Peering Through Censorship: Tackling Internet Censorship Using P2P Networking

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

By

Eliana V. Troper, A.B.

Washington, DC
April 11, 2023

Peering Through Censorship: Tackling Internet Censorship Using P2P Networking

Eliana V. Troper, A.B.

Thesis Advisor: Micah Sherr

Abstract

TODO: Optional for master's thesis but these are nice (max 350 words)

Index words: [TODO], Theses (academic)

Dedication

TODO: Optional

Acknowledgments

TODO

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The internet has become a staple of the modern world, used by nearly every person on Earth.[CITE] The internet is a collection of independently managed networks which can route arbitrary traffic between said networks. It is built upon numerous *protocols* which have been designed and iterated upon both academically and through practical usage over decades. Traffic on the internet travels from the edge of one network into another.[1] This high level view loses some key context: networks are run by entities who have their own desires and are free to operate their network as they see fit. In an ideal world, all of these entities would run the same algorithms and would move traffic in a content-agnostic fashion, but we do not live in an ideal world. Various confounding factors result in a web that looks drastically different in practice from idealized theory: malicious entities use the web to perform attacks, bugs cause unforeseen issues to arise, entities want to tightly control traffic and content in their network, among other complicating factors.

The complicated, real internet leads to issues that do not arise in pure networking theory, one of which is *censorship*. Censorship has become a loaded term in the current political context, and we do *not* mean the colloquial usage of censorship. Censorship, as we discuss it, refers to entities censoring web traffic (either through complete blocking or other manipulation) at nodes (frequently routers) in the network(s) that they control. Censorship may be done for a variety of reasons - most censors claim

[1] *See* 2.1 for more on networking and key networking concepts.

they are solely removing malicious traffic, however, an entity may use their own definition of malicious which differs from colloquial usage of the term. We see some entities that choose to block social media sites, news sources, or any site that they deem block-worthy. Censors have a variety of justifications for their actions such as economic protectionism, removing unsavory content, blocking dangerous content, and maintaining law and order. We see this and ask our key research question:

Key Research Question. *Can we circumvent censorship?*

This is a very broad question - to be able to address it in a concrete and scientific manner we must refine it and break out more specific and actionable research questions. To circumvent censorship, we must address what censorship is:

Research Question 1. *What do censorship systems look like?*

We explore previous work to answer this question. In the real world, we frequently see adversaries perform basic censorship actions, such as blocking IPs and DNS resolutions. We also see adversaries taking more extreme actions, such as performing fingerprinting attacks or other techniques involving deep packet inspection (DPI).[2] In theory, we see formulations that involve either attacking specific protocols [TODO e.g. xref Wang discussion] or employing a specific technique of attack applicable to several systems. [TODO: xref Salmon discussion] This allows us to refine our key question to focus on concrete systems:

Key Research Question. *Can we circumvent censorship systems? If so, how?*

As we discussed above, we note that some entity is performing censorship - censorship is not occurring spontaneously. Systems are not deployed magically, and do not arise without a concerted action undertaken by an entity. These systems are being

[2] *See* 3.2 for our exploration of real-world adversaries.

designed and implemented by some entity to achieve a specific purpose. This leads us to our next research question:

Research Question 2. *Can we define an adversary and model their censorship systems?*

Previous work has spent significant time defining adversaries in order to posit new methods of circumvention, and we define our adversary in a similar fashion. Our adversary has unlimited resources, constrained only by physics and algorithmic limitations.[3] In our model, we also assume there is at least 1 location *not* under the control of any censoring adversary.[XREF sphere of influence] Our theoretical adversary is informed by both real world observations and theories developed in prior work. In prior work, most adversarial models are highly focused - rather than looking at how an adversary may choose the full setup of a censorship suite, adversary modelling focuses on an adversary trying to tackle just one method of circumvention, and the broader picture is ignored. We formulate a model for evaluating an adversaries actions that looks at the *whole* of their censorship apparatus:[4]

$$[TODO : insert the final equation after cleaning notation] \tag{1.1}$$

This model of an adversary is a novel contribution and allows for rooting research in the field to both the real world and existing theory. With an adversary and model defined, we can further refine the key research question tackled in this paper:

Key Research Question. *Can we circumvent our adversary's censorship systems? If so, how?*

[3] *See* 2.2.2 for our full definition and further discussion.

[4] *See* 4 for more details and the derivation.

We cannot immediately answer this question - as we see in our model, an adversary has some utility function. We start by bounding our potential utility functions with one of the [edge possibilites]: Consider a utility function where an adversary does not care about blocking access to sites they deem legitimate (i.e., blocking an allowed site has a utility to the adversary of 0), but they feel strongly that no user should access a non-legitimate site (i.e. a user accessing a non-legitimate site has a utility of $-\infty$). This adversary would simply shut down *all* traffic. This would provide them with 0 utility, which is their maximal utility. In fact, in some instances censors have had real-world utility functions that result in actions like this, though those instances have been short lived.[CITE] However, in practice, we also see that no censor has continually provided 0 internet access - indicating that real world adversaries place a positive, non-zero value on allowing access to legitimate sites, even if they place a very negative value on some non-legitimate sites. While we can come up with infinite theoretical utility functions, we should focus on real and realistically possible utility functions, leading us to another research question:

Research Question 3. *What does a realistic utility function look like for a censor?*

We rely on looking at previous work focusing on real world adversaries to tackle this question, as well as novel contemporary observations. We find that most censors, in most circumstances, wish to censor users while providing access to acceptable sites. We also see that censors frequently fail to block some traffic that they could block using known methods. This brings us to the key concept of *collateral damage*.[TODO: xref] A censor can employ censorship techniques to stop some traffic that they wish to stop, [but many methods of censorship would result in shutting down access to some allowed traffic]. We see several real world adversaries each employ unique censorship

systems with varying degrees of collateral damage. [TODO: Add a bit more] As such, we refine our key research question:

Key Research Question. *Can we circumvent our adversaries censorship system, given a realistic adversarial utility function? If so, how?*

First, if we *only* look at *current* censorship apparatuses and real-world adversarial utility functions, we could conclude and say yes - based entirely on previous work. In practice, we see that the circumvention system Snowflake is currently working under all real-world adversaries. However, adversaries *do* change their utility functions, and new censorship systems arise. Adversaries are trying to develop new systems, and so we cannot assume a system will function as well as it does currently indefinitely.[5] We look ahead to a potential adversary either adopting a different utility function with a higher tolerance for collateral damage or the development of a new censorship system that is more optimal than currently deployed systems. This leads us to our next research question:

Research Question 4. *What might censorship look like in the near future?*

We review related work and synthesize some possibilities for actions an adversary may take in the future. We focus on currently successful circumvention techniques, and provide some methods of censorship an adversary may engage in to further restrict these systems using existing techniques that might increase collateral damage. We also explore censorship techniques, and we observe that deep packet inspection and bridge enumeration are the two broad techniques an adversary may use in a more

[5]We note that a circumvention system which is provably able to survive could arise, however existing tools either do not have a strong scientific proof (e.g. Snowflake [XREF]) or proofs are frequently constrained to such an extent that they work in theory, but neglect an attack outside of the scope of that theory (e.g. Obfs and the bridge distribution problem [XREF]).

advanced censorship system. We discuss some directions that these techniques may proceed in. [TODO: More + be more concrete] We use this to refine our overarching research question into an approachable question:

Research Question 5. *Can we circumvent our adversaries existing or potential censorship systems, given a realistic adversarial utility function? If so, how?*

We address this by combining several observations which result from the answers to our previous questions:

- A realistic censorship system should be based upon *real* censorship systems and *real* adversaries, with some further constraining beyond real-world adversaries allowed.
- Circumvention systems that thwart real censorship systems exist both in theory and the real world, but some functional systems *primarily* exist in theory.
  - There must be some limitation(s) to systems that only exist in theory.
  - The practicality of real-world systems is crucial.
- Circumvention systems function by combining a bootstrapping channel and a circumvention channel.
  - We see that successful bootstrapping is generally done using a high value channel that has reasonable latency, but may have low bandwidth.
  - We see that the circumvention channels that are used in the real world must have reasonable latency *and* bandwidth.

We build our system based upon the following principles derived from those observations:

- A novel circumvention system should be functional.
  - Functional means both not blocked, and having practical latency and bandwidth.
  - The system must function both when bootstrapping and relaying traffic.
- A novel circumvention system should improve upon an existing system in at least one of the following ways:
  - Increase collateral damage[6]
  - Scalability
  - Usability
- A novel circumvention system should not face the limitations that primarily theoretical systems face

We choose to build a system that iterates on the success of Snowflake. We do this by using the same type of channel (WebRTC) to relay our main traffic. We also use a bootstrapping channel in a similar fashion to Snowflake.[7] This is where the similarities end. We build our system as a standalone P2P system, which allows us to improve on Snowflake in several ways:

- Increase collateral damage
  - In existing systems, adversaries need to perform censorship solely at international gateways, but our system requires them to expand their censorship apparatus to cover intranational traffic as well. This drastically increases

[6] This also includes using a novel method of circumvention, though that is not required - a system may be iterative.

[7] We use a stand-in in our demo system.

collateral damage, as adversaries have not been observed performing notable on-the-wire censorship within their borders. Adding intranational censorship will take any collateral damage that would have occurred solely to international endpoints and expands that to intranational end points as well.[TODO: mention/cite that most traffic is intranational]

  - We use IPFS as a library for our channel, and any blocking of our system would result in blocking IPFS as well, blocking a protocol that is in use across the globe.

- Scalability

  - Snowflake is entirely volunteer run, and has encountered scalability issues in the real world in the last year as a result. There is some evidence that the volunteer base is largely saturated, and growing the network further is difficult. We address this by having every user of our network *also* act as a provider of our network. This means that each node in our network does not solely drain system bandwidth, instead they use some bandwidth and also provide some bandwidth.[8]

- Usability

  - We build our system as a standalone system, allowing users to use our system without the reduction of usability that comes with using Tor.[9]

  - Our system can be transferred and run as a singular HTML file in any relatively up to date web browser.

[8] A user may provide more bandwidth than they use, however, an exploration of the bandwidth provided vs the bandwidth used of users of our network would require numerous real users.

[9] We emphasize that users could potentially use our system to access Tor, but they are not required to, increasing usability in instances where a user does not need anonymity.

[END TODO: If any of these remain single bullet points, switch to 'ITEM: TEXT' instead of nested lists]

In practical terms, we build a system by using LibP2P and IPFS as libraries, and building a dynamically updating tree which contains metadata and pointers to hashes of files - which can then be retrieved over IPFS. We also outline some attacks a censor may take against IPFS, and we modify our usage of IPFS to mitigate these attacks. We build both Golang and Javascript libraries which are used to access and update the aforementioned tree, as well as to establish uplink channels. We build a benchmarking app on top of our network, as well as some demo apps [TODO: list final apps].

We then use our benchmarking app to demonstrate how our system works in a variety of conditions, based upon some of the potential actions a censor may take:

- No censorship
- Censoring fixed nodes
- Censoring fixed nodes *and* all traffic except WebRTC and email.

The second test is in line with how adversaries currently function in the real world, and the third test posits an adversary which has a utility function more extreme than existing adversaries. We note that we do not spend extensive time looking at DPI attacks on WebRTC - this is because these are generally fingerprinting attacks, and because we use the built in WebRTC from standard web browsers, there should not be any fingerprint distinct from standard usage of WebRTC.[10]

[TODO: Explore speed results]

[10] An adversary may perform *content* fingerprinting attacks, however, we relegate these attacks to future work, and note that adversaries do not appear to do these in practice in the realm of censorship, and they have false positive rates too high for usage. [CITE] [XREF false positives]

We then conclude and offer a discussion on ethics of our system and potential future work.[XREF]

[END TODO: Check paragraph indentation] [END TODO: Add xrefs throughout] [END TODO: Add cites throughout]

Chapter 2

Background

The term censorship has taken on a life of it's own in the modern social and political context. We are *not* exploring the general connotation of censorship, rather, we are exploring a specific type of censorship that existing in a networking (particularly inter-networking) context. We will explore some key concepts and present a short model of networking.

2.1 Networking concepts

We are going to consider the internet as it currently exists, rather than historical states of the internet. Some key concepts:

- Internet Protocol (IP) addresses: These are used to represent endpoints of a connection, and are used for routing. Within a network, other identifiers, such as MAC addresses, are used for routing, however, we are focused on routing between networks and will only consider IP based routing as we explore censorship circumvention. We will use IPv4 addresses throughout as sample addresses, though conceptually IPv6 addresses function similarly for the focuses of this paper.

- Domain Name System (DNS): DNS resolution is used to map from human-readable and memorable addresses that map to IP addresses (e.g. georgetown.edu resolved to 23.185.0.2 at the time of writing). [TODO: Explore the transparency of these, DNSSEC, etc]

- Router: Routers perform routing for networking, as well as other functions that act on network traffic. They may be the home of a NAT, a firewall, or censorship software. The controller of a router may access and manipulate any traffic which passes through it.

- Firewalls: These are software that may restrict traffic based on rules. These rules may be simple or complex. These have positive uses, such as securing a network from attacks.

- Network address translation (NAT): A NAT is used to map network addresses seen on one side of a router to those seen on the other side of a router. These are frequently used to house potentially thousands of devices behind a single internet facing IP address. Most humans use devices located behind a NAT.

2.2 What is censorship circumvention?

When we discuss the subfield of censorship circumvention when discussing networking and security, we are *not* discussing all forms of censorship that a web user might face - we are discussing censorship on-the-wire of active internet traffic, which usually consists of blocking access to specific services. We do not consider a service blocking certain conduct (e.g. a social network restricting hate speech) the censorship that we are circumventing. We aim to allow a use to connect to any service they would like (notwithstanding technical limitations) - not to allow a user to use any service

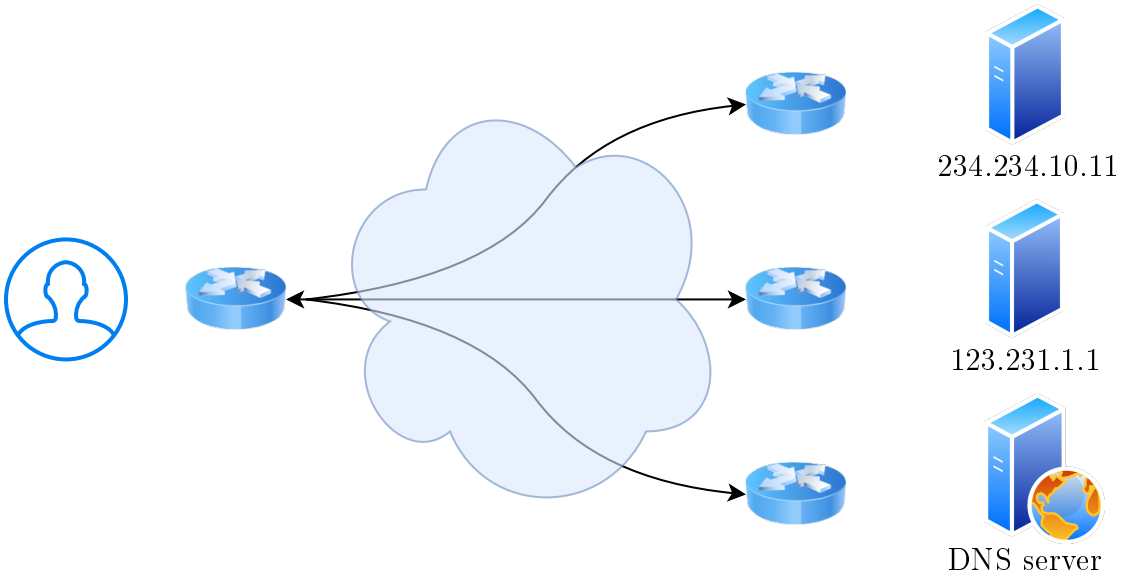

Figure 2.1: A model of networking, sufficient for exploring censorship circumvention [TODO: Add final arrows, some sample traffic]

how they would like. Throughout this paper, when we mention censorship, we mean on-the-wire censorship of reaching services.

2.2.1 Ethics

We recognize that some users run heinous services on the web - most of which are illegal in any existing jurisdiction. We believe that the best approach for ceasing access to these heinous services is best done through a legal system - IE, investigating and identifying the perpetrator(s), turning off the service directly, and prosecuting any perpetrator(s) of said acts. We note that most morally reprehensible acts that are frequently associated with anonymity and censorship circumvention systems involve actions that occur in the real world (e.g. exploitation of children, weapon/drug/human

trafficking all involve actions that occur in the real world either prior to distribution on the internet or triggered by actions on the internet).

The only notable class of crime which can occur without physical, real-world actions is piracy/IP violations. We believe that targeted legal actions (e.g. DMCA take-downs) can mitigate the issue, and that the cost of further mitigation through censorship would lead to large collateral damage of legal content and a chilling effect of legal speech and usage of the web, and as such the possibility of a circumvention system being used for piracy/IP violations is an acceptable risk given the ability to mitigate damage through the legal system.

2.2.2 Threat model and adversary

Censorship circumvention is an abstract concept, and approaching it in a scientific way requires properly defining and scoping the task. We have already defined the task, [TODO: xref] and must now scope the problem. We will scope our problem by creating a **threat model** of the censor we are aiming to thwart, termed **the adversary**.

An adversary has an **area of influence**. This is an area an adversary controls the network of, and can control (either directly or through compulsion) any service providers within. We assume an adversary may have an area of influence consisting of several nations, with at least one nation outside of the area of influence of *any* censor.[1]

Our adversary has effectively infinite resources, but is bound by the laws of physics and as such is bounded by the difficulty of computational problems (and the known solutions to these problems, e.g. even if P=NP, the solution is not known, and as such

[1] This does not mean that this nation does not have the technical capability to censor, but rather does not perform any censorship either directly or via threatening a start of censorship.

cannot be used). These resources may include labor, capital, technical expertise, and total control of the legal system.

We also assume that our adversary has access to the source code of any circumvention system available, and knowledge of the existence of our circumvention system. We discuss why security through obscurity cannot be used in an effective circumvention system in [TODO: xref].

We also assume that our adversary only takes actions **on-the-wire** and at **service providers** within their area of influence - effectively, the adversary isn't interested in any individual in particular and is performing a dragnet operation, rather than an operation targeting a specific entity. We can use a thought experiment to justify this: if an adversary had a person looking over your shoulder or using video surveillance, they could simply look at what you are doing and punish you directly, no need to expend the resources needed to censor the web.

We assume our adversary has some interest in the utility of the internet and certain technologies on the internet. This means two things: an adversary will not shut down the internet in it's entirety[2] and an adversary has some threshold of unacceptable collateral damage caused by potential censorship techniques.[3]

Finally, we note that any technique that subverts this adversary will also work when weakening the strength of these assumptions - e.g. if the adversary controls one nation instead of several, we have more area to run a circumvention system from; if

[2]Internet shutdowns are a trivial way to censor forbidden internet usage - at the expense of shutting down any non-forbidden usage. [TODO: Discuss prior shut downs in another section and xref]

[3]Similar to a full internet shutdown, if an adversary wanted to block all access to a specific HTTP site, they could block *all* HTTP traffic, at the expense of blocking every HTTP site. This piecemeal web censorship could even be thwarted with a circumvention system that simply proxies HTTP traffic outside of the area of influence within an unblocked protocol and connects to HTTP sites from outside the area of influence. This starts to show that any non-complete shutdown of internet traffic may have gaps for circumvention techniques. We discuss collateral damage further in [TODO: xref].

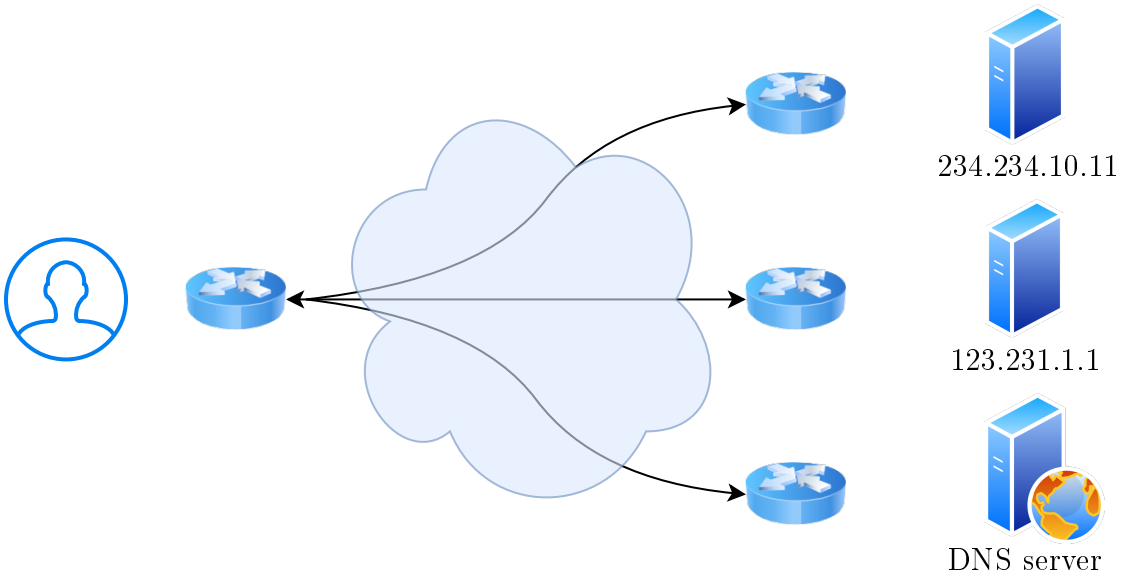

Figure 2.2: A model of networking, with our adversary model overlaid [TODO: Once networking diagram is finalized, add an adversary]

the adversary is only willing to take actions at service providers and not on-the-wire, anything that can thwart an adversary that takes both actions could trivially thwart the adversary that only performs actions at one. We also note that a system could detect the type of actions an adversary is taking and maximize performance based upon what subset of possible actions an adversary is taking - e.g. an adversary that is not touching services leaves an easy path for a circumvention system to use - those services themself.

[TODO: Basic model + adversary]

2.2.3 User model

[TODO]

[TODO: We should outline that we do not assume unlimited resources for our system. We should outline that we expect users to run the software on a phone or basic computer, and that we expect the system to be run by volunteers (who may or may not have significant resource)]

2.2.4 System provider model

[TODO]

2.2.5 History and basics

The web arose in a relatively haphazard fashion [TODO: discuss a bit more]. The web was not a place where adversaries engaged in censorship (if they paid any mind to the web whatsoever). Instead, as the web became more prominent and more global, individuals realized that they could be tracked, and that they desired anonymity. This led to the development of basic and advanced anonymity software, with proxying being a basic tool for this and Tor being the most prominent example of full fledged anonymity software.

As this occurred, the web continued to grow in prominence, and the concept of "Web 2.0" arose - regular users could post content to sites such as message boards and social media sites. People could organize quickly and effectively across larger distances and with people they had never met in the past. Governments took notice - and censorship began.[4]

When internet censorship arose, basic techniques were used. These include:

[4]The main justifications censors employ are maintaining law and order, preventing the spread of disinformation, preventing access to "unwholesome" content, or economic protectionism. [TODO: Would be cool to cite one or several instances of these justifications coming from the horses mouth]

- Blocking specific IPs: Traffic to or from various IP addresses could be dropped. This is simple and low cost.

- Blocking DNS resolution: DNS resolution traffic for specific requests could be dropped (or replaced). This is simple and low cost. [TODO: address DNS encryption and authentication]

- Legal mechanisms: If a service exists within the area an adversary controls, they could simply take down the service itself. Since we define the adversary to not control the whole world, we will ignore this form of censorship throughout this paper, as a restricted service could avoid this by relocating out of the sphere of influence of an adversary.

These techniques are sufficient to cut off access to services for users who are either unaware of a censor, unaware that circumvention is possible, users who do not care about the actions of a censor, or out of laziness. A user would have to actively use something that circumvents these simple blocks. The solution to avoid IP blocking and censoring DNS results is to proxy a users raw traffic outside of a censored area (either through a basic proxy, or something that provides other features, such as Tor). From the view of a censor, a user would not be accessing a restricted service, they would be accessing a proxy service.

Censorship circumvention arose as a by-product of tools made for purposes that were not focused on censorship circumvention. As primitive censorship began, people realized that anonymity software such as Tor or even a basic proxy could be used to access services that would otherwise be blocked. The entrances to these proxy services and Tor were accessed via well known IP addresses or DNS names.[5] Censors began

[5]The reason these were well known is that, if you want the public to access your service, they must know how to access these services, including the addresses of your service. These

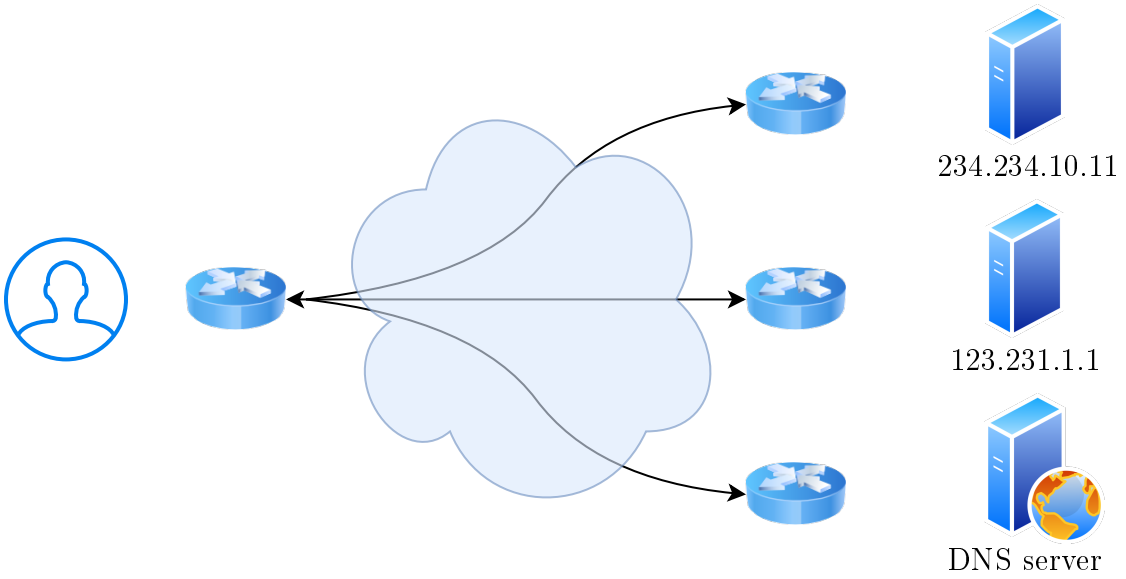

Figure 2.3: A model of networking, with our adversary performing basic censorship [TODO: Once networking diagram is finalized, show blocking]

being blocking access to these services, in addition to their original censorship targets. This set off a "censorship arms race."

At this point, those who used and built the anonymity software recognized that censorship circumvention would be crucial for maintaining access to these tools, and others realized that censorship circumvention in-and-of itself would be useful. This effectively led to the rise of a subdiscipline that draws heavily from networking, security, and even game theory, but also exists independently from those subdisciplines, with a variety of stakeholders including governments, NGOs, non-profits, independent programmers, academics, and others interested in developing censorship circumvention techniques. This also leads to a relatively unique feature of this discipline - some

must be made public to allow people to access your service. A censor has access to public information.

techniques were developed in academia, some techniques were developed in the real world, and some were developed in both, and ignoring the results from either thread of research and observation can lead to the development of flawed systems.

2.2.6 Current theory of censorship circumvention

[TODO: Maybe cut? Or just point to next section]

The bedrock of the theory underlying successful provision of access to proxies is collateral damage.

[TODO: Discuss collateral damage more]

Chapter 3

Related Work

Censorship circumvention has developed over several decades. Originating out of anonymity research, censorship resistance is a less difficult problem than full anonymity,[1] and has become a topic studied on it's own independent of anonymity.[2] Censorship circumvention generally looks at one of two problems: proxying traffic *out* of an adversaries sphere of influence, and providing access to those proxies to those within an adversaries sphere of influence. Additionally, observing real world censors are frequently observed to see how theory plays out in the real world, as well as if other censorship techniques are in place that have not been suggested in the literature, or were not significantly explored in the literature. We discuss real world observations and censorship techniques in 3.2.

3.1 Prior proxying theory and techniques

Most proxying techniques fall into one of three broad categories: looking like an unknown protocol, mimicking an existing protocol/service, or using an existing pro-

[1] Anonymity generally requires censorship resistance (or a non-censoring adversary) to bootstrap.

[2] Though anonymity and censorship are distinct problems and are often approached independently in research, there is notable interaction between the two fields - many censorship circumvention techniques target the pluggable transport (PT) spec for deployment of censorship systems. The original PT spec was made for Tor, an anonymity network and protocol,[5] though later iterations of the spec are more general and are in use by tools including Tor, Lantern, and Psiphon.[**?**] In practice, current functional real world techniques all use the PT spec - see **??** and 3.1.2 for details.

tocol/service as intended. Of these, mimicry was eliminated as a functional technique. Houmansadr et al established requirements that a mimicry system must meet to succeed - the imitated protocol must be mimicked in full, the imitation protocol must react properly to errors and network conditions, the imitation protocol must have traffic typical of the imitated protocol, and the imitation protocol must imitate artifacts from implementations of the imitated protocol properly.[9] They continued by showing that none of the cutting edge mimicry protocols of the time were able to fulfill these requirements.[3] The authors finally show that even with the initial attacks they find, hypothetical, improved transports that patch those would still fail as there are attacks that could not be prevented when imitating a proprietary protocol. The authors have a damning conclusion that has held until today: "unobservability by imitation is a fundamentally flawed approach, unlikely to ever succeed due to the daunting list of requirements that an imitator must satisfy. The failure of all proposed [imitation] circumvention systems to achiever unobservability confirms this conclusion."[9]

Beyond mimicry, most systems fall into either looking like an unknown protocol, or using an existing protocol as intended. We can also break down proxying techniques into low-bandwidth and high-bandwidth techniques. High-bandwidth, on it's own, is generally preferred to lower bandwidth, however, in censorship circumvention other trade-offs exist - most notably, the cost of censoring many low-bandwidth channels is higher than many high bandwidth channels. These channels frequently use an existing protocol/service as intended, which means that censoring these channels may require taking down a high value service, e.g. if a circumvention system uses email effectively,

[3]The authors explore SkypeMorph, StegoTorus, and CensorSpoofer - they quickly find passive attacks for these transports.[12][**?**][**?**][9]

it may only be possible to block by blocking email. We discuss several examples of both low and high bandwidth channels in the following sections.

Lower and higher bandwidth transports can be combined to work together and gain benefits of both. Most higher-bandwidth channels require conveying some less-fixed *out-of-band* information (e.g. an IP address) to use a higher bandwidth channel. Low bandwidth channels, generally being more costly to block, still require out-of-band information, but frequently this information is valid as long as the channel is functional. For example, an email-based channel will require a recipient email address, but this email address could be effectively constant due to the way email itself functions. As a result, an email based channel may effectively hard-code this information in the transport itself, allowing the channel to be used as long as you have the code and an adversary that does not block email. This highly resilient but low-bandwidth setup may be used to bootstrap higher bandwidth channels. An email channel may be used to convey an IP address of a point to point high bandwidth channel, and could be used for important messages or to get another channel if the received point-to-point channel is blocked. With this synergy between the low and high bandwidth channels, discussing and understanding both is crucial to a modern circumvention system.

[TODO: Discuss DPI, xref real world section]

[TODO: Discuss IP blocking]

[TODO: Discuss fingerprinting, including contemporary examples w/ snowflake]

[TODO: For each of these be sure to mention if they work IRL]

[TODO: Do mention some things like shadowsocks that are in use but are even further from academics]

[TODO: Discuss false positives and base-rate fallacy]

3.1.1 High-bandwidth channels

Several high-bandwidth channels have been created, and at least two are in active use in real world situations. We first discuss channels that attempt to look like an unknown protocol (UP), and then discuss ones that properly use protocols (PUP).

Unknown protocols

As various censorship apparatuses developed, DPI began being used. DPI caused issues for several protocols - as it was able to selectively censor unencrypted or partially encrypted data streams based on parameters that a censor sets. [TODO: xref DPI] obfs2 was developed by the Tor project to begin providing protection to access to the Tor network.[15] This protocol added an obfuscation layer, which encrypted the full stream and made the stream look like a stream of unknown bytes, in order to get past basic DPI. Obfs2 was vulnerable however, as a passive MitM [TODO: make sure this is defined earlier] could listen and decrypt the transport.[16] In addition to DPI, active probing arose as a technique that earlier circumvention techniques could be detected (and subsequently blocked) using. Active probing involves an adversary sending data of their choice to a server of their choice, and observing the response. They may potentially continue probing depending on the response. [TODO: Give an example] obfs2 was vulnerable to active probes as well.[4]

obfs3 was developed to patch issues with obfs2. The main issue it addressed was the ability to perform passive decryption of the flow. This was done by using a custom Diffie-Hellman handshake.[16] obfs3 was still vulnerable to active probing.[22] Scramblesuit was designed to address this flaw, as well as several other identified tells of obfs3.[22] Scramblesuit defined a strong adversary, similar to the one proposed in this

[4] And, indeed, it was actively probed by China.[**?**]

paper, and aimed to tackle active probing while also introducing basic traffic shaping via padding as well as inter-arrival times - introducing the concept of a polymorphic traffic shape for a protocol.[5] Scramblesuit introduced the addition of a shared secret distributed *out-of-band* to enable these improvements. From a usability and deployment standpoint, this was not a drastic departure from previous techniques: the older transports required basic info such as the bridge's IP address to be transmitted out of band, and this simply extended the amount of out-of-band information that needs to be transmitted by several characters.[6] Scramblesuit consists of two phases: a uniform Diffie-Hellman key exchange, and then a data exchange phases.

Scramblesuit performed some evaluation, however, they identified a key problem with evaluating circumvention techniques that affects all protocols that attempt to look like unknown protocols: evaluation is very difficult due to the lack of good datasets and effective experimental setups. The authors opted to show that the transport differs from the carried protocol (in this case, Tor).[22] Regardless, Scramblesuit became widely deployed.[7]

Scramblesuit had notable overhead from it's handshake phases. The UniformDH used modular exponentiation, which is an expensive operation.[1] Additionally, Scramblesuit was vulnerable to some MitM attacks if the attacker knew the shared secret.[1] obfs4 replaced the UniformDH with use of Tor's ntor handshake, with obfuscation

[5]Polymorphic shaping is a promising defense for circumventors. The basic concept is that if every server has a unique traffic shape, then even if an adversary is able to identify a single server effectively, they will be unable to use that information in blocking other servers. This concept has not been fully exploited by a published circumvention technique at the time of writing.

[6]Usability and deployment were specific considerations in the development of Scramblesuit - it was designed to be used by Tor and in the real world.[22]

[7]Notably, inter-arrival time manipulation was off by default and could optionally be turned on. This was because inter-arrival time manipulation introduced significant overhead. With hindsight, we have seen other issues with Scramblesuit and have not seen issues caused by inter-arrival time manipulation being disable. These other issues are discussed directly below.

done by Elligator 2.[1] obfs4 is still in use today, and is one of the main two circumvention techniques used for circumventing censorship in the Tor browser. [TODO: cite] obfs4 requires some out of band info to function - which is done through hard-coding, email, telegram, or domain fronting. [TODO: xref]

Even though obfs4 is still functional and in use, that does not mean it is not vulnerable to attack. Wang et al demonstrated that obfs4 is vulnerable to entropy based attacks: conventional and often used web protocols, even if encrypted, usually have some control messages unencrypted - e.g. TLS and SSH both have unencrypted portions of the header. This means that the entropy of these flows is lower than that of obfs4 flows - because obfs4 attempts to look-like-nothing, it ends up looking like a fully encrypted stream with no plaintext, resulting in very high entropy - but, no well known protocol does this. By attempting to look like a hypothetical unknown protocol, obfs4 may actually look unique.[21] Obfs4 can be reliably detected and blocked as a result, with relatively low computational costs. There are some drawbacks to this - most notably the base-rate fallacy, discussed in Section **??**. This paper was published in 2015, though for over 5 years this attack was not observed by any adversaries. In 2021, these attacks were observed on the live internet, discussed further in Section **??**.

Obfs4 also faces another problem: the bridge distribution problem, discussed further in Section **??**. Bridges, once known by an adversary, can be blocked at the IP level, and the software is difficult for basic users to spin up. Some of these issues are addressed by Snowflake in Section 3.1.2.

[TODO: Add in new paper re: obfs]

[TODO: Others?]

3.1.2 Properly using protocols

Several transports attempt to use protocols as intended. Snowflake is in real world usage, and several others have been developed in the literature.

Snowflake

Snowflake is a transport designed for usage by the Tor browser,[8] and is currently in use in the real world. Snowflake seeks to offer an alternative way of spinning up proxies compare to obfs and other transports that rely on proxies with fixed IP addresses, as well as using a different transport. Snowflake does this by taking advantage of how modern browsers are setup to ease proxy deployment and transport development. These concepts were introduced by Flashproxy and other related work. Flashproxy defined three key problems that this class of transport must address: how to create proxies, how to rendezvous, and how to actually hide when in use.[**?**] Flashproxy addresses proxy creation by allowing volunteers to embed code in their sites that turn any visitor into a proxy for censored users. These proxies are ephemeral, and are regular web users, changing up the cost and benefits a censor reaps if they were to be blocked. Rendezvous is handled by setting up a well-known connection broker. Hiding in use is not explored deeply, but is justified by simply using a frequently used web browser transport (WebSockets).

Snowflake took the concepts presented in Flashproxy and iterated on them and deployed them widely. Snowflake switched the transport in use from Websockets to WebRTC.[9] WebRTC was designed for P2P usage, and an ecosystem of utility proto-

[8] Though, it conforms to the PT spec spec, and as such could be used by other systems. See Section **??** for more on PTs.

[9] WebRTC is a commonly used standard across a variety of digital applications. Notably, it is supported in the web browser via Javascript. WebRTC is used for a variety of tasks, including video, audio, and other data, and is used by most video/audio chatting applications.

cols and standards has arisen around it, including tools to do NAT puncturing and establishing connections between users in various, differing networks, which provides significant utility to Snowflake compared to Websockets.[10] Snowflake iterated on the rendezvous by creating a broker. The broker is reached by using a high-reliability, low-bandwidth channel (domain fronting in practice, see Section 3.1.3 for further details). The broker is able to match volunteers and users to each other, in order to provide service. Snowflake then makes several usability improvements, setting up usage such that a client can fail over to a backup volunteer should their primary one fail, as well as maintaining a session across numerous volunteers using Turbotunnel.[11]

[TODO: Add a diagram which will be used later to start to show "modifications" from the snowflake model to get our system]

[TODO: Iran case study to highlight scalability]

[TODO: add stuff on bridge enumeration]

Decoy routing

[TODO: Tapdance] [TODO: Cirripede] [TODO: Telex]

[10] Snowflake uses Interactive Connectivity Establishment (ICE) to connect users behind NATs. ICE uses STUN and TURN relays as tools to ensure a connection can be established. STUN is commonly used and is available widely on the web, and is required for innocuous services to function as well. TURN services are less common, as they require significantly more infrastructure and are not generally permissionless like STUN.[6]

[11] Turbotunnel is a concept most completely implemented in Snowflake. It is a session/reliability layer that exists above an obfuscation layer.[7] Many transports exist above a reliability layer (e.g. obfs4 uses TCP), so adding an additional reliability layer adds notable overhead. In some cases, such as Snowflake, where the transports used are very unreliable, the benefits outweigh the costs. Turbotunnel also addresses several other issues - for example, a censor could close any long-running connections to disable obfs4.[7] Without any internal reliability layer, the connection must restart from scratch, effectively preventing usage of the transport, even if an adversary did not identify a specific bridge.

CacheBrowsing

[TODO: Discuss cachebrowser]

MassBrowser

Massbrowser is a P2P circumvention system. Massbrowser was designed to address flaws of existing systems - specifically, that systems that rely on limited, well known bridges (such as obfs4) are easily blocked, [TODO: xref] certain systems are expensive (such as using domain fronting for a full web session), many systems offer poor quality of service due to poor scalability, several systems force a user to add additional overhead to their web session by using another tool such as Tor,[12] and many systems (particularly decoy routing) are difficult to deploy.[14]

Massbrowser attempts to address these flaws by relying on volunteer proxies in both censored and uncensored regions to circumvent censorship. These proxies are supposed to be regular web users. A Massbrowser session begins by connecting to a central broker (via domain fronting) to match users with a proxy. These proxies then connect (after performing NAT punching) and use a modified version of obfs to communicate.

On top of this, because proxies may be in censored regions, MassBrowser maintains a complex whitelist on a per proxy basis - if a proxy is in a censored region, it cannot acquire certain content for the connected circumvention user.[13] This is costly - measurements must be performed to maintain this whitelist, and it assumes accessibility to CacheBrowser and Tor, and the maintinence of this whitelist does not benefit from economies of scale. On top of this, proxy providers may heavily tailor the loads they

[12] Many web users do not care about anonymity and would prefer simply to get around web blocks regardless of anonymity guarantees.

[13] MassBrowser simply tunnels through Tor in this case - creating a chicken vs egg problem.

wish to support, employing whitelisting, destination whitelisting (which varies from the basic whitelisting behavior previously mentioned), loads, and other options.[14] On top of this, MassBrowser uses custom software - either GUI software or a modified version of Firefox.

3.1.3 High-reliability channels

Most of the previously mentioned channels require either a shared secret (e.g. obfs4) or a rendezvous (e.g. snowflake). Distributing a secret or setting up a rendezvous cannot be punted indefinitely, and as such there must be a way to perform these actions, even in a censored region. High-reliability channels may be used for these actions. Though all high-reliability channels vary in numerous ways, they can generally be characterized as low-bandwidth, highly available, and sticky addresses. Each of these characteristics is important for differing reasons. Most of these channels are considered low-bandwidth - were they not, we could simply use the channel as the primary transport.[14] A channel must be highly available, or it could not be relied upon. Finally, sticky addresses (e.g. an email address, a social media handle) are important, as a more temporary address would leave us punting on distributing a secret or setting up a rendezvous channel - a sticky address allows one (or several) addresses to be hard-coded in a transports default configuration for the long run, allowing for usage without acquiring more info out-of-band.

Domain fronting

Domain fronting arose through exploiting the differences between the plaintext and encrypted portions of an HTTPS connection sequence. Though implementation varies

[14] And, this can still be done, see section **??** for some discussion of using one for a full connection.

slightly depending on the provider, at a high level domain fronting is accomplished by setting up a server behind the networks of a key piece of internet infrastructure,[15] and initializing DNS and server name indication (SNI) using a whitelisted domain from that same provider. After that, the HTTP host header is set to the proper destination (as this field is encrypted by TLS, the censor is unable to see it). Domain fronting is relatively simple to setup, and does not require direct participation from the provider (though, they can explicitly disallow it).[?] Domain fronting is used extensively, from full sessions,[8] to bootstrapping in Snowflake, or passing out of band secrets for Obfs4.

Domain fronting works well for small tasks, however, domain fronting costs a significant amount of money, proportional to bandwidth and machines used, limiting it's uses for more heavyweight tasks (particularly proxying full web sessions). Additionally, while it can be setup without cooperation from providers, they may choose to disable it. Google, Cloudflare, Amazon, and others have disabled it or take actions against users performing it. Currently, Azure (Microsoft) does not block meek, and Snowflake uses both fastly and Google's AMP cache to perform signalling.

Email

Email has become essential to modern life in a way few other protocols have. It functions as a good channel because it is generally encrypted using TLS, and it is widely available with third party clients allowed for most major servers, including companies such as Google and Yahoo. A censor would have to block a major email provider entirely if they wanted to prevent users using it as a channel. This has been exploited by several circumvention systems, including Sweet, Raven, and Tor's Bridgedb.

[15] Usually large providers of either direct services (e.g. Google) or CDNs (e.g. Cloudflare).

Sweet introduced the concept of channels with high availability guarantees. Sweet chooses to use email to build on the concepts presented in "The Parrot is Dead" by Housmandr et al.[9] Using the email protocol properly is trivial to do given the wide variety of available codebases and the numerous email servers in existence on the web. Sweet tunnels a full connection via email - however, it comes with increased latencies as a result. Sweet is explicitly vulnerable to traffic analysis, as most users do not tunnel a full web connection via email.[10]

Raven introduces the concept of behavioral realism to a circumvention channel. Raven offers an exemplar implementation written using email as the transport. This was done by using the authors actual emailing patterns and machine learning to craft a realistic sending pattern for the transport. As a result, Raven suffers from being drastically slower and higher latency than Sweet, to the extent it is usable for very little besides basic signalling.[16][20]

In reality, no censors appear to be doing behavioral analysis of web traffic - however, that does not mean they could not start doing so. In reality, it's likely that email could be used in a manner more strict than Sweet and less strict than Raven for small messages correlated with a real person accessing a system - IE, when a user is actually touching their web browser, shooting off and receiving a few email might not conform to a precisely real behavior, but it might be close enough that a censor could not detect it without significant false positives causing the blocking of regular email traffic.

In the real world, Tor uses email as a channel for distributing obfs4 bridges. This works to successfully provide users with bridges, though this runs into the bridge distribution problem limiting effectiveness, discussed in 3.2.2.

[16]Realistic behaviour is likely necessary for more transports beyond email to remain secure against a strong adversary in the long run, however, this is an area of open research.

Steganography

[TODO]

Other channels

Tor uses telegram to distribute obfs4 bridges.[**?**]

3.2 Real world observations

[TODO - a bunch of case studies - lox has several]

[TODO: Important to hit: recent OBFS entropy attack, Iran study that shows slowdown, China, DNS/IP censorship, DNSSEC failure]

3.2.1 Observing adversaries

[TODO - may cut this, but this will be drawing conclusions from our examples above, and comparing those adversaries to the adversary model we outlined, and will transition our discussion to OBFS/the bridge distribution problem]

Attacks on specific channels

3.2.2 The bridge distribution problem

[This subsection is being substatially revised]

We have discussed how channels work on the wire, and how specific protocols may be blocked, but we've deferred discussing the main problem with how many transports like obfs are frequently blocked: an adversary attempts to pose as a non-malicious user and requests access to an obfs bridge. They then attempt to probe the bridge, and if the connection succeeds, they block the bridge. This can be done quickly and easily by an adversary with resources, allowing for the blocking of numerous bridges if no

countermeasures are put in place. This problem arises because connection info must be disseminated to users who have no connection to those providing the circumvention techniques. There is no trust and effectively no communication between the parties. Circumvention must allow all users, even those who do not have technical contacts in areas where the internet is not censored, to access the web to succeed. We term this problem *the bridge distribution problem.*

Salmon, proposed by Douglas et al., formalized this notion of bridge distribution , and formulated one potential solution.[13] Nasr et al build on the *college admissions game* framework - a framework used in game theory to analyze and optimize solutions to certain problems of a certain form. The game is formulated such that there are effectively two entities - the bridge distributor, and the adversary. Nasr et al. show that the optimal distribution strategy is to provide clients with arbitrary proxies, but then track which clients request proxies again, and then new proxies are distributed based upon the *deferred acceptance algorithm.* This is done until all bridges are full or all clients have a bridge and are no longer requesting bridges.[13] The authors then show that an optimal censor will attempt to enumerate bridges, but will delay blocking a bridge to observe it for some time. The authors then demonstrate several bridge distributions strategies, but ultimately conclude that if new bridges do not come online often enough, then a censor will inevitably be able to block the full network of proxies.[13]

[TODO: Fix the above]

Lox, by Tulloch and Goldberg, builds on Salmon by introducing a social graph based bridge-distribution scheme. They also provide protection for the topology of the social network used.[19] [TODO - Salmon + that followup paper]

3.3 P2P networking

Peer to peer networking has existed since the early web, but has fallen into and out of favor over time. The web was originally envisioned as many small networks interlinking to form the full web, but as the internet developed, many services such as email, web hosting, server provision, etc. have become centralized by large companies. For example, if two users message on a site like Twitter, their messages are going to Twitter, being stored on Twitter, and passed to the recipient via Twitter, rather than directly (even if both users have no desire for Twitter to have access to the content). Generally, unless a user is deliberately using a specific P2P protocol, only select activities are augmented with P2P connections (usually voice/videochat and gaming). The most notable uses of P2P networks are torrents, some cryptocurrencies, and IPFS.

3.3.1 BitTorrent

BitTorrent originated in 2001.[17] It was designed to make provision of common, large files more efficient. "When a file is made available using HTTP, all upload cost is placed on the hosting machine. With BitTorrent, when multiple people are downloading the same file at the same time, they upload pieces of the file to each other. This redistributes the cost of upload to downloaders, (where it is often not even metered), this making hosting a file with a potentially unlimited number of downloaders affordable."[4] Publishers of files choose to make their file available as a torrent, where anyone with the proper software can download it.[18] At launch, users would make .torrent files available - these files contained basic info about a file - the size,

[17] We use torrent and BitTorrent interchangeably throughout this paper.

[18] Nowadays, even web browsers can use extensions to access torrents.[17]

name, a tracker,[19] and hashing info.[20][4] The original uploader must send out at least one copy of the file being shared (or it must already be distributed out of band), but after that, the file can be reconstructed from anyone sharing the torrent rather than the original provider. Users acquire peers in a random manner from trackers, ensuring robustness. BitTorrent divides files into small pieces of fixed size, which allows users to upload before a download is completed, and these pieces are known via their SHA1 hash, ensuring file integrity. Peers announce what files they have, resulting "in less than a tenth of a percent bandwidth overhead [while] reliably utiliz[ing] all available upload capacity.[4] BitTorrent generally prioritizes the "rarest" pieces of a file first, based upon which peers have which pieces.[21] BitTorrent reaches a relatively optimum end result through a mechanism known as choking: BitTorrent downloads from several peers - the fastest known peers - and optimistically probes for faster peers as well.[4] BitTorrent effectively uses "tit-for-tat" exchange mechanisms to allow for a relatively distributed system which generally reaches optimal or near-optimal outcomes.

[TODO: mention more swap algorithms, see ipfs paper]

[TODO: Torrents, Crypto, ???]

[TODO: MainlineDHT]

[19] "Trackers are responsible for helping downloaders find each other. They speak a very simple protocol layered on top of HTTP in which a downloader sends information about what file it's downloading, what port it's listening on, and similar information, and the tracker responds with a list of contact information for peers which are downloading the same file."[4]

[20] Since then, magnet links have become an option as well. These are specifically formatted links that provide the data needed to bootstrap a torrent download without pulling a file separate from the magnet link.

[21] Though, users can customize the ordering if they choose to do so. Selecting the rarest first both ensures a full file is available, and it maximizes the sharing speed of a torrent, as once the rarest piece is acquired by a second peer, that peer can share that piece further.

3.3.2 Crypto currencies

[TODO]

3.3.3 IPFS

IPFS is a P2P system where files are shared based on the content of the files, rather than URLs (uniform resource locators). URLs point to specific locations, which may host whatever those in charge of the location choose to serve, assuming you can route to the specific location to acquire a file.[22] Users of IPFS (either directly or an app that performs content acquisition under the hood using IPFS) request specific files, based upon the hash of the files. A key assumption of IPFS is that no nodes are privileged or trusted specially, which leads to several design decisions based upon prior work. At it's core, IPFS provides basic utilities for identities, networking, routing, file exchange, object acquisition, file system management, and basic naming capabilities.[3]

A node in IPFS can be thought of as having an identity.[23] This identity is very basic: it's simply a lone node ID, which is the hash of a nodes public key. These keys are built using primitives from the S/Kademlia DHT.[24] IPFS is designed to be transport agnostic - this is done using a *multiaddr*, which is an identifier similar

[22]Routing to blocked content is indeed the primary problem that we seek to solve in the censorship circumvention sphere.

[23]This identity can be refreshed as often as a user wants, though there is some benefit to using the same identity in the long run.

[24]S/Kademlia is an iteration of the Kademlia DHT. Kademlia DHT is designed to minimize the number of hops needed to learn the location of a desired address.[11] A large network of 10,000,000 nodes could be searched with 20 hops on average - queries are resolved in $\lceil log_2(n) \rceil$, where n is the number of nodes in the network.[3] Additionally, Kademlia has low overhead for control messages, and had prior large usage.[3] S/Kademlia builds on Kademlia to reduce Sybil attacks by requiring some proof of work work (signatures) and allows for lookups in a more adversarial network - "S/Kademlia achieves a [lookup success] rate of .85 even with an adversarial fraction as large as half of the nodes."[3] S/Kademlia assumes an adversary similar to one which we do - an adversary can manipulate packets, do observation, etc. They can also generate an arbitrary number of adversarial nodes, and might attempt an eclipse attack.[2] The proof of work for generating a node-id is based upon putting some

to a URL in terms of hierarchy, but providing networking information. For routing, IPFS must allow nodes to find other peers, and the location of specific content. IPFS draws on S/Kademlia and Coral DSHT's (distributed sloppy hash table) extensions to Kademlia for this - Coral DSHT notably reduces the overhead needed to find content by allowing for acquisition of just a single peer to acquire content, rather than needing to acquire every peer (hence, the *sloppy* in the name). Coral also organizes content into clusters, which allows for more focused searching for content and reducing the latency of the network.

Content acquisition builds on BitTorrent. (See Section 3.3.1 for more.) IPFS effectively uses the file sharing technology of torrenting, but instead of focusing on the pieces contained in a singular torrent, IPFS simply announces all pieces it has, regardless of the original file. This means that files which share content may be acquired from multiple users, even if they don't have the exact same file as the user looking for the content (e.g. a user may be seeking a music album, but they may acquire a song from that album from a user who only has that song). A notable change from BitTorrent, BitSwap (the name of the content swapping protocol of IPFS) also encourages acquisition of generally rare files that users want, as by decoupling desired files from a torrent, users are incentivized to provide content that providers of the content they want are in search of, which results in a spread of rarer, more desired files in order to receive better download speeds.[3]

IPFS builds a Merkle DAG on top of the DHT and content swapping technology. This DAG links via hashes of content - a generalization of the Git data structure.[3] Using this dag of hashes, IPFS has *unique* identification of any objects, it is tamper resistant, and it allows for easy deduplication.[3] Content can be acquired using a

basic constraints on node-ids (e.g. certain bits must be 0), and therefore generating a valid node-id is doable for users, but is a configurable variable to stop an adversary.[2]

very basic path format: /ipfs/<HASH>/<PATH TO OBJECT>. This format allows for easy integration into existing file systems, and for minor changes to allow for web browser usage (set up as: ipfs://<HASH>/<PATH TO OBJECT>). Users can add any object to the DHT, and may choose to *pin* an object - marking it as an object that should be kept for the long run. Files are represented similar to Git files, with some minor additions added to improve performance. These changes include fast size lookups, deduplication of large files, and organizing commits into a tree.[3] A Git object can actually be converted to IPFS without losing any info, but not vice-versa.

So far, all of IPFS has been presented as immutable objects with a mutable networking structure, however, many applications require mutable objects to function properly, but IPFS would not support this, and would require extensive out-of-band communication in order to run anything using IPFS. A basic, mutable file addressing schema was introduces to provide every node-key to publish a single mutable file. This is done by allowing nodes to insert an arbitrary IPFS hash as /ipns/<Node ID>. The only mutable portion of IPFS is the routing system, which is used to publish this by adding the data as a singular, small metadata value.[3]

IPFS has been in use for several years, and has been used to run some services and provide some data on the web. [TODO: any censorship circumvention uses?] It has been implemented in both Golang and Javascript (with other languages in various states of completion), and is available in some prominent web browsers (Brave and Opera).[18] The current version of IPFS uses a CID (content identifier) to point to objects - this is a variant of the hashes used to point to object. These CIDs include a multibase prefix - a prefix used to indicate the base the encoding used, a version identifier, a codec identifier (e.g. protobuf, JSON, etc), and a *multihash* - a hash that indicates the hash algorithm used, and the length of the hash.[18] IPFS uses a multiaddress to provide network addresses of

peers - a flexible structure that allows conveying a full network address by first outlining the peer, their network protocol (and any data needed to use), the transport layer, and any additional transports that may reside on top of that.[18] A couple sample multiaddresses are /ip4/<ip address>/tcp/<PORT>/ipfs/<Node ID> and /dns4/site.com/tcp/<PORT>/tls/ipfs/<Node ID>.

[TODO: Add stuff about current transports]

[TODO: Add section on uploading/retrieval]

[TODO: Mention gateways]

[TODO: SFS - self certified filesystems]

[TODO: Add chart from measurement paper]

Chapter 4

Theory

We begin with a basic model - an adversary will engage in the censorship of service x if $B_x - C_x > 0$, where B_x is the average benefit per user of engaging in censorship of service x and C_x is the average cost of engaging in censorship of service x. We will refine this formula by switching to a focus on censorship techniques - $(b_{x,1} + b_{x,2} + ... + b_{x,n}) - C_x > 0$, where $b_{x,n}$ is the average benefit per user of censoring service n using method x. A censor will engage in any censorship method where the average benefit reaped by censoring *any* desired services using that method is greater than the cost. Also note that these benefits and costs are based on how they are *perceived* by the censor, not by an objective, bird-eye view of good and bad, and they may account for a variety of benefits/cost, including direct financial costs, but also including less tangible value such as "stability".

We now note that each method may only stop a fraction of users accessing a service using a method - so a more realistic formulation would be $(u_{x,1} * b_{x,1} + u_{x,2} * b_{x,2} + ... + u_{x,n} * b_{x,n}) - C_x > 0$, where $u_{x,n}$ is the fraction of users accessing service n that are blocked from accessing service n using method x.

We will now break down cost further. $C_x = f_x + m_x$ - where f_x is the fixed cost per user of setting up method x and m_x is the marginal cost of actively engaging in x per user. We will drop f_x from further consideration (IE, $C_x = m_x$) - the cost of setting up a method is a one time cost, and to an adversary with unlimited resources, expending a one time cost averages to 0 over the long run. We will now explore the

marginal cost of x. Censorship tools may slow down "good" traffic, and they may erroneously block access to good services. Therefore, $C_x = s_x * i_x + (u_{x,1} * c_{x,1} + u_{x,2} * c_{x,2} + ... + u_{x,n} * c_{x,n})$, where $c_{x,n}$ is the average cost of blocking service n per user, $u_{x,n}$ is the same as above, s_x is the average fraction of the total value of unblocked traffic retained (due to the reduction in speed caused by enabling method x),[1], and i_x is the value per user of access to all unblocked sites. Combining the above, we see $(u_{x,1} * b_{x,1} + u_{x,1} * b_{x,2} + ... + u_{x,n} * b_{x,n}) - s_x * i_x + (c_{x,1} + c_{x,2} + ... + c_{x,n}) > 0$, or, more succinctly:

$$-s_x * i_x + \sum_{s=1}^{n} u_{x,s} * v_{x,s} > 0 \tag{4.1}$$

where $v_{x,s} = b_{x,s} - c_{x,s}$.

We could conclude here if a censor engaged in censorship solely using a singular method of censorship - however, censors engage in multiple methods of censorship, and the benefit of successfully blocking a site is not distributed multiple times if it's blocked by multiple methods, the benefit is only achieved based on new users blocked by a method, or, in short, a singular censorship system x will be implemented if:

$$-s_x * i_x + \sum_{s=1}^{n} \Delta u_{x,s} * v_{x,s} > 0 \tag{4.2}$$

Note that all variables are the same, except i_x is now the value per user of access to all unblocked sites prior to enabling censorship system x and $\Delta u_{x,s}$ is the change in the fraction of users who can access service s after enabling x.

We have now established an equation used to determine if a censor will enable an individual system, taking existing systems (or lack thereof) as a given. This is not reflective of reality, where a rational censor will evaluate *all* of their active and potential circumvention systems in conjunction.

[1]Note that an adversary may engage in multiple stages of filtering, so some subsets of unblocked traffic may experience no slowdown, while some might experience more. For the purposes of our analysis, we can use the average across all allowed traffic.

The valuation of a system consisting of censorship systems 1 through m is as follows:

$$\sum_{x=1}^{m}\sum_{s=1}^{n}\Delta u_{x,s} * v_{x,s} - i_x * \prod_{x=1}^{m} s_x \tag{4.3}$$

All variables are the same as previously, except i_x is the value per user of access to all unblocked sites again, s_x is the average fraction of the total value of unblocked traffic retained after enabling all systems y with $0 \leq y < x$, and $\Delta u_{x,s}$ change in fraction of all users blocked using method x that were not already blocked after enabling all systems y with $0 \leq y < x$. We will define $x = 0$ such that this "censorship" system is the null (non-existent) censorship system, and therefore $s_0 = 1$, and $u_{0,s} = 0$.

We assert that a censor is picking from a number of censorship systems that is small enough such that the censor can evaluate the above equation for every possible set of censorship systems in every permutation such that each set only has one of each censorship system in a reasonable amount of time. [TODO: Maybe justify a bit more?]

A rational censor will enable the set of censorship systems such that the above equation is maximized given every potential set of circumvention systems.

[TODO: Dig into the base rate fallacy, and expound on that with an equation based expansion that fits in the above model]

[TODO: Walk through one example using this]

Chapter 5

System design

We saw that several circumvention systems work given the constraints they were designed for, however, these constraints are either too loose and have known real-world failures under our adversary model,[1] the systems are difficult to deploy, or they have scalability issues.

Additionally, usability is also a crucial issue [TODO: COVER THIS IN RELATED WORK], and we identify two crucial issues where at least one of these affects every previously discussed system:

- Low bandwidth/poor performance: Users want a good experience on the web. Some of these methods are completely unusable for proxying the entirety of a web session. Additionally, many of these items are tied to usage of Tor. While we support usage of Tor, it slows down web browsing even without circumvention tools in use, and usage can [result in broken behavior on some websites/some sites block Tor TODO: explore and cite either in fn or related work].

- Barriers to entry: Every system we've explored that are usable by a layperson require usage in conjunction with Tor - and a layperson must be aware Tor

[1] The bridge distribution problem being the most notable issue for several otherwise effective systems.

exists, not be biased against it,[2] know how to get it, and not be censored when getting it.[3]

Below, we propose a system with a clear theoretical backing and achievable goals based on both theory and real-world observations. [TODO: xref] We also address these other key issues.

5.1 Theory and goals

First, we want to define system goal and highlight some non-goals that are close to censorship circumvention, but are not directly used for them.

We aim to thwart the adversary we outline in [TODO: xref]. We will rely heavily on the concept of collateral damage. [TODO: xref]

We are *not* aiming to provide anonymity.[4] We feel the need to emphasize this, as most current systems that are functional in the real world are used purely as a PT by Tor.[5]

We will break down our goals into two sets: accomplishing censorship circumvention, and providing a more usable system for a layperson.

[2] Some individuals believe that Tor is only used by criminals, and while we do not believe this is true, dispelling misinformation at a large scale is a herculean task that we will assume is not possible in the short and medium term [TODO: Citing this would be good]

[3] Acquiring Tor is frequently censored for users, which means that a user *must* acquire it prior to being under a censorship regime, or must have some out of band channel for acquisition. [TODO: Worth citing, may be worth bumping up importance]

[4] Perhaps our system could be used to access an anonymity system, but we do not want to guarantee anonymity or guarantee that the system will not leak information that may break anonymity, though we are not deliberately trying to break anonymity, we are also not evaluating our system for it at this point.

[5] Our system could be used as a PT, as we do have a Golang implementation, however, evaluating this system for usage as a PT is reserved for future work. [TODO: XREF PTs]

Table 5.1: Comparison of existing systems.

[TODO: Make table take whole page sideways, make in latex when finalized]

		Approach			Bootstrapping			Limitations				Active usage	Source	Notes [TODO: Make footnote]
		Look-like-nothing	Maximize collateral damage	Refraction Networking	Domain fronting	Email	Out of band	Bridge distribution problem	Scalability	Deployment Difficulty	Low bandwidth			
System	obfs4	X (Custom)			X	X	X	X				PT (Tor)	[TODO]	
	Snowflake		P2P-like (WebRTC)		X				X			PT (Tor)	[TODO]	
	MassBrowser		P2P [TODO]					[TODO]	[TODO]	[TODO]	[TODO]		[TODO]	
	[Refraction Networking]			X [TODO]	[TODO]	[TODO]	[TODO]		X	X			[TODO]	
	Meek		Domain fronting (HTTPS)		X				X		*	PT (Tor)	[TODO]	Poor scalability causes low bandwidth
	Raven		Realistic behavior (Email)			X					X		[TODO]	
	[TODO: More]													

5.1.1 Censorship circumvention

In Section 3.1 we discussed a variety of censorship circumvention systems. Observing how and why each of these work, and the drawbacks associated with each of them will allow us to formulate goals we wish to achieve, working techniques we may use, and a theory for our systems function. We present a short summary of previous systems in [TODO: xref table], comparing and contrasting their successful techniques and transports, and their individual drawbacks.

We see that existing systems are limited by at least one of the following:

- The bridge distribution problem (*see* Section 3.2.2)
- Scalability
- Deployment difficulty
- Low bandwidth

Another key observation is that *collateral damage* is the bedrock of the theory behind *all* circumvention systems - as such, any system must be based around maximizing this - the ideal system will make the only feasible way to shut down the system effectively shutting down a large portion of internet traffic.

[TODO: Discuss false positives + the overhead of blocking traffic]

[TODO: More]

We first note that systems with deployment difficulty (refraction networking) as their primary limitation rely on either a single or handful of *large*, *continuously cooperative* entities (all very large tech corporations). This makes the circumvention method brittle - IE, a change of heart of very few entities could negate the effectiveness of this method. We believe that not only is this a possibility, but it is a *likely* possibility - we note that adversaries may be as large as a collective of nations in our model - and given the profit maximization of corporations, these entities will want to perform business in the sphere of influence of our adversary. The adversary could pressure the entity to shut down the system in order to perform business within, with, and with citizens of the sphere of influence of the adversary. Economic theory implies that this will occur in the long run. [TODO: shore this up with both more clear wording and with some cites to examples] Due to this limitation we will not be building on refraction networking for our system.

[TODO: We will not use systems with the bridge distribution problem as their flaw, xref salmon section]

We are now effectively left with techniques with scalability and low bandwidth as our primary limitations to consider. Systems with low bandwidth are either directly limited through maximum channel bandwidth,[6] or through financial cost. Because of

[6]Even without system overheads - e.g. Raven reduces bandwidth from an already low level to an even *lower* level.

our system and user model, where the system is run with potentially limited resources and users are not charged for usage, we cannot require users to pay to access the system or expect large financial expenditures from our system provider. This means we may not use these techniques for the majority of our systems operation (though we may use them for low-bandwidth usage, such as bootstrapping).

We are left with systems whose primary limitation is scalability. We see that Snowflake is the only remaining existing system. Snowflake was developed primarily in the real world - and we know it works in the real world - but to build off of this success we want to formalize that success and that system design.

We will return to our model of censorship, reprinted below [XREF]:

$$\sum_{x=1}^{m}\sum_{s=1}^{n}\Delta u_{x,s} * v_{x,s} - i_x * \prod_{x=1}^{m} s_x \tag{5.1}$$

We will now refine our adversary definition further. We will assume our adversary has access to any censorship system proposed in the literature *or* in use (either actively, historically, or proposed for use) in the real world. We will break down our adversaries into two classes:

- Has a high tolerance for a low $\prod_{x=1}^{m} s_x$.
- Has a low tolerance for a low $\prod_{x=1}^{m} s_x$.

We note that both adversaries have been observed in the real world, with [XREF Iran] slowing down *all* traffic notably, and all other adversaries slowing down only limited subsets of traffic at most (e.g. [XREF Russia with Twitter]).

We will now assume that all adversaries are rational. This allows us to conclude that an adversary has optimized their censorship apparatus by evaluating all possible combinations of censorship systems. [TODO: dive a bit more into this in the earlier section, and discuss game theory a bit more] We will let $V_existing$ =

$\sum_{x=1}^{m_e existing} \sum_{s=1}^{n} \Delta u_{x,s} * v_{x,s} - i_x * \prod_{x=1}^{m_e existing} s_x$ for the optimized set S of censorship apparatuses of size $m_e existing$.

We propose a theoretical censorship system, s_{new}. We will consider $S \cup s_{new}$. Our censor will reevaluate the utility of their system using every permutation of this larger set. This means that they will evaluate adding s_{new} to their system, and keeping all existing systems or removing some of them and adding this new one. We will let this new set be S_{new}. We have two potential cases:

- S_{new} does not include s_{new} - IE, $S_{new} = S$. In this case, the censor is not engaging any new censorship systems.

- S_{new} includes s_{new} - IE, they engaged the new system. In this case we will assume that they will never disable an old system. [TODO: Can we prove that they will never disable an old system?] This means we can evaluate a new system using a simplified formula from earlier:

$$-s_x * i_x + \sum_{s=1}^{n} \Delta u_{x,s} * v_{x,s} > 0 \tag{5.2}$$

  [TODO: XXX: FIX S REUSE IN FORMULAS]

[TODO: APPLY THE ECONOMIC MODEL OF CENSORSHIP CIRCUMVENTION TO SNOWFLAKE]

We know censors do not censor Snowflake. We can assert the

[We have a value, though abstract, that we can relatively compare to - so long as adjustments to our system have a net 0 or an increase in censorship costs

[TODO: MAKE AN ASSUMPTION - OUR THEORETICAL ADVERSARY CAN BE THOUGHT OF AS HAVING THE CAPABILITIES OF ALL REAL WORLD ADVERSARIES + ACADEMIC ATTACKS - BUT THEY HAVE A UTILITY MODEL BASED ON REAL WORLD COUNTRIES - EXPLORE AN

ADVERSARY A LA IRAN WILLING TO SLOW DOWN GENERAL TRAFFIC AND BLOCK A LOT, AND ONE THAT IS MORE SIMILAR TO CHINA - VERY WILLING TO BLOCK, BUT NOT WILLING TO SLOW DOWN TRAFFIC - USE FALSE POSITIVE RATES OF REAL COUNTRIES TO GET A THEORETICAL ADVERSARIES FALSE POSITIVE FUNCTION - JUSTIFY WITH THE RELATIVE LACK OF INTERNET SHUTDOWNS AND UNWILLINGNESS TO USE WANG ET AL FOR OBFS OR ANYTHING WITH SNOWFLAKE]

5.1.2 Usability

We also note that existing systems require a user to download and use specialized software, and that systems usually require using anonymity software such as Tor to reap the benefits of censorship circuvmention. While anonymity is a useful [thing] to have in some circumstances on the web, getting the drawbacks of common anonymity software [TODO: outline] is not ideal in all use cases, particularly getting users to adopt software.

[TODO: More - draw from usability studies]

5.2 Implementation

We outlined certain goals we want to achieve based upon successes and limitations. Notably:

- Building off of a succesful technique, increase collateral damage. This can be done by increasing the number of services that would be blocked should an adversary take action, or increasing the slowdown to other traffic.

- Increasing usability, particularly by lowering the barrier to entry for users, enabling easier acquision of the system, and increasing bandwidth for circumvention focused usage.

As we noted above [xref], Snowflake seems to be the most promising style of circumvention system to build off of. This leaves us two approaches: stick with WebRTC or build a system with another available transport that models the topology of Snowflake.

[TODO: Find a web study about most used protocols, spend some time discussing why there are not many viable alternatives to WebRTC]

We choose to stick with WebRTC as our primary transport. We do this because, as we discussed previously, no adversary present in the real world has a utility function that allows them to censor Snowflake. [TODO: Spend some time discussing fingerprinting and why that isn't a long term issue] From this, we can see that the utility of WebRTC must outweigh the drawbacks of restricting it further. As such, to build on the success of Snowflake, we choose to instead change the topology

[TODO: Finish this section]

Chapter 6

System evaluation

Having built our system, we will perform an evaluation of our system. Demonstrating that the system as designed is viable [shows that a system that improves on Snowflake is not just theoretical, but also practicle].

[More]

6.1 Performance

Demonstrating performance is a key measure to show our system works and that it is possible to bootstrap with limited resources.

We first show that our system *under no censorship* functions [TODO: good description of the final results]

[TODO: Describe how the benchmarking app works]

[TODO: a table, likely a full page sideways, will exist here outlining the results of our benchmarking app]

[TODO: Walk through results]

6.2 Performance under censorship

Having demonstrated basic functionality, we will show that our system not only works, but works [TODO: well (verify)] under some of the more extreme censorship that could occur. To do this, we take steps a censor would likely take to attack this system.

6.2.1 Blocking bootstrapping nodes

The easiest vector of attack a censor could take to attempt to block our system is to block the IPs and DNS records of the bootstrapping nodes used for the system. We discuss this method of attack in the real world more in [TODO: xref bootstrapping availability AND system design AND related work], as well as our mitigations for the attack.

We perform the same benchmarking test without using any default bootstrapping nodes. We do this by using a stand-in for a high value channel, which connects to the system and performs bootstrapping *without* using the default nodes. [POTENTIAL EXPANSION: Don't just do a stand-in, actually do it using raven] [POTENTIAL EXPANSION: FIREWALL THE NODES as extra proof] We perform the same tests as above.

[TODO: a table, likely a full page sideways, will exist here outlining the results of our benchmarking app]

[TODO: Walk through results]

6.2.2 Blocking *all* transports except WebRTC and a high value channel

Instead of iterating through all potential censorship apparatuses that are more powerful than simply blocking the bootstrapping nodes, we outline a *very* strong adversary willing to engage in massive censorship and show that our system still survives.

Adversary Model 1. *An adversary is dealing with serious turmoil within their borders. They decide that all non-essential web traffic is to be shutdown. They designate email and video chats as essential for their own communication, and as such allow*

access to email and WebRTC. They block all other traffic, excluding some auxilary traffic that are used related to those (e.g. DNS, STUN, TURN).

Note that this means that our adversary is blocking bootstrapping nodes as well.

We perform the same benchmarking test as before, without using any default bootstrapping nodes *and* without using any transport besides our stand-in for a high value channel and WebRTC. [POTENTIAL EXPANSION: Don't just do a stand-in, actually do it using raven] [POTENTIAL EXPANSION: FIREWALL THIS - will be harder than before since will need to whitelist DNS/STUN/TURN/ETC]

[TODO: a table, likely a full page sideways, will exist here outlining the results of our benchmarking app]

[TODO: Walk through results]

6.2.3 A more advanced adversary?

We see that no real-world adversary has a utility function which allows them to block Snowflake. This does not mean that an adversary may not be able to build a more advanced censorship system. We believe that the changes that we have made to the model setup in Snowflake drastically increases the costs [TODO: Either outline or xref] to censor our system over Snowflake, and as such an adversary would not just have to build a system that could censor Snowflake with acceptable collateral damage, but they would also have to build one *significantly better* [TODO: Describe more precisely] than that to stop our system. That may happen in the future, but [discuss current literature on traffic analysis + false positives rate or xref those].

6.3 Bootstrapping availability

[TODO: Section using RIPE probes]

[TODO: Tie into getting the network bootstrapped/security through obscurity]

[TODO: Cite raven email study]

Chapter 7

Conclusions

TODO

7.1 Ethics

In the course of this research, we developped and tested a system performing legal activity without a real-world adversary attempting to censor our traffic, using our own resources or public resources in an acceptable fashion. We *emulated* an adversary by taking the most logical steps to censoring our system while mitigating collateral damage. As such, performing this research was ethical.

Further exploration of deploying a system like this would be necessary. We previously discussed that adversaries may be able to deploy legal apparatuses to censor services within a country, and while we do not expect an adversary to have the ability to do this against most users, we cannot exclude the possibility that an adversary may use legal tools against *some* users of our system.

Our system is distinct from any deployed system: users of our system are also providers of our system. An adversary may not distinguish between users and providers of a system, but it is possible that they might, and if they did, they would likely punish "providers" more harshly than simple users.

We see two potential paths towards potentially resolving this issue that would need to be considered before deployment to users in a censored area:

- Plausibly deniability: If the system performs these actions without a users input, without informing a user, this provides them with plausible deniability. This would allow users to plead ignorance should an adversary attempt to punish them. Additionally, having a "killer app" that is useful to users *regardless* of the ability to use circumvention would further enhance this path: if users use this system for reasons besides censorship circumvention, then users would be able to say they didn't know or care that the system provided censorship circumvention - they thought they were just using another app, not a circumvention system.

- Clear disclosure to users: Clearly disclosing to users how the system works means that users, who know their circumstances better than we could, could make the ideal decision for themselves to use the system. This may turn some users off of using the system, but it pushes the full decision to use the system onto users themselves, and would eliminate any uncertainty for users about what they were doing, and that they may be at risk doing it. The downside, of course, is that an adversary who catches a user using the system could take action against a user and the user would be unable to argue that they *unintentionally* were providing a circumvention service.

Beyond these two paths, we could also provide users with the ability to disable provision of services to other users. While this path could be explored further, the notable benefits (both scalability and increased collateral damage) provided by this system over Snowflake partially or fully dissapear if users in a censored region opt out of being providers.

While we believe that one of these paths (clear disclosure) would allow for an ethical deployment of our system, a deeper exploration of the alternative would be worthwhile to evaluate.

7.2 Future work

[Evaluation of the average "value" of a user in the network, and ideal node balances, etc]

[Plausable deniability]

[Analyze what is lost when doing circumvention without anonymity]

[Roll items into IPFS]

[IPFS improvements: Authenticated pubsub]

[Add support for PTs]

[Add support for usage as a PT]

[Add a Tor download]

[Privacy enhancements]

[Could use salmon stuff still, even if not perfect, especially since we are not an anonymity system]

[Killer app/other uses/etc]

Bibliography

[1] Yawning Angel. obfs4 Specification. https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt. Accessed 2021.05.18.

[2] Ingmar Baumgart and Sebastian Mies. S/kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*, pages 1–8, 2007.

[3] Juan Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[4] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72. Berkeley, CA, USA, 2003.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[6] Serene Han et al. Snowflake Technical Overview. https://keroserene.net/snowflake/technical/, 2017. Accessed: 2021.05.18.

[7] David Fifield. Turbo tunnel, a good way to design censorship circumvention protocols. In *FOCI@ USENIX Security Symposium*, 2020.

[8] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015, 2015.

[9] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *Symposium on Security & Privacy*. IEEE, 2013.

[10] Amir Houmansadr, Wenxuan Zhou, Matthew Caesar, and Nikita Borisov. Sweet: Serving the web by exploiting email tunnels. *IEEE/ACM Transactions on Networking*, 25(3):1517–1527, 2017.

[11] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems: First International-Workshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers*, pages 53–65. Springer, 2002.

[12] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges, 2012.

[13] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. Enemy at the gateways: Censorship-resilient proxy distribution using game theory. In *NDSS*, 2019.

[14] Milad Nasr, Hadi Zolfaghari, Amir Houmansadr, and Amirhossein Ghafari. MassBrowser: Unblocking the censored web for the masses, by the masses. In *Network and Distributed System Security*. The Internet Society, 2020.

[15] The Tor Project. obfs2 Protocol Spec. https://github.com/NullHypothesis/obfsproxy/blob/master/doc/obfs2/obfs2-protocol-spec.txt, 2013. Accessed: 2023.06.01.

[16] The Tor Project. obfs3 Protocol Spec. https://github.com/NullHypothesis/obfsproxy/blob/master/doc/obfs3/obfs3-protocol-spec.txt, 2013. Accessed: 2023.06.01.

[17] WebTorrent Team. Webtorrent. https://webtorrent.io/. Accessed 2023.06.01.

[18] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 739–752, 2022.

[19] Lindsey Tulloch and Ian Goldberg. Lox: Protecting the social graph in bridge distribution. *Proceedings on Privacy Enhancing Technologies 2023*, 1:16.

[20] Ryan Wails, Andrew Stange, Eliana Troper, Aylin Caliskan, Roger Dingledine, Rob Jansen, and Micah Sherr. Learning to behave: Improving covert channel security with behavior-based designs. *Proceedings on Privacy Enhancing Technologies*, 3:179–199, 2022.

[21] Liang Wang, Kevin P Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. Seeing through network-protocol obfuscation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 57–69, 2015.

[22] Philipp Winter, Tobias Pulls, and Juergen Fuss. Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 213–224, 2013.