 7.1. This threshold defines the circuit's tolerance to perceived congestion.

$$T = (1 - \alpha) \times rtt_{min} + \alpha \times rtt_{max} \tag{7.1}$$

Choosing a small α value ensures that the threshold is close to the base RTT, and any increases beyond the threshold implies the presence of congestion along the circuit.[4] For each RTT measurement, *e.g.*, each received circuit-level `SENDME` , the circuit window size (in cells) is adjusted according to Equation 7.2.

[2] Due to the aforementioned bug, in practice, the window size should be 101 cells.
[3] Similar to the 50 KiB windows, `SENDME` cells may be available after 101 cells.
[4] For our experiments, we use $\alpha = 0.25$.

$$new\_window(rtt) = \begin{cases} old\_window + 100 & \text{if } rtt \leq T \\ \lfloor old\_window/2 \rfloor & \text{otherwise} \end{cases} \tag{7.2}$$

Finally, we explicitly cap the minimum and maximum circuit window sizes at 100 and 1000 cells, respectively.[5]

7.2.2 ATM-style Congestion and Flow Control for Tor

Since Tor's flow control works at the circuit's edges–the client and the exit router–it may require up to one full circuit round-trip time to react to perceived congestion along the circuit. In addition, window-based solutions may not offer optimal flow control, as cells are often forwarded in groups of up to one full circuit-window in size, following by periods of silence on the circuit while acknowledgments are in flight. To address these problems, we apply an Asynchronous Transfer Mode (ATM)-style per-link flow control to ensure a steady flow of cells while reducing congestion at the intermediate switches. ATM is a circuit-switching data link layer that is widely used by the telecommunications industry to relay real-time voice traffic [178].

While Tor and ATM certainly have different purposes and goals, they have many similarities. First, both Tor and ATM are connection-oriented, in the sense that before higher-level applications can send or receive data, virtual circuits must be constructed across multiple routers or switches through a signaling mechanism. Second, both Tor and ATM encode data in fixed-sized cells, rather than variable length packets as in IP networks. While ATM networks drop cells in the event of a buffer overflow at an intermediate switch, due to Tor's use of TCP between routers, Tor cannot drop cells because there is no re-transmission mechanism. Similarly, ATM networks do not typically re-transmit cells end-to-end; to do so, would incur an end-to-end latency that would have detrimental effects on the real-time voice traffic being carried. To eliminate the possibility of cell loss, credit-based flow control approaches explicitly bound the number of cells that can be received at each

[5] Note that a selfish Tor client could attempt to increase their circuit window by pre-emptively acknowledging data segments before they are actually received. Prior work in mitigating similar behavior in selfish TCP receivers may be applied here [194, 203].

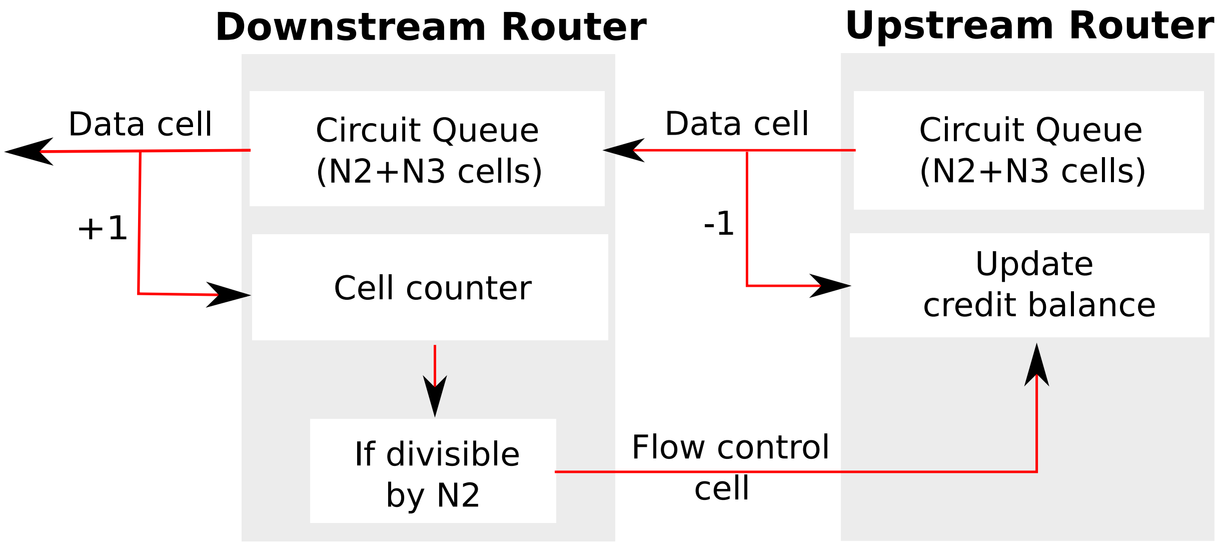


Figure 7.3: N23 credit-based flow control in Tor

switch by the size of the switch's buffer [132]. A similar per-link approach that explicitly caps Tor's circuit queue sizes could enable better congestion control. Thus, given the many similarities between Tor and ATM, we propose to adapt a standard per-link ATM-style flow control algorithm to Tor.

N23 flow control for Tor. Figure 7.3 depicts the N23 scheme that we integrated into Tor, and it works as follows. First, when a circuit is built, each router along the circuit is assigned an initial *credit balance* of $N2+N3$ cells, where $N2$ and $N3$ are system parameters. $N2$ cells is the available steady state buffering per circuit, $N3$ cells is the allowed excess buffering, and the circuit's queue length is strictly bounded by $N2+N3$ cells. In general, $N2$ is fixed at the system's configuration time, but $N3$ may change over a circuit's lifetime.

When a router forwards a cell, it decrements its credit balance by one for that cell's circuit. Each router stops forwarding cells if its credit balance reaches zero. Thus, routers' circuit queues are bounded by $N2+N3$ cells, and congestion is indicated to upstream routers through this back-pressure. Next, for every $N2$ cells forwarded, the downstream router sends a *flow control cell* to the upstream router that contains credit information reflecting its available circuit queue space.

On receiving a flow control cell, the upstream router updates the circuit's credit balance and may forward cells only if the credit balance is greater than zero.

Adaptive buffer sizes and congestion control. The algorithm as described assumes a static $N3$. We also developed an adaptive algorithm that adjusts the $N3$ value when there is downstream congestion, which is detected by monitoring the delay that cells experience in the connection's output buffer. When the congestion subsides, $N3$ can increase again. The value of $N3$ is updated periodically and is bounded by a minimum and a maximum value (100 and 500 cells, respectively).

Advantages. The N23 algorithm has two important advantages over Tor's current flow control. First, the size of the circuit queue is explicitly capped, and guaranteed to be no more than $N2+N3$ cells. This also ensures steady flow control, as routers typically have cells available to forward. Tor's current flow control algorithm allows the circuit queue of a circuit's intermediate routers to grow up to one circuit window in size, which not only wastes memory, but also results in unnecessary delays due to congestion. In contrast, for typical parameter values ($N3 = 500$ and $N2 = 10$), N23 ensures a strict circuit queue bound of 510 cells, while these queues currently can grow up to 1000 cells in length.

The second advantage is that adaptive N3 reacts to congestion within a single link RTT. When congestion occurs at a router, the preceding router in the circuit will run out of credit and must stop forwarding until it gets a flow control cell.

7.3 Experiments and Results

To empirically demonstrate the efficacy of our proposed improvements, we offer a whole-network evaluation of our congestion and flow control algorithms using the ModelNet network emulation platform [226]. Briefly, ModelNet enables the experimenter to specify realistic network topologies annotated with bandwidth, delay and other link properties, and run real code on the emulated network.

Our evaluation focuses on performance metrics that are particularly important to the end-user's quality of service. First, we measure *time-to-first-byte*, which is how long the user must wait

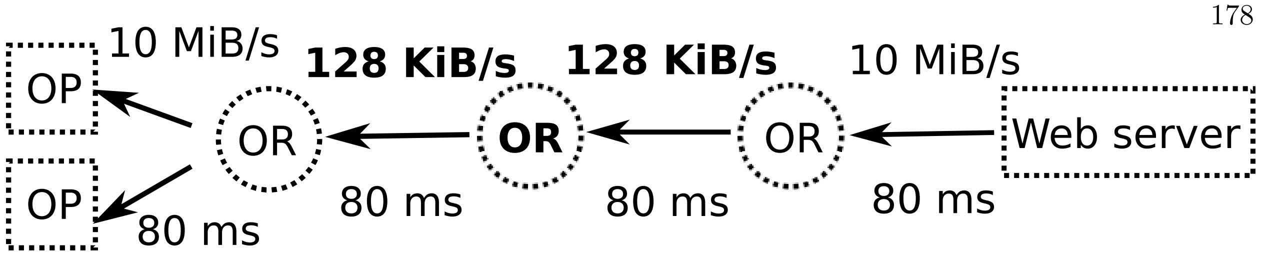


Figure 7.4: A simple topology with a middle router bandwidth bottleneck and 80 ms link RTTs

from the time they issue a request for data until they receive the first byte. The time-to-first-byte is two end-to-end circuit RTTs: one RTT to connect to the destination web server, and a second RTT to issue a request for data (*e.g.*, HTTP `GET`) and receive the first byte of data in response.[6]

Second, we measure *overall download time* (including time-to-first-byte). For all experiments, we use the latest development branch of the Tor source code (version `0.2.3.0-alpha-dev`).[7]

7.3.1 Small-scale Experiments

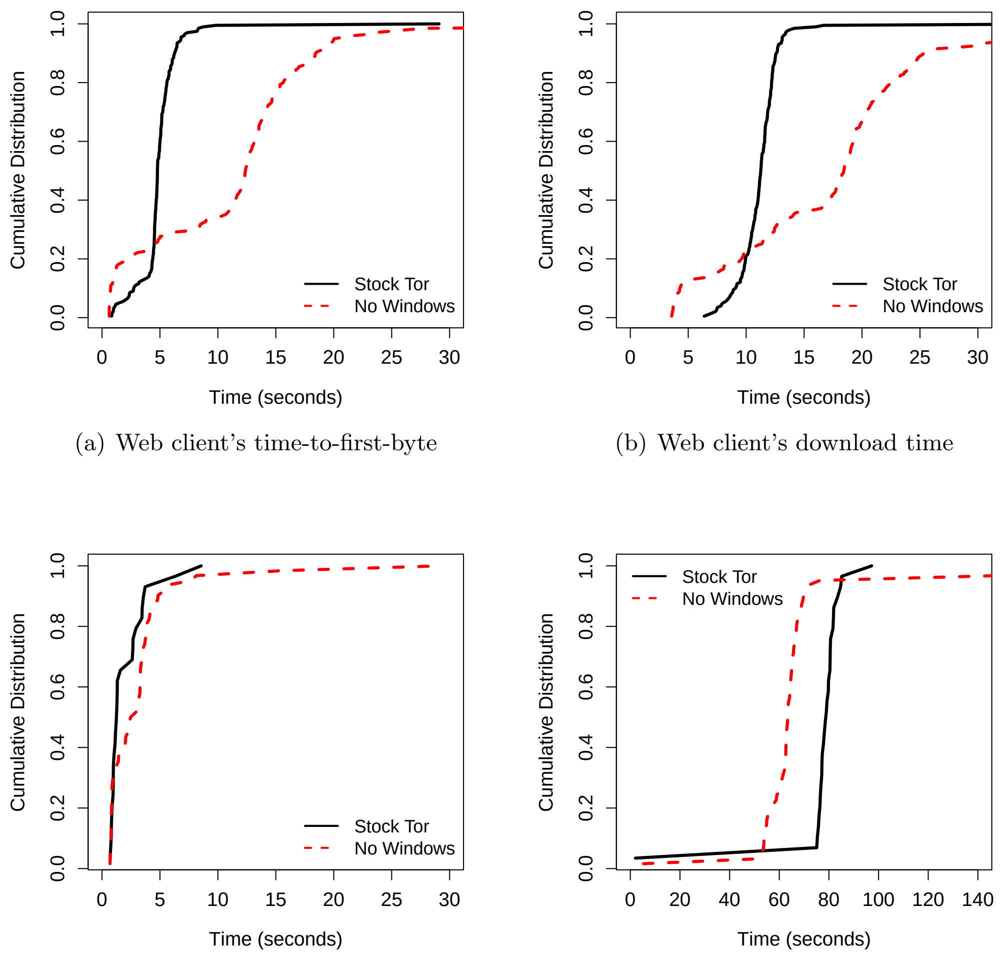
Setup. We emulate the topology depicted in Figure 7.4 on ModelNet where two Tor clients compete for service on the same set of routers with a bandwidth bottleneck at the middle router.[8]

The objective of this experiment is to analyze our congestion and flow control proposals in the presence of congestion. One client downloads 300 KiB files, which roughly correspond to the size of an average web page [186]. The second client, a bulk downloader, fetches 5 MiB files. To avoid synchronization, both clients pause for a random amount of time between one and three seconds, and repeat their downloads. Each experiment concludes after the web client completes 200 downloads. Each client uses the `wget` web browser and the destination runs the `lighthttpd` web server.

[6] Note that there is a proposal being considered to eliminate one of these RTTs [120].

[7] In our evaluation, we refer to unmodified Tor version `0.2.3.0-alpha-dev` as *stock Tor*, 50 KiB (100 cell) fixed windows as *50 KiB window*, the dynamic window scaling algorithm as *dynamic window*, and the N23 algorithm as *N23*.

[8] Note that a 128 KiB/s router corresponds to the 65th percentile of routers ranked by observed bandwidth, as reported by the directory authorities. Thus, it is likely to be chosen fairly often by clients. We additionally used the iPlane query interface [21] to estimate the pairwise latencies between all Tor routers marked as `active` and `valid`. We estimate that an RTT of 80 ms is at the 25%-percentile of all pairwise links between Tor routers. Thus, this is a fast topology in terms of network latency.



(a) Web client's time-to-first-byte

(b) Web client's download time

(c) Bulk client's time-to-first-byte

(d) Bulk client's download time

Figure 7.5: Bottleneck topology performance comparisons between stock Tor and Tor without any congestion and flow control

No congestion and flow control. Before we present the performance results for our proposed congestion and flow control algorithms, we first seek to motivate the need for a layer of congestion and flow control beyond that which is provided by Tor's use of pairwise TCP between routers. In this experiment, we eliminate Tor's stream and circuit windows and configure the Tor exit router to ignore all `SENDME` cells received. This allows the Tor exit router to read from the destination as quickly as possible, as regulated by the exit router's TCP connection with the destination. The exit router subsequently forwards these cells as quickly as possible down the circuit.

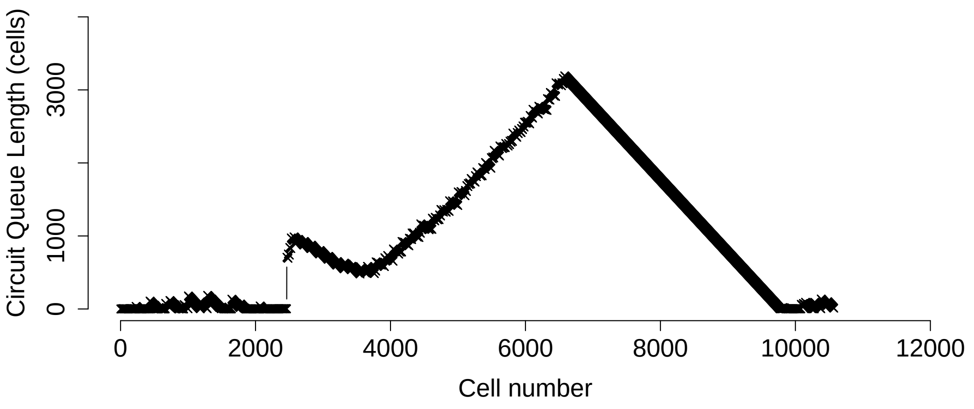


Figure 7.6: Bulk circuit's circuit queue length at the exit router observed over the course of a download with no congestion and flow control in a bottleneck topology

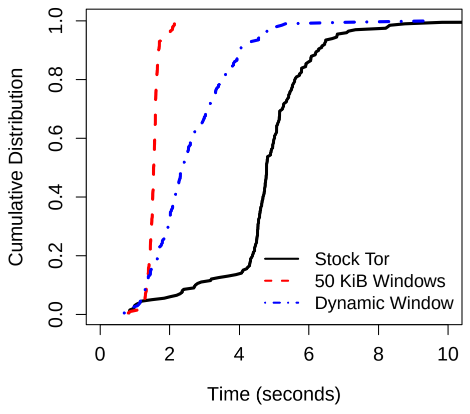
We consider clients that disable Tor's end-to-end window-based congestion and flow control in the bottleneck topology illustrated in Figure 7.4. Figure 7.5(a) shows that the web client's web page response time (or time-to-first-byte) is significantly worse without any congestion and flow control compared to stock Tor's window-based mechanisms. At the median, stock Tor has 4.5 seconds of delay, while Tor without congestion and flow control requires roughly 13 seconds of delay. As even stock Tor's delay is unacceptable for delay-sensitive web clients, removing Tor's windows increases this delay by nearly three times. Similarly, as shown in Figure 7.5(b), removing Tor's congestion and flow control increases total web page load time from a median of 10 seconds with stock Tor to 19 seconds without windows. To improve fairness when bulk downloaders compete for router bandwidth with web clients, these results highlight the need for an additional layer of congestion and flow control beyond that which is provided by Tor's use of pairwise TCP between routers.

Interestingly, the bulk client's performance improves when windows are eliminated. The bulk client's time-to-first byte is slightly worse without windows (see Figure 7.5(c)), but, as shown in Figure 7.5(d), the overall download time is improved from roughly 80 seconds at the median as offered by stock Tor's to under 60 seconds at the median. However, while bulk clients may benefit from improved throughput without Tor's end-to-end windows, in the absence of a mechanism to restrict the amount of data in flight, routers must buffer arbitrarily large amounts of data. For example, Figure 7.6 shows that a sample circuit's queue lengths steadily grow over 3 000 cells in length, which is over 1.4 MiB that must be stored in memory. Clearly, this is an excessive amount of memory to allocate to a single circuit, and worse, if many such circuits exist, the router may ultimately exhaust its physical memory.

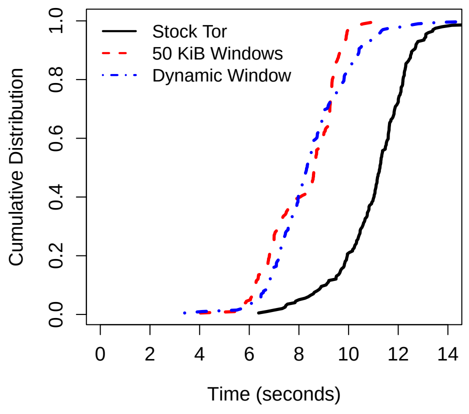
These results indicate that eliminating Tor's end-to-end window-based mechanisms improves throughput, at the cost of high router memory usage (due to unbounded buffering requirements). Perhaps more importantly, the devastating cost to delay-sensitive web clients in the form of significantly increased web page response time and longer web page load time is far too high a price to pay for better throughput. Furthermore, Tor is targeted at delay-sensitive web users, who use Tor to browse the web anonymously and/or to resist censorship. Indeed, the majority of real Tor traffic is interactive web browsing, as we show in Chapter 3. Thus, it is unfair and unwise to harm their performance while improving performance for bulk downloaders.

End-to-end window-based solutions. We next present the performance results for the window-based flow control solutions. Figure 7.7(a) shows that the time-to-first-byte for a typical web client using stock Tor is 4.5 seconds at the median, which is unacceptably high for delay-sensitive, interactive web users who must incur this delay for each web request. In addition, stock Tor's circuit queues fluctuate in length, growing up to 250 cells long, and remaining long for many seconds, indicating queuing delays, as shown in Figure 7.8(a). Reducing the circuit window size to 50 KiB (*e.g.*, one circuit `SENDME` interval) offers a median time-to-first-byte of less than 1.5 seconds, and dynamic windows offer a median time-to-first-byte of two seconds. In Figure 7.7(b), we see that the web client's download time is influenced by the high time-to-first-byte, and is roughly 40% faster with 50 KiB and dynamic windows relative to stock Tor. Also, the circuit queues are smaller with the 50 KiB and dynamic windows (see Figures 7.8(b) and 7.8(c)).

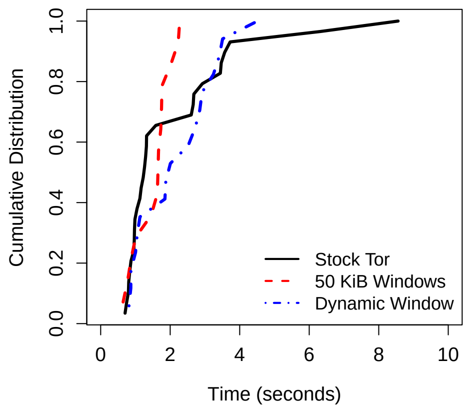
The bulk client experiences significantly less time-to-first-byte delays (in Figure 7.7(c)) than the web client using stock Tor. This highlights an inherent unfairness during congestion: web clients' traffic is queued behind the bulk traffic and, consequently, delay-sensitive clients must wait



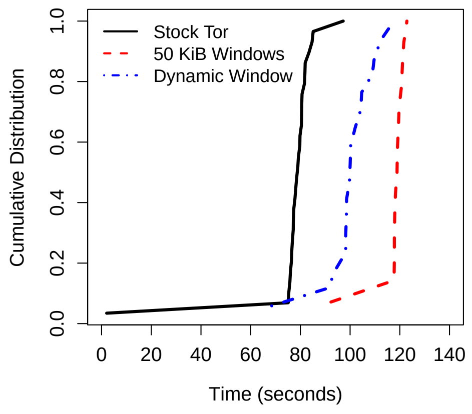
(a) Web client's time-to-first byte



(b) Web client's download time

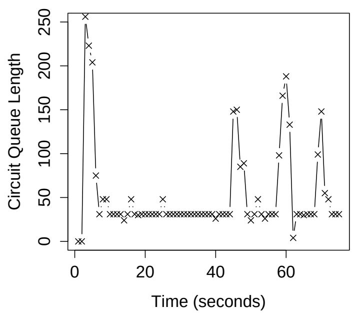


(c) Bulk client's time-to-first-byte

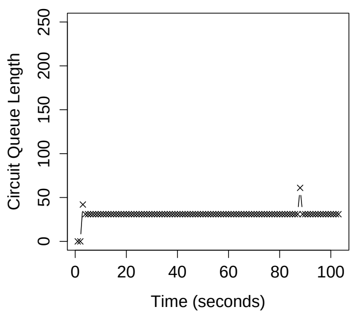


(d) Bulk client's download time

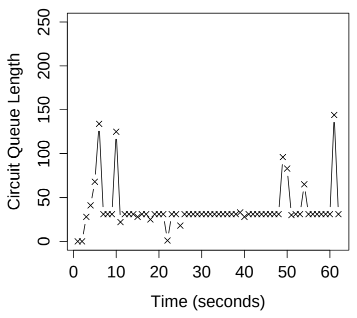
Figure 7.7: Performance comparisons for window approaches in a bottleneck topology



(a) Stock Tor

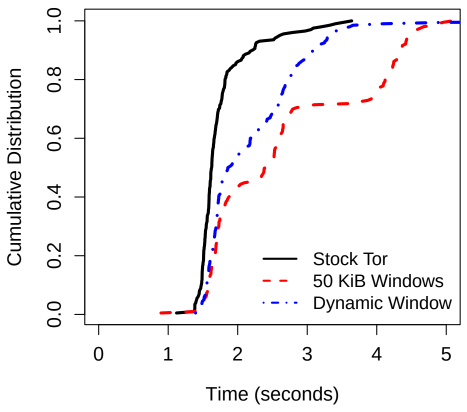


(b) 50 KiB window

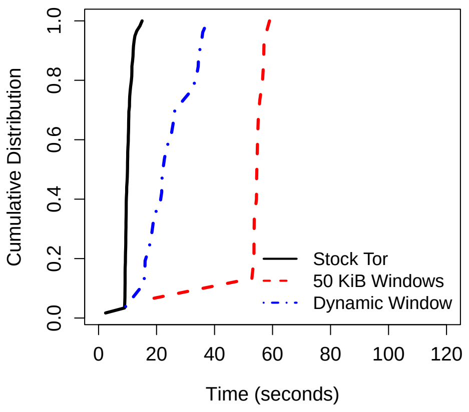


(c) Dynamic window

Figure 7.8: Bulk client's circuit queues at the exit router over the course of a download



(a) Web client's download time



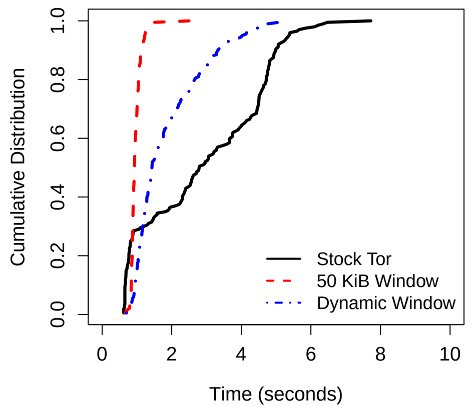
(b) Bulk client's download time

Figure 7.9: Performance comparisons for window approaches in a non-bottleneck topology

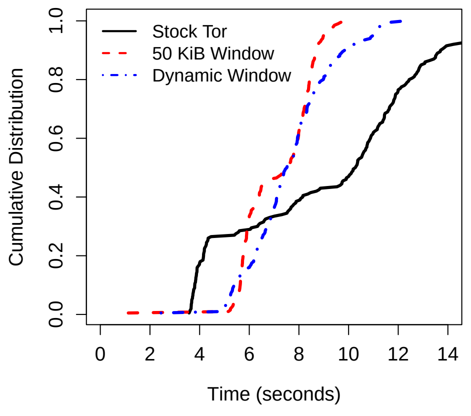
longer than delay-insensitive bulk downloaders to receive their first byte of data. Using a small or dynamic window reduces this unfairness, since the bound on the number of unacknowledged cells allowed to be in flight is lower.

However, Figure 7.7(d) indicates that the bulk client's download actually takes significantly longer to complete with 50 KiB windows relative to stock Tor. Thus, 50 KiB windows enhance performance for web clients at the cost of slower downloads for bulk clients. The bulk clients experience slower downloads because they keep less data in flight and, consequently, must incur additional round-trip time delays to complete the download. Dynamic windows offer a middle-ground solution, as they ameliorate this limitation by offering an improvement in download time for web clients while penalizing bulk clients less than small windows, but bulk clients are still penalized relative to stock Tor's performance.

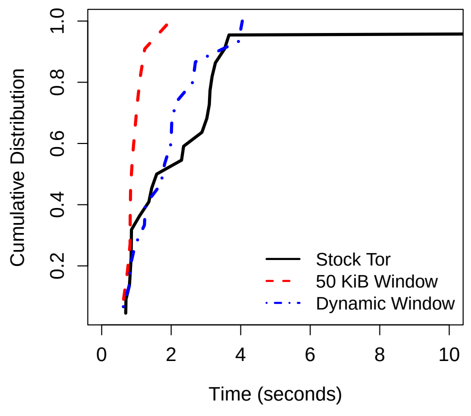
We next consider the same topology shown in Figure 7.4, except we replace the bottleneck middle router with a 10 MiB/s router. In such a topology, congestion is minimal, as evidenced by a median time-to-first-byte of 0.75 s for both the web and bulk clients (regardless of the window size). However, because the 50 KiB and dynamic windows generally keep less data in flight, these solutions offer slower downloads relative to stock Tor, as shown in Figures 7.9(a) and 7.9(b).



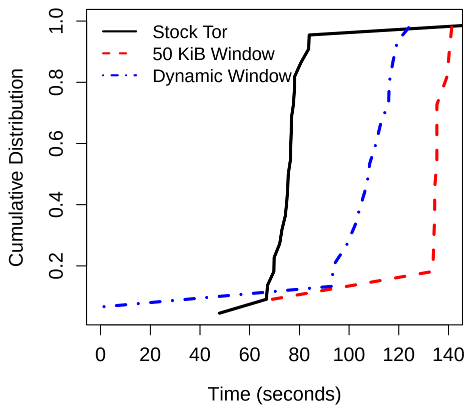
(a) Web client's time-to-first byte



(b) Web client's download time



(c) Bulk client's time-to-first-byte



(d) Bulk client's download time

Figure 7.10: Performance comparisons for window-based congestion and flow control in combination with circuit scheduling prioritization

To mitigate the unfairness that may exist when bursty web circuits compete with bulk transfer circuits for router bandwidth, circuit-level prioritization has been proposed [221] to enable routers to process bursty circuits ahead of bulk circuits. Here, we combine small and dynamic circuit window with circuit scheduling prioritization and evaluate performance in the bottleneck topology from Figure 7.4.[10] For the web client using stock Tor, the time-to-first-byte is reduced from 4.5 seconds (see Figure 7.7(a)) to 3 seconds, and the time-to-first-byte for 50 KiB and dynamic

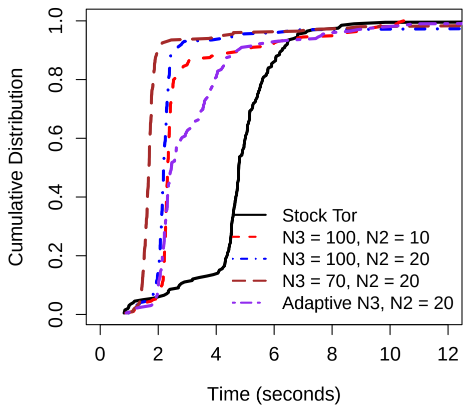
[10] For this experiment, we set `CircuitPriorityHalflifeMsec` to 30 seconds, the current value used on the live Tor network.

windows are roughly the same. However, as shown in Figure 7.10(a), roughly 25% of requests experience no significant improvement when using small or dynamic circuit windows. For these same requests, stock Tor's large window allows more data in flight without acknowledgment and, as shown in Figure 7.10(b), induces faster downloads (compared to Figure 7.7(b)). However, for the remaining 75%, small and dynamic windows offer faster downloads. The bulk client's time-to-first-byte and overall download times are not significantly altered by the circuit prioritization, as shown in Figures 7.10(c) and 7.10(d), relative to non-prioritized circuit scheduling (see Figures 7.7(c) and 7.7(d)). This is consistent with the claims made by Tang and Goldberg [221] that priority-based circuit scheduling does not significantly effect bulk clients' performance.

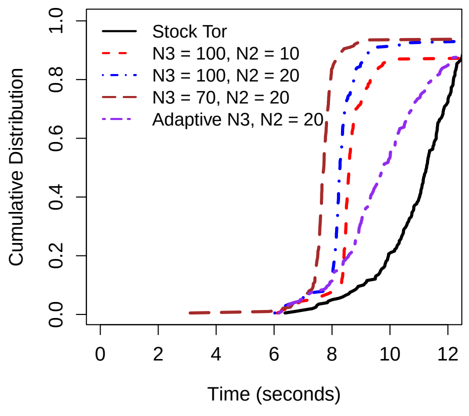
Despite the improvements in time-to-first-byte in the presence of bandwidth bottlenecks, we find that smaller circuit windows tend to under-utilize the available bandwidth and the dynamic window scaling algorithm is unable to adjust the window size fast enough, as it receives congestion feedback infrequently (only every 100 cells). Also, even in the non-bottleneck topology, the 50 KiB window web client's time-to-first-byte is higher than the optimal delay from two circuit RTTs, which is 0.64 s. Lastly, 50 KiB windows offer worse flow control than Tor's current design, since only 50 KiB can be in flight, and the exit router must wait for a full circuit RTT until more data can be read and sent down the circuit.

Based on these drawbacks, we conclude that in order to achieve an improvement in both time-to-first-byte and download speed, it is necessary to re-design Tor's fundamental congestion and flow control mechanisms. We next offer an evaluation of per-link congestion and flow control for Tor.

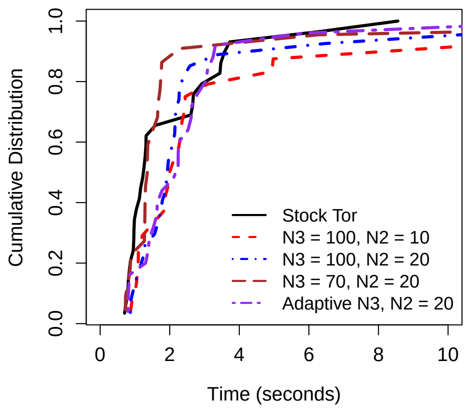
Per-link congestion and flow control. We first present experiments where we configure N23 with fixed values for both N2 and N3 (*static N23*). Next, we experiment with a dynamic N3 value that react to network feedback (*adaptive N3*). For these experiments, we disable Tor's window-based flow control entirely, so that exit routers simply discard `SENDME` cells that they receive from clients. In this section, we present the results of N23 for both the bottleneck and non-bottleneck topologies.



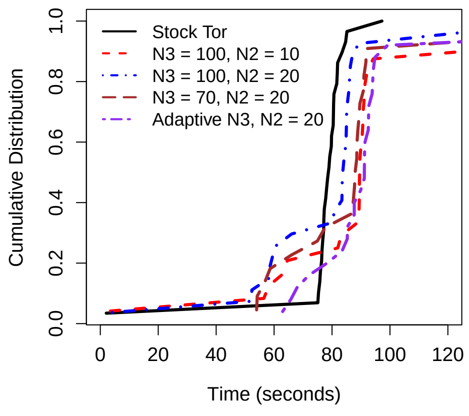
(a) Web client's time-to-first-byte



(b) Web client's download time



(c) Bulk client's time-to-first-byte

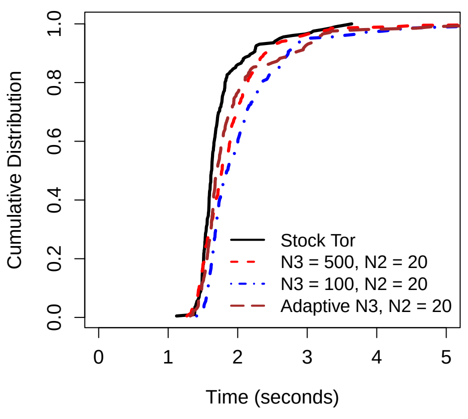


(d) Bulk client's download time

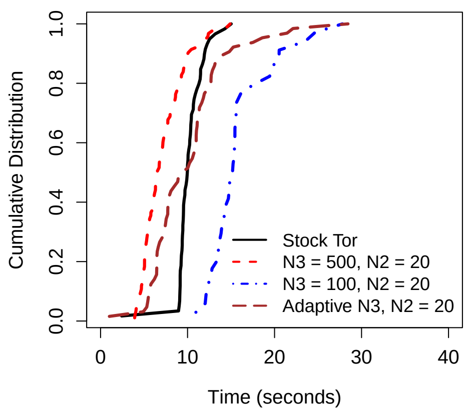
Figure 7.11: Performance comparisons for Tor and N23 in a bottleneck topology

For bottleneck scenarios, Figures 7.11(a) and 7.11(b) show that smaller values of N3 improve both the download time and time-to-first-byte for the bursty web traffic. For example, the web browsing client experiences a 20% decrease in download time for 80% of the requests when N23 is used. Also, the web client's time-to-first-byte is only two seconds for 90% of the requests, whereas for the stock Tor client, 80% of web requests take more than four seconds to receive the first byte. Figure 7.13 shows that the circuit queue length is upper bounded by $N2 + N3 = 90$ cells.

In the non-bottleneck topology, we see in Figure 7.12(b) that N23 provides a substantial improvement in download time for the 5 MiB downloads compared to stock Tor only for higher



(a) Web client's download time



(b) Bulk client's download time

Figure 7.12: Download time comparison for Tor and N23 in a non-bottleneck network

values of N3 — 500 cells, comparable to stock Tor's stream window size. The graph shows that there is a 25% decrease in delay for 50% of the bulk downloads when N23 is used. Since the maximum throughput is bounded by W/RTT, where W is the link's TCP window size and RTT is the link's round-trip time, and since N23's per-link RTT is significantly smaller than a stock Tor's complete circuit RTT, throughput is increased when N23 is used. This improvement suggests that in non-bottleneck scenarios, bulk traffic data cells are unnecessarily slowed down by Tor's flow control at the edges of the circuit. For bursty web traffic, both Tor's current flow control and N23 have similar performance for fixed and adaptive N3, as shown in Figure 7.12(a). Also, the median time-to-first-byte is the same for the web and bulk clients at 0.75 s.

To understand how N23 performs with different $N2$ values, we repeated the bottleneck experiments while varying that parameter. Although a higher value for $N2$ has the undesirable effect of enlarging the circuit buffer, it can be seen in Figures 7.11(a) and 7.11(b) that when $N3$ is fixed at 100 cells, increasing $N2$ to 20 cells slightly improves both download time and time-to-first-byte. It can be observed from Figure 7.11(a) that time-to-first-byte is significantly improved by keeping a smaller $N3 = 70$ and a larger $N2 = 20$. Decreasing $N3$ to 70 cells makes up for the increase in the $N2$ zone of the buffer, which means we gain the benefits of less flow control overhead, and

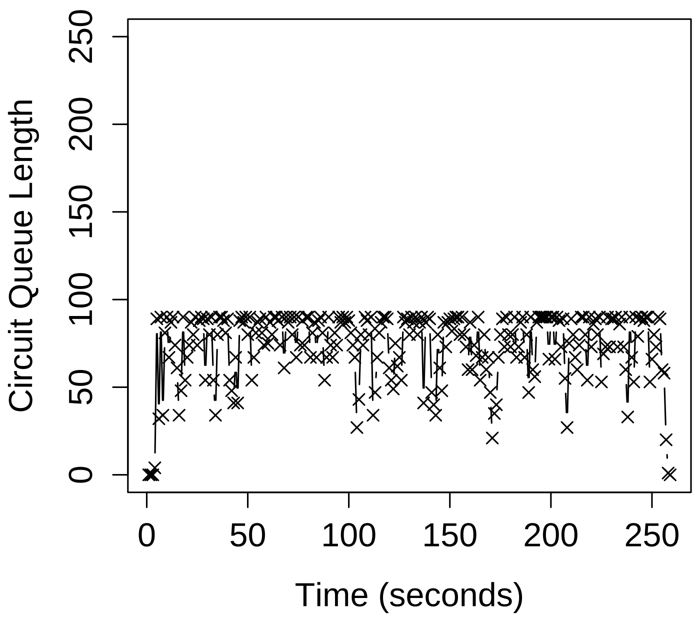


Figure 7.13: Circuit queue length with bottleneck: N3 = 70, N2 = 20

the benefits of a small buffer of $N2 + N3 = 90$ cells. While performance is improved for the web client, the bulk client's time-to-first-byte is not affected greatly, as seen in Figure 7.11(c), but its downloads generally take longer to complete, as we see in Figure 7.11(d). In addition, adaptive N3 offers improved time-to-first-byte and download times for the web client, while slowing downloads for the bulk client. By N23 restricting the amount of data in flight, the bandwidth consumed by bulk clients is reduced, improving time-to-first-byte and download time for delay-sensitive web clients.

Finally, the bandwidth cost associated with the N23 scheme is relatively low. For instance, with $N2 = 10$, a flow control cell must be sent by each router on the circuit for every 10 data cells forwarded, which requires a 10% bandwidth overhead per router. For $N2 = 20$, a flow control cell is sent for every 20 data cells, which is only a 5% overhead per router. While this cost is higher than Tor's window-based flow control (*e.g.*, one stream-level `SENDME` for every 50 data cells is only a 2% overhead per steram and one circuit-level `SENDME` for every 100 data cells is only 1% overhead per circuit), the cost of N23 is nonetheless modest.

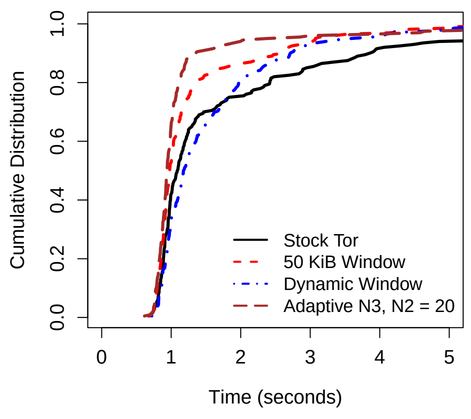
7.3.2 Larger-scale Experiments

Setup. We next evaluate the window-based solutions and N23 with adaptive N3 in a larger network topology.[9] We deploy 20 Tor routers on a random ModelNet topology whose bandwidths are assigned by sampling from the live Tor network. Each link's round-trip time is set to 80 ms. Next, to generate a traffic workload, we run 200 Tor clients. Of these, ten clients are bulk downloaders who fetch files between 1–5 MiB, pausing for up to two seconds between fetches. The remaining 190 clients are web clients, who download files between 100–500 KiB (typical web page sizes), pausing for up to 30 seconds between fetches. This proportion of bulk-to-non-bulk clients approximates the proportion observed on the live Tor network [158]. To isolate the improvements due to our proposals, circuit-level prioritization is disabled for this experiment.

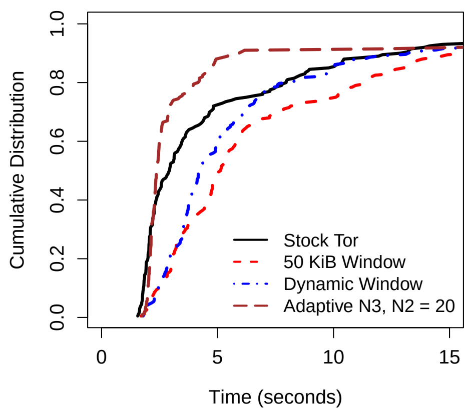
Results. For web clients, Figure 7.14(a) shows that both the 50 KiB fixed and dynamic windows still offer improved time-to-first-byte. However, both algorithms perform worse than stock Tor in terms of overall download time, as shown in Figure 7.14(b). Because smaller windows provide less throughput than larger windows when there is no bottleneck, non-bottlenecked circuits are under-utilized.

N23 with the adaptive N3 algorithm, in contrast, has the ability to react to congestion quickly by reducing routers' queue lengths, causing back pressure to build up. Consequently, our results indicate that N23 offers an improvement in both time-to-first-byte *and* overall download time. This experiment again highlights the potential negative impact of 50 KiB and small dynamic windows, since even in a larger network with a realistic traffic load, smaller windows offer worse performance for typical delay-sensitive web requests relative to Tor's current window size. Thus, to achieve maximal improvements, we suggest that Tor adopt N23 congestion and flow control.

[9] In this experiment, we only consider N23 with adaptive N3 because in practice, N23 should discover the right buffer size for the given network conditions.



(a) Web client's time-to-first-byte



(b) Web client's download time

Figure 7.14: Performance results for large-scale experiments

7.4 Discussion

Having empirically evaluated our proposed congestion and flow control approaches, we next discuss a variety of open issues.

7.4.1 Optimality and Comparison to an Alternative Transport Design

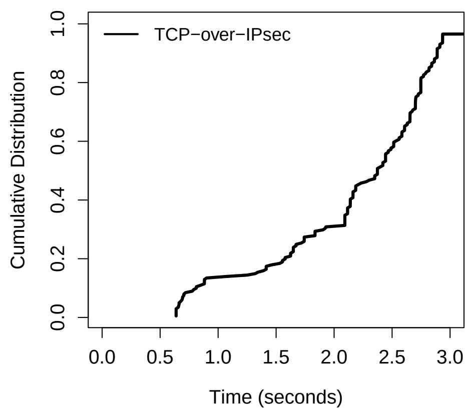
Given the results of our performance analysis showing an improvement in web page response time and web page load time enabled by re-engineering Tor's congestion and flow control mechanics, it is reasonable to ask: *Are these results optimal?* and *Can we do better?*. To begin to answer these questions, we offer a performance comparison between our congestion and flow control proposals within Tor's current transport architecture and a proposed transport re-design.

TCP-over-IPsec. We compare our results to a proposed anonymous overlay based on a layer three approach called *IPpriv* that leverages end-to-end TCP (*e.g.*, between the client and destination) over an IPsec [139] layer three path of anonymizing IPsec routers [142, 143]. In short, this system offers source-destination unlinkability similar to Tor, but it uses IPsec link security with layered onion routing-style encryption using IPsec's Encapsulating Security Payload (ESP) [140] and a telescoping key establishment protocol based on an extension to IKEv2 [70]. This system architecture uses the end-systems' TCP congestion and flow control algorithms end-to-end; thus, there is no need for any additional higher layer of congestion and flow control.

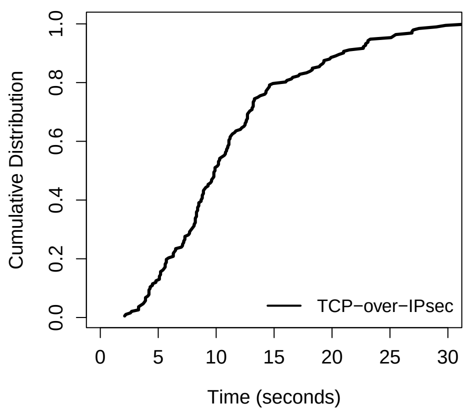
Setup. This experiment's setup is similar to the experiments presented in Chapter 7.3.1, except we assume an IPsec anonymizing network instead of Tor. We construct an IPsec path between two end-hosts with a 128 KiB/s bottleneck IPsec router in the middle. This topology is similar to the small bottleneck topology from Figure 7.4, with the same end-to-end round-trip time of 320 ms (80 ms per link with 4 links). Next, two clients compete for bandwidth over this IPsec path. One client, a web user, downloads 300 KiB and the second client, a bulk downloader, fetches 5 MiB. The experiment completes after the web client finishes 200 downloads. For simplicity, we abstract way the computational and performance overhead associated with the IPsec cryptography, since any cryptography overhead will be dominated by the high propagation delays and the bandwidth bottleneck. Thus, we consider our results an upper bound on attainable performance.

Results. Figure 7.15(a) shows that the web client's time-to-first-byte is about 2.25 s at the median, which is less than stock Tor's time-to-first-byte from Figure 7.7(a). However, this delay in web page response is over two times the delay offered by small windows, dynamic windows, and N23 with $N3 = 100$ cells (see Figure 7.11(a)). Similar to Tor, TCP-over-IPsec still suffers from queuing delays at intermediate routers where bulk clients' traffic fills the queues and bursty web traffic must incur significant delays while the bulk traffic is slowly forwarded. In addition, Figure 7.15(b) shows that the web client's page load time is roughly 10 s at the median with TCP-over-IPsec transport. While this is an improvement over stock Tor, N23 with $N3 = 100$ offers web page load times that are less than 8 s at the median (see Figure 7.11(b)).

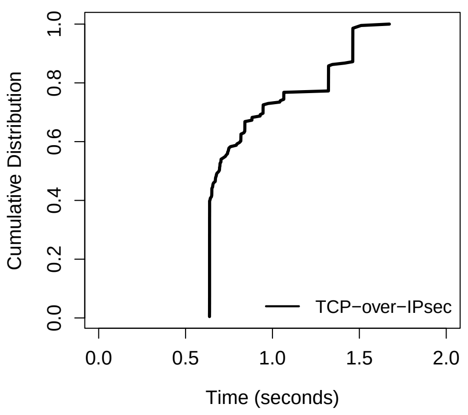
For bulk clients, Figure 7.15(c) shows that they have a significantly faster time-to-first-byte than the web client, as they only compete for router queue space with the 300 KiB web client, which doesn't require large amount of queuing. The same trend was observed in stock Tor (see Figures 7.7(c) and 7.11(c)). Also, the bulk client experiences better download time – with a median of about 50 s – with TCP-over-IPsec than with stock Tor or any of our congestion and flow control proposals (see Figures 7.7(c) and 7.11(d)). This improvement in download time for bulk



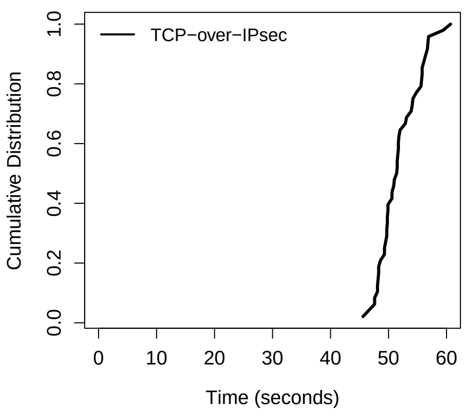
(a) Web client's time-to-first-byte



(b) Web client's download time



(c) Bulk client's time-to-first-byte



(d) Bulk client's download time

Figure 7.15: Small-scale bottleneck performance results for TCP-over-IPsec

downloaders is the result of using TCP's native congestion and flow control algorithms, rather than artificially restricting the amount of data in flight, as in stock Tor's layer seven congestion and flow control. Kiraly *et al.* also observed that TCP-over-IPsec offers improved throughput relative to Tor [142], however, they do not consider the effects of bandwidth bottlenecks and router queuing on the performance of small flows (*e.g.*, web clients) when there is contention with large flows (*e.g.*, bulk downloaders).

Discussion. In summary, our proposed congestion and flow control mechanisms offer better time-to-first-byte and overall page load time for web clients compared to TCP-over-IPsec because they effectively restrict the amount of data that the bulk client can add to the network, while freeing scarce bandwidth and router queue space for smaller flows that likely correspond to interactive, delay-sensitive web traffic. While is not clear that our approach offers the optimal performance, this experiment indicates that it does offer improved performance for web users compared to one alternative transport design. The TCP-over-IPsec design does, however, offer improved throughput for bulk downloaders. To improve web client's performance, it may be possible to adopt a queuing strategy at the intermediate routers that prioritizes bursty, web traffic over bulk downloads. Such a queuing policy has been demonstrated be effective in low bandwidth access links to generally mitigate the harmful performance effects that result from bursty traffic competing with bulk flows [216].

According to our experiments, TCP-over-IPsec transport offers the best throughput, however, exposing the client's raw TCP/IP packets to the destination may enable a new side channel for information leakage that is not currently possible in Tor's design. A malicious destination server could learn information about the client's networking stack and/or operating system type and/or version using standard open-source fingerprinting tools such as p0f [173], PRADS [32], and Nmap [28]. These tools leverage such features that tend to vary between operating system and TCP/IP implementations including the initial packet size, initial time-to-live (TTL), window size, maximum segment size, and others. Identifying characteristic differences in these features may enable an adversary to learn information about the client, but they typically cannot identify the client uniquely. However, if TCP timestamps are enabled, then remote device fingerprinting techniques that analyze variations in clock skew can be used to uniquely identify the client's specific hardware [144] (and de-anonymize the client). Given that these attacks have been proved successful in the wild and that there exist free, open-source tools to implement them, the client's TCP/IP packets should not be exposed to any party other than the entry point into the anonymizing network (*e.g.*, the entry guard, in Tor's case). Thus, the decrease in throughput offered by Tor relative to TCP-over-IPsec is the inherent cost of protection from these attacks.

7.4.2 Incremental Deployment

In order for our proposed congestion and flow control mechanisms to be practical and easily deployable on the live Tor network, it is important that any modifications to Tor's router infrastructure be incrementally deployable. Any solutions based on Tor's existing window-based flow control require upgrades only to the exit routers; thus they can be slowly deployed as router operators upgrade. N23 may also be deployed incrementally, however, clients may not see substantial performance benefits until a large fraction of the routers have upgraded.

7.4.3 Anonymity Implications

A key question to answer is whether improving Tor's performance and reducing congestion enables any attack that was not previously possible. It is well known that Tor is vulnerable to congestion attacks wherein an attacker constructs circuits through a number of different routers, floods them with traffic, and observes if there is an increase in latency on a target circuit, which would indicate a shared router on both paths [166]. More recent work has suggested a solution that would mitigate bandwidth amplification variants of this attack, but not the shared router inference part of the attack [113]. We believe that by reducing congestion (and specifically, by bounding queue lengths), our proposed techniques may increase the difficulty of mounting congestion attacks.

However, if only a fraction of the routers upgrade to our proposals and if clients only choose routers that support the new flow control, then an adversary may be able to narrow down the set of potential routers that a client is using. Thus, it is important to deploy any new flow control technique after a large fraction of the network has upgraded. Such an incremental deployment can be controlled by setting a flag in the authoritative directory servers' consensus document, indicating that it is safe for clients to use the new flow control.

Another well-studied class of attack is end-to-end traffic correlation. Such attacks endeavor to link a client with its destination when the entry and exit points are compromised, and these attacks have been shown to be highly accurate [61,168,172,197,206]. Reducing latency might improve this attack; however, Tor is already highly vulnerable, so there is little possibility for additional risk.

Finally, previous work has shown that network latency can be used as a side channel to infer a possible set of client locations [127]. By reducing the variability in circuits' latencies, we might expose more accurate latency measurements, thus improving the effectiveness of this attack. However, reducing congestion does not enable a new attack, but rather may potentially increase the effectiveness of a known attack. To put this attack in perspective, Tor's design has already made many performance/anonymity trade-offs, and thus, we believe that our performance improvements outweigh any potential decrease in anonymity brought about by reducing the variance in latency.

7.4.4 Performance over Asymmetric Links

Given the performance degradation that we empirically demonstrate in Chapter 7.3.1 due to the presence of bandwidth bottlenecks and congestion, another question to answer is whether asymmetric links contribute additional dynamics during congestion. Ensuring high Tor performance over asymmetric links is important, as emerging residential broadband access network technologies such as DOCSIS 3.0 [12] offer significantly more bandwidth than has been previously available to residential Internet users. For example, the DOCSIS 3.0 standard supports downstream bandwidth up to 160 Mb/s and upstream bandwidth up to 120 Mb/s [40]. The prevalence of high bandwidth Internet connectivity in both the downstream and upstream directions could be an untapped source for valuable relay bandwidth for anonymizing networks such as Tor. Thus, it is necessary to ensure that systems like Tor still offer acceptable performance in asymmetric bandwidth environments.

To understand how asymmetric links can cause performance problems when there are bi-directional TCP flows, consider the following scenario. Suppose that an asymmetric link has 50 Mb/s downstream bandwidth and 25 Mb/s upstream bandwidth (these are reasonable values given DOCSIS 3.0). Further suppose that there are two flows utilizing the link, where one is a 10 MiB download and the other a 10 MiB upload. In such a scenario, it has been shown that the download's throughput may decrease because its TCP acknowledgments are queued behind the upload's data packets in the access network's intermediate router's queues [136]. Furthermore, sharing a common buffer for TCP data segments and acknowledgments in end-systems results in *ACK compression*, in which acknowledgments arrive in groups leading to loss of throughput, and the negative performance effects of ACK compression have been shown to be more severe with asymmetric links [238].

While bi-directional TCP performance degradation over an asymmetric link is not a new problem specific to systems like Tor, but rather an inherent problem in TCP, it is nonetheless important to reduce such performance loss in order to enable residential broadband subscribers to make significant bandwidth contributions to Tor. To reduce the harmful effects of ACK compression and queuing delay, ACK prioritization and back-pressure have been proposed [53, 136]. The former approach schedules acknowledgments ahead of data packets, to reduce their delays in IP router queues. The later approach restricts the IP routers' queue lengths by applying back-pressure to the TCP layer. Note that ACK prioritization is not possible if the acknowledgments are piggy-backed on top of data packets (which is typically the case for bi-directional traffic). To address this limitation, it has been suggested that overlay networks that use a single TCP connection for bi-directional traffic should separate each direction of traffic into two distinct TCP connections, and subsequently prioritize ACKs [156].

7.5 Summary

We seek to improve Tor's performance by reducing unnecessary delays due to poor flow control and excessive queuing at intermediate routers. By improving performance – more specifically, by improving expected web page response time and web page load time – we enhance Tor's usability, making it more appealing to a wider audience, and ultimately, improving Tor's anonymity properties by growing its user base. In this chapter, we proposed two broad classes of congestion and flow control. First, we tune Tor's existing circuit windows to effectively reduce the amount of data in flight. However, our experiments indicate that while window-based solutions do reduce queuing delays, they tend to suffer from poor flow control, under-utilizing the available bandwidth, and consequently, smaller windows provide slower downloads than unmodified Tor.

To solve this problem, we offer a fresh approach to congestion and flow control in Tor by adapting, implementing, and experimentally evaluating a per-link congestion and flow control algorithm from ATM networks. Our experiments indicate that this approach offers reduced web page response times and faster overall web page downloads.

Chapter 8

Conclusions and Future Work

This thesis seeks to improve the security and performance offered by Tor, the most widely used privacy enhancing technology for achieving online anonymity and resisting censorship. To that end, we offer a detailed analysis of attacks on Tor's anonymity and an in depth study aimed at diagnosing and improving Tor's performance problems. At the beginning of this dissertation, we asserted the following thesis statement:

> *Low latency anonymity networks can offer greater anonymity and better performance than provided by existing systems.*

We next explain how we addressed this statement by summarizing this dissertation's fundamental contributions.

8.1 Fundamental Contributions

This dissertation's contributions can be summarized as follows.

8.1.1 First Characterization of a Live Anonymity Network

In order to understand how a popular low latency anonymity network is used in practice, we provided the first characterization of the live Tor network. In particular, we showed that Tor transports a large amount of non-TLS protected exit traffic, which enables a Tor exit router operator, a destination server, or an eavesdropper along the network path between the Tor exit router and the destination to learn information about clients, even including uniquely identifiable user names or login credentials. To reduce the risk of inadvertently exposing identifying information within application-layer traffic, we proposed that Tor provide a mechanism to explicitly block the TCP ports that are commonly used by insecure protocols such POP3, IMAP, and telnet [59]. We also showed that the majority of Tor exit traffic is interactive HTTP by connection and volume, but file sharing traffic consumes a disproportionate amount of Tor's scarce bandwidth. We also found Tor users tend to originate in over 126 countries around the world, but router bandwidth is concentrated primarily in Germany and the United States. Lastly, we identified a variety of malicious behaviors both in Tor clients and routers.

8.1.2 Low-resource Traffic Confirmation Attacks and Defenses

Since low latency anonymity networks generally do not employ padding or perturb the traffic's temporal properties, these systems are vulnerable to end-to-end traffic confirmation attacks. We show that Tor's use of bandwidth-weighted router selection allows an adversary who controls several high-bandwidth routers to attract a large amount of clients' traffic and compromise the circuits' end-points with high probability. However, we show that because Tor routers self-advertise their own bandwidth capacities, it is possible for an adversary with a few low bandwidth routers to falsely inflate their bandwidth claims to draw traffic and thereby compromise circuits. We also develop a novel circuit linking algorithm that uses only circuit construction traffic to correlate clients with their respective destinations before any data is sent. In combination with selective disruption tactics, our Planetlab experiments show that an adversary who controls only six low bandwidth Tor routers who advertise high bandwidths in a network with 66 total routers can compromise up to 46% of all circuits constructed. In addition, we improve Tor's entry guard design by placing restrictions on their believable resource claims. We offer a variety of mitigation strategies to increase the difficulty of launching these attacks, but ultimately, without secure bandwidth verification, these attacks still remain possible.

8.1.3 Security and Performance Evaluation of Path Length

Systems for low latency anonymous communications generally employ a multi-hop architecture to limit the amount of information that the anonymizing routers can learn about communicating parties. In Tor's design, precisely three routers are used for clients' circuits. We challenge the assumption that three routers is the optimal path length. In particular, we show that two-hop paths offer improved performance in terms of bandwidth savings and an overall reduction in clients' delays. Also, we show that two-hop paths are more resilient to end-to-end traffic correlation attacks where the adversary employs selective disruption tactics. However, two-hop paths trivially reveal the client's chosen entry guards to the exit router, which may enable profiling. To reduce this risk, we introduce the notion of path length blending, where the client attempts to conceal the length of their circuit by generating dummy traffic that looks like three-hop path construction traffic. Ultimately, we argue that two-hop paths offer significant performance benefits and are more resilient to realistic attacks.

8.1.4 Improving Performance by Offering an Alternative for BitTorrent

Given our observation that BitTorrent consumes a disproportionate amount of Tor's scarce bandwidth, we design, implement, and empirically evaluate a Crowds-style anonymizing solution tailored specifically for the BitTorrent file sharing protocol. We first offer a case study the many avenues of information leakage in BitTorrent and then we present BitBlender, which is a dynamic relay network that provides a degree of plausible deniability for peers listed by the BitTorrent tracker servers. Our performance results indicate that BitBlender offers significantly faster downloads and uploads relative to Tor, while simultaneously improving Tor's performance for delay-sensitive web users by potentially eliminating a significant source of congestion.

8.1.5 Improving Tor's Congestion and Flow Control

To further improve Tor's performance, we diagnose a source of performance degradation caused by Tor's use of an end-to-end window-based congestion and flow control mechanism with large and static windows. We first evaluate the performance benefits of using small fixed-size windows and a dynamic window scaling algorithm that re-sizes the window in response to end-to-end circuit latency-based congestion signals. We find that while web page response time is improved, overall throughput suffers because less data is allowed to be in flight. To solve this problem, we propose a per-link congestion and flow control algorithm inspired by standard techniques from ATM networks. We find that this approach offers an improvement in both web page response time and web page load time.

8.2 Future Work

We next enumerate a variety of avenues for future work to extend the fundamental contributions of this thesis.

8.2.1 Additional Performance Improvements

To build on the work presented in this thesis, we plan to continue diagnosing Tor's performance problems and offering improvements. In particular, there are a variety of open questions from our performance analysis in Chapter 7. First, we wish to explore how the proposed congestion and flow control mechanisms perform in practice on the live Tor network. In addition, Tor's current design provides a layer of flow control within the circuit-level, ostensibly to ensure some notion of fairness among multiple streams within the same circuit. Our analysis does not consider fairness within a circuit, but rather, we focus on fairness among multiple competing circuits. Future work should investigate whether Tor needs an additional mechanism to ensure fairness among streams. Lastly, we plan to work toward devising and migrating to a next-generation Tor-like anonymity network with an improved transport design.

8.2.2 Improving Router Selection with Link-based Metrics

To improve both security and performance, link-based router selection has been proposed as an alternative to so-called node-based (*e.g.*, bandwidth) router selection [200, 202]. In short, link-based router selection allows the client to select a circuit with desired end-to-end properties satisfying any combination of latency, jitter, AS-traversal, country-traversal, or other constraints. In the case of latency-based router selection, virtual coordinate systems can be applied to scalably estimate pairwise latencies between routers [84].

One important question is how link-based router selection will scale in practice when users begin to request paths with certain specific constraints. For example, if a fraction of users desire paths with low end-to-end latencies, can these paths still offer such low latency when they begin to experience higher congestion due to increased selection? Also, another interesting avenue to explore is whether link-based router selection is compatible with bandwidth-weighted selection. Finally, in addition to virtual coordinate systems, it may be possible to leverage additional information sources such as IDMaps [115] or iPlane [21, 154, 155] to make routing decisions based on additional link properties beyond latency.

8.2.3 Secure Bandwidth Verification

To reduce the risk of the low resource attacks we present in Chapter 4, Tor has adopted a bandwidth-weighted router selection scheme that uses active bandwidth verification from a set of trusted bandwidth measurement authorities [176]. In addition, other secure bandwidth verification methods have been proposed [212]. We plan to investigate the security and overhead of these approaches and others to finally secure Tor's router selection process.

8.3 Final Remarks

To conclude, we have presented a variety of results that collectively aim to improve the security, anonymity, and performance of low latency anonymity networks. By enhancing Tor's performance and security, more users will participate in Tor (both as clients and as router operators) and, thereby, Tor's user base will grow, strengthening all users' anonymity. Our hope is that this work helps to enable more people to achieve online anonymity, enhance the privacy of their online activities, and resist censorship.

Bibliography

[1] 17 United States Code Section 512. http://www4.law.cornell.edu/uscode/17/512.html.

[2] 2nd Workshop on Ethics in Computer Security Research (WECSR 2011). http://www.cs.stevens.edu/~spock/wecsr2011.

[3] African network information centre. http://www.afrinic.net.

[4] Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2.

[5] American registry for Internet numbers. http://www.arin.net/index.shtml.

[6] The anonymizer. http://www.anonymizer.com.

[7] Asia pacific network information centre. http://www.apnic.net.

[8] Azureus BitTorrent client. http://azureus.sourceforge.net.

[9] BayTSP. http://www.baytsp.com.

[10] BitTorrent protocol specification. http://wiki.theory.org/BitTorrentSpecification.

[11] BTGuard - BitTorrent Anonymously. http://btguard.com.

[12] DOCSIS 3.0. http://www.bci.eu.com/wp-content/uploads/2009/12/docsis-30-11.pdf.

[13] Echoping performance measurement. http://echoping.sourceforge.net.

[14] EFF: AOL's data valdez. http://w2.eff.org/Privacy/AOL.

[15] Enhanced CTorrent. http://www.rahul.net/dholmes/ctorrent.

[16] Ethereal. http://www.ethereal.com.

[17] Field v. Google, Inc., 412 F. Supp 2d. 1106 (D. Nev. 2006).

[18] First Workshop on Ethics in Computer Security Research (WECSR 2010). http://www.cs.stevens.edu/~spock/wecsr2010.

[19] The GNU privacy guard. http://www.gnupg.org.

[20] I2P Anonymous Network. http://www.i2p2.de.

[21] iPlane: An information plane for distributed services. http://iplane.cs.washington.edu.

[22] Ipredator. http://ipredator.se.

[23] JonDonym - The Anonymisation service. http://anonymous-proxy-servers.net.

[24] Latin american & caribbean Internet addresses registry. http://lacnic.net/en.

[25] Mandatory Data Retention. http://www.eff.org/issues/mandatory-data-retention.

[26] Media defender – P2P anti-piracy and P2P marketing solutions. http://www.mediadefender.com.

[27] MGM Studios Inc. v. Grokster, ltd., 545 U.S. 913 (Supreme Court 2005).

[28] Nmap. http://nmap.org.

[29] NSA Spying. http://www.eff.org/issues/nsa-spying.

[30] Panopticlick. https://panopticlick.eff.org.

[31] Polipo. http://www.pps.jussieu.fr/~jch/software/polipo/.

[32] PRADS. http://gamelinux.github.com/prads.

[33] Privoxy. http://www.privoxy.org.

[34] Response template for Tor node maintainer to ISP. http://www.torproject.org/eff/tor-dmca-response.html.

[35] Ripe network coordination centre. http://www.ripe.net.

[36] Safenet Inc: The foundation for information security. http://www.safenet-inc.com.

[37] The Pirate Bay. http://thepiratebay.org.

[38] The Wall Street Journal: What They Know. http://blogs.wsj.com/wtk.

[39] Transparent SOCKS Proxying Library. http://tsocks.sourceforge.net.

[40] CableLabs Issues DOCSIS 3.0 Specifications Enabling 160 Mbps. CableLabs Press Release. http://www.cablelabs.com/news/pr/2006/06_pr_docsis30_080706.html, August 2006.

[41] A face is exposed for AOL search no. 4417749. http://www.nytimes.com/2006/08/09/technology/09aol.html, August 2006.

[42] Tor researcher who exposed embassy e-mail passwords gets raided by Swedish FBI and CIA. http://blog.wired.com/27bstroke6/2007/11/swedish-researc.html, November 2007.

[43] Pirate bay tricks anti-pirates with fake peers. http://torrentfreak.com/the-pirate-bay-tricks-anti-pirates-with-fake-peers-081020, October 2008.

[44] Measuring Tor and Iran. https://blog.torproject.org/blog/measuring-tor-and-iran, June 2009.

[45] China blocking Tor: Round Two. https://blog.torproject.org/blog/china-blocking-tor-round-two, March 2010.

[46] Recent Events in Egypt. https://blog.torproject.org/blog/recent-events-egypt, January 2011.

[47] T.G. Abbott, K.J. Lai, M.R. Lieberman, and E.C. Price. Browser-based attacks on Tor. In Privacy Enhancing Technologies, volume 4776 of Lecture Notes in Computer Science, page 184. Springer, 2007.

[48] Alessandro Acquisti and Ralph Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In Privacy Enhancing Technologies, pages 36–58, 2006.

[49] Mashael AlSabah, Kevin Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey Voelker. DefenestraTor: Throwing out Windows in Tor. University of Waterloo Centre For Applied Cryptographic Research Technical Report CACR 2011-06. http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-06.pdf, March 2011.

[50] Elli Androulaki, Mariana Raykova, Shreyas Srivatsan, Angelos Stavrou, and Steven M. Bellovin. PAR: Payment for Anonymous Routing. In Nikita Borisov and Ian Goldberg, editors, Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), pages 219–236, Leuven, Belgium, July 2008. Springer.

[51] Adam Back, Ian Goldberg, and Adam Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[52] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, Proceedings of Information Hiding Workshop (IH 2001), pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[53] H. Balakrishnan, V. N. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. RFC 3449: TCP Performance Implications of Network Path Asymmetry. http://tools.ietf.org/html/rfc3449, December 2002.

[54] Kevin Bauer, Dirk Grunwald, and Douglas Sicker. The arms race in P2P. In Proceedings of the 37th Research Conference on Communication, Information and Internet Policy, September 2009.

[55] Kevin Bauer, Dirk Grunwald, and Douglas Sicker. Predicting Tor path compromise by exit port. In Proceedings of the 2nd IEEE International Workshop on Information and Data Assurance, 2009.

[56] Kevin Bauer, Joshua Juen, Nikita Borisov, Dirk Grunwald, Douglas Sicker, and Damon McCoy. On the optimal path length for Tor. In HotPets in conjunction with Tenth International Symposium on Privacy Enhancing Technologies (PETS 2010), Berlin, Germany, July 2010.

[57] Kevin Bauer and Damon McCoy. Tor specification proposal 109: No more than one server per IP address. https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/109-no-sharing-ips.txt, March 2007.

[58] Kevin Bauer and Damon McCoy. Uptime Sanity Checking. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/107-uptime-sanity-checking.txt`, March 2007.

[59] Kevin Bauer and Damon McCoy. Block insecure protocols by default. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/129-reject-plaintext-ports.txt`, January 2008.

[60] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against anonymous systems. Computer Science Technical Report CU-CS-1025-07, University of Colorado, February 2007.

[61] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against Tor. In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007), Washington, DC, USA, October 2007.

[62] Kevin Bauer, Damon McCoy, Dirk Grunwald, and Douglas Sicker. BitBlender: Light-weight anonymity for BitTorrent. In Proceedings of the Workshop on Applications of Private and Anonymous Communications, Istanbul, Turkey, September 2008.

[63] Kevin Bauer, Damon McCoy, Dirk Grunwald, and Douglas Sicker. BitStalker: Accurately and efficiently monitoring BitTorrent traffic. In First IEEE International Workshop on Information Forensics and Security, London, United Kingdom, December 2009.

[64] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In Roger Dingledine and Paul Syverson, editors, Proceedings of Privacy Enhancing Technologies workshop (PET 2002). Springer-Verlag, LNCS 2482, April 2002.

[65] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping Internet sensors with probe response attacks. In Proceedings of the 14th conference on USENIX Security Symposium, Baltimore, MD, July 2005. USENIX Association.

[66] Stevens Le Blond, Arnaud Legout, Fabrice Lefessant, Walid Dabbous, and Mohamed Ali Kaafar. Spying the world from your laptop: Identifying and profiling content providers and big downloaders in BitTorrent. In Proceedings of 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '10), April 2010.

[67] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In Proceedings of CCS 2007, October 2007.

[68] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[69] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In Proceedings of the conference on Communications architectures, protocols and applications, SIGCOMM '94, pages 24–35, New York, NY, USA, 1994. ACM.

[70] Ed. C. Kaufman. RFC 4306: Internet key exchange (IKEv2) protocol, December 2005.

[71] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. RFC 4880: OpenPGP message format. `http://www.ietf.org/rfc/rfc4880.txt`, 2007.

[72] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In OSDI 2002.

[73] Paul Cesarini. Caught in the Network. In The Chronicle of Higher Education, volume 53. Washington, D.C., February 2007.

[74] A. Chaabane, P. Manils, and M.A. Kaafar. Digging into Anonymous Traffic: A deep Analysis of the Tor Anonymizing Network. In 4th International Conference on Network and System Security (NSS 2010), Melbourne, Australia, September 2010.

[75] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 4(2), February 1981.

[76] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology, 1988.

[77] Fallon Chen and Mike Perry. Improving Tor path selection. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/151-path-selection-improvements.txt`, July 2008.

[78] David Choffnes, Jordi Duch, Dean Malmgren, Roger Guimerà, Fabián Bustamante, and Luís A. Nunes Amaral. Strange bedfellows: Community identification in BitTorrent. In Proceedings of the 9th international conference on Peer-to-peer systems, IPTPS'10, pages 13–13, Berkeley, CA, USA, 2010. USENIX Association.

[79] David R. Choffnes, Jordi Duch, Dean Malmgren, Roger Guierma, Fabian E. Bustamante, and Luis Amaral. SwarmScreen: Privacy through plausible deniability for P2P systems. Northwestern EECS Technical Report, March 2009.

[80] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pages 46–66, July 2000.

[81] Richard Clayton, George Danezis, and Markus G. Kuhn. Real world patterns of failure in anonymity systems. In Ira S. Moskowitz, editor, Proceedings of Information Hiding Workshop (IH 2001), pages 230–244. Springer-Verlag, LNCS 2137, April 2001.

[82] Henry Corrigan-Gibbs and Bryan Ford. Dissent: Accountable anonymous group messaging. In Angelos D. Keromytis and Vitaly Shmatikov, editors, Proceedings of the 2010 ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. ACM, 2010.

[83] Cypherpunks Remailer. `http://www.cypherpunks.to`.

[84] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In Proceedings of the ACM SIGCOMM '04 Conference, Portland, Oregon, August 2004.

[85] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003), pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[86] George Danezis, Claudia Diaz, and Paul Syverson. Systems for anonymous communication. In CRC Handbook of Financial Cryptography and Security, 2009.

[87] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 2003.

[88] George Danezis and Ben Laurie. Minx: A simple and efficient anonymous packet format. In WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society, pages 59–65. ACM, 2004.

[89] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks: Red-green-black mixes. In WPES '03: Proceedings of the 2003 ACM workshop on privacy in the electronic society, pages 89–93. ACM, 2003.

[90] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In Proceedings of 6th Information Hiding Workshop (IH 2004), LNCS, Toronto, May 2004.

[91] Norman Danner, Danny Krizanc, and Marc Liberatore. Detecting denial of service attacks in Tor. In Financial Cryptography and Data Security, pages 273–284, Berlin, Heidelberg, 2009. Springer-Verlag.

[92] Prithula Dhungel, Moritz Steiner, Ivinko Rimac, Volker Hilt, and Keith W. Ross. Waiting for anonymity: Understanding delays in the Tor overlay. In Peer-to-Peer Computing, pages 1–4. IEEE, 2010.

[93] Prithula Dhungel, Di Wu, Brad Schonhorst, and Keith W. Ross. A measurement study of attacks on BitTorrent leechers. In International Workshop on Peer-to-Peer Systems (IPTPS), February 2008.

[94] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, Proceedings of Privacy Enhancing Technologies Workshop (PET 2002). Springer-Verlag, LNCS 2482, April 2002.

[95] T. Dierks. RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1, April 2006.

[96] Roger Dingledine. EFF is looking for Tor DMCA test case volunteers. `http://archives.seul.org/or/talk/Oct-2005/msg00208.html`.

[97] Roger Dingledine. Personal communication.

[98] Roger Dingledine. Prop 168: Reduce default circuit window. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/168-reduce-circwindow.txt`, August 2009.

[99] Roger Dingledine. Research problem: adaptive throttling of Tor clients by entry guards. `https://blog.torproject.org/blog/research-problem-adaptive-throttling-tor-clients-entry-guards`, September 2010.

[100] Roger Dingledine and Nick Mathewson. Tor directory protocol, version 3. `https://www.torproject.org/svn/trunk/doc/spec/dir-spec.txt`.

[101] Roger Dingledine and Nick Mathewson. Tor path specification. `http://tor.eff.org/cvs/doc/path-spec.txt`.

[102] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In Workshop on the Economics of Information Security, June 2006.

[103] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In Proceedings of the 13th USENIX Security Symposium, August 2004.

[104] Roger Dingledine, Nick Mathewson, and Paul Syverson. Challenges in deploying low-latency anonymity. NRL CHACS Report 5540-625, 2005.

[105] Roger Dingledine and Steven Murdoch. Performance improvements on Tor or, why Tor is slow and what we're going to do about it. `http://www.torproject.org/press/presskit/2009-03-11-performance.pdf`, March 2009.

[106] Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, Proceedings of Financial Cryptography (FC '02). Springer-Verlag, LNCS 2357, March 2002.

[107] David Dittrich, Michael Bailey, and Sven Dietrich. Towards community standards for ethical behavior in computer security research. Technical Report 2009-01, Stevens Institute of Technology, Hoboken, NJ, USA, April 2009.

[108] John Douceur. The Sybil Attack. In Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002), March 2002.

[109] Peter Eckersley. How Unique Is Your Browser? In Proceedings of the Privacy Enhancing Technologies Symposium, July 2010.

[110] Matthew Edman, Fikret Sivrikaya, and Bülent Yener. A combinatorial approach to measuring anonymity. In Proceedings of the 2007 IEEE International Conference on Intelligence and Security Informatics (ISI '07), pages 356–363, 2007.

[111] Matthew Edman and Paul F. Syverson. AS-awareness in Tor path selection. In Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS), pages 380–389, 2009.

[112] Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. ACM Computing Surveys, 42(1), 2010.

[113] Nathan Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on Tor using long paths. In Proceedings of the 18th USENIX Security Symposium, August 2009.

[114] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004), Washington, DC, USA, October 2004.

[115] 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 Trans. Netw., 9:525–540, October 2001.

[116] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, November 2002.

[117] Simson L. Garfinkel and Lorrie Faith Cranor. Institutional review boards and your research. Commun. ACM, 53:38–40, June 2009.

[118] Thomer M. Gil, Frans Kaashoek, Jinyang Li, Robert Morris, and Jeremy Stribling. King data set. `http://pdos.csail.mit.edu/p2psim/kingdata`.

[119] Corrado Gini. Measurement of inequality and incomes. The Economic Journal, 1921.

[120] Ian Goldberg. Prop 174: Optimistic data for Tor: Server side. `https://trac.torproject.org/projects/tor/ticket/1795`.

[121] Ian Goldberg. On the security of the Tor authentication protocol. In Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006), Cambridge, UK, June 2006. Springer.

[122] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In Proceedings of Information Hiding: First International Workshop. Springer-Verlag, LNCS 1174, May 1996.

[123] Ryan Henry, Kevin Henry, and Ian Goldberg. Making a Nymbler Nymble using VERBS. In Privacy Enhancing Technologies Symposium, July 2010.

[124] Dominik Herrmann and Rolf Wendolsky. Effectivity of Various Data Retention Schemes for Single-Hop Proxy Servers. In Proceedings of PET-CON, Regensburg, Germany, October 2009.

[125] Thomas Heydt-Benjamin, Andrei Serjantov, and Benessa Defend. Nonesuch: A mix network with sender unobservability. In WPES '06: Proceedings of the 5th ACM workshop on privacy in the electronic society, pages 1–8. ACM, 2006.

[126] Andrew Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul Syverson, editors, Proceedings of Privacy Enhancing Technologies workshop (PET 2002). Springer-Verlag, LNCS 2482, April 2002.

[127] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? In Proceedings of CCS 2007, October 2007.

[128] Internet World Stats. `http://www.internetworldstats.com`.

[129] Iperf – The TCP/UDP bandwidth measurement tool. `http://dast.nlanr.net/Projects/Iperf`.

[130] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving P2P data sharing with OneSwarm. In Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, SIGCOMM '10, pages 111–122, New York, NY, USA, 2010. ACM.

[131] Raj Jain. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, 1991.

[132] Raj Jain. Congestion control and traffic management in ATM networks: Recent advances and a survey. Computer Networks and ISDN Systems, 28:1723–1738, 1995.

[133] Rob Jansen, Nicholas Hopper, and Yongdae Kim. Recruiting new Tor relays with BRAIDS. In Angelos D. Keromytis and Vitaly Shmatikov, editors, Proceedings of the 2010 ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. ACM, 2010.

[134] JAP-Anonymity and Privacy. `http://anon.inf.tu-dresden.de`.

[135] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous IP-address blocking. In Nikita Borisov and Philippe Golle, editors, Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, June 2007. Springer.

[136] Lampros Kalampoukas, Anujan Varma, and K. K. Ramakrishnan. Improving tcp throughput over two-way asymmetric links: analysis and solutions. In Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, SIGMETRICS '98/PERFORMANCE '98, pages 78–89, New York, NY, USA, 1998. ACM.

[137] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In In Proceedings of the Twelfth International World Wide Web Conference, pages 640–651. ACM, 2003.

[138] Sachin Katti, Jeffery Cohen, and Dina Katabi. Information slicing: Anonymity using unreliable overlays. In Proceedings of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI), April 2007.

[139] S. Kent. RFC 2401: Security architecture for the Internet protocol, November 1998.

[140] S. Kent. RFC 2406: IP encapsulating security payload (ESP), November 1998.

[141] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien Petitcolas, editor, Proceedings of Information Hiding Workshop (IH 2002). Springer-Verlag, LNCS 2578, October 2002.

[142] C. Kiraly, G. Bianchi, and R. Lo Cigno. Solving performance issues in anonymiziation overlays with a L3 approach. University of Trento Information Engineering and Computer Science Department Technical Report DISI-08-041, Ver. 1.1, September 2008.

[143] Csaba Kiraly and Renato Lo Cigno. IPsec-based anonymous networking: A working implementation. In Proceedings of the 2009 IEEE International Conference on Communications, pages 2146–2150, Piscataway, NJ, USA, 2009. IEEE Press.

[144] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. Remote physical device fingerprinting. In SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy, pages 211–225, Washington, DC, USA, 2005. IEEE Computer Society.

[145] H. T. Kung, Trevor Blackwell, and Alan Chapman. Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing. SIGCOMM Comput. Commun. Rev., 24:101–114, October 1994.

[146] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. ACM Trans. Program. Lang. Syst., 4(3):382–401, 1982.

[147] Jörg Lenhard, Karsten Loesing, and Guido Wirtz. Performance Measurements of Tor Hidden Services in Low-Bandwidth Access Networks. In Proceedings of the 7th International Conference on Applied Cryptography and Network Security, June 2009.

[148] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, Proceedings of Financial Cryptography (FC '04). Springer-Verlag, LNCS 3110, February 2004.

[149] Marc Liberatore and Brian Neil Levine. Inferring the Source of Encrypted HTTP Connections. In Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006), pages 255–263, October 2006.

[150] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia. A new cell counter based attack against Tor. In Proceedings of the CCS 2009. ACM.

[151] Karsten Loesing. Measuring the Tor network: Evaluation of client requests to the directories. Tor Project Technical Report, June 2009.

[152] Karsten Loesing, Steven Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In Workshop on Ethics in Computer Security Research, January 2010.

[153] Karsten Loesing, Werner Sandmann, Christian Wilms, and Guido Wirtz. Performance Measurements and Statistics of Tor Hidden Services. In Proceedings of the 2008 International Symposium on Applications and the Internet (SAINT). IEEE CS Press, July 2008.

[154] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In OSDI, pages 367–380. USENIX Association, 2006.

[155] Harsha V. Madhyastha, Ethan Katz-Bassett, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In Jennifer Rexford and Emin Gün Sirer, editors, NSDI, pages 137–152. USENIX Association, 2009.

[156] Daniel Marks, Florian Tschorsch, and Bjorn Scheuermann. Unleashing Tor, BitTorrent & Co.: How to Relieve TCP Deficiencies in Overlays (Extended Version). Technical Report TR-2010-001. Computer Science Department, Heinrich Heine University. http://www.cn.uni-duesseldorf.de/publications/library/Marks2010b.pdf, August 2010.

[157] Nick Matthewson. Base "stable" flag on mean time between failures. `http://git.torproject.org/checkout/tor/master/doc/spec/proposals/108-mtbf-based-stability.txt`.

[158] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In Proceedings of the 8th Privacy Enhancing Technologies Symposium, July 2008.

[159] Damon McCoy, Kevin Bauer, Dirk Grunwald, Parisa Tabriz, and Douglas Sicker. Shining light in dark places: A study of anonymous network usage. University of Colorado Technical Report CU-CS-1032-07, August 2007.

[160] Aleecia M. McDonald and Lorrie Faith Cranor. Americans' attitudes about Internet behavioral advertising practices. In Proceedings of the 9th annual ACM workshop on Privacy in the Electronic Society, pages 63–72, New York, NY, USA, 2010. ACM.

[161] Jon McLachlan and Nicholas Hopper. On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design. In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009). ACM, November 2009.

[162] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. AP3: cooperative, decentralized anonymous communication. In EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop, page 30, New York, NY, USA, 2004. ACM.

[163] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. IETF Internet Draft, July 2003.

[164] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, November 2006.

[165] Steven J. Murdoch and Ross Anderson. Tools and Technology of Internet Filtering. In Access Denied: The Practice and Policy of Global Internet Filtering. MIT Press, 2008.

[166] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In Proceedings of the 2005 IEEE Symposium on Security and Privacy. IEEE CS, May 2005.

[167] Steven J. Murdoch and Robert N. M. Watson. Metrics for security and performance in low-latency anonymity systems. In Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, July 2008.

[168] Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In Proceedings of Privacy Enhancing Technologies Workshop (PET 2007), June 2007.

[169] Arjun Nambiar and Matthew Wright. Salsa: A structured approach to large-scale anonymity. In Proceedings of CCS 2006, October 2006.

[170] Tsuen-Wan "Johnny" Ngan, Roger Dingledine, and Dan S. Wallach. Building Incentives into Tor. In Radu Sion, editor, Proceedings of Financial Cryptography (FC '10), January 2010.

[171] OpenDNS. `http://www.opendns.com`.

[172] Lasse Øverlier and Paul Syverson. Locating hidden servers. In Proceedings of the 2006 IEEE Symposium on Security and Privacy. IEEE CS, May 2006.

[173] p0f. `http://lcamtuf.coredump.cx/p0f.shtml`.

[174] Mike Perry. Securing the Tor network. Defcon 2007. `http://fscked.org/transient/SecuringTheTorNetwork.pdf`.

[175] Mike Perry. Exit Scanning. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/159-exit-scanning.txt`, February 2009.

[176] Mike Perry. TorFlow: Tor Network Analysis. In HotPets, August 2009.

[177] Larry Peterson, Steve Muir, Timothy Roscoe, and Aaron Klingaman. PlanetLab Architecture: An Overview. (PDN–06–031), May 2006.

[178] Larry L. Peterson and Bruce S. Davie. Computer Networks: A Systems Approach. Morgan Kaufmann, San Francisco, 2003.

[179] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`, February 2008.

[180] Michael Piatek, Tadayoshi Kohno, and Arvind Krishnamurthy. Challenges and directions for monitoring P2P file sharing networks – or – Why my printer received a DMCA takedown notice. In 3rd USENIX Workshop on Hot Topics in Security (HotSec), July 2008.

[181] Bogdan C. Popescu, Bruno Crispo, and Andrew S. Tanenbaum. Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System. In Proceedings of the 12th Cambridge Intl. Workshop on Security Protocols, 2004.

[182] Rahul Potharaju, Jeff Seibert, Sonia Fahmy, and Cristina Nita-Rotaru. Omnify: Investigating the Visibility and Effectiveness of Copyright Monitors. In Passive and Active Measurement Conference, 2011.

[183] Ryan Pries, Wei Yu, Xinwen Fu, and Wei Zhao. A new replay attack against anonymous communication networks. In ICC, pages 1578–1582. IEEE, 2008.

[184] Ryan Pries, Wei Yu, Steve Graham, and Xinwen Fu. On performance bottleneck of anonymous communication networks. In Parallel and Distributed Processing (IPDPS), 2008.

[185] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, pages 433–444, London, UK, 1992. Springer-Verlag.

[186] Sreeram Ramachandran. Web metrics: Size and number of resources. `https://code.google.com/speed/articles/web-metrics.html`.

[187] Joel Reardon. Improving Tor using a TCP-over-DTLS tunnel. Unversity of Waterloo Master's Thesis, October 2008.

[188] Joel Reardon and Ian Goldberg. Improving Tor using a TCP-over-DTLS tunnel. In Proceedings of the 18th USENIX Security Symposium, August 2009.

[189] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security, 1(1), June 1998.

[190] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC, USA, November 2002.

[191] Ron Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 21(2):120–126, February 1978.

[192] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329–350, November 2001.

[193] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In Proc. 16th USENIX Security Symposium, 2007.

[194] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. SIGCOMM Comput. Commun. Rev., 29:71–78, 1999.

[195] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Proceedings of Privacy Enhancing Technologies Workshop (PET 2002). Springer-Verlag, LNCS 2482, April 2002.

[196] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Proceedings of Information Hiding Workshop, 2002.

[197] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In Proceedings of ESORICS 2003, October 2003.

[198] Claude Shannon. A Mathematical Theory of Communication. In Bell System Technical Journal, volume 27, pages 379–656, 1948.

[199] Micah Sherr. Coordinate-based routing for high performance anonymity. University of Pennsylvania Doctoral Dissertation, 2009.

[200] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable link-based relay selection for anonymous routing. In PETS '09: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies, pages 73–93, Berlin, Heidelberg, 2009. Springer-Verlag.

[201] Micah Sherr, Boon Thau Loo, and Matt Blaze. Towards application-aware anonymous routing. In Proceedings of the Second Workshop on Hot Topics in Security (HotSec). USENIX, August 2007.

[202] Micah Sherr, Andrew Mao, William R. Marczak, Wenchao Zhou, Boon Thau Loo, and Matt Blaze. A3: An Extensible Platform for Application-Aware Anonymity. In 17th Annual Network and Distributed System Security Symposium, February 2010.

[203] Rob Sherwood, Bobby Bhattacharjee, and Ryan Braud. Misbehaving TCP receivers can cause Internet-wide congestion collapse. In Proceedings of the 12th ACM conference on Computer and communications security, CCS '05, pages 383–392, New York, NY, USA, 2005. ACM.

[204] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. p^5: A protocol for scalable anonymous communication. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, May 2002.

[205] Yoichi Shinoda, Ko Ikai, and Motomu Itoh. Vulnerabilities of passive Internet threat monitors. In Proceedings of the 14th conference on USENIX Security Symposium, Baltimore, MD, July 2005. USENIX Association.

[206] Vitaly Shmatikov and Ming-Hsui Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In Proceedings of ESORICS 2006, September 2006.

[207] Douglas C. Sicker, Paul Ohm, and Dirk Grunwald. Legal issues surrounding monitoring during network research. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, October 2007.

[208] Georgos Siganos, Josep M. Pujol, and Pablo Rodriguez. Monitoring the BitTorrent monitors: A bird's eye view. In PAM, pages 175–184, 2009.

[209] Atul Singh, Peter Druschel, and Dan S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In IEEE INFOCOM, 2006.

[210] Emin Gün Sirer, Sharad Goel, Mark Robson, and Doğan Engin. Eluding carnivores: File sharing with strong anonymity. In Proceedings of the 11th ACM SIGOPS European workshop. ACM, 2004.

[211] Robin Snader and Nikita Borisov. A tune-up for Tor: Improving security and performance in the Tor network. In Proceedings of the Network and Distributed Security Symposium (NDSS), February 2008.

[212] Robin Snader and Nikita Borisov. Eigenspeed: Secure peer-to-peer bandwidth evaluation. In Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS), 2009.

[213] Robin Snader and Nikita Borisov. Improving Security and Performance in the Tor Network through Tunable Path Selection. IEEE Transactions on Dependable and Secure Computing, 2010.

[214] Daniel J. Solove. 'I've Got Nothing to Hide' and Other Misunderstandings of Privacy. In San Diego Law Review, volume 44, 2007.

[215] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In 10th USENIX Security Symposium, 2001.

[216] Neil T. Spring, Maureen Chesire, Mark Berryman, Vivek Sahasranaman, Thomas E. Anderson, and Brian N. Bershad. Receiver based management of low bandwidth access links. In INFOCOM, pages 245–254, 2000.

[217] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In IEEE Symposium on Security and Privacy, 2002.

[218] L. Sweeney. *k*-Anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):557–570, 2002.

[219] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[220] Paul F. Syverson, Stuart G. Stubblebine, and David M. Goldschlag. Unlinkable serial transactions. In Proceedings of the First International Conference on Financial Cryptography, FC '97, pages 39–56, London, UK, 1997. Springer-Verlag.

[221] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. In Proceedings of ACM Conference on Computer and Communications Security, October 2010.

[222] Tor metrics portal: Data. `http://metrics.torproject.org/data.html#stats`.

[223] A. Tran, N. Hopper, and Y. Kim. Hashing it out in public: Common failure modes of DHT-based anonymity schemes. In ACM Workshop on Privacy in Electronic Society, 2009.

[224] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. PEREA: Towards practical TTP-free revocation in anonymous authentication. In Proceedings of the 15th ACM conference on Computer and communications security, CCS '08, pages 333–344, New York, NY, USA, 2008. ACM.

[225] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. ACM Trans. Inf. Syst. Secur., 13:39:1–39:33, December 2010.

[226] 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 Oper. Syst. Rev., 36:271–284, December 2002.

[227] Camilo Viecco. UDP-OR: A fair onion transport. Technical report presented at the First HotPETS, July 2008.

[228] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Tracking anonymous peer-to-peer VoIP calls on the Internet. In Proceedings of the ACM Conference on Computer and Communications Security, pages 81–91, November 2005.

[229] Zheng Wang and Jon Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. SIGCOMM Comput. Commun. Rev., 22:9–16, April 1992.

[230] Rolf Wendolsky, Dominik Herrmann, and Hannes Federrath. Performance comparison of low-latency anonymisation services from a user perspective. In Nikita Borisov and Philippe Golle, editors, Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, June 2007. Springer.

[231] Alan Westin. Privacy and Freedom. Atheneum, New York, 1967.

[232] Charles Wright, Lucas Ballard, Fabian Monrose, and Gerald Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In Proceedings of the 16th USENIX Security Symposium, 2007.

[233] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In Proceedings of the IEEE Symposium on Security and Privacy, 2008.

[234] C.V. Wright, F. Monrose, and G.M. Masson. On inferring application protocol behaviors in encrypted network traffic. Journal of Machine Learning Research, 2006.

[235] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In Proceedings of the Network and Distributed Security Symposium - NDSS '02. IEEE, February 2002.

[236] Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. ACM Trans. Inf. Syst. Secur., 7(4):489–522, 2004.

[237] Sebastian Zander and Steven J. Murdoch. An improved clock-skew measurement technique for revealing hidden services. In Proceedings of the 17th USENIX Security Symposium, San Jose, CA, US, July 2008.

[238] Lixia Zhang, Scott Shenker, and David D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In Proceedings of the conference on Communications architecture & protocols, SIGCOMM '91, pages 133–147, New York, NY, USA, 1991. ACM.

[239] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: Resilient anonymous routing. In Proc. of NSDI, Boston, MA, May 2005. ACM/USENIX.

Appendix A

Extended Circuit Compromise Results from Simulations

The full path compromise results discussed in Chapter 4.4 are presented here. Due to space constraints, the results begin on the next page.

Table A.1: Path compromise rate for each protocol's default port as the number of passive malicious routers increases

Malicious Routers	**Application-layer Protocol**										
Total (BW)	*FTP*	*SSH*	*Telnet*	*SMTP*†	*HTTP*	*POP3*	*HTTPS*	*Kazaa*†	*BitTorrent*†	*Gnut ella*†	*eDonkey*†
6 (60 MiB/s)	9.6%	9.5%	8.0%	21.8%	7.0%	7.0%	6.3%	21.7%	18.5%	20.7%	21.3%
16 (160 MiB/s)	30.4%	29.7%	28.0%	42.8%	24.2%	25.3%	24.1%	43.3%	40.7%	41.7%	43.2%
26 (260 MiB/s)	44.2%	42.7%	41.3%	54.2%	37.7%	39.1%	38.0%	54.4%	53.5%	54.8%	54.5%
36 (360 MiB/s)	54.4%	52.5%	49.7%	63.2%	47.2%	48.8%	46.9%	62.8%	62.1%	62.8%	62.7%
46 (460 MiB/s)	59.7%	58.6%	57.4%	69.2%	54.4%	55.2%	54.2%	67.9%	67.9%	67.8%	68.6%
56 (560 MiB/s)	66.0%	64.1%	61.9%	72.6%	60.0%	61.2%	59.4%	72.2%	71.1%	72.8%	73.2%
66 (660 MiB/s)	69.4%	69.1%	67.1%	75.8%	64.4%	64.5%	63.9%	76.1%	74.7%	75.5%	75.1%
76 (760 MiB/s)	72.5%	71.9%	69.8%	77.7%	68.4%	69.2%	68.0%	77.9%	77.3%	77.7%	77.8%
86 (860 MiB/s)	75.8%	74.5%	73.1%	80.7%	71.0%	71.8%	70.0%	80.6%	79.5%	80.3%	80.0%
96 (960 MiB/s)	77.7%	76.3%	74.4%	81.1%	73.1%	73.7%	72.4%	81.7%	81.2%	82.3%	82.0%
106 (1060 MiB/s)	78.5%	78.5%	76.6%	83.4%	75.5%	76.0%	74.9%	83.4%	82.4%	82.7%	83.2%

Table A.2: Tor's distribution of exit bandwidth by each protocol's default port

	FTP	*SSH*	*Telnet*	*SMTP*	*HTTP*	*POP3*	*HTTPS*	*Kazaa*	*BitTorrent*	*Gnutella*	*eDonkey*
Number of routers	184	197	500	13	625	552	629	19	23	20	19
Total bandwidth (in MiB/s)	65.4	73.7	90.1	1.4	116.9	107.9	122.7	2.8	9.1	3.4	2.8