The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

A HIGH-PERFORMANCE SYSTEM FOR ANONYMITY IN PEER-TO-PEER FILE TRANSFER NETWORKS

A Thesis in

Information Sciences and Technology

by

Tyler James Frederick

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2017

The thesis of Tyler James Frederick was reviewed and approved* by the following:

Sencun Zhu

Associate Professor of Information Sciences and Technology

Thesis Adviser

Dinghao Wu

Assistant Professor of Information Sciences and Technology

Gerald Santoro

Senior Lecturer of Information Sciences and Technology

Andrea Tapia

Associate Professor of Information Sciences and Technology

Director of Graduate Programs, College of Information Sciences and Technology

*Signatures are on file in the Graduate School.

Abstract

In peer-to-peer file transfer, the properties of privacy and performance are often mutually exclusive. Although there are a wide variety of peer-to-peer applications and networks which attempt to close this gap, even the most elegant systems still present privacy and performance as start trade-offs to one another. Amidst continually increasing concerns about privacy, as well as ever growing amount of data being transferred, a peer-to-peer system which provides a more balanced approach to anonymity and performance is desperately needed. In this research, I propose a new system called Excelsior which embraces both of these properties in a hybrid network utilizing DC-nets, entanglement, and BitTorrent-style content distribution. This system is designed to provide a simple and straightforward means for the ubiquitous BitTorrent to vastly increase user privacy with the minimum possible performance sacrifice.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview

Peer-to-peer (P2P) communication is important. Spanning a wide variety of systems – everything from low-powered wireless sensor nets, to bandwidth-saving technologies in corporate networks such as Microsoft's BranchCache[1], to popular public file sharing software like BitTorrent - P2P communication is everywhere. As a mature technology, P2P is increasingly being used to facilitate the efficient transmission of large volumes of content.

The latter example above, BitTorrent, is a high-performance P2P file sharing protocol. BitTorrent accounts for a substantial percentage of global internet traffic, and is capable of disseminating a wide range of content, including but not limited to: software applications, audio, video, and textual data [4, 9, 16]. Although this protocol has gained popularity due to its efficiency and open-source nature, it lacks virtually any concept of user privacy or anonymity, as BitTorrent readily shares a large amount of information about a user, such as their IP address and what content they are uploading or downloading [1, 9, 13].

A number of extensions or workarounds to the protocol have been proposed to address these shortcomings. One such example is routing BitTorrent traffic through anonymous overlay networks. Solutions like Tor and I2P provide a relatively safe haven, but at a significant cost in terms of bandwidth and latency [15]. In 2015, estimates place global BitTorrent bandwidth usage at around 1.8% of all traffic, or 4.97TB/s, while the total

[1] BranchCache is a trademark of Microsoft Corporation in the United States and/or other countries.

advertised bandwidth across the entire Tor network is only in the range of approximately 30GB/s [4, 16]. Limitations such as this prevent the wide-scale adoption of these measures.

Alternate approaches utilizing non-BitTorrent protocols, networks, and clients exist that provide the required user privacy/anonymity without the performance limitations that routing BitTorrent traffic through traditional overlay networks incurs, however these systems are generally standalone in that they do not communicate with other networks [15]. Since there is no benefit to using these systems if one does not require privacy or anonymity, it unfortunately becomes easy to label those users as "having something to hide".

An ideal solution would protect user privacy/anonymity without sacrificing performance or functionality, all while allowing a hybrid network consisting of both standard and augmented peers [14]. Although this "silver bullet" is likely not a realistic goal, it is still possible to improve the current state of affairs by partially limiting the scope of the privacy or anonymity required to include only the elements deemed absolutely necessary to protect the user.

The approach to protecting user privacy/anonymity in a BitTorrent network is often the same as that used in other networks: an emphasis is placed on guaranteeing anonymity by preventing other users from associating in-network identities or activity with public identities, such as an IP address or name [6, 7, 10]. For some types of networks, this approach is appropriate, as the network exists for a singular purpose, which is the inferred purpose of any users of that network.

As an example of the above concept, one could generally infer that a user of an internet forum dedicated to mathematics is interested in learning about or discussing math. As

an alternate example, the reputation of a user participating on a forum whose topic material was potentially illegal could be damaged simply by being a member of that forum.

BitTorrent networks are different, however. Although there are many types of content available, some of which is sensitive, there is plenty of content that is not [30]. For example, many open-source and freely available Linux distributions offer downloads through the BitTorrent network [30]. It is therefore not possible to accurately infer a user's purpose simply by their participation, but rather by what the user does as a member of the network, and more specifically, what content they upload or download. This is an important distinction, as it allows for targeting the privacy measures in place to specifically address protecting the identity of data being transferred, rather than the much broader and more complex task of protecting the identity of a participant in said transfer.

1.2 Research Objectives

1. To perform a survey of currently available methods for providing privacy/anonymity in BitTorrent networks.
2. To devise a method by which to decrease the tradeoff between privacy/anonymity, and performance in BitTorrent networks.
3. To develop an implementation of this method, conduct testing, and provide an analysis of the results.

Chapter 2: Literature Review

2.1 Communication Models

Every computer network is built around the concept of two or more devices communicating with one another for the purposes of exchanging information. There are a wide variety of different ways to categorize types of networks [25]. In terms of overall architecture, networks can be classified into one of two logical models: client/server or peer-to-peer, with the possibility of hybrid models as well [25].

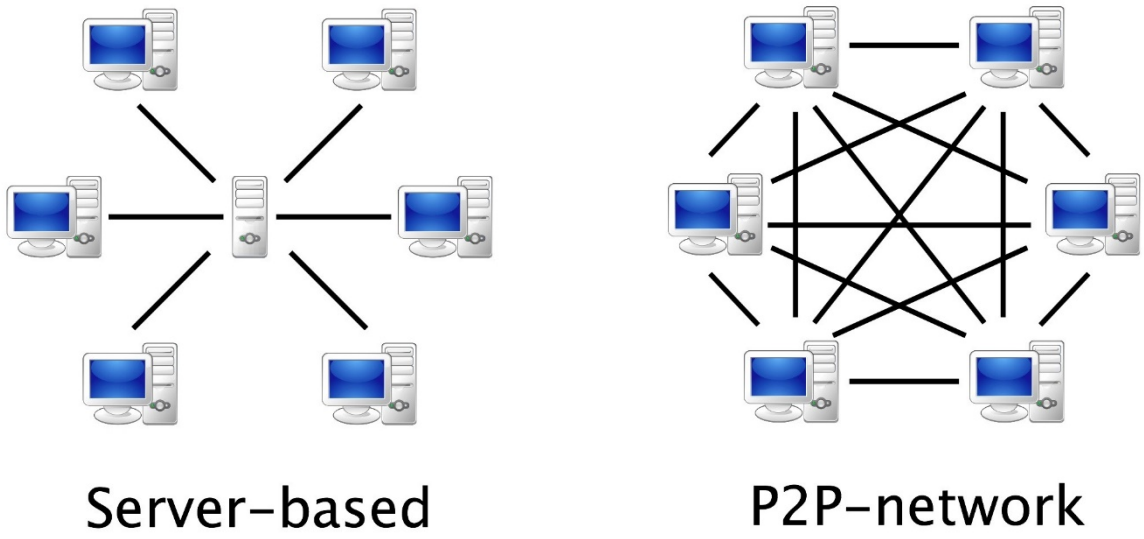

Figure 2.1: Client/Server versus Peer-to-Peer

Client/server networks are likely the most commonly recognizable type. Although not exclusively, this model supports a substantial amount of network infrastructure today, such as the World Wide Web, many applications on corporate and academic networks, and some of the core internet infrastructure (such as DNS) [25]. Client/server networks contain one or more servers, devices that hosts resources such as data or services, and

one or more clients, devices that make use of said resources. Put simply, in a client/server network, servers act as "providers", while clients act as "consumers" [25].

Peer-to-peer networks, although somewhat less well known, are also a critical component in modern communications infrastructure. In contrast to client/server networks, which paint a very "black and white" picture with respect to provider and consumer roles, P2P networks take the opposite approach and allow these roles to be flexible. A P2P network is composed of two or more peers, some or all of which contain resources which may be shared amongst some, all, or none of the other peers in the network [9, 25].

2.2 Peer-to-Peer Networking

Given the vast amount of deployed infrastructure using the client/server model, it is helpful to understand why P2P networks remain important. Client/server networks, and the underlying model of strict provider and consumer roles remain ubiquitous due to a number of key advantages. Specifically, client/server networks are [19, 20, 25]:

- Simpler: Strict separation of roles means software can be compartmentalized.
- Easily managed: All infrastructure providing services is logically (and often physically) grouped together and/or under the control of a single entity.
- More secure: Although not inherently more secure, client/server networks are generally less complex, with simpler software and less entities to protect.
- Reminiscent of real-world workflows: Resources on the internet and local networks are generally controlled by a single entity (the provider) and utilized by many unrelated entities (the consumers).

In spite of these advantages however, client/server networks are prone to several disadvantages as well, specifically [19, 20, 25]:

- Single point of failure: If the server fails, the entire network is disrupted. Redundant systems add a layer of protection, but any complete failure of the server-side disrupts the network for all clients.
- Scalability: Client/server networks require an allocation of resources at the server sufficient to support the full set of clients.

The second item, scalability, is especially important as it is one of the primary advantages of peer-to-peer networks. In any network, communication requires some amount of bandwidth. For a client/server network, each pair of client and server require the same amount of bandwidth, e.g. the combined upstream and downstream bandwidth used by the client is the same as that used by the server [18, 25]. A server may have multiple clients, however, which means the server bandwidth required is a function of the bandwidth per-client multiplied by the number of clients [18, 25].

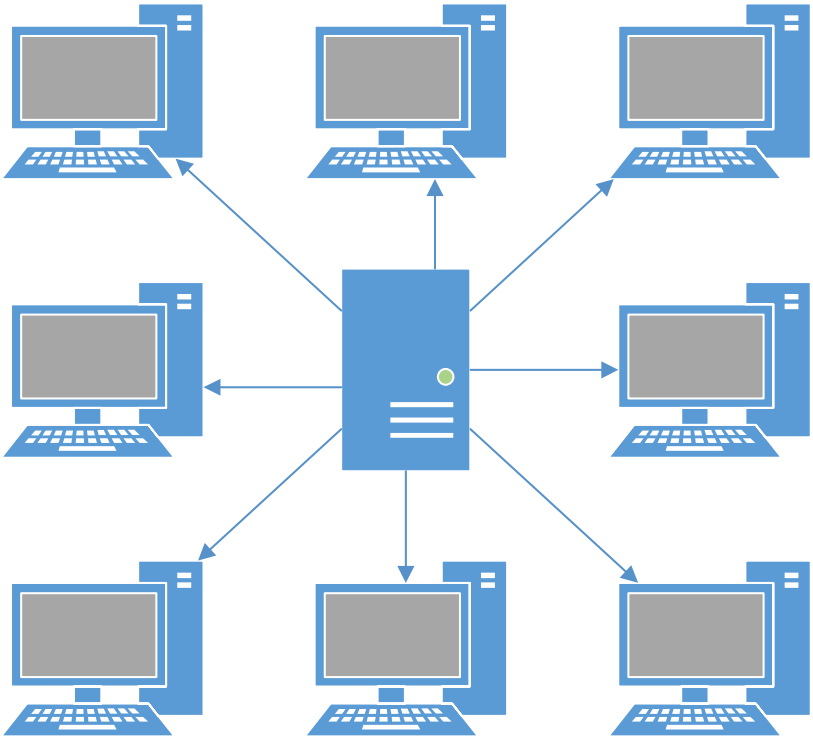

Figure 2.2: Client/Server Content Delivery

An example of this is presented below, describing an environment where a single server is expected to serve a variable number of peers a 100MB file:

Downstream Bandwidth for Downloading a 100MB File		
Clients	**Per-Server Bandwidth**	**Per-Peer Bandwidth**
1	10MB	10MB
10	100MB	10MB
100	1,000MB	10MB
1000	10,000MB	10MB

Table 2.1: Downstream Bandwidth Comparison

As is shown in the table, the bandwidth required for each peer remains fixed, while the server bandwidth required increases linearly. Although not every situation involves the full complement of clients to be requesting the same resource simultaneously, it should be relatively apparent that large resources or numbers of clients can warrant a substantial resource allocation to the server, which is not always practical or feasible. [18, 19, 25].

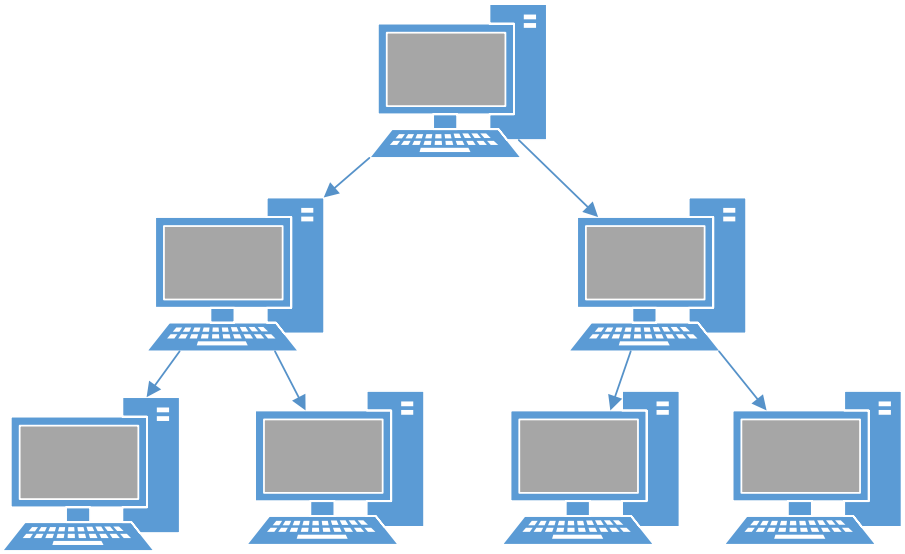

Figure 2.3: Peer-to-Peer Content Delivery

Although more complex, less easily managed, and generally harder to secure, peer-to-peer networks excel where the client/server model falls short. As all peers in the network can potentially serve as both providers and consumers, roles can be distributed amongst the network, and dynamically redistributed if necessary, allowing for a dramatic increase in potential redundancy [18, 20, 25]. This ability to distribute roles amongst the network also empowers a level of scalability not possible in the client/server model – as all peers can fulfill the role of a server, resource requirements can be distributed amongst multiple peers instead of concentrated at a single server [18, 19, 20, 25].

2.3 File Sharing

File sharing is practice of or ability to transmit files from one computer to another over a network or the internet. Due to the broad nature of the definition, and the many uses

of file sharing, it is important to properly define the scope of this process with respect to its use in this paper.

Specifically, as describe here, file sharing is distinct from the related concept of file transfer. While both refer to the process of moving files or data from one networked system to another, file transfer refers only to the transfer itself, while file sharing describes a broader process that includes file transfer, content search, and dissemination of metadata [17, 21, 26]. In general, a complete file sharing system, as viewed from an end user's perspective, is a software application allowing the user to publish, retrieve, search for, and share content with other users of that application [1, 17, 21, 26].

As with most processes, file sharing can be accomplished over both client/server and P2P networks, and typically inherits the same advantages and disadvantages of each network type [1, 27]. Peer-to-peer networks tend to be favored for consumer file sharing for two reasons [2, 15, 20]:

- The use of such networks is often of an ad-hoc nature where no centralized entity exists to manage a server infrastructure.
- As the size of the content and number of peers involved grows, the resources required to support a client/server application can easily scale beyond what a typical end user could support or afford.

Peer-to-peer file sharing networks today are relatively common, and include systems such as Gnutella, Freenet, eDonkey, GNUnet, and BitTorrent, as well as many others [2]. Of all the available systems, the most ubiquitous is BitTorrent [2].

2.4 BitTorrent

BitTorrent is a protocol for P2P file sharing. Designed in 2001, it is currently the most popular such protocol. In common usage, BitTorrent refers to the protocol itself, as well as other components required to facilitate the transfer, such as a client application and metadata search and retrieval services [8, 9].

BitTorrent uses TCP or UDP protocols to communicate over any IP-based network, such as the internet or most local area networks. Users of BitTorrent use a software application, called a BitTorrent client, which implements the protocol and typically provides for management of uploads and downloads. The collection of all users with a BitTorrent client implementing the protocol is known as the BitTorrent network [8, 9].

BitTorrent is based around the concept of a torrent, a set of one or more files shared to or acquired from the network, and a collection of metadata describing those files. Users which hold a complete copy of the torrent are called seeds, while those with an incomplete copy are called peers. The set of all seeds and peers for a given torrent is called a swarm [8, 9].

The user which initially introduces content to the network is called the initial seed. This user gathers a set of files to be included in the torrent and divides those files up into smaller blocks, called pieces. Each block is hashed and added to a list. That list, along with a description of the torrent, its contents, and at least one method for locating seeds make up the metadata for that torrent. Any user wishing to acquire the torrent must first obtain this metadata. Metadata may be disseminated in any way, however typically it is either published to a website in a text file, typically with a .torrent extension, or acquired through the use of a URI pointing to an entry in a distributed hash table (DHT) [8, 9, 23].

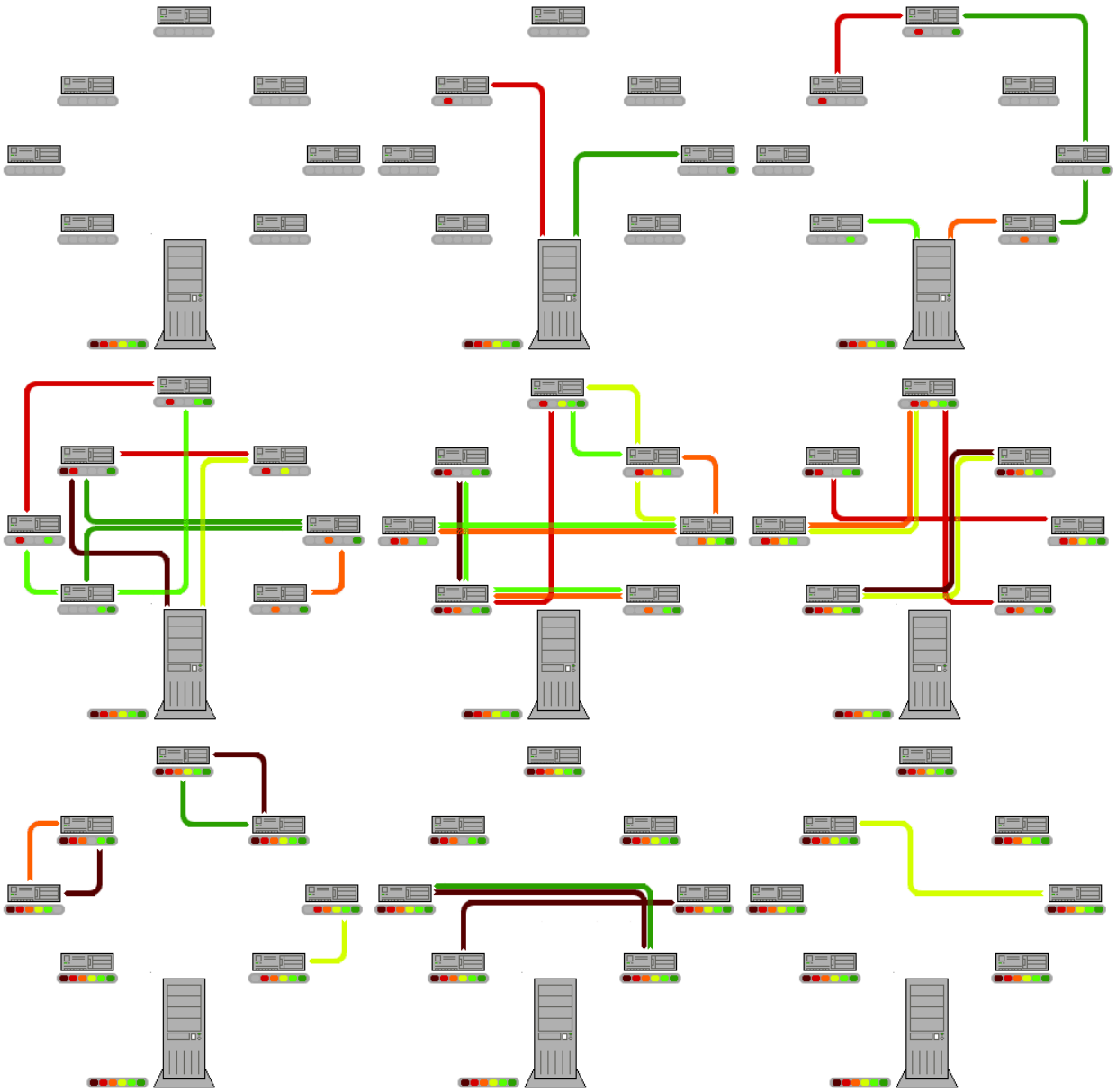

Figure 2.4: BitTorrent Content Distribution[2]

A user acquiring the metadata is able to join the swarm and become a peer. As a peer, the user then attempts to connect to other users in the swarm to download the contents of the files. Unlike traditional file transfer, BitTorrent allows downloading data out of order, from multiple users simultaneously, and from other users with an incomplete set

[2] Image courtesy of Wikimedia Commons. Licensed under Creative Commons Attribution-Share Alike 3.0.

of data. A BitTorrent client may connect to many peers concurrently and begin downloading different pieces of the torrent. When a peer has acquired all of the pieces, it becomes a seed for that torrent [8, 9].

This strategy for distribution is the key element which makes BitTorrent such a powerful protocol – resource requirements are distributed amongst many users in a swarm allowing for simple scalability, and pieces are distributed out-of-order and acquired in a rarest-first manner, greatly increasing redundancy. Together, both of these features allow a peer to acquire content far more quickly than if they were to download from a standalone server [8, 30].

Unfortunately, these advantages do come at a price: user privacy/anonymity. BitTorrent is, by design, a "talkative" protocol. The methods used to connect to other users in a swarm readily make available that user's IP address, while the methods used to determine availability of pieces expose which files from the torrent a user possesses [15].

2.5 Privacy and Anonymity: Defined

Thus far, the terms *privacy* and *anonymity* have been used somewhat interchangeably. While these are two similar and related concepts with somewhat broad definitions in the realm of technology, it is important to distinguish between them.

A general definition of privacy is "the state or condition of being free from being observed or disturbed by other people". As applied to technology, privacy generally refers to a condition where access to a set of data is controlled such that only the owner, or those authorized by the owner can view or manipulate it. In the context of this paper, the definition is narrowed even further, to mean simply that the contents of a set

of data cannot be positively identified by outside individuals. A simple example of privacy would be submitting personal information using an encrypted web page – the encryption prevents the average user from eavesdropping on the communication and viewing the contents, but not necessarily from observing that the communication has occurred.

Anonymity on the other hand, is a state in which a user's identity cannot be established. Traditionally, anonymity describes a state in which one cannot be identified at all. For two-way internet communications, however, it is often infeasible to become completely anonymous [6, 14]. Even in overlay networks such as Tor, a user's public IP address is still visible to immediately adjacent nodes to which they are connected [6, 10, 14]. For this reason, anonymity in an internet domain commonly refers to a state where a one's assumed identity cannot be associated with specific actions they perform. In the previous Tor example, a user would still be considered anonymous because other users both inside and outside the network would be unable to associate any actions by the user with the user's public IP address [6, 7, 22].

Privacy and anonymity are both extremely relevant concepts when discussing any type of communications on the internet. Relative to situations where sensitive information is maintained in physical form, the interconnected nature of most internet services greatly increases the degree to which personal or sensitive information could potentially be exposed. In addition, the variety of information and the frequency with which it is sent and received places it at additional risk.

These are especially relevant concerns in P2P file sharing due to the relative ease of gathering information about network users and their actions [29]. Many P2P file sharing applications take little or no steps to ensure privacy or anonymity. BitTorrent, in

particular, allows any peer or seed in a swarm to obtain a wide variety of information about other peers, such as their public IP address, communication port, BitTorrent software version, connected peers, download progress, and most importantly what files they possess [29].

The ease at which this information can be obtained presents a serious threat to the privacy/anonymity of BitTorrent users [11, 15]. This threat is compounded when the wide variety of content available through BitTorrent is considered. Although it is true that BitTorrent is used to facilitate the (often illegal) distribution of copyrighted content, there are many other, more legitimate scenarios which warrant concern for privacy/anonymity, such as where:

- The content could be illegal in the user's location. Although this covers distribution of copyrighted materials, it also refers to acquiring literature or educational materials which are outlawed in a particular location [5[.
- The content might be sensitive, such that any association with the user would have potential social or political ramifications. Academics researching sensitive topics often desire to keep their research activity separate from their public identity [6, 7].
- The content might be the target of censorship efforts by a corporate or government entity. This could be as benign as simply blocking access to internet resources, or as serious as an oppressive government regime seeking out those who speak critically of leadership [6, 7].

These scenarios should make it apparent that BitTorrent and other P2P file sharing services present legitimate concerns for users other than those simply distributing copyrighted materials, and therefore worthwhile of a solution to protect the

privacy/anonymity of those users. As described in the next section, however, implementing privacy/anonymity comes at a price.

2.6 Privacy and Anonymity: Trade-Offs

In the technology world, the expression "there is no free lunch" is especially apt. Although improvements are constantly being made to increase performance, security, usability, and reliability, there will always exist some trade-offs between these items. An excellent example of this can be seen between security and usability – for example, passwords that are longer and more complex are generally more secure, but become very unwieldy to remember and input [14].

Privacy and anonymity are no exception to this rule, as they often present with an inverse relationship to performance. One great example of this is encryption – it's well know that encrypted communications help to increase privacy by preventing eavesdropping, but encryption itself always incurs some level of performance impact [14]. Many anonymity solutions present inverse relationships with respect to performance as well. Tor, for example, provides a high degree of anonymity, but sacrifices throughput and latency to do so [6].

Finding the appropriate harmony between performance and privacy/anonymity is a balancing act that both designers and users of software must take into consideration. It is important to recognize that there is no "one size fits all" solution – different types of software and different user requirements will yield different balances between the two [14]. For example, a user casually browsing a forum under a pseudonym has very different requirements than a user using social media to report on an oppressive government regime. Both users will expect vastly different levels of anonymity, and will

also be willing to tolerate very different levels of performance degradation to achieve that anonymity [5, 14].

Achieving the correct balance is extremely important in file sharing as well, as there a similarly wide variety of use cases for this technology. While some of these cases demand extremely strict privacy/anonymity and are willing to accept significant performance segregation to achieve this, there are many others that have no need for anonymity at all, and are strictly concerned with achieving the maximum performance [14].

2.7 Existing Solutions

At present, users have very few realistic options for maintaining privacy/anonymity while using BitTorrent. Although there are a number of existing networks that provide this type of protection, such as Publius, OFF, Tangler, and Dagster, to name a few, none outperform or are able to inter-operate with standard BitTorrent [24, 26]. With no performance gain, and a substantially smaller group of users, the only tangible benefits are to those who explicitly desire privacy. Unfortunately, by using these networks, those users might actually be assumed to have "something to hide" [31]. Although this may not always present an issue, consider a scenario where a user lives in a country with an oppressive government regime – being singled out as possible having "something to hide" might have serious consequences. It's therefore important for these networks to either provide benefits other than privacy/anonymity to being them into mainstream usage, or to interoperate with an existing mainstream network, such as BitTorrent.

Currently, there is one solution for anonymity that is fully interoperable with BitTorrent: Tor. As described previously, Tor is an anonymizing overlay network that can be used to provide users with anonymity by encrypting and routing their traffic

over multiple hops within the network [6, 7]. Tor is designed as a general-purpose overlay network, and is thus capable of accepting traffic from most applications without any modification, including BitTorrent traffic [6, 7]. This model provides a complete anonymity solution to BitTorrent, however it does so at the expense of a substantial performance penalty, primarily in speed, but also in latency as well [6, 7, 28].

Not only is Tor not designed to efficiently route the large, continuous streams of traffic produced in BitTorrent, but the network simply lacks the capacity to handle the volume of traffic - in 2015, estimates place global BitTorrent bandwidth usage at around 1.8% of all traffic, or 4.97TB/s, while the total advertised bandwidth across the entire Tor network is only in the range of approximately 30GB/s [4, 6, 16]. This means that while running BitTorrent over Tor is a feasible option for those who value anonymity above all else, the performance trade-off is unacceptable for most users [6, 7].

With no mainstream solution available to provide privacy/anonymity for P2P file sharing, it is reasonable to explore developing one. As the remainder of this paper will describe, it is possible to adapt some of the techniques used in BitTorrent over Tor and other P2P file sharing networks which provide privacy/anonymity to create a hybrid system, interoperable with traditional BitTorrent, which provides an acceptable level of both privacy/anonymity and performance.

Chapter 3: Excelsior

3.1 Overview

Given the two extremes represented by plain BitTorrent and BitTorrent over Tor with respect to performance and privacy/anonymity, a third option is clearly needed. This paper introduces a system called Excelsior, which is designed to take a middle road. Excelsior is a system which builds on top of BitTorrent, and provides privacy by focusing on anonymizing the content rather than the participants. Since only the content being distributed in a BitTorrent swarm is sensitive, the complexity and pitfalls of anonymizing overlay networks can be avoided. As the only other mainstream privacy/anonymity solution for P2P file sharing, this paper will frequently call on BitTorrent over Tor as a baseline or reference for the functionality implemented in Excelsior.

3.2 Objectives for Resulting System

- Provide user privacy in the form of plausible deniability via content anonymity for all network participants.
- Provide a decrease in overall transfer time as compared to running unmodified BitTorrent over Tor under real-world conditions.
- Utilize the existing BitTorrent protocol as the final method of distribution, to allow for a seamless transition.
- Allow the trade-off between performance and privacy to be customized by the network participants.

3.3 General Theory of Operation

Excelsior is a peer-to-peer file sharing network communicating over a LAN or the internet using traditional networking protocols, such as TCP/IP. This network consists of two distinct segments, or phases. Phase one encompasses a dining cryptographers network (DC net), while phase two is comprised of a traditional BitTorrent network.

Hosts in either phase are grouped into seeds and peers. A seed is a provider; the node which initially introduces a complete piece of content they wish to make available to the network. Peers are consumers; nodes who wish to obtain content from the network. Both seeds and clients may participate in the distribution of content.

As with most peer-to-peer networks, hosts may physically participate in many different networks simultaneously, however these communications are logically differentiated via the concept of a swarm. A swarm is a logical grouping of three or more seeds, each with a discrete piece of content, and zero or more peers, with a boundary corresponding to the distribution of aforementioned content.

In Excelsior, every swarm begins with a minimum of three hosts, at least one of which is a seed. These hosts can be selected through an automated matchmaking process, or by the end user operators. They agree to collaborate throughout the first phase to prepare their collective set of content for distribution to a larger set of peers in phase two. The purpose of this preparation is to allow for privacy for all of the initial hosts, as well as optionally to all peers in phase two.

Once a swarm has been established, participating hosts begin the process of distribution. Seeds must exchange content with all other hosts in a manner that provides each a copy of all content files, without revealing which seed originally provided that content. In

order to accomplish this task, a full-mesh DC net[3] connected to all hosts is established. DC nets provide a way for multiple parties to exchange messages with one another anonymously. All parties are able to send and receive over the network, but no individual party can ascertain the source of a message.

Once the DC net is established, participating seeds begin to disseminate the content they wish to share amongst one another. All members of the DC net receive a copy of the content. This transfer process repeats for each seed that is a member and has content to share, and concludes when all hosts have a complete copy of all content. At this stage, all hosts in the DC net are now seeds.

Next, each seed must prepare the content they hold for distribution to the rest of the network. Since this network relies on anonymity of the content to protect all peers, clients cannot simply download one of the source files directly, as this would reveal their intentions. Instead, entanglement is used to "intertwine" all of the content. This process, which is accomplished through the use of the simple exclusive OR (XOR) operator, reversibly combines multiple files such it is impossible to determine which file a downloading peer is actually attempting to acquire, thereby providing plausible deniability with respect to the peer's intentions.

After all seeds have completed the entanglement operation, they publish metadata for each piece of content to an external location. This metadata provides a set of

---

[3] It should be noted that this scheme can theoretically accommodate both full and partial-mesh DC nets with any number of users each. For the purposes of simplicity, however, all examples in the paper refer to a three member, full mesh group.

instructions to clients detailing how to acquire a specific piece of content. It contains a description of the content, identification for all of the file pieces, instructions for reconstructing the content from the entangled data, and connection information for all peers. Once metadata has been published, the network moves to phase two, which allows for the actual acquisition of content by peers.

Phase two is carried out entirely through the use of unmodified BitTorrent. It begins with a peer wishing to acquire content described in the metadata from phase one. A peer obtaining this metadata uses the list of seeds specified in the metadata to bootstrap and make initial connections. The peer determines which files are necessary to obtain in order to reconstruct the original content, and then requests them from the seeds. As with any other BitTorrent implementation, peers can hold any number of files or pieces of files, and share them with other members in the swarm. This process allows extremely efficient distribution to all network members.

Once a peer has acquired the necessary files, as listed in the metadata, the peer performs the reconstruction process (simply reversing the steps performed during entanglement), to restore the original content. Since the downloaded files are an amalgamation of multiple pieces of source content, and can be reconstructed in several different ways, it is impossible for an observer to determine which content the peer was attempting to obtain. As it has only obtained the metadata for a single piece of source content, the peer itself is unaware of what other content are resident in the files obtained. This scheme protects provides the peer with privacy through plausible deniability.

3.4 Modes of Operation

3.4.1 Overview

Even in a system which is specifically designed to maximize performance, anonymity still represents a significant trade-off. In order for such a system to enjoy widespread adoption, it should allow for the tradeoff between performance and anonymity to be adjusted to some extent by the end user to best suit their particular requirements. To this end, Excelsior includes two distinct modes of operation: single source and multi-source.

3.4.2 Single Source

In single source mode, the phase one group consists of multiple peers, but only one seed (only one peer with content to share). This initial group is formed through a matchmaking process, and conducts the entanglement process. Once completed, the entangled files are distributed via traditional BitTorrent. This mode provides higher transfer performance at the cost of anonymity – peers must download less data, but only the privacy of initial seed group is protected.

Due to the DC-net based phase one exchange, no entity in the network is able to perform a mapping between a piece of content and the seed who originally uploaded it. Source files are only distributed within the DC-net, protecting the nodes in the phase one group. Although nodes in this group could potentially acquire the file metadata and determine what content they are hosting, without doing so, each node is only aware of the content they introduced, allowing for plausible deniability for this potentially sensitive content.

In the single source model, peers in the BitTorrent based exchange must download three files, each of which represents one-third of the original file size. Since only a single piece

of content is available in the network, participation infers acquisition of that content. The entanglement process does provide a minor advantage to peers from a privacy perspective, as they would no longer be acquiring source content, and thus proof of a complete download would be required to substantiate the ability to reconstruct the source data.

The single source mode is highly efficient. As compared to a pure BitTorrent network, the only overhead incurred is the time required for the phase one group to exchange content. Since this does not directly impact the distribution time (only the time to publish), it is considered to have no impact on performance. Additionally, since the phase two exchange begins with at least three seeds as opposed to one, a performance gain is actually realized for all peers in phase two.

Peers in phase two must only download three files, each representing a third of the original file, which causes no overhead whatsoever for this group. A negligible amount of time is expended by peers to reconstruct the source files, however as this falls after delivery has completed, it is considered to have no impact on performance either.

3.4.3 Multi-Source

In multi-source mode, the phase one group consists of three or more peers, each with content to share. As with the single content model, this initial group is formed through matchmaking and conducts the entanglement process. Once completed, the entangled files are distributed via traditional BitTorrent in phase two. This mode provides the highest level of privacy at the cost of transfer performance – peers must download more data, but all network members are protected.

As with the single content model, no entity in the network is able to perform a mapping between a piece of content and the seed who originally uploaded it. Source files are only

distributed within the DC-net, protecting the nodes in the phase one group. Although nodes in this group could potentially acquire the published metadata and determine what content they are hosting, without doing so, each node is only aware of the content they introduced, allowing for plausible deniability this potentially sensitive content.

Unlike the single content model however, the BitTorrent based exchange is protected as well. Clients in the BitTorrent network download only entangled files. In an example network with three pieces of content, and a client downloading any three of the four files, it is impossible to prove which file they intend to reconstruct. Further, since all exchanges are performed with entangled files, a peer uploading files it has downloaded does not necessarily have knowledge of what any of those files are, which indemnifies them.

The multi-source mode provides maximum anonymity at the cost of performance. As with the single source mode, there is overhead incurred at the time required for the phase one group to exchange content. Since this does not directly impact the distribution time (only the time to publish), it is considered to have no impact on performance. Additionally, since the phase two BitTorrent exchange begins with at least three seeds as opposed to one, a performance gain is actually realized for all peers in phase two.

Peers in phase two must download multiple blocks, each representing the size of the original file. For a network containing three sources, two blocks would be required to reconstruct a single piece of content, while three blocks would be required to ensure anonymity and reconstruct all of the content. The total bandwidth required would be equal to the number of blocks required times the block size. A negligible amount of time

is expended by peers to reconstruct the source files, however as this falls after delivery has completed, it is considered to have no impact on performance.

3.5 Technical Description

3.5.1 Common Elements

3.5.1.1 Dining Cryptographers Network

DC-nets are used exclusively to facilitate the distribution of the source content to the phase one group. The method of operation for a DC-net is well documented in security literature, however a brief example of how the network is applied to this research is shown below.

There exists an example set of three seeds:

$$\{S_1, S_2, S_3\}.$$

One seed is in possession of its own content:

$$S_1 : \{F_1\}$$

$$S_2 : \{F_2\}$$

$$S_3 : \{F_3\}$$

Each pair of seeds negotiates a shared Boolean secret.

$$\{S_1, S_2\} = 0$$

$$\{S_1, S_3\} = 1$$

$$\{S_2, S_3\} = 1$$

This can also more easily be visualized through a diagram such as the following:

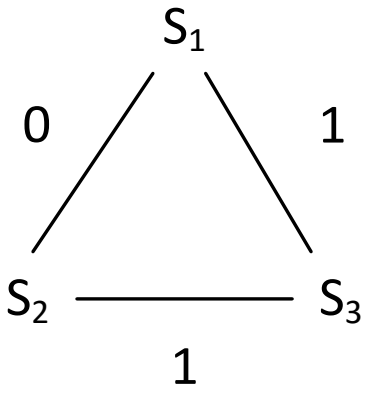

Figure 3.1: DC Net Exchange

A file from the set $\{F_1, F_2, F_3\}$ is randomly selected by the network as the first to be transferred. Only the seed with this file is aware that it is the file holder. Each member of the network then attempts to answer the question:

Where $Bit_x(n)$ is the nth bit in the selected file, F_x,

Is the value of $Bit_x(n) = 0$?

Example:

The selected file is F_1.

The contents of file F_1 are 01

Therefore, $Bit_1(1) = 0$, $Bit_1(2) = 1$

Each seed must answer the question "Does $Bit_1(1) = 0$"

The value of each seed's answer is given as an operation on the two shared secrets that seed holds. Those operations are as follows:

True: $\neg(Shared\ Secret\ \#1 \oplus Shared\ Secret\ \#2)$

False or Don't Know: $Shared\ Secret\ \#1 \oplus Shared\ Secret\ \#2$

Example:

Shared Secrets: $\{S_1, S_2\} = 0$, $\{S_1, S_3\} = 1$, $\{S_2, S_3\} = 1$

Selected File: F_1

Bit Values: $Bit_1(1) = 0$, $Bit_1(2) = 1$

Question: "Does $Bit_1(1) = 0$?"

Seed S_1 knows this is true, answering $\neg(0 \oplus 1) = \neg(1) = 0$

Seed S_2 doesn't know, answering $(0 \oplus 1) = 1$

Seed S_3 doesn't know, answering $(1 \oplus 1) = 0$

Each seed then computes the logical XOR of all three answers, and determines the answer to the question based on the following key:

Where A_x represents the answer from seed x,

If $A_1 \oplus A_2 \oplus A_3 = 0$, then FALSE

If $A_1 \oplus A_2 \oplus A_3 = 1$, then TRUE

Example:

$A_1 = 0$, $A_2 = 1$, $A_3 = 0$

$0 \oplus 1 \oplus 0 = (0 \oplus 1) \oplus 0 = 1 \oplus 0 = 1$

Therefore, the statement "Does $Bit_1(1) = 0$" is TRUE.

Which is correct, as $Bit_1(1) = 0$

Both seeds not possessing the file have determined that the first bit is a zero, and save this value to disk. The process is repeated again, incrementing the position in the file

each time until the entire file has been transferred. If there are multiple files, a new file is then selected, and the process continues until all files have been shared amongst all seeds. Using this process, every seed can obtain all source files without revealing which seed introduced which file.

It should be noted that shared secrets must be exchanged securely (such as over a channel secured using public key cryptography), and that they should be rotated after each response to maintain the secrecy of the communication. Since renegotiating the secret value each time is very expensive in terms of time and bandwidth, an alternative is to generate shared secrets en masse through the use of a common cryptographically secure pseudo-random number generator (CSPRNG) and a shared seed value exchanged over the secure channel. For example, if all seeds agree upon the use of CSPRNG C, and each pair of seeds exchanges an initial seed value S, the pair can generate a large amount of shared secrets without the need for any additional communication.

3.5.1.2 Block Identification and Validation

With the exception of the original source files on the introducing seed, every block used throughout the network, whether it contain source content or entangled data, must be uniquely identified. The integrity of these blocks must also be validated before and after transfer or manipulation to protect against corruption or a malicious network participant. To simplify things, blocks are identified by a cryptographic hash of their contents, which can also be used to validate the integrity of the block.

3.5.1.3 Publishing and Advertisement

Each seed is responsible for publishing metadata which allows BitTorrent clients to connect and acquire the content they originally introduce. This metadata must include, at a minimum:

- Unique identifier for each block
- A description of the content
- A URI for each of the seeds
- Mapping information to reassemble the blocks correctly

As is common in the BitTorrent environment, metadata can be published in a wide variety of formats, such as a .torrent file, info hash, data URI, or many other formats. Excelsior utilizes the standard .torrent file with the following modifications:

- A dictionary of key/value pairs showing reassembly combinations is included.
- A DHT table with connection information for all of the initial seeds is included.
- Each block held on the seeds is included as a 'file', with the name equal to the block's identifier/hash.

3.5.2 Single Source

3.5.2.1 Entanglement Process

The process of preparing the source files such that a download remains private is known as entanglement. This process uses the *exclusive or* operation to combine or *entangle* source files.

There exists an example network consisting of three seeds:

$$\{S_1, S_2, S_3\}.$$

One seed is in possession of content, while the other two have no content:

$$S_1: \{F\}$$

$$S_2: \{\quad\}$$

$$S_3: \{\quad\}$$

The seed holding content, S_1, splits its content into three identically-sized pieces, padding the content if necessary.

$$F \rightarrow \{F_1, F_2, F_3\}$$

The seed performs the following operations, saving the results:

$$F_1 \oplus F_2 = A$$

$$F_1 \oplus F_3 = B$$

$$F_2 \oplus F_3 = C$$

$$F_1 \oplus F_2 \oplus F_3 = M$$

The seed discards the following files:

$$F_1, F_2, F_3$$

The network state is now as follows:

$$S_1: \{A, B, C, M\}$$

$$S_2: \{\ \ \}$$

$$S_3: \{\ \ \}$$

3.5.2.2 Initial Seed-to-Seed Distribution

The DC-net file transfer procedure described in 3.5.1.1 is utilized to transfer pieces to achieve the following state:

$$S_1: \{A, B, C, M\}$$

$$S_2: \{A, B, C, M\}$$

$$S_3: \{A, B, C, M\}$$

Following this exchange, S_1, being the only seed to introduce content, proceeds to publish the metadata for its content.

3.5.2.3 Client Downloads

Transfer of content among peers is accomplished using the standard BitTorrent protocol. The entire set of files $\{M, A, B, C\}$ is treated like an individual torrent, with clients selecting files they wish to acquire.

3.5.2.4 Reassembly

A client wishing to acquire content in this network is, of course, ultimately attempting to acquire file F, and thus, by extension, must the set:

$$\{F_1, F_2, F_3\}$$

In order to reassemble the above files, the client must always acquire:

$$M$$

The peer must also acquire two of the following.

$$A, B, C$$

With M and two of $\{A, B, C\}$, any source file can be reassembled. The following reassembly operations are possible:

With $\{M, A, B\}$:

$$F_1 = M \oplus A \oplus B$$

$$F_2 = M \oplus B$$

$$F_3 = M \oplus A$$

With $\{M, A, C\}$

$$F_1 = M \oplus C$$

$$F_2 = M \oplus A \oplus C$$

$$F_3 = M \oplus A$$

With $\{M, B, C\}$:

$$F_1 = M \oplus C$$

$$F_2 = M \oplus B$$

$$F_3 = M \oplus B \oplus C$$

With $F_1, F_2, and\ F_3$, the client can reassemble F:

$$F_1 + F_2 + F_3 = F$$

3.5.3 Multi-Source

3.5.3.1 Initial Seed-to-Seed Distribution

There exists a network of three seeds:

$$\{S_1, S_2, S_3\}.$$

Each seed begins in possession of its own content:

$$S_1: \{F_1\}$$

$$S_2: \{F_2\}$$

$$S_3: \{F_3\}$$

The DC-net file transfer procedure described in 3.5.1 is utilized to achieve the following state, where each seed is in possession of its own content, as well as the content of the other two seeds:

$$S_1: \{F_1, F_2, F_3\}$$

$$S_2: \{F_1, F_2, F_3\}$$

$$S_3: \{F_1, F_2, F_3\}$$

3.5.3.2 Entanglement Process

The process of preparing the source files such that a download is rendered private is known as entanglement. This process uses the *exclusive or* operation to combine or *entangle* source files.

Each seed performs the following operations, saving the results:

$$F_1 \oplus F_2 = A$$

$$F_1 \oplus F_3 = B$$

$$F_2 \oplus F_3 = C$$

$$F_1 \oplus F_2 \oplus F_3 = M$$

Each seed discards the following files:

$$F_1, F_2, F_3$$

Each seed is now in possession of the following data:

$$S_1 : \{A, B, C, M\}$$

$$S_2 : \{A, B, C, M\}$$

$$S_3 : \{A, B, C, M\}$$

Following this exchange, all seeds proceed to publish the metadata for their respective content.

3.5.3.3 Client Downloads

Transfer of content among peers is accomplished using the standard BitTorrent protocol. The entire set of files $\{M, A, B, C\}$ is treated like an individual torrent, with clients selecting the files they wish to acquire.

3.5.3.4 Reassembly

A client wishing to acquire content in this network is, of course, ultimately attempting to acquire one or more files in the set:

$$\{F_1, F_2, F_3\}$$

In order to reassemble any of the above files, the client must always acquire:

$$M$$

The peer may choose to acquire one or more of the following. A single acquisition provides the ability to reassemble a single source file with less of a performance impact, but fails to provide privacy, as the intended reassembly can be determined. Two or more acquisitions require additional bandwidth, but preserve privacy.

$$A, B, C$$

With M and one of $\{A, B, C\}$, any source file can be reassembled. The following reassemble operations are possible:

With $\{M, A\}$:

$$F_3 = M \oplus A$$

With $\{M, B\}$

$$F_2 = M \oplus B$$

With $\{M, C\}$:

$$F_1 = M \oplus C$$

With M and two or more of $\{A, B, C\}$, any source file can be reassembled. The following reassemble operations are possible:

With $\{M, A, B\}$:

$$F_1 = M \oplus A \oplus B$$

$$F_2 = M \oplus B$$

$$F_3 = M \oplus A$$

With $\{M, A, C\}$

$$F_1 = M \oplus C$$

$$F_2 = M \oplus A \oplus C$$

$$F_3 = M \oplus A$$

With $\{M, B, C\}$:

$$F_1 = M \oplus C$$

$$F_2 = M \oplus B$$

$$F_3 = M \oplus B \oplus C$$

Chapter 4: Performance Analysis

4.1 Overview and Methodology

There are many approaches to providing privacy/anonymity in the realm of file sharing, and as shown in this paper, not all solutions are created equal. Excelsior's primary goal is to improve transfer performance while still maintaining a level of privacy/anonymity comparable to BitTorrent over Tor. While the hypothetical performance of Excelsior exceeds that of BitTorrent over Tor, this chapter describes a suite of testing done to validate that that performance holds in the real word.

Specifically, the following items are discussed below:

- Minimum performance goals for Excelsior
- The configuration of the testing environment
- A list of metrics gathered during testing
- The specific configuration of each test
- The outcomes of the performance analysis

4.2 Goals

As stated in the research objectives, one of the primary goals for this project is to decrease the tradeoff between privacy/anonymity and performance in a BitTorrent network. It is therefore a reasonable goal for any new solution to outperform the existing solution, BitTorrent over Tor. With respect to specific metrics, Excelsior aims to achieve the following outcomes:

- Provide an overall transfer time (combined DC net and BitTorrent metrics) that is similar or less than that of BitTorrent over Tor.
- Provide a second phase (BitTorrent only) transfer time that is less than that of BitTorrent over Tor.

4.3 Test Environment

All of the tests conducted to produce the benchmarks described in this chapter were carried out in a laboratory environment. This setting consisted of multiple virtual machines hosted in a hypervisor running on a single physical server built using off-the-shell hardware. The use of a fully virtualized environment allowed all networking to be handled in a virtual domain, substantially reducing complexity and eliminating the need to address network connections between physical servers.

4.3.1 Physical Hardware Configuration

The physical system used to host the virtual lab environment is a Dell PowerEdge T605 server. This is an off-the-shelf Dell system with relatively modest commodity hardware. VMware's free ESXi hypervisor was used to support the virtual environment. The system was connected to a workstation used to control and monitor the virtual machines through a web interface to the hypervisor.

- Dell PowerEdge T605 Tower Server
- 2x AMD Opteron 2000 Quad-core Processors
- 32GB DDR2 SDRAM
- 1TB Samsung 850 Pro SSD
- PERC 6/i RAID Controller
- 2x 1Gbps Network Interfaces
- 2x 675W Power Supplies
- VMware ESXi 6.5.0a

Figure 4.1: Dell PowerEdge T605 Server

4.3.2 Virtual Machine Configuration

All testing was complete on virtual machines (VMs) running within the ESXi hypervisor. A total of eight VMs were configured, all with identical specifications. Windows 10 was used as the operating system, and was updated with all available patches as of February 2017. Any configuration not explicitly specified below was left at the default value:

- Windows 10 Enterprise LTSB 2016
- 8 Virtual Processors
- 4GB of RAM
- VMware Paravirtual SCSI Adapter
- VMXnet3 (VMware Paravirtual) 10Gbps Network Adapter
- Virtual Hardware Version 13

Once a VM was configured, a snapshot was taken to preserve the system state, and allow for two separate configurations to be tested. Once testing was complete for the

first configuration, the virtual machine state was rolled back to the snapshot, and the second configuration was instituted.

- In the first configuration, Tor version 0.2.9.9 was installed, along with version 3.4.9 of the uTorrent BitTorrent client. This was used to establish the reference performance of running BitTorrent over Tor.
- In the second configuration, the excelsior application was installed, along with version 3.4.9 of the uTorrent BitTorrent client. This was used to establish the performance of the Excelsior application.

4.4 Metrics

The table below lists six metrics collected during each round of testing. Together, these metrics provide an accurate picture of the performance characteristics of both Excelsior and BitTorrent over Tor. It is important to note that not all metrics apply to every test – for example, there is no DC net time present in BitTorrent over Tor.

Name	**Unit**	**Description**
DC Net Overhead	Percent	How much traffic is used for non-data transfer
DC Net Time	Seconds	How long does the DC net transfer take
BitTorrent Overhead	Percent	How much traffic is used for non-data transfer
BitTorrent Time	Seconds	How long does the BitTorrent transfer take
Preparation Time	Seconds	How long does the entanglement process take
Reassembly Time	Seconds	How long does the reassembly process take

Table 4.1: Performance Metrics

4.5 Test Configuration

To effectively illustrate how performance can be adjusted using the single source and multi-source modes of Excelsior, three separate test configurations were used: Excelsior in multi-source mode, Excelsior in single-source mode, and the reference BitTorrent over Tor. In all three configurations, a total of eight virtual machines were used, each participating in every test. For BitTorrent over Tor, a single VM served as a seed, while the remaining seven served as peers. For both modes of Excelsior, three VMs served as members of the initial seed group, while the remaining five VMs served as peers.

To accurately reflect how real-world variables affect performance, each configuration was tested with seven different size files, with sizes of 128MB, 256MB, 512MB, 1GB, 2GB, 5GB, and 10GB. For multi-source modes, three of the same file were given to the initial seed group. Finally, each configuration type and file size was tested with ten separate iterations to account for any variability in the environment. Although only the averages are presented in this section, the full set of data can be found in the appendix.

Each test was conducted in the same manner as these systems are used in real world file sharing: the designated seed VM(s) were configured with a file to share, and the designated peer VMs were configured to acquire that file. Time and bandwidth data were collected using debugging counters in the Excelsior code. All VMs began each test simultaneously, and the tests concluded when the last peer VM had acquired the target file.

4.6 Test Results

4.6.1 Phase Two Transfer Time

The phase two transfer time analysis is intended to compare metrics on the BitTorrent network only. In order to achieve anonymity, the reference system, BitTorrent over Tor, routes BitTorrent traffic through a Tor overlay network. This is the primary bottleneck of this approach which Excelsior attempts to overcome. Excelsior instead guarantees privacy by using entangled file. While not subject to the bottlenecks of an overlay network, entanglement requires each peer to download substantially more data. This comparison shows how Tor's bottleneck compares to Excelsior's overhead.

The following three tables show the raw averages from ten individual iterations for overhead and transfer time across all three configurations and all seven file sizes. The file table shows BitTorrent over Tor, while the latter two show Excelsior single and multi-source modes, respectively.

BitTorrent over Tor							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	83.62%	83.58%	83.99%	83.43%	83.60%	83.50%	83.71%
Seconds	0.268	0.534	1.094	2.116	4.275	10.615	21.515

Table 4.2: BitTorrent over Tor Benchmarks

Excelsior BitTorrent (Single Source Mode)							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	12.11%	11.00%	11.91%	11.86%	13.37%	11.77%	10.84%
Seconds	0.036	0.07	0.142	0.284	0.578	1.418	2.807

Table 4.3: Excelsior BitTorrent Single Source Benchmarks

Excelsior BitTorrent (Multi-Source Mode)							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	12.03%	11.11%	11.70%	11.04%	11.58%	11.43%	13.90%
Seconds	0.107	0.211	0.425	0.845	1.7	4.24	8.72

Table 4.4: Excelsior BitTorrent Single Source Benchmarks

While not necessarily the easiest to visualize in tabular format, plotting the information in a line graph allows one too easily observe trends across the different test configurations. The chart below shows that while transfer times are very similar for smaller files, as file size grows, so does the gap between BitTorrent over Tor and Excelsior, with Excelsior showing a clear advantage above the 1GB size.

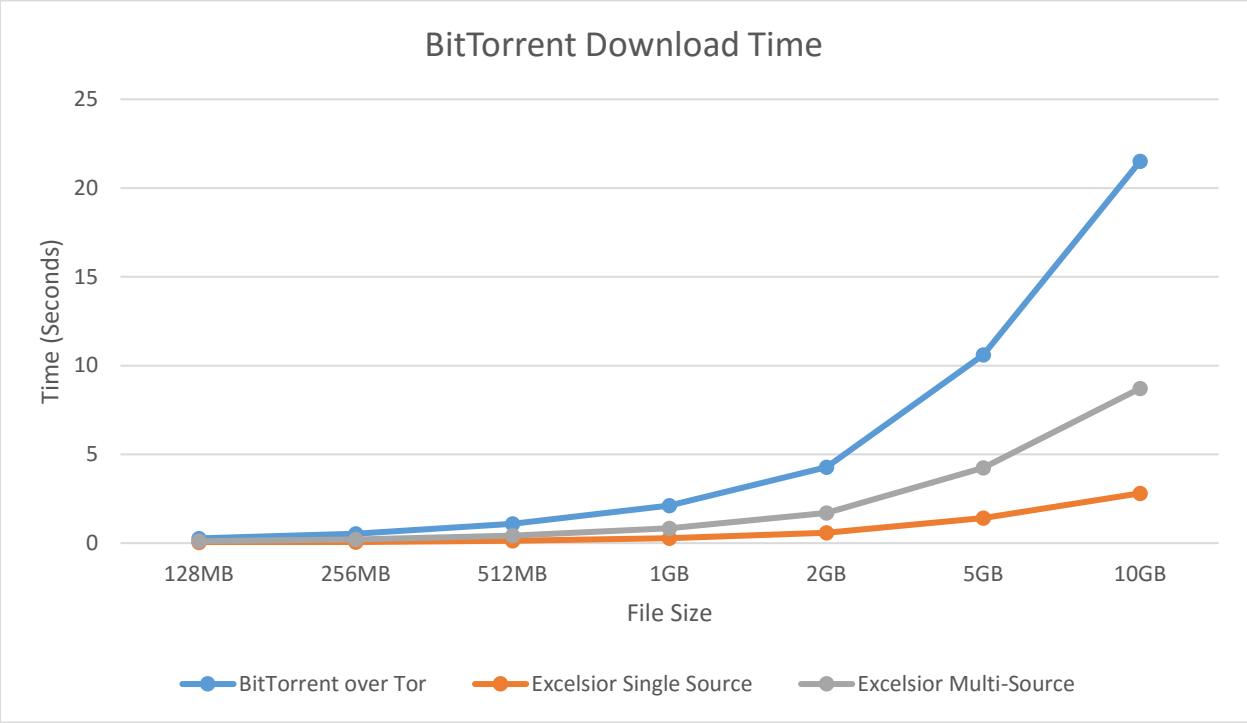

Figure 4.2: BitTorrent Network Download Time

The network overhead chart below, although not as interesting, shows that both solutions produce a fairly a constant percentage of extraneous data across different file sizes. This metric emphasizes the actual percentage of bandwidth not directly used for moving files for each solution, not the overall amount of bandwidth consumed. Excelsior has a clear advantage in overhead since it's not encumbered by the multiple hops of an overlay network.

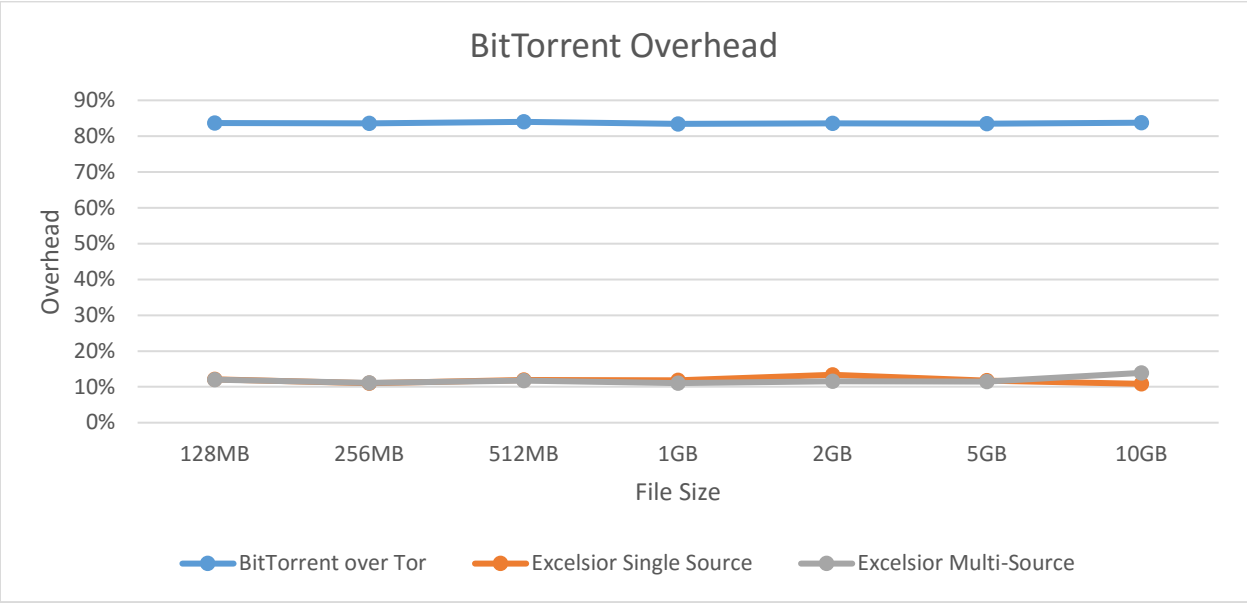

Figure 4.3: BitTorrent Network Overhead

4.6.2 Overall Transfer Time

The overall transfer time analysis is intended to compare metrics across both BitTorrent and DC/overlay networks. While the previous comparison looked only at the final distribution, or the portion of file sharing that involved peers acquiring content, this test also includes the time required to prepare the content. In Excelsior, this involves sharing the file among the initial see group. For this comparison, BitTorrent over Tor has a distinct advantage coming in to the test, as there is no setup time required prior to seeding.

The following two tables show the raw averages from ten individual iterations for bandwidth utilization and transfer time across all three configurations and all seven file sizes.

BitTorrent over Tor							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	83.62%	83.58%	83.99%	83.43%	83.60%	83.50%	83.71%
Seconds	0.268	0.534	1.094	2.116	4.275	10.615	21.515

Table 4.5: BitTorrent-over-Tor Overall Benchmarks

Excelsior Overall Benchmarks (Single Source Mode)							
Speed	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	53.27%	52.69%	53.15%	53.16%	53.84%	53.07%	52.65%
Time	0.261	0.515	1.034	2.094	4.099	10.332	20.851

Table 4.6: Excelsior Overall Single Source Benchmarks

Excelsior Overall Benchmarks (Multi- Source Mode)							
Speed	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Overhead	53.20%	52.73%	53.07%	52.79%	53.01%	52.94%	54.24%
Time	0.774	1.541	3.127	6.342	12.524	31.235	64.021

Table 4.7: Excelsior Overall Single Source Benchmarks

As with the previous charts, plotting the information in a line graph allows one too easily observe trends across the different test configurations. The chart below shows that while transfer times are very similar for smaller files, as file size grows, so does the gap in performance. As expected, Excelsior is at a disadvantage in this comparison, as it includes the "non-peer distribution times".

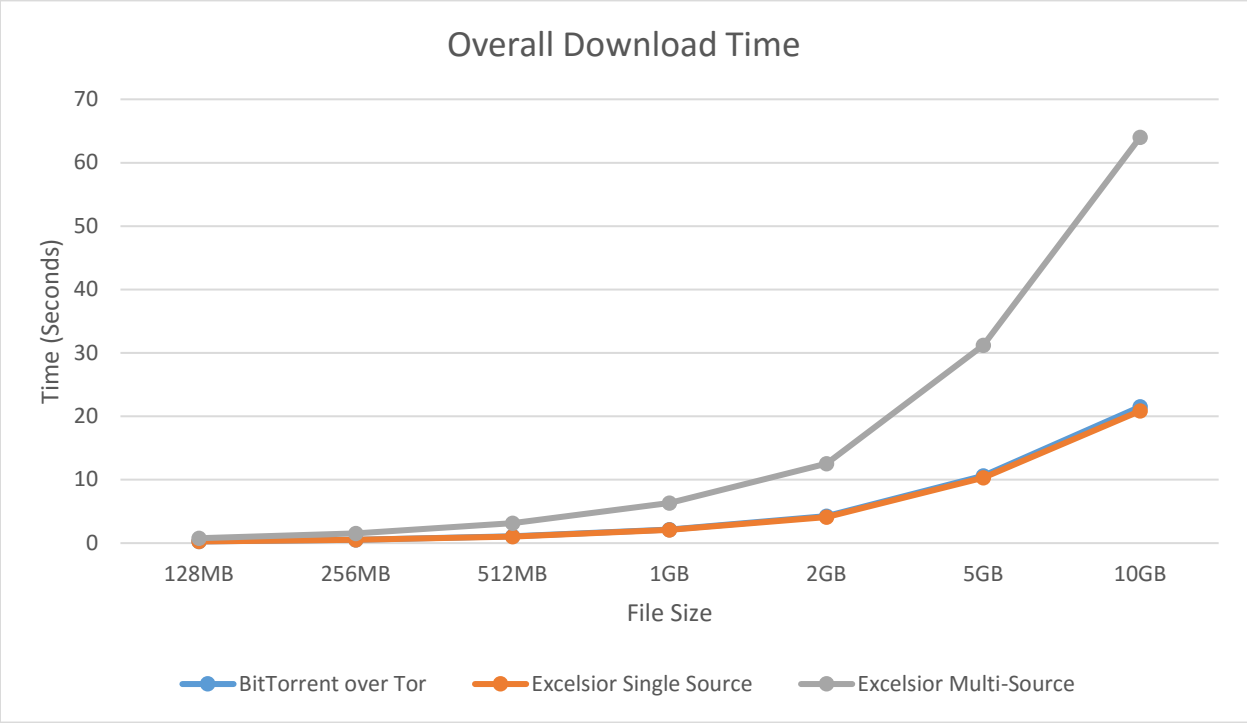

Figure 4.4: Overall Download Time

The network overhead chart shown below remains almost identical to that of the BitTorrent-only comparison. Both solutions produce a fairly constant amount of overhead across all modes of operation. As mentioned previously, this metric emphasizes the actual percentage of bandwidth not directly used for moving files for each solution, not the overall amount of bandwidth consumed.

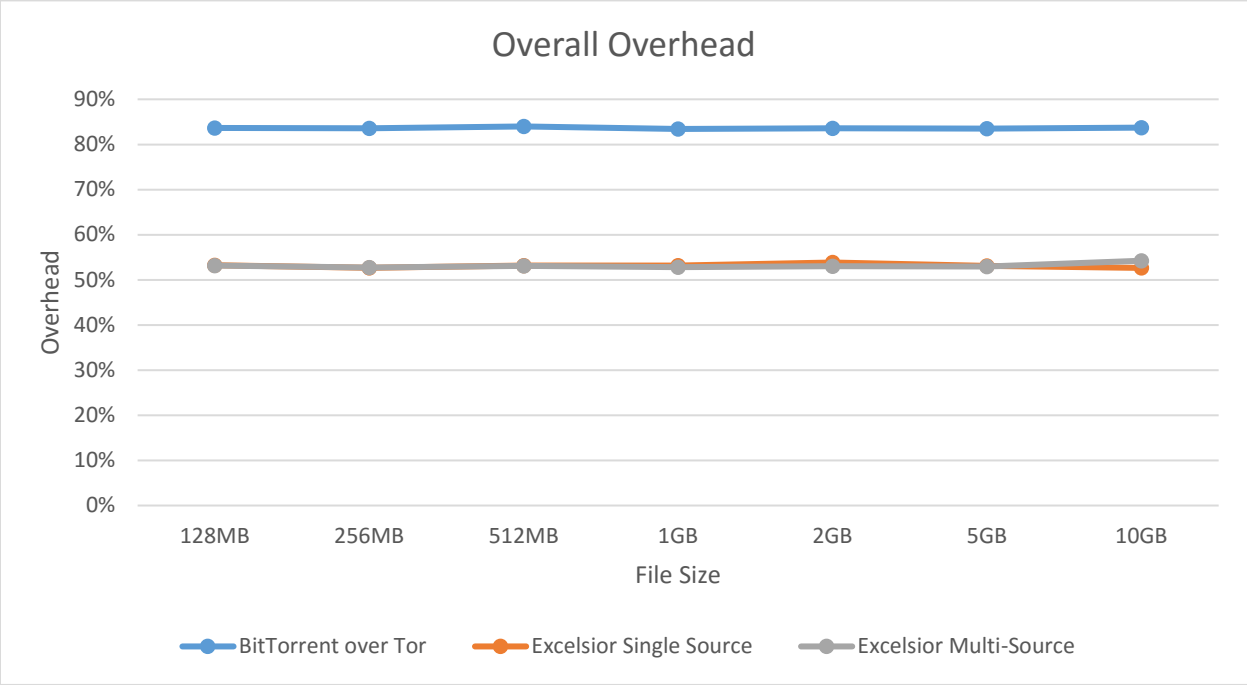

Figure 4.5: Overall Network Overhead

4.6.3 Preparation and Reassembly

One final metric gathered during the testing process was preparation and reassembly time. In Excelsior, preparation time equates to time spent performing file entanglement, while reassemble is the opposite – putting files back together. Although these have little to do with file transfer performance, as they are performed entirely on each host, it is helpful to illustrate that there can be additional overhead present not related to the transfer itself.

Excelsior Preparation & Reassemble (Single Source Mode)							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Preparation	0.618	1.237	2.467	4.905	9.836	24.556	49.108
Reassembly	0.533	1.036	2.139	4.237	8.483	21.175	42.350

Figure 4.8: Preparation and Reassembly Times for Single Source Excelsior

Excelsior Preparation & Reassemble (Multi-Source Mode)							
File Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Preparation	1.843	3.678	7.362	14.707	29.420	73.566	147.107
Reassembly	0.781	1.591	3.174	6.384	12.565	31.633	63.597

Figure 4.9: Preparation and Reassembly Times for Multi-Source Excelsior

Chapter 5: Security Analysis

5.1 Overview and Methodology

Along with performance, security is a fundamental and critical metric by which Excelsior should be evaluated. This security analysis attempts to objectively examine the degree to which Excelsior accomplishes the overall project goal of providing privacy for BitTorrent users, while also evaluating its ability to maintain the same general level of security as the reference system, BitTorrent over Tor.

While reading this analysis, it is important to keep in mind that Excelsior offers two modes that control the level of user privacy provided: single source and multi-source. As mentioned earlier in the paper, in the single-source mode, only seeds obtain privacy via plausible deniability, and there is little additional protection to peers, as there is only one piece of content available in a given swarm. This mode is intended to be used to achieve maximum performance where only the uploader desires or requires privacy.

In multi-source mode, the principle of plausible deniability applies to both seed and peer file transfers, providing privacy for both. Peers still have the option to choose the level of trade-off between anonymity and performance by acquiring a larger or smaller number of blocks. At the smallest number of blocks, there is only a single reassembly solution, negating the effects of the privacy solution. Any level above this, however, there are at least two reassembly solutions, and plausible deniability applies to the peer's acquisition.

5.2 Design Goals

To further clarify the overall project goal of providing privacy to users, Excelsior strives to meet or exceed the following four security-specific goals:

- Offer privacy to users in the form of protection from an internal or external entity associating the user's public identity (such as an IP address), with actions performed against identifiable shared content in the network.
- For all users in multi-source network, provide a level of privacy equivalent to or better than the level of anonymity provided by using BitTorrent over Tor.
- For seeds in a single-source network, provide a level of privacy greater than that of traditional BitTorrent.
- Provide a file sharing network design and client application with similar security to that of BitTorrent and common BitTorrent clients.

5.3 Components

In order to perform an effective analysis, it is helpful to review each of the key components that make up Excelsior in a security context. The complete system can be divided up into the following four functional processes:

- Matchmaking
- DC Nets & Entanglement
- Metadata distribution
- BitTorrent & Reassembly

5.3.1 Matchmaking

At present, automatic matchmaking functionality is not implemented in Excelsior. In principle, this functionality should make it easy for seeds with content to share to

quickly locate other seeds with similarly sized content. Any type of intelligent matchmaking process relies on obtaining certain information to base a pairing off of. A hypothetical matchmaking service for Excelsior might use information such as file size and geographic location to pair up seeds. Until such functionality is implemented internally, an external service can be utilized to provide this functionality, or it can be done manually, as was the case in the testing environment. The initiation of the matchmaking process marks the beginning point at which the privacy protection in Excelsior takes effect.

5.3.2 DC Nets and Entanglement

The DC net exchange described in this paper is based on the classic dining cryptographers problem. This 1988 paper by David Chaum establishes that an exchange based on this problem is unconditionally, or cryptographically secure, depending on whether it is based on one-time use keys, or public keys, respectively. In the reference implementation of Excelsior, public key cryptography is initially used to establish a secure channel between each pair of hosts in the DC net. This secure channel is used to exchange the seed for a cryptographically secure pseudorandom number generator (CSPRNG). The seeded CSPRNG is then used as the source for the DC net secret keys for the remainder of the exchange.

Seeds use the DC net to communicate the contents of their source file with one another, without revealing ownership for any of these files. Once all files have been disseminated to the phase one group, the seeds entangle the files. This process reversibly combines the source files such that a peer downloading an entangled file could conceivably be attempting to obtain any of the constituent files, thus providing plausible deniability with respect to the peer's intentions.

5.3.3 Metadata Distribution

As with traditional BitTorrent, the metadata that allows users to join a swarm is distributed outside the file transfer network. This metadata contains information about both the source content and the resulting entangled files. At present, neither traditional BitTorrent nor Excelsior provide any built-in method by which to facilitate metadata exchange. This process is completed out-of-band through other means, such as DHT or traditional .torrent file exchange.

5.3.4 BitTorrent and Reassembly

Excelsior uses unmodified BitTorrent to allow users to download files from the phase one group of seeds. Once a torrent is brought into phase two, the functionality and security considerations with respect to the transfer itself are virtually identical to that of traditional BitTorrent. Excelsior relies on the plausible deniability of file contents to provide user privacy in the public BitTorrent network. Peers perform the final process, reassembly, after a complete BitTorrent download.

5.4 Security Model

5.4.1 Trust Model

Unlike many client/server applications, where there is a clear division between provider and consumer entities, P2P networks generally treat all participating entities equally, as all entities can fulfill both provider and consumer roles. This is especially true in BitTorrent (and therefore in Excelsior), as the key performance advantages stem from this multi-role benefit. As a result, it is only appropriate to treat all entities participating on the network with the same level of trust. Even in P2P networks, however, there must be some form of control to allow the network to function.

BitTorrent and Excelsior use a set of predefined behavioral rules, along with the information contained in the torrent metadata to exercise control over the network.

The metadata file provides a final ruling on the content that should be exchange in the network. It details the contents of the torrent, as well as hashes that can be used to validate said contents. When combined with a predefined set of behavioral expectations and responses, the network essentially becomes self-governing. BitTorrent and Excelsior both define a protocol for network communications, rules that state that information sent must be valid according to the metadata, and limits on the resources any individual user can consume. If a user fails to abide by any of these rules, they can be banned from participation in the network.

This self-governing nature allows a zero-trust model to exist across both BitTorrent and Excelsior. Every user is responsible for regulating their own actions and to abide by the rules, and every user checks to ensure that all other users with which they interact are abiding by the rules, taking action if they are not. This eliminates the need to establish any specific trust relationships among peers.

5.4.2 Threat and Attack Model

There are three primary threats to operation and security in Excelsior:

- Privacy Compromise
- Content Alteration
- Network Disruption

5.4.2.1 Privacy Compromise

The most critical threat Excelsior faces is a failure of user privacy. The ability to ensure that content remains unidentifiable, that dissemination and acquisition of content remains plausibly deniable, and that a participant's identity cannot be associated with

their activity in the network is a fundamental design goal of the project. A successful exploitation of this threat would occur when:

- A user is identified as uploading an identifiable piece of content
- A user is identified as downloading an identifiable piece of content

There are several ways in which this type of failure could occur:

- Compromised DC Net
- Compromised Metadata Exchange
- Compromised Matchmaking Process
- Failure to download minimum number of pieces

The first item, a compromised DC net, is the most obvious attack against Excelsior. As the DC net is used by initial seeds to exchange source content, any attack allowing the source of the content to be revealed would identify a user as uploading identifiable content, thus violating the user's privacy. Given the configuration of the reference implementation, the underlying dining cryptographers problem is still unconditionally secure, while the overall security of the DC net is dependent on the security of both the public key cryptography (PKC) used to establish the secure channels and the security of the CSPRNG itself, making it cryptographically secure (assuming appropriate selections/implementation for both PKC and CSPRNG). De-anonymizing information exchanged over the DC net would therefore only be possible if the PKC channel or CSPRNG were broken or otherwise compromised.

The only significant threat to the anonymity provided by the DC net is collusion. If a substantial portion of the DC net members cooperated to pool their keys, they would be able to de-anonymize traffic from other nodes in the network. The number of colluding nodes required varies depending on the configuration of the DC net, but in general, for a

full mesh, a collusion requires n-1 nodes. For less than a full mesh, a collusion must isolate all connections between any two nodes or groups of nodes.

The ease of performing a collusion attack thus relies on the number of nodes comprising the DC net, as well as the configuration of the net. As is common in this space, there exists a performance-anonymity trade off in these parameters – more nodes per group takes longer to initially seed, but reduces the chances of a successful collusion. Likewise, a full mesh network reduces the available bandwidth, but requires a greater percentage of nodes for a successful collusion.

A compromise of the metadata exchange is another route by which user privacy could be breached. As mentioned earlier, metadata files identify a specific piece of content, and contain information on both the source and entangled files. As each seed only publishes metadata for the files it initially contributed, and peers only acquire metadata for content they wish to download, observing either entity interact with metadata can effectively reveal their intentions. While this exchange is outside the scope of the protections provided by Excelsior, it is important to recognize that both seeds and peers must take steps to prevent details of a specific metadata exchange from being linked back to them.

Additionally, while the metadata lists all of the seeds involved in the exchange, it only references entangled blocks held by each. Since no seed holds the original file blocks, and descriptive metadata is only distributed after the exchange, there is no way for any seed except for the publisher to know the details of the content they are holding beforehand. It remains possible for other members of the phase one group to search through published metadata in an attempt to locate any which references blocks they hold, so as to determine what the content of those blocks is. Although this is always

possible assuming the metadata has been published publicly, it does not reveal which seed introduced the content, rather only serves to remove the element of plausible deniability on the part of that host, as they are now aware of the content they hold.

The third item, a compromised matchmaking process, is yet another route by which privacy could theoretically be compromised. Exploiting this weakness would require an entity to use information provided during the matchmaking process to determine which piece of content was introduced by which seed. Although difficult to do perform if the information provided is sufficient generic, specific details, such as a very granular file size, may allow for a member of the initial seed group to use process of elimination to determine which seed introduced which content. Although the matchmaking process is not presently implemented in Excelsior, it is important to consider when providing matchmaking information to a third-party service. In any event, this threat exclusively affects seeds.

The fourth item, failure to download the minimum number of pieces, is a threat affecting only peers. The plausible deniability provided by entanglement relies on the ability of the downloaded content to be reconstructed to multiple files. In a simple network with three pieces of content, there are four resulting files. A peer must only download two of these pieces to successfully reconstruct the original file, however three pieces are required for the peer to reconstruct multiple files, thus providing plausible deniability. Successfully exploiting this threat would require a peer intentionally overriding the default download settings to select fewer files.

5.4.2.2: Content Alteration

Content alteration is a threat affecting the integrity of the files shared through Excelsior. In proper operation, all of the source files introduced by the initial seed group

can be successfully reconstructed to their original, unaltered form by all peers. A successful content alteration attack would occur when a file reconstructed by a peer is not identical to the corresponding source file introduced by a seed. An attack such as this might be executed either to introduce a malicious payload, or simply to disrupt the normal functionality of the network. There are two ways in which this type of failure could occur:

- Content Alteration by a User
- Metadata Alteration by a Seed

As with traditional BitTorrent, any user in an Excelsior network has direct access to the files they host. This gives any user the opportunity to alter the content of those files freely. Although this can be a problem if not checked, standard BitTorrent provides a checksum mechanism which validates all data received using hashed contained in the metadata file. Seeds or peers which consistently server content failing the validation checks can be banned from the network.

The type of validation method described above works based on the assumption that the metadata files have not been tampered with. In traditional BitTorrent, only a single seed initially hosts files, and that seed is the only entity capable of publishing metadata for that file. In Excelsior, however, initial content is hosted by multiple seeds, which allows any seed to generate a metadata file for any piece of content. This opens the door for an attack in which one seed alters the content of the files it hosts, and publishes false metadata for that content. Although this attack successfully allows for content alteration, it is functionally the same as a single seed publishing altered or malicious content in a traditional BitTorrent environment.

One final type of attack exists using content that has been altered. Although this attack does not alter content within the network, it is worth mentioning as a potential threat to network users. As with traditional BitTorrent, a seed can publish any type of content to the network, including content with malicious payloads. In traditional BitTorrent, the contents of all files in a given swarm are apparent, making it relatively easy for users to identify atypical or unwanted content.

In Excelsior, however, peers only know about the swarm content for which they have obtained metadata. Since peers can reconstruct content other than that for which they have obtained metadata, a "curious" peer may reconstruct the other source files in addition to the one for which it has obtained metadata. As the peer has no information regarding what the contents of these other files are, they may be subject to a false sense of security regarding its nature.

5.4.2.3 Network Disruption

Network disruption is a threat which prevents the proper operation of the network. This is similar to a denial of service attack, in that the primary glad is to prevent others from using the network or other services which depend on it. Network disruptions can occur in both DC net and BitTorrent network. There are two ways in which this type of failure could occur:

- DC Net Disruption by Colluding Peers
- BitTorrent Network Disruption

Disruption of the DC net is the easiest network disruption attack to perpetrate against Excelsior. DC nets are vulnerable to a phenomenon known as collisions, which can occur when an even number of nodes attempts to reply to a group. Using collisions, a malicious user can "jam" a DC net, preventing communication from occurring. In

Excelsior's implementation, where control is carried out using an out-of-band channel, the effect would be to cause an incorrect value in the transferred data. There are a variety of methods to detect and mitigate collisions, with various tradeoffs in bandwidth. In any event, the potential damage is limited only to a temporary disruption in the DC net and a need to restart the transfer, as file hashes are verified at every stage to detect accidental or malicious corruption

Network disruption could also occur in the BitTorrent network. As Excelsior uses an unmodified version of the protocol, any attacks which work against traditional BitTorrent would also be effective against Excelsior. As with a disruption of the DC net, this effect would only be temporary. Any errors in data transmission would be detected by the hashing process and those pieces re-downloaded.

Although not network disruption per-say, it is worth mentioning that the matchmaking and metadata exchange processes could be vulnerable to disruption as well. As these processes are not provided by Excelsior, and therefor rely on an external service to complete, their vulnerability to implementation would depend highly on how the security and redundancy measures in place on the external service.

5.4.3 Adversaries

The threat model earlier in this section describes a number of theoretical attacks on Excelsior. Adversaries who would most likely be perpetrating these attacks fall into one of two categories:

- State-sponsored Agencies
- Commercial Sector Agencies

A state-sponsored entity is an umbrella term for any organization working with, or on behalf of government interests. Common examples of state-sponsored agencies include

intelligence services and police organizations. There are multiple reason why these agencies would have interests in undermining the privacy in Excelsior. As one example, police or security agencies may wish to investigate illegal activity being conducted though P2P file sharing. An alternate example would be intelligence services working for a corrupt government regime attempting to identify dissidents base on their internet activity.

The category of commercial sector agencies primarily includes organizations with financial interests in content being distributed over Excelsior. These are commonly art and media industry trade groups, such as the Motion Picture Association of America (MPAA) or Recording Industry Association of America (RIAA). Groups such as these work to monitor and report copyright and trademark infringement on behalf of other organization in their respective industries. The user privacy present in Excelsior would prevent groups such as these from definitively linking infringing activity with a public IP address, thus removing their ability to combat such infringement.

Chapter 6: Conclusions

6.1: Overview

This research was intended to be an initial examination of a different way of approaching user privacy in P2P file sharing networks. Although Excelsior is hardly the first system to utilize DC nets and content entanglement to provide privacy, it is the first to incorporate these concepts into an existing, mainstream system like BitTorrent. The primary function of exploring this topic was to demonstrate that the basic concept of content anonymity paired with BitTorrent was indeed a feasible endeavor

6.2 Performance

The performance analysis in chapter four describes two specific performance goals for Excelsior:

- Provide an overall transfer time (combined DC net and BitTorrent metrics) that is similar or less than that of BitTorrent over Tor.
- Provide a second phase (BitTorrent only) transfer time that is less than that of BitTorrent over Tor.

As shown in the performance analysis, the second phase transfer times in every test of Excelsior far outperformed BitTorrent over Tor, showing a clear success for this goal. The overall transfer time results for Excelsior, although slightly longer than BitTorrent over Tor when sharing larger files, were still well within the definition of similar.

With regard to performance, it's important to remember that the phase one process in Excelsior, which was responsible for a significant portion of the time consumed in the overall test, represents a one-time process that occurs only when a swarm is first

created, and does not actually involve any peer transfer. Although this long initial setup time detract from overall performance, all subsequent data transfers to peers complete significantly faster than that of BitTorrent over Tor. As such, it seems fitting to describe Excelsior as having successfully met the goals.

6.3: Security

There are two key types of security metrics that have been discussed throughout this paper: those outlined in the security analysis chapter, which detail the level of privacy required to be present for both modes of operation, and the requirement that Excelsior maintain the same general level of security as the reference system, BitTorrent over Tor. From the discussion in the security analysis, it should be relatively clear that, through the design itself, the former goals are achieved. With regard to the latter item, maintaining the same level of security as BitTorrent over Tor, the following observations can be made:

- Both Excelsior and BitTorrent over Tor rely on the existing BitTorrent protocol as a transport mechanism, and are therefore subject to potential disruption from any attack which is effective against BitTorrent
- Both Excelsior and BitTorrent over Tor rely on a second network to provide an additional stage of transport. Excelsior relies on DC nets, while Tor relies on an anonymizing overlay network. Both systems are only as secure as the least secure network.
- Both Excelsior and Tor are subject to collusion attacks. In both DC nets and overlay networks, any entity which controls a sufficiently large number of nodes can circumvent the privacy or anonymity measures in place.

- Both Excelsior and BitTorrent over Tor allow for an adjustable level of privacy. Excelsior provides single source and multi-source modes to provide users with a choice as to where privacy is required. Tor allows the user to adjust the minimum number of hops for each connection. Both of these adjustments adjust a privacy/anonymity to performance balance.

From a security perspective, Excelsior accomplishes all of goals defined in this project, and comes out relatively on par with the reference system, BitTorrent over Tor. A user of this system can expect to enjoy a resulting level of privacy that is sufficient for most tasks online, and which in similar to the level of anonymity provided by BitTorrent over Tor.

6.4 Final Thoughts

As demonstrate in the performance and security analyses, Excelsior lives up to the system goals defined throughout this paper:

- Provides user privacy in the form of plausible deniability for all network participants.
- Provides a decrease in the time required for a peer to privately acquire a file share in the network as compared to BitTorrent over Tor.
- Utilize the existing BitTorrent protocol as the final method of delivery, thus taking advantage of the substantial existing installation base.
- Allow an end-user customizable trade-off between performance and anonymity.

6.5 Future Work

As with any research project, due to time constraints and the scope of the topic, there were, of course, a number of aspects which were not fully explored. These items, all of

which would be excellent candidates for future research in this area, fall into three key categories:

- Functionality
    - Integrated Matchmaking System
    - Integrated Metadata Exchange
- Networking
    - Larger DC Nets
    - Partial Mesh DC Nets
- Real-world Benchmarks

The first category, functionality, is comprised of two processes essential to the operation of this and other P2P file transfer systems, but which were not critical in designing or evaluating Excelsior. The first process is an integrated system for matchmaking among seeds in phase one. This system would allow for functionality that utilizes DHT or a peer exchange-like process (PEX) to automatically find other seeds and build DC nets. Although seed group were successfully assigned manually in the testing environment, any production software targeted at casual users would all but demand it exists.

The second is an integrated system to allow for metadata to be published and retrieved automatically. This would also likely utilize DHT in order to make metadata accessible via a simple DHT URI. Controlling this aspect of the metadata exchange would also allow for better protection against privacy breaches through improper metadata exchange than are currently possible.

In the networking category, the ability to support DC nets with larger than three seeds was of particular interest. Although theoretically possible, designing the application to scale to an arbitrary DC net size proved to be too time consuming of an effort at this

juncture. The primary benefit of larger DC nets is improved resistance against collusion. In a full mesh DC net, a collusion of all but one node is necessary to break the net's privacy. In a larger net size, the number of colluding seeds required also increases, making the attack more difficult to execute.

Partial-mesh DC nets go hand-in-hand with nets of larger size. Although a full mesh net can theoretically support any number of seeds, there would be a significant performance impact on very large full mesh nets due to the need to send a complete file copy to every connected seed. Partial-mesh nets, although complex to implement, would allow solve this issue.

Finally, one additional area that would benefit from additional research is the inclusion of real world benchmarks. All of the test results described in this paper were generate in a lab environment. Although consistent and reproducible, this environment does not necessarily accurately simulate all of the possible variables that can be encountered when running a P2P network over the public internet. For the purposes of this research, simply determining that the system was generally "in the ballpark" faster than BitTorrent over Tor was sufficient to validate its usefulness, however formal benchmarks for all aspects would certainly be preferable.

Appendix: Performance Benchmarks

BitTorrent over Tor							
Speed	**128**	**256**	**512**	**1024**	**2048**	**5120**	**10240**
Effective Bandwidth	3435.6	3558.45	3160.85	3563.35	3463.95	3393.95	3293.15
	3298.75	3292.1	3224.55	3225.6	3246.6	3416.7	3291.75
	3264.8	3530.8	3362.45	3512.6	3477.25	3367.7	3254.3
	3503.5	3326.75	3406.55	3242.75	3246.25	3414.25	3152.45
	3188.5	3255.7	3435.25	3486	3502.45	3206	3378.9
	3374.35	3168.2	3165.4	3375.75	3184.3	3417.05	3195.15
	3554.25	3497.55	3156.3	3478.3	3203.55	3368.75	3527.65
	3224.9	3390.8	3247.3	3500.35	3482.85	3515.4	3557.75
	3298.05	3209.15	3189.9	3181.5	3218.6	3235.05	3455.55
	3395.7	3396.05	3443.3	3367.35	3568.95	3454.5	3260.25
Time	0.261	0.504	1.134	2.012	4.139	10.56	21.766
	0.272	0.544	1.111	2.222	4.416	10.49	21.776
	0.274	0.508	1.066	2.041	4.123	10.642	22.026
	0.256	0.539	1.052	2.21	4.416	10.497	22.738
	0.281	0.55	1.043	2.056	4.093	11.179	21.214
	0.266	0.566	1.132	2.123	4.502	10.489	22.434
	0.252	0.512	1.136	2.061	4.475	10.639	20.319
	0.278	0.528	1.104	2.048	4.116	10.195	20.148
	0.272	0.558	1.124	2.253	4.454	11.079	20.743
	0.264	0.528	1.041	2.129	4.017	10.375	21.986

Table A.1: BitTorrent over Tor

Single Source DC Net							
Speed	**128**	**256**	**512**	**1024**	**2048**	**5120**	**10240**
Effective Bandwidth	1132.76	1146.41	1095.59	1114.47	1090.36	1198.19	1174.44
	1130.38	1174.44	1178.12	1208.4	1161.38	1201.16	1167.79
	1155.91	1200.56	1175.27	1121.71	1186.79	1196.53	1116.25
	1142.14	1162.92	1201.75	1145.58	1184.53	1164.7	1113.04
	1167.43	1191.66	1173.01	1085.26	1187.98	1103.9	1078.37
	1152.83	1104.14	1131.33	1135.73	1169.09	1172.42	1109.48
	1137.98	1167.91	1145.94	1075.99	1208.16	1175.98	1202.11
	1116.49	1121.95	1108.41	1198.78	1151.16	1105.44	1094.52
	1181.92	1119.22	1169.69	1170.76	1179.43	1102	1174.32
	1077.3	1125.75	1106.75	1078.25	1123.85	1083.59	1132.88
Time	0.226	0.447	0.935	1.838	3.757	8.546	17.438
	0.226	0.436	0.869	1.695	3.527	8.525	17.537
	0.221	0.426	0.871	1.826	3.451	8.558	18.347
	0.224	0.44	0.852	1.788	3.458	8.792	18.4
	0.219	0.43	0.873	1.887	3.448	9.276	18.992
	0.222	0.464	0.905	1.803	3.504	8.734	18.459
	0.225	0.438	0.894	1.903	3.39	8.708	17.037
	0.229	0.456	0.924	1.708	3.558	9.263	18.711
	0.217	0.457	0.875	1.749	3.473	9.292	17.44
	0.238	0.455	0.925	1.899	3.645	9.45	18.078

Table A.2: Single Source DC Net

Multi-Source DC Net							
Speed	**128**	**256**	**512**	**1024**	**2048**	**5120**	**10240**
Effective Bandwidth	1102.83	1165.65	1113.99	1133.71	1192.13	1166.36	1139.29
	1087.63	1140.36	1134.06	1168.26	1185.48	1105.8	1193.44
	1206.98	1150.21	1080.74	1104.85	1161.14	1127.29	1084.19
	1196.88	1142.61	1150.09	1109.72	1112.57	1191.66	1068.87
	1125.75	1094.64	1138.22	1121.12	1093.57	1190.59	1099.74
	1204.24	1175.39	1150.33	1073.5	1153.89	1164.7	1118.74
	1206.86	1180.26	1129.55	1137.86	1098.32	1084.54	1072.91
	1168.62	1143.21	1120.41	1079.44	1083.12	1143.33	1080.63
	1101.41	1163.51	1169.21	1080.63	1123.14	1076.59	1091.91
	1129.91	1197.12	1189.4	1179.07	1162.09	1142.26	1175.39
Time	0.696	1.318	2.758	5.419	10.308	26.338	53.928
	0.706	1.347	2.709	5.259	10.365	27.781	51.482
	0.636	1.335	2.842	5.561	10.583	27.251	56.669
	0.642	1.344	2.671	5.537	11.045	25.779	57.481
	0.682	1.403	2.699	5.48	11.237	25.802	55.868
	0.638	1.307	2.671	5.723	10.649	26.376	54.919
	0.636	1.301	2.72	5.4	11.188	28.325	57.265
	0.657	1.344	2.742	5.692	11.345	26.869	56.856
	0.697	1.32	2.627	5.686	10.941	28.535	56.269
	0.68	1.283	2.583	5.211	10.574	26.894	52.272

Table A.3: Multi-Source DC Net

Single Source BitTorrent							
Speed	**128**	**256**	**512**	**1024**	**2048**	**5120**	**10240**
Effective Bandwidth	19023.75	19006.88	18729.38	17330.63	17145	18470.63	19110
	17360.63	19033.13	17358.75	19042.5	16876.88	18729.38	18916.88
	18631.88	17823.75	17602.5	17325	18971.25	18256.88	17730
	18041.25	17323.13	16899.38	17673.75	19068.75	18581.25	18073.13
	18903.75	18900	18806.25	19057.5	17566.88	17068.13	18834.38
	16931.25	18988.13	17184.38	18562.5	16929.38	18875.63	17681.25
	16972.5	16876.88	18157.5	17430	17821.88	18301.88	17422.5
	18611.25	18001.88	17998.13	17075.63	17341.88	17925	17673.75
	17868.75	18313.13	18939.38	18226.88	17835	17364.38	18350.63
	17660.63	18007.5	18723.75	18787.5	17863.13	17131.88	18802.5
Time	0.034	0.067	0.137	0.295	0.597	1.386	2.679
	0.037	0.067	0.147	0.269	0.607	1.367	2.707
	0.034	0.072	0.145	0.296	0.54	1.402	2.888
	0.035	0.074	0.151	0.29	0.537	1.378	2.833
	0.034	0.068	0.136	0.269	0.583	1.5	2.718
	0.038	0.067	0.149	0.276	0.605	1.356	2.896
	0.038	0.076	0.141	0.294	0.575	1.399	2.939
	0.034	0.071	0.142	0.3	0.59	1.428	2.897
	0.036	0.07	0.135	0.281	0.574	1.474	2.79
	0.036	0.071	0.137	0.273	0.573	1.494	2.723

Table A.4: Single Source BitTorrent

Multi-Source BitTorrent							
Speed	**128**	**256**	**512**	**1024**	**2048**	**5120**	**10240**
Effective Bandwidth	17148.75	17737.5	18410.63	17026.88	18665.63	17705.63	17658.75
	19053.75	18275.63	18339.38	18650.63	18815.63	18588.75	17068.13
	18399.38	18532.5	17375.63	17055	18900	17521.88	17131.88
	17268.75	19123.13	17767.5	17169.38	17257.5	19074.38	17490
	19057.5	17821.88	17315.63	18796.88	18382.5	18751.88	18350.63
	18843.75	17655	18476.25	18399.38	18371.25	18101.25	17190
	18973.13	18294.38	17006.25	19033.13	16890	19076.25	18770.63
	17261.25	18714.38	18819.38	19003.13	19098.75	17986.88	17585.63
	17281.88	17638.13	18525	19021.88	16936.88	17332.5	18172.5
	16884.38	18260.63	18804.38	18026.25	17767.5	17242.5	16908.75
Time	0.112	0.216	0.417	0.902	1.646	4.338	8.698
	0.101	0.21	0.419	0.824	1.633	4.132	8.999
	0.104	0.207	0.442	0.901	1.625	4.383	8.966
	0.111	0.201	0.432	0.895	1.78	4.026	8.782
	0.101	0.215	0.444	0.817	1.671	4.096	8.37
	0.102	0.218	0.416	0.835	1.672	4.243	8.935
	0.101	0.21	0.452	0.807	1.819	4.026	8.183
	0.111	0.205	0.408	0.808	1.608	4.27	8.734
	0.111	0.218	0.415	0.807	1.814	4.431	8.452
	0.114	0.21	0.408	0.852	1.729	4.454	9.084

Table A.5: Multi-Source BitTorrent

Preparation and Reassembly							
Size	**128MB**	**256MB**	**512MB**	**1GB**	**2GB**	**5GB**	**10GB**
Single Seed Preparation	0.587	1.319	2.438	5.096	9.762	25.612	46.077
	0.600	1.204	2.359	4.664	10.351	23.491	46.912
	0.640	1.249	2.448	5.116	10.223	23.029	47.989
	0.610	1.229	2.558	4.594	10.243	24.381	48.742
	0.592	1.205	2.610	4.876	9.832	23.853	51.585
	0.655	1.131	2.556	5.121	10.166	25.773	48.922
	0.644	1.268	2.332	5.113	9.187	23.999	50.759
	0.650	1.264	2.377	4.799	9.567	24.521	51.203
	0.627	1.189	2.568	4.808	9.626	25.344	51.133
	0.575	1.310	2.419	4.866	9.398	25.552	47.757
Single Seed Reassembly	0.574	1.039	2.002	4.328	8.887	20.878	41.776
	0.565	0.957	2.118	4.463	8.716	20.964	43.814
	0.548	1.086	2.252	4.355	8.771	21.777	43.854
	0.533	1.056	2.169	4.317	8.265	20.088	40.236
	0.525	1.005	2.003	4.092	8.705	19.750	41.124
	0.483	0.986	2.204	4.014	8.731	21.927	41.957
	0.484	1.077	2.223	4.210	8.433	22.117	39.461
	0.563	0.998	2.120	3.956	8.047	21.781	43.564
	0.518	1.089	2.203	4.217	7.872	20.522	44.233
	0.532	1.066	2.096	4.415	8.399	21.947	43.482
Multi-Seed Preparation	1.800	3.642	7.761	15.472	27.632	76.775	146.855
	1.912	3.598	7.599	14.110	28.185	70.443	146.256
	1.939	3.531	7.705	14.719	30.407	73.443	152.217
	1.769	3.825	7.685	15.329	29.287	73.173	143.976
	1.746	3.518	7.384	14.589	30.718	76.093	153.480
	1.915	3.797	7.061	14.063	28.757	77.378	141.082
	1.959	3.677	7.207	15.329	28.190	76.765	138.072
	1.819	3.890	6.906	14.404	30.698	71.998	153.500
	1.754	3.807	7.284	15.229	29.397	70.521	140.795
	1.816	3.493	7.025	13.821	30.933	69.066	154.836
Multi-Seed Reassembly	0.714	1.539	3.279	6.139	12.091	33.135	66.340
	0.805	1.557	3.125	6.568	13.166	30.217	62.664
	0.784	1.522	3.000	5.939	13.146	31.317	60.384
	0.791	1.627	3.124	6.542	12.357	32.871	65.812
	0.782	1.635	3.329	6.055	12.590	29.585	61.647
	0.780	1.500	3.048	6.229	13.260	32.591	65.762
	0.809	1.685	3.128	6.832	11.708	31.435	59.231
	0.780	1.651	3.074	6.305	11.798	30.863	65.923
	0.802	1.629	3.301	6.598	12.519	31.401	62.971
	0.767	1.564	3.327	6.635	13.015	32.911	65.233

Table A.6: Preparation and Reassembly

Bibliography

[1] Arthur, D., & Panigrahy, R. (2006). Analyzing the Efficiency of BitTorrent and Related Peer-to-Peer Networks. *SODA*.

[2] Battula, B. P., & Jaideep, G. (2016). Survey on the Present State-of-the-Art of P2P Networks, Their Security Issues and Counter Measures. *International Journal of Applied Engineering Research*, 616-620.

[3] Chaum, D. L. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 89-90.

[4] Cisco Systems. (2017). *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021.* Cisco Systems.

[5] Comb, M., & Watters, P. A. (2016). Peeking Behind the Great Firewall: Privacy on Chinese File Sharing Networks. *Privacy, Security and Trust (PST), 2016 14th Annual Conference on* (pp. 650-656). IEEE.

[6] Dingledine, R., & Murdoch, S. J. (2009). Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it. *Online: http://www.torproject.org/press/presskit/2009-03-11-performance.pdf.*

[7] Dingledine, R., Mathewson, N., & Syverson, P. (2004). *Tor: The Second-Generation Onion Router.* Naval Research Lab.

[8] Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P., Al Hamra, A., & Garces-Erice, L. (2004). Dissecting BitTorrent: Five Months in a Torrent's Lifetime. *Passive and Active Network Measurement*, 1-11.

[9] Johnsen, J. A., Karlsen, L. E., & Birkeland, S. S. (2005). Peer-to-peer networking with BitTorrent. *Department of Telematics, NTNU.*

[10] Kiraly, C., Bianchi, G., & Cigno, R. L. (2008). *Solving Performance Issues in Anonymization Overlays with a L3 approach.* University of Toronto.

[11] Le Blond, S., Choffnes, D., Zhou, W., Druschel, P., Ballani, H., & Francis, P. (2013). Towards Efficient Traffic-analysis Resistant Anonymity Networks. *ACM SIGCOMM Computer Communication Review* (pp. 303-314). ACM.

[12] Le Blond, S., Legout, A., Lefessant, F., Dabbous, W., & Kaafar, A. M. (2010). Spying the World from your Laptop. *LEET'10.*

[13] Le Blond, S., Manils, P., Chaabane, A., Kaafar, C. A., & Legout, A. (2010). De-anonymizing BitTorrent Users on Tor. *7th USENIX Symposium on Network Design and Implementation (NSDI '10).*

[14] Murdoch, S. J., & Watson, M. N. (2008). Metrics for Security and Performance in Low-Latency Anonymity Systems. *Privacy Enhancing Technologies Worshop.*

[15] Nielson, S. J., & Wallach, D. S. (2011). The BitTorrent Anonymity Marketplace. *arXiv preprint arXiv:1108.2718.*

[16] Palo Alto Networks. (2015). *Application Usage and Threat Report.* Palo Alto Networks.

[17] Pouwelse, J. A., Garbacki, P., Epema, D., & Sips, H. J. (2004). *A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System.* Delft University of Technology.

[18] Saroiu, S., Gummadi, K. P., & Gribble, S. D. (2002). A Measurement Study of Peer-to-Peer File Sharing Systems. *Proceedings of Multimedia Computing and Networking*, (p. 152).

[19] Schollmeier, R. (2001). A Definition of Peer-to-Peer Networking for the Classification of Peer-to-. *Proceedings of the First International Conference on Peer-to-Peer Computing* (pp. 101-102). IEEE.

[20] Sen, S., & Wang, J. (2004). Analyzing Peer-To-Peer Traffic Analyzing Peer-To-Peer Traffic. *IEEE/ACM Transactions on Networking (ToN)*, 219-232.

[21] Sirer, E. G., Goel, S., Robson, M., & Engin, D. (2004). Eluding Carnivores: File Sharing with Strong Anonymity. *Proceedings of the 11th workshop on ACM SIGOPS European workshop* (p. 19). ACM.

[22] Snader, R., & Borisov, N. (2008). A Tune-up for Tor: Improving Security and Performance in the Tor Network.

[23] Steinmetz, R., & Wehrle, K. (2005). *Peer-to-Peer Systems and Applications.* Springer.

[24] Stubblefield, A., & Wallach, D. S. (2001). Dagster: Censorship-Resistant Publishing Without. *Rice University, Technical Report TR0l-380.*

[25] Waidner, M., & Pfitzmann, B. (1989). The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. *Advances in Cryptology*, 690.

[26] Waldman, M., & Mazieres, D. (2001). Tangler: A Censorship-Resistant Publishing System Based. *Proceedings of the 8th ACM conference on Computer and Communications Security* (pp. 126-135). ACM.

[27] Waldman, M., Rubin, A. D., & Cranor, L. F. (2000). Publius: A Robust, Tamper-Evident Censorship-Resistant Web Publishing System. *9th USENIX Security Symposium*, (pp. 59-72).

[28] Wendolsky, R., Herrmann, D., & Federrath, H. (2007). Performance Comparison of Low-Latency Anonymisation Services from a User Perspective. *PET*.

[29] Wolchok, S., & Halderman, A. J. (2010). Crawling BitTorrent DHTs for Fun and Profit. *WOOT*.

[30] Wu, G., & Chiueh, T.-c. (2006). How Efficient is BitTorrent? *Electronic Imaging* (p. 60710). International Society for Optics and Photonics.

[31] Yanet, M., & Yasinsac, ,. A. (2001). Policies to Enhance Computer and Network Forensics. *Proceedings of the 2001 IEEE* (pp. 289-295). IEEE.