Measuring Performance Effect of Tor Bandwidth Publishing Intervals

Evangelos Dimitriou

MInf Project Part 1
School of Informatics
University of Edinburgh

2020

Abstract

This paper consists of an investigation on the Tor network. More precisely, it explores the effects of shorter consensus publishing intervals on the divergence between client-known relay-bandwidth information, and their real-time levels. Because of the relay-selection algorithm Tor uses, differences between the two can hurt user performance through the network and decrease throughput precision. This research consists of collecting live bandwidth data from Tor relays and using analysis techniques, in order to infer how accurately the information is captured by different sampling periods. Continuing, the results, along with other factors, must be evaluated in terms of user-perceived performance.

Tor uses a 60-minute bandwidth publishing interval, and the results of this dissertation show that to significantly decrease the information error present in Tor clients, the interval would have to be decreased below 4 minutes. Measuring all relays in the network, at such high frequency, is a difficult task to perform due to the infrastructure available. However, since many aspects of Tor functionality depend on its publishing interval, the effects towards user performance are not straight forward and further evaluation is required to draw definite conclusions.

Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the work that will be performed throughout this Masters thesis. Firstly, we will introduce the context in which this research takes place, and then, we will go on to describe why any findings along the way would prove useful in today's world. This is a key part of this dissertation since its motivation is based on needs stemming from the ever-growing digitization of our society.

Continuing, we will clearly state the objectives this paper aims to meet and give a brief description of the methodologies used for collecting the required data. Lastly, we will provide a layout of the paper's structure and an overview of each chapter.

1.1 Context

This paper involves research in the field of computer science, which includes numerous possible study branches such as Artificial Intelligence, Cryptography, Software engineering etc. Here, we are focusing on the study of computer networks. A network can be defined as a group of two or more computer systems linked together, with the biggest one in existence being the internet. Its goal is to seamlessly transfer information between two hosts and it enables many (if not most) of our everyday activities, namely: web browsing, instant chatting, social media and numerous others.

This dissertation is focusing on the Tor network, a special type of system which allows its users to remain anonymous when performing online transactions. Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays, to conceal a client's location and usage from anyone conducting network surveillance or traffic analysis. However, due to its architecture, users of the network experience significant delays in their traffic when compared to standard direct connections, discouraging its use.

There have been several previous research papers aiming to improve the performance and speed of Tor and we will talk about some of them in chapter 3. In this paper, we will explore how random user traffic affects the network state and see how this relates to Tor's operation.

1.2 Motivation

Over the years, since its launch in 1983, the internet has become a central focus for many institutions, including large and small companies, universities and government agencies. An increasing number of people rely on the internet for professional, social and many other activities, therefore placing their trust on online platforms to keep their personal data safe. However, when the biggest computer network was being created, security was not as fundamental a priority as speed and reliability, leading throughout the years to numerous types of attacks performed by malicious users aiming to steal sensitive data, monitor transactions, etc. Such incidents gave rise to the field of Network Security [1].

When two computer hosts perform an online transaction, their IP (Internet Protocol) addresses are mutually known. These addresses are unique to every device connected on the internet and they provide specific information about their holder, including their location. This has the unfortunate side effect of allowing any web site or online platform owner to track the locations and identities of any clients connecting to it. For example, when you go on a Forum (such as Reddit or Quora) and post a question, the site's administrator will know who you are and where you connected from.

However, this begs the question, why is this unfavourable for the standard user? Why would we care if Facebook, YouTube or any other online platform we visit can track our location? Well, the issue does not stop there. A connection between two hosts is rarely direct, meaning that it must be routed through a very large and complicated network of switches and routers. These internet routers also have access to the receiver and sender IP addresses of each package they forward, giving them the ability to track possible users as well. Furthermore, the majority of internet routers are run by ISPs (Internet Service Providers), which in some nations, are owned and/or controlled by governing bodies.

Should a government be able to overhear and control the most widely used media of information? This is a controversial topic with many implications in basic human rights such as freedom of opinion and expression [2].

How can we mitigate the risks mentioned above when going online? How can we remain anonymous and secure personal information? The Tor network is one of the most widely used solutions to the aforementioned issues. It uses a number of volunteer 'relays' (routers) in order to anonymize its users before their connections reach their desired destination. This process makes it extremely difficult for passively listening parties to connect a client's signature to a visited site. However, the proportion of Tor users to standard internet users remains negligible since its release in 2002 (end of 2019 there is an estimate of 4 million Tor users). Why is that so?

While there are numerous possible factors for Tor's lack of distributed use, a defining one is the slow connection speed that Tor clients face. Interactive connections, which account for over 90% of connections in the Tor network [3], need low latency and high bandwidth to provide a smooth and pleasant experience for the user. However, because of its architecture, a network package must travel dramatically increased distances to reach its destination, when compared to traditional routing. This certainly discourages ordinary people from using the Tor network for their everyday activities.

In the years following Tor's introduction, there have been many improvements on its functionality and much research was conducted with the aim of improving its connection speeds, such as the Low-Latency AS-Aware Tor client which we will discuss in later chapters. This paper will attempt to contribute further by investigating the random network traffic caused by client connections and its effect on the current Tor browser operation algorithm's performance.

1.3 Objectives

In the section above, we described the facts acting as motivation for our current work. We hope for our contribution to lead to a better understanding of Tor network traffic by collecting real-time data and using statistical analysis techniques to explore the deviations of real-time bandwidth compared to the published consensus, which is sampled every hour in real life. Below, we clearly define the objectives this paper aims to achieve.

1. Choose an arbitrary number of Tor nodes which best represent the network as a whole.

2. Collect live relay data in order to infer the network traffic going through them.
3. Perform statistical analysis to explore the deviation of Tor node real-time bandwidth levels from the published consensus bandwidths which are sampled hourly.
4. Evaluate the relationship between the deviation described above and user-perceived performance.

1.4 Methodology

To achieve the objectives above, we will have to adhere to a systematic methodology, consisting of the following steps.

1. Come up with an appropriate technique to model the Tor network and use it to choose an arbitrary number of nodes which best represent the network.
2. Record the latencies of these nodes over appropriate time frames using live measurements so that available TCP bandwidth can be inferred.
3. Process the collected data using statistical analysis techniques and calculate the deviation of real-time to published available bandwidth.
4. Evaluate results in terms of user-perceived performance taking into account extra induced traffic on the network and available infrastructure.

1.5 Structure of Document

This section includes a break-down of this paper, giving an overview of each chapter.

Chapter 2 - Theoretical Background. In this chapter, we will be giving some background information on the Tor network, so that the reader becomes familiar with concepts which we refer to throughout this research.

Chapter 3 - Data Collection. Chapter 3 will be outlining the choices made in terms of the data collection part of this project. This includes how we pick specific relays to measure, along with the use of metrics and tools.

Chapter 4 - Data Processing. To use the raw data collected for performing analysis and drawing conclusions, it first must be reformatted and processed into a useful format. This section outlines the steps taken to achieve this.

Chapter 5 - Analysis and Evaluation. This is the final chapter of this paper. It describes and explains the processes used for analysis and visually presents our findings. It then goes on to evaluate the results in terms of Tor users' performance and formulates essential formulas which can be used for further research.

Chapter 2

Background

Having a basic understanding of the field we are discussing is very important in order to properly follow the steps described in this paper and their implications. Hence, in this chapter we are going to give an overview of the Tor network operation and introduce readers to onion routing and anonymity.

2.1 History of Tor

In the early 1990s, when the internet started getting some traction with the public and becoming 'mainstream', its lack of security and its ability to be used for surveillance and tracking was beginning to be apparent. Thus, in 1995, David Goldschlag, Mike Reed, and Paul Syverson at the U.S. Naval Research Lab (NRL) started researching ways of communicating using the internet, without revealing who is talking to who [4]. They shortly came up with the concept of onion routing.

Onion routing's objective was to offer a degree of privacy to internet users when they perform online transactions by hiding the identity of the sender (and the receiver in some cases eg. in onion URLs). It worked based on the principle of routing internet traffic through multiple servers with many layers of encryption. This way, the receiver of a packet would only know the location of the server which relayed it last and that is basically how Tor still works today.

To distinguish this work at the Naval Research Lab from other relay-based network research, the project was named Tor, The Onion Routing. After its development, the Tor network was initially deployed in October 2002 and its code was released under a free and open software licence. This was necessary because onion routing was meant to operate as a decentralised network and thus run by entities with diverse interests and trust assumptions.

In the first two years of its operation Tor only had a few dedicated volunteer nodes, reaching about a dozen in the end of 2003. Fortunately, by 2006, its apparent benefits towards digital rights had secured funding by the Electronic Frontier Foundation (EFF) and the Tor Project Inc, a non-profit organization was founded. During that time, the network started gaining rapid popularity among technology experts, but it was still difficult to advertise it to the public due to the technical knowledge required to set up for personal use.

In 2008, this technology barrier was brought down with the introduction of the Tor Browser, an easy to use internet browser which automatically routed users' traffic through the relay network, achieving anonymity. In the years to follow, especially after Snowden revelations in 2013 which turned mass internet surveillance into a mainstream concern, the popularity of Tor increased immensely. Today, the network has, on average, two million users active at any single time [5], who are being supported by nearly 7000 volunteer run relays [6].

2.2 Tor Basics

Now that we are more familiar with Tor, knowing how it was created and why it gained popularity, we will go on to explain the basic Tor network functionality and hence give some further context on the focus of this thesis. As we stated in the previous chapter, the Tor network consists of nearly 7000 volunteer operated nodes. However, not all of them have the same responsibilities. We can differentiate these servers in three different super-classes, namely; guard, middle and exit nodes. Each of these types has its own responsibilities and functions, which will be explained further later.

We have also mentioned previously that the core of Tor lies in routing internet traffic through predetermined servers, so users can stay anonymous. This is achieved by the creation of circuits. Circuits can be described as network paths which, by default, consist of 3 nodes;

1. A guard node, which acts as the clients' entry point to the tor network, is responsible for forwarding network packets from the user to the next node in the previously established circuit. Hence, the only information available to this server are the IP addresses of the user and that of the next node.

2. The middle node, which acts as a 'middleman', is responsible for receiving traffic from a guard node and forward it to an exit node. The only

information available to this server are the addresses of the guard and exit nodes.

3. Finally, the exit node is responsible for forwarding packets received from middle nodes to their final destination (eg. facebook.com). Following the pattern, this Tor relay only knows the addresses of the middle node and the packet's destination.

Using the protocol described above to route internet traffic, the final target of a user's network exchange (eg. facebook.com) is only aware of the IP address belonging to the Tor exit node which was used in the corresponding circuit. This is what provides Tor users anonymity, since any malicious party which might be overhearing internet traffic will not be able to connect the Tor client with its traffic destination.

However, how is such a circuit established? Who decides what nodes are going to be used for a given user's connection? Since Tor operates as a decentralised network, there is no trusted authority that can establish these circuits and keep them secure from eavesdroppers. Therefore, it is up to the client (Tor Browser) to choose relays and set-up the path used for upcoming network transactions. For this to be feasible however, a client must always be aware of the IP address of every single node in the network, and that is where the Tor Directory Authority servers come in.

The Tor network has around five to ten dedicated Directory Authority servers which are responsible for creating a network 'snapshot'. Essentially, they serve signed "directory" documents, called consensuses, containing a list of "server descriptors", along with short summary of the status of each router. When clients request and receive that document, they get up-to-date information on the state of the network, and are certain that the list they were getting was attested by semi-trusted directory authorities. (Here we use "semi-trusted" because an authority could potentially be compromised. Tor mitigates this risk by using multiple authorities which vote and agree on a mutual consensus document.) Having this information, users have the ability to choose relays and construct network paths which are used for their anonymization.

2.3 Bandwidth and Anonymity Issues

Having described the basic workings of The Tor Network, we should have an overall image of how onion routing works in theory. The clients pick a path, consisting of three Tor nodes, and redirect their internet traffic through it to remain anonymous during their online transactions. There are however certain issues which need to be addressed when implementing the described protocol and deploying it for public use, in order to be user friendly. We are going to focus on the network "speed" Tor provides its clients and shortly describe why its performance is worse than a typical web browser.

When a typical internet user wants to visit his favourite webpage or stream online content, he is limited by his "internet speed" or, more specifically, his active bandwidth. This is usually determined by the infrastructure offered by the local ISP, and sets a standard for the user's browsing habits. Unfortunately, traffic through the Tor network is highly unlikely to maintain a client's bandwidth due to the usage of multiple intermediary servers. These nodes, as stated in the previous sections, are volunteer-run, non-profit servers and hence do not necessarily provide a constant high-speed connection. In fact, the majority (as we will see in later sections) offers very low available bandwidth, creating a path "bottleneck". If any of the three Tor relays between a client and his destination provides a lower bandwidth than that of the client, then the connection's speed is going to be limited to that value. As a result, when using the Tor network, a user is very likely going to experience slower loading times than the standard he is used to.

Fortunately, The Tor Project has chosen to address this issue by having specific Bandwidth Authority servers perform systematic bandwidth measurements, which are published every hour inside the consensus document. This way, clients will theoretically know which nodes are "fast" and only use them to create their circuits, thus maximizing their available bandwidth through the network. However, this cannot be realistically done, since it would cause two major problems. Firstly, if all users chose to use the high-bandwidth relays (a small fraction of the network), they would end up flooding the servers with network traffic which would bring their effective bandwidth to zero. We can therefore see why this would be counter-productive in the long-run. Secondly, and most importantly, if all clients routed their internet traffic through the same predetermined circuit, they would actually de-anonymize themselves to any party that is overhearing their connections. This is possible since any network adversaries would know exactly what relays their traffic would go through and thus could connect a client to his traffic destination.

The solution given to the problems described above, is the usage of bandwidth probability weights in the consensus document. Instead of choosing the fastest, clients now pick a relay at random and the probability of one getting picked for circuit construction, becomes proportional to its available bandwidth. As a result, a user can choose a relatively fast circuit, for the majority of the time, without compromising his anonymity.

2.4 Previous Work

In the previous chapter, we talked about the effects of Tor node available bandwidth on users' perceived network speed. We also saw how the current Tor algorithm improves on the aforementioned issues, providing its users maximum performance, while not compromising their anonymity.

However, bandwidth is not the sole parameter affecting the perceived speed of a network. There has also been previous work in the field which aimed to lower Tor waiting times by trying to minimize latency induced by relaying network traffic. LastTor focuses on decreasing the latency by using geolocation clustering [3], while PredicTor focuses on predicting more efficient circuits using machine learning [7].

2.5 Proposal

In this paper, we are going to explore how live Tor network traffic affects the available bandwidth of circuit nodes and how much it deviates from the hourly sampled bandwidth weights.

When Bandwidth Authorities perform measurements on relays, they capture a 'snapshot' of the available bandwidth the target can offer at that specific point in time. Since the available bandwidth of a server highly depends on the amount of incoming network packets, and network traffic is induced by clients connecting randomly throughout the day, it is valid to assume that the server's response time is not constant. Therefore, when the last published consensus weights are used by incoming Tor clients to build circuits, their accuracy cannot be guaranteed.

So how can we know what the real available bandwidth of a relay is, at any given point in time? To do this, we would have to constantly perform measurements on the specific server and record the TTD (time to download) for a known file. However, this is not realistic, as performing such a task for all nodes in the Tor network would dramatically increase our network traffic and use most (if not all) of our available bandwidth.

Since we cannot always know the real available bandwidth of a given server, we are going to explore its deviation from the last sampled consensus value.

If the deviation is minimal, it would mean that consensus bandwidth weights remain valid for a long period of time. Hence, proposing an increase to the bandwidth publishing intervals to lower the Bandwidth Authorities' measuring load induced on the network, could improve performance. However, if we find that there are significant levels of deviation between the relay live available bandwidths and their sampled values, a smaller sample window could prove beneficial towards performance.

This thesis therefore, focuses on the effect of the Tor nodes' bandwidth publishing interval to perceived client network performance.

Chapter 3

Data Collection

This chapter outlines the decisions made concerning the data gathering stage of the project. In order to gather live available bandwidth levels across different Tor relays, several factors must be taken into account. Firstly, the number of relays to be measured must be decided, as well as which specific ones. Within this there are time and infrastructure constraints to account for. Secondly, an appropriate measuring metric and tool must be chosen, along with the volume of data required. Finally, the method and program created to tackle each of these factors must be formulated while taking into account the project's requirements.

3.1 Choosing Tor Relays

As with any project requiring real-world data, a sample must be chosen from the population for data collection. In this case the Tor network consists of 7000 nodes, and recording live data for all of them is difficult to perform. To ensure an appropriate sample is taken, one that represents the real distribution of bandwidth across the Tor network, several methods of statistical sampling were considered, ranging from naive approaches to more sophisticated and accurate methods. Each method assumes a different distribution for the data gathered, and this chapter outlines their benefits and drawbacks as well as the final choice of sampling method. It must be taken into account that the number of relays we are able to measure depends on multiple factors such as; the measuring software, the available network bandwidth and the available computing power.

3.1.1 Naive Approach

A baseline for statistical sampling is the assumption of a random distribution in the population. This means that the chosen data set would consist of randomly chosen relays from the Tor network, making this approach an easy-to-use fast sampling technique. However, not all relays in the network are comparable or equal as there is a natural inequality across nodes in terms of bandwidth. For example, a few nodes will always have a higher amount of bandwidth available due to high-quality hardware, whereas others will not be available online for most of the time. Therefore, assuming a random sample of the population is taken, there are no guarantees that all relays will be represented. This could result in the data consisting mainly of extreme cases: of the higher-end relays, or on the other hand, of the slowest or unavailable nodes.

For these reasons, the naive approach was not considered as an appropriate and truly representative sampling technique for this project.

3.1.2 Ordered Approach

In order to ensure our sample will be representative of the different Tor relays, we must solve the problem arising from the use of a random approach. A way to achieve this would be to perform uniform selection from an array containing all relays in order of their available bandwidth. Irrespective of the number of relays we choose, this technique mitigates the risk introduced by the naive approach and ensures the representation of both high and low performing Tor nodes in the chosen sample. However, since this project aims to explore the Tor users' perceived network speed, the resulting sample must also accurately represent the relay choice of an average Tor user when building a circuit.

As described in chapter 2.3, for a Tor client to create a circuit through the network, it must randomly choose a specific number of nodes to route its traffic through. Since each publicly available relay has an assigned selection bias proportional to its last known available bandwidth, a faster relay has more probability of being chosen than a slower one. Consequentially, we need the chosen relay sample to reflect this weighted bias applied on Tor circuit building and thus, uniform selection would not be an appropriate approach for sampling our population.

3.1.3 Matching Probability Distribution

To solve the problem described above, we will need to explore the bandwidth distribution of the Tor network relays. Fortunately, the data we need in order

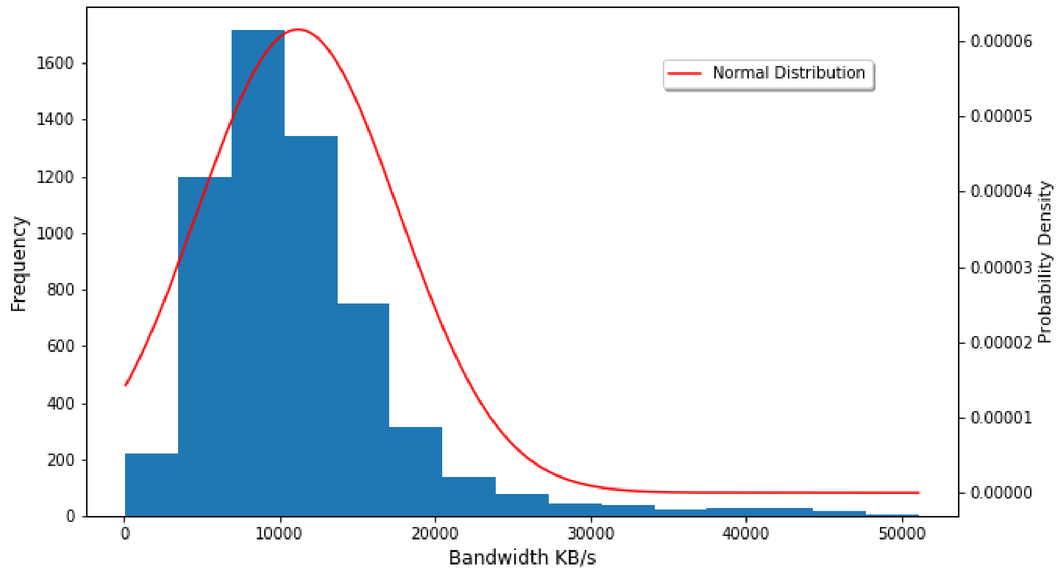


Figure 3.1: *February 2020 mean Tor bandwidth weights distribution*

to do this is publicly available as part of archived Tor consensuses.

Figure 3.1 includes data from relays that were continuously online throughout February 2020. We can see the available bandwidth weights of the Tor nodes range from 0 to more than 50 MB/s. Even though the majority of them have a bandwidth lower than 25 MB/s, the relays in the top percentile are chosen much more often than the rest due to the bias in the circuit building algorithm used by Tor clients. This behavior would not be represented if we decided to choose relays uniformly according to their bandwidths. In order to solve this problem, the sample we choose from the population must include a higher proportion of high-bandwidth relays than the population itself, imitating the behaviour of Tor clients when building circuits through the network. Furthermore, our solution must be independent of the final sample size since it may need changing due to software/hardware limitations.

To achieve this we will have to formulate an algorithm which outputs N Tor node identifiers taking into account the previously discussed constraints. We start by forming a cumulative distribution of all mean consensus weights ordered by magnitude, assigning the total sum of the probability weights to 1. This curve, seen in figure 3.2, describes the inequality of the available bandwidths inside the Tor network and it can be used directly to choose an arbitrary number of relays for our sample. Firstly, N points are chosen uniformly in the range [0,1], which represents the cumulative weight distribution range shown in the y-axis of figure 3.2. We then calculate the corresponding coordinates of the chosen points along the x-axis, which represent relay identifiers ordered by decreasing bandwidth.

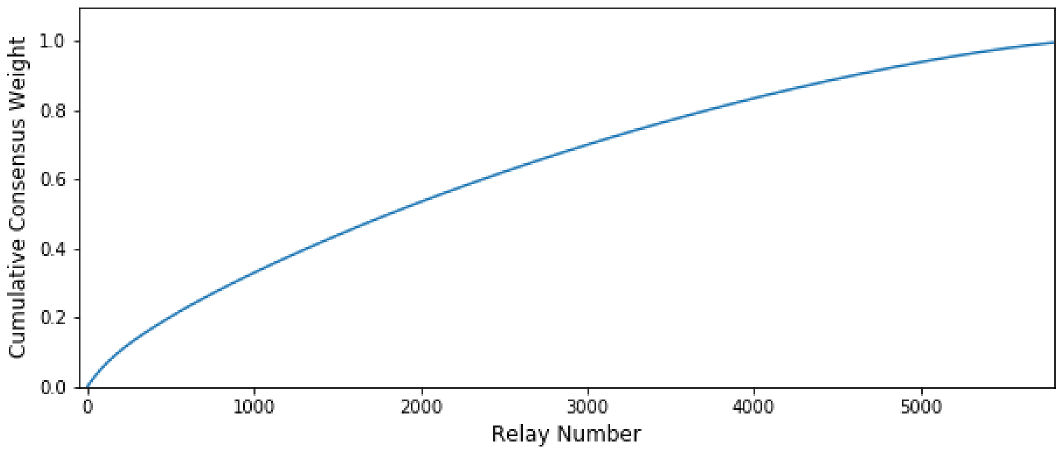


Figure 3.2: *Cumulative consensus weight distribution in Tor network*

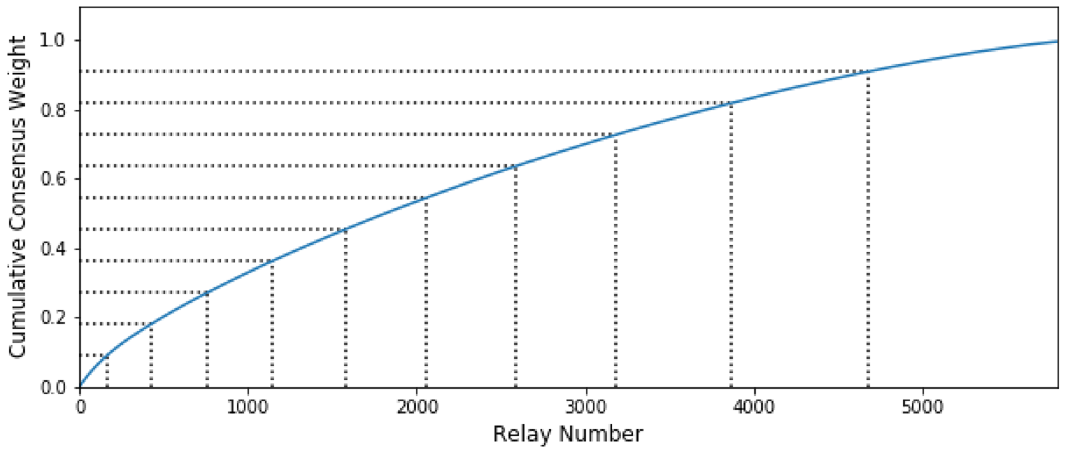


Figure 3.3: *Choosing 10 relays using cumulative distribution*

This algorithm was implemented and used to choose an example of 10 relays from the Tor network and the results are depicted in figure 3.3. Visually inspecting the graph, we see the distance between the chosen nodes becomes greater as their bandwidth weight decreases, following the pattern of the cumulative weight distribution. This achieves the previously stated goal of our chosen sample including a higher proportion of high-bandwidth relays than the population, while following the circuit building algorithm probability curve.

3.2 Measuring metric and tool

The technique used for sampling the population for data collection has now been finalised. This section focuses on two further important aspects of the data collection process; the metric and tool used. Firstly, we need to define a metric that provides useful information about the network state of a Tor relay at a given point in time, but also one we can accurately record. This is a critical step in this project, as the accuracy of our results and conclusions will be largely based on the metric chosen. Secondly, we must choose an appropriate tool for the recording of the measurements and evaluate it according to the project objectives. It is also worth noting that our available computing power and network bandwidth need be taken into account when choosing a software to use as different tools have different requirements.

3.2.1 Metric

Throughout this paper, we have been using bandwidth to rate a server's connection speed, so it would make sense to attempt directly recording the live Tor relays' available bandwidth. To perform this accurately we would have to record the time taken to download a file, of known size, directly from the server in question. Unfortunately, most Tor relays do not host a publicly accessible web server with available files, rendering this technique infeasible. Another possible approach for inferring the live bandwidth of a relay would be to use it in a circuit along with two other nodes which are known to have a higher consensus bandwidth weight. This would force the relay in question to become the bottleneck of the connection through the constructed circuit. Knowing this allows us to measure the relay's available bandwidth through the effective bandwidth of the connection. However, this technique would introduce many possible sources of noise to the recorded data such as other relays and any external web server used. Therefore, it is not deemed appropriate for this project's purpose.

Due to the properties of the TCP protocol, there is another method of accurately inferring bandwidth without measuring it directly. TCP is a network protocol that defines specifications on how data is transmitted between two network hosts in order for it to reach the destination without any loss or corruption. For this to happen, the receiving host must acknowledge received packets by sending periodic ACK replies. If an ACK reply is late, the sender must pause after sending the first 65KB (by default) until acknowledgement is received. This results in the latency between two communicating hosts, dictating their maximum possible TCP throughput.

Latency is defined as the time taken for data to propagate from one network node to another and the most common way of measuring it is called round-trip time (RTT). This is the time it takes for a data packet to travel from one point to another on the network and for a response to be sent back to the source. Since Tor uses TLS encryption [8], which in turn works over TCP, we can measure the temporal latency of relays in order to infer their maximum live TCP throughput. Since throughput is measured in B/s, we can calculate the maximum bandwidth using the following formula.

$$Bandwidth < \frac{BUF}{RTT}$$

Where **BUF** is the maximum available TCP buffer size and **RTT** is the round-trip time of one single network packet. As a result, continuous measurements of RTT for a data packet between our measuring PC and the relay in question, would provide useful information about the relay's temporary network load and allow us to record its available bandwidth.

3.2.2 Tool

In the previous section we have discussed verified approaches for accurately recording a Tor node's temporary available bandwidth and hence estimating network load. Due to limitations presented, we have decided to record the latency between two hosts in order to extract the data necessary for completing our objectives. Next step in the process is to choose an appropriate tool for our measurements to be valid in the context of this research. This tool must also be well documented and able to run on our available hardware.

The simplest way to measure latency between two nodes in a network, is to use a tool named Ping. This is an extremely lightweight program in terms of memory footprint and is also built-in on most operating system that have network capabilities. Ping works on the Internet Control Message Protocol (ICMP) and operates on the third layer of the OSI model, namely the network layer. When the command is run, the computer sends 32 bytes of data to the specified network host and then reports the time taken to receive a reply in milliseconds. This time is the previously mentioned RTT between the local and the target host. These properties make Ping a very fast and accurate tool for recording latency of a network path. It does however, present a few drawbacks that render it inappropriate for use within our specific domain.

The Tor network has very specific properties which differentiate it from traditional AS routing systems. When data packets are routed through it, each Tor relay must unpack part of the application-layer data of each packet, in order to know where to forward it next. This differs from normal IP routers that use lower-layer packet headers to route passing network traffic. This Tor behaviour introduces delays and noise to the latency of a network route which Ping would not be able to capture. Hence, any results we record using it would not accurately represent the network state through the perspective of a Tor user, which ultimately is our objective. We must therefore find an alternative

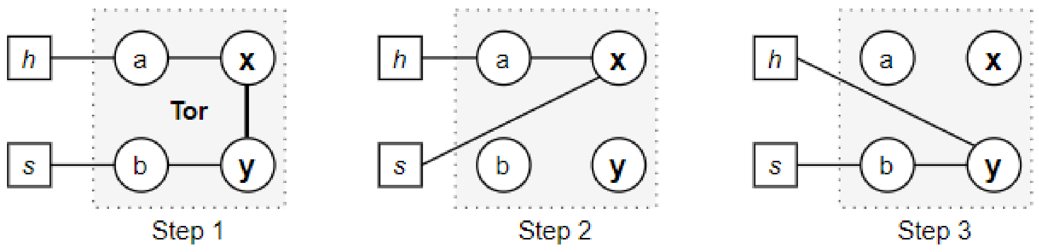


Figure 3.4: *Ting technique for measuring latency between two Tor relays (x,y) using relays (a,b) and a local host and server (h,s)*

tool to use for recording the required data.

Fortunately, there is a publicly available software which measures latency while also capturing the Tor network's behavior. Ting was developed in 2015 with the goal of measuring latency between two arbitrary Tor relays [9] and runs on the Tor protocol. Its primary purpose was to give researchers and developers the ability to measure the network path latency between two nodes they do not directly control. As mentioned above, Ping outputs the time taken for a packet to reach a destination host and travel back to the local computer. However, it does not provide functionality to measure the latency of a route in which neither of the hosts are controlled by the researcher.

Figure 3.4 depicts the steps utilized by the Ting algorithm in order to calculate the path latency between Tor relays **x** and **y**. Firstly, the software creates a simple server, **s**, on the local machine, **h**, and configures it to listen for network packets at a user-specified port. It then constructs a Tor circuit through relays **a, x, y, b**, where **a, b** are two high-speed, pre-specified Tor relays. Finally, Ting attaches a TCP connection between **h** and **s** through the Tor circuit and measures the RTT. This first step records the time taken for a data packet to travel through all four intermediate nodes. The two further measurements, depicted in steps 2 and 3 of figure 3.4, allow Ting to isolate the latency between the two required nodes with a reasonably high accuracy [9].

However, our objective is to measure the available bandwidth of one specific relay, while Ting measures the latency between two of them. So how can we use it to infer the response latency of a single node? To achieve this, the fastest Tor node will be chosen to act as an 'anchor' to all measurements. We choose a high-bandwidth node, so that we force the second relay to be the latency bottleneck of the network path in-between. This is a similar technique to the one used by the Tor bandwidth authorities, which construct a circuit through 2 relays; The one being measured, and a known faster one. Consequentially, the result of the measurement will depend on the available bandwidth of the 'slow' node.

Ting, in contrast to Ping, utilizes the Tor network to record latency measurements between specified nodes, and hence any Tor-specific behavior will be captured in terms of delays or noise. This way we can be certain we are looking at the network through a client's perspective and not just testing the underlying infrastructure. Therefore, in this paper we will be using Ting to record latency of a specified path towards the relays we have chosen, in order to infer their available bandwidth due to network traffic.

3.3 Data Collection

Up to this point, we have chosen the metric and tool that will be used for the data recording part of this project. The following section will be focusing on the requirements and formulation of the data collection methodology and evaluate the choices made in terms of the paper's objectives. The implementation of the software responsible for recording the data will also be presented.

3.3.1 Requirements

Before we decide on specific methods and techniques for bandwidth recording, we must make sure the results will be useful in regards to the project's initial objectives. To do this, we will be setting out necessary requirements the collected data must meet.

- The current Tor consensus publishing interval is 1 hour long. Hence, Tor clients only receive an update on the available bandwidth level of each relay every 60 minutes. We want to capture the live change of available bandwidth over shorter periods of time, to test how shorter publishing intervals would affect a client's performance. Therefore, the live relay latency measurements should be performed throughout a session of at least 1 hour for each node.

- To rate how different consensus publishing intervals affect Tor clients' performance, we are going to simulate shorter intervals by sampling the live latency measurements we collect. Then we will calculate the area between the original and sampled signals, getting a metric of variation from the real value. Shorter sampling intervals should theoretically yield smaller variation from the original data and hence would provide better network performance. Since the larger interval tested will be 60 minutes, we must make measurements using longer time frames so that we can study its effects as well. A minimum length of 3-hours is therefore chosen for each live measurement.

- It is clear that just a one off 180-min live measurement session cannot be representative of the overall behaviour of a certain relay. We must therefore make repeated measurements for each relay in order to be able to recognise possible outliers. However, due to hardware and network limitations, we cannot parallelize the data collection process, ie. we cannot record latencies for more than 1 relay at once. Taking this into account along with time constraints, we should aim for at least 3 latency measurement sessions for each of the relays in our chosen sample.

3.3.2 Recording Method and Software

To execute this section of the project, measuring software needs to be constructed and deployed on a Raspberry Pi 3b+. This software will be responsible for collecting the latency levels of pre-specified relays while following the specifications set out in the previous paragraphs.

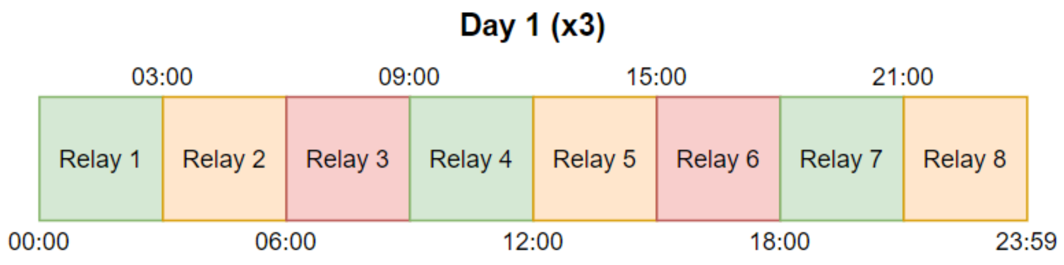


Figure 3.5: *Splitting of a day in 3-hour sessions for recording latencies of 8 different relays*

Initially we will choose a sample of 32 relays from the Tor network using the technique described in the beginning of this chapter. This includes collecting their unique fingerprints for Ting to have the ability to construct circuits using them. As there are 24 hours in a day, and we are measuring each relay for 3 hours, we can only measure 8 relays per full day. For each relay there must be 3 measurements, or 3 sessions, so as to provide more accurate data. Therefore this section of data collection will take 3 days for each group of 8 relays (from the original 32), resulting in a total testing effort of 12 days, assuming no interruptions occur. This can be seen pictorially in Figure 3.5. An arbitrary number of 32 was chosen to avoid faulty data which can often occur when using the tool Ting. Of course it would be more beneficial to this research if we could measure more relays, but due to limitations, 32 was thought to be the maximum number we can test.

Using Python3, we create a program to manage the gathering of session data for all relays automatically. Firstly, the program divides the selected relays into their respective groups and calculates and keeps track of the time slots for each relay so as to switch between sessions. The explanation for repeating the experiments three times, meaning testing the same relay in the exact same time slot, is to enable us to identify time-dependant network traffic patterns while limiting data outliers. Theoretically it is possible to perform parallel measurements so as to limit the experiment time, but this was unachievable with the available hardware, something that should be considered in any potential continuation of the project. Once the relays are separated and time slots are assigned, a web server is started for the purpose of using Ting (**s** in Figure 3.4). This covers the initialisation of the program.

With each measurement, a single instance of Ting is created with the required relay as input. The pre-specified relays, necessary for the Ting experiment as explained in Chapter 3.2.2, are also set by the program. This process is repeated for all relays' measurements until all sessions have been completed. Ideally this would have taken 12 days, but due to many factors such as network availability and problems with Ting functionality, the process had to be restarted several times resulting in a data set of 19 rather than the initial estimate of 32.

Chapter 4

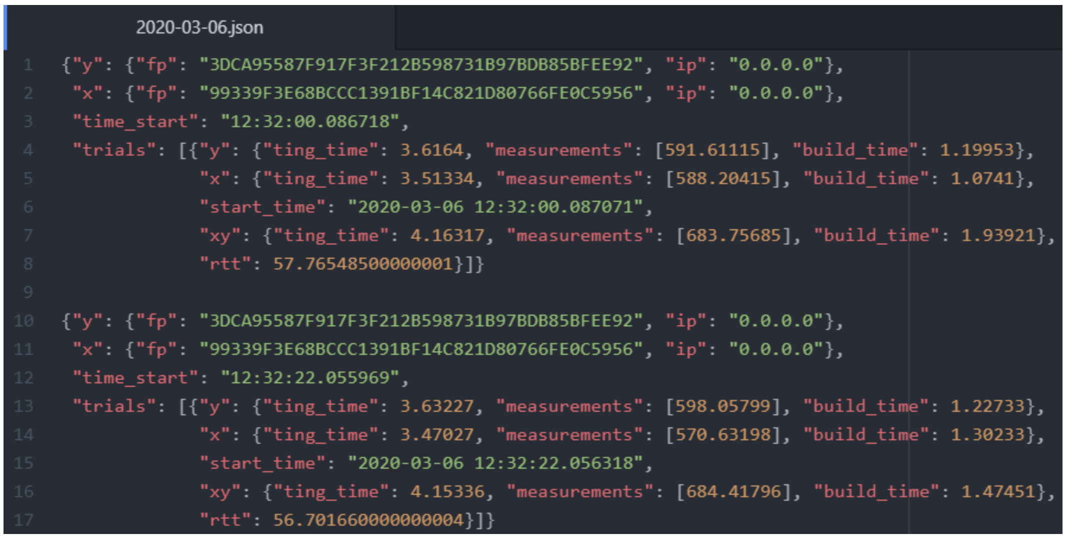
Data Processing

This chapter will talk about the methodology followed to analyse the Tor relays' live latencies collected as described in the previous section. We will be exploring and justifying the data processing steps and the decisions made in relations to this paper's initial goals. As in most projects involving statistical manipulation [10], our objective for data processing is to code and capture data from its raw state to a custom format, so it can be used directly for further analysis. This includes creating CSV files using a pre-specified format, filtering the data for noise reduction and performing other necessary operations to achieve the required results.

4.1 Raw Data Presentation

In this chapter, the data collected will be presented in its output format. When a latency measurement is performed, Ting automatically appends the results to a pre-specified JSON file, the format of which can be seen in figure 4.1. The two measurements depicted include; the fingerprints of the relays that make up the current network path under test, a time stamp and the measured latencies in milliseconds. As seen in figure 4.1, Ting has already calculated the isolated path between relays **x, y**, which can be found in the last **"rtt"** field of each record.

Measurements are taken back-to-back and are mostly 22 seconds apart, as can be seen in figure 4.1 looking at the 'start_time' variable. There is however, a noticeable variance in the interval between measurements since it can range from 20 to 40 seconds (in edge cases), depending on the time taken for Ting to set up the respective Tor circuits. It must also be mentioned that many measurements taken failed due to relay availability, as some nodes were not online when the Ting experiment was running and so several sessions were missed or corrupted.


2020-03-06.json

```
{"y": {"fp": "3DCA95587F917F3F212B598731B97BDB85BFEE92", "ip": "0.0.0.0"},
 "x": {"fp": "99339F3E68BCCC1391BF14C821D80766FE0C5956", "ip": "0.0.0.0"},
 "time_start": "12:32:00.086718",
 "trials": [{"y": {"ting_time": 3.6164, "measurements": [591.61115], "build_time": 1.19953},
             "x": {"ting_time": 3.51334, "measurements": [588.20415], "build_time": 1.0741},
             "start_time": "2020-03-06 12:32:00.087071",
             "xy": {"ting_time": 4.16317, "measurements": [683.75685], "build_time": 1.93921},
             "rtt": 57.76548500000001}]}

{"y": {"fp": "3DCA95587F917F3F212B598731B97BDB85BFEE92", "ip": "0.0.0.0"},
 "x": {"fp": "99339F3E68BCCC1391BF14C821D80766FE0C5956", "ip": "0.0.0.0"},
 "time_start": "12:32:22.055969",
 "trials": [{"y": {"ting_time": 3.63227, "measurements": [598.05799], "build_time": 1.22733},
             "x": {"ting_time": 3.47027, "measurements": [570.63198], "build_time": 1.30233},
             "start_time": "2020-03-06 12:32:22.056318",
             "xy": {"ting_time": 4.15336, "measurements": [684.41796], "build_time": 1.47451},
             "rtt": 56.701660000000004}]}
```

Figure 4.1: *Example Ting measurements in raw JSON format*

4.2 Data Reformatting and Visualisation

When handling large amounts of data, its visualisation is key since it allows trends and patterns to be easier to spot [11]. Just reading the JSON file depicted in figure 4.1, does not provide a lot of useful information. Our priority is transforming the data to a standard format which enables the use of well-known Python graphing library; Matplotlib [12]. This meant taking all the information included in the big JSON files provided by Ting and separating all relay 3-hour recording sessions. The program created was responsible for keeping track of the different sessions, their respective time slots and the unique relay identifier, so as to separate the data correctly into different CSV files.

Using the newly created files, a graph can now be created to demonstrate latency measurements for a relay across different sessions, seen in figure 4.2. Here, just 1 hour of latency data is shown, so that the subtleties in the short-term patterns can be seen clearly. High latency spikes represent a surge in network traffic and requests to the particular Tor node, while negatively pointing spikes denote instances with decreased traffic. This happens due to increased network load causing the server's reply to be slower than usual, and thus the temporary latency measured is higher. While there is no visible daily pattern in the reply times of this particular Tor relay, its latency seems to oscillate around a stable level of 50ms. Applying the formula shown in 3.2.1, with a default TCP buffer of 64kB, the node's maximum TCP throughput is calculated to be 6.5Mb/s, compared to an assigned consensus weight of 66200 on the same day.

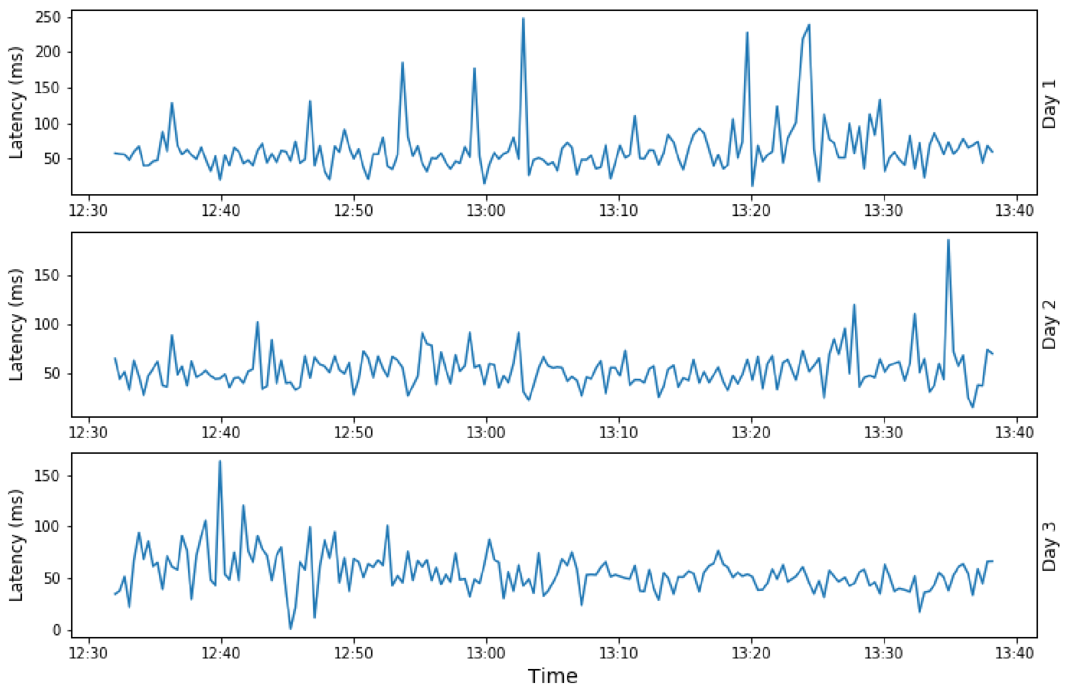


Figure 4.2: *Example Relay Latency measurements over the same time slot across 3 consecutive days*

4.3 Signal Filtering

In section 3.2.2 we introduced and described Ting as the tool used to collect the data needed for this project. Since latency is the network path delay between two nodes, the data collected not only depend on the relay under observation, but on the 'anchor' relay as well. While the anchor was specifically chosen from a small pool of high-bandwidth nodes, meaning it is not likely for it to be the bottleneck in any circuit through the Tor network, its dependency introduces a possible source of noise added to the observed signal in the form of short-term fluctuations. This can have a negative impact on the consensus publishing simulation later on, since it can largely affect the signal sampling process due to the aliasing effect [13].

Our goal is to try and minimize the short-term fluctuations caused by noise and hence improve the accuracy of the sampling process, by employing a processing technique called signal filtering. Filtering is often used in electronics to eliminate undesired frequencies from a signal and can significantly improve the visibility of any existing patterns [14]. Without such filtering there can be multiple signals which, when sampled, can generate the exact same result. Furthermore, the type of filter and its parameters must be chosen carefully, as incorrect settings can potentially distort a signal and eliminate its properties. In cases where sampling is required, a low pass anti-aliasing filter is commonly used with a bandwidth of **fs**/2 Hertz where **fs** is the number of samples taken

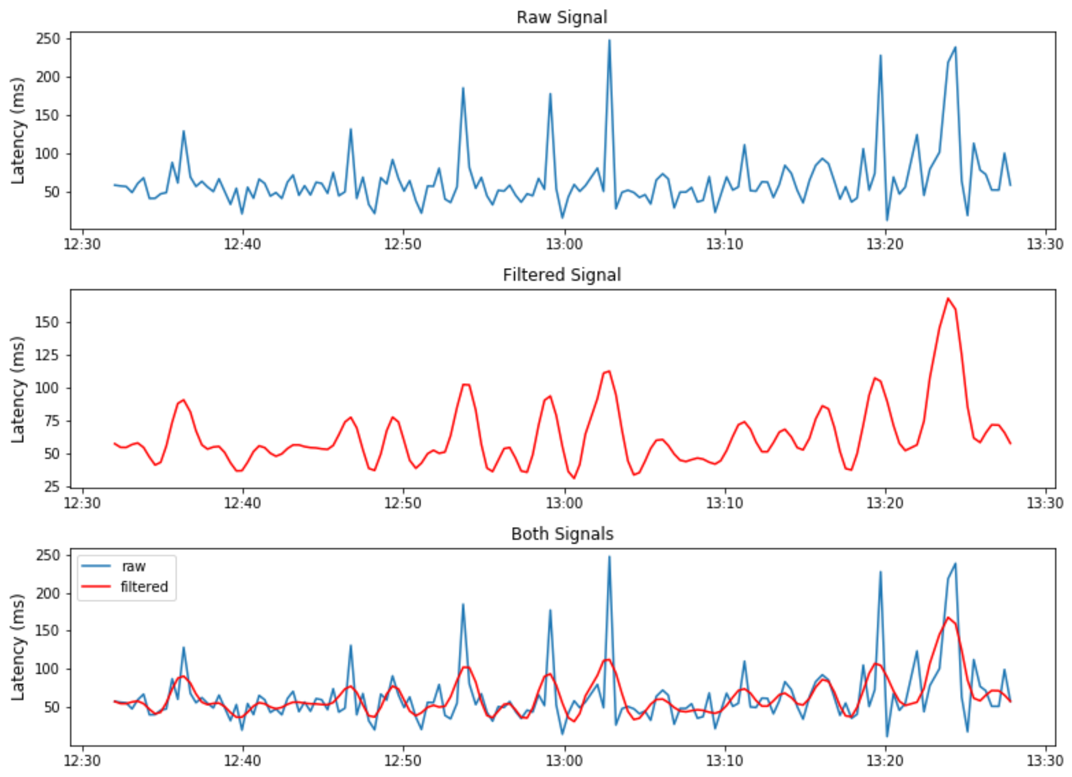


Figure 4.3: *Filtering the collected latency data using a lowpass filter*

in one second. This satisfies the Nyquist–Shannon sampling theorem [15].

The latency signal depicted in figure 4.2 will be sampled at a maximum frequency of 1 per minute (minimum publishing interval we are testing for), making the sampling frequency $\frac{1}{60}$Hz. Following the rules stated above, we will be using a lowpass filter with a cutoff frequency of $\frac{1}{120}$Hz to reduce noise in our sampling process. The outcome can be seen in figure 4.3. The first pair of axes contains the raw latencies collected from a Tor relay throughout 1 hour, while the second contains its filtered counterpart. Looking at the third plot, we can directly see how the original and processed signals compare. The raw latencies have many more spikes which are larger in magnitude throughout the 60 minutes, while the smoother curve tends to follow the overall pattern without reaching the extrema points of the blue line.

4.4 Signal Sampling

The purpose for this section is to simulate Tor's bandwidth publishing by sampling the collected results using various intervals. Currently the Tor bandwidth authorities take measurements every hour and publish them in the hourly-updated consensus document, even though the relays' available bandwidth does not remain constant. It is this project's objective to explore how shorten-

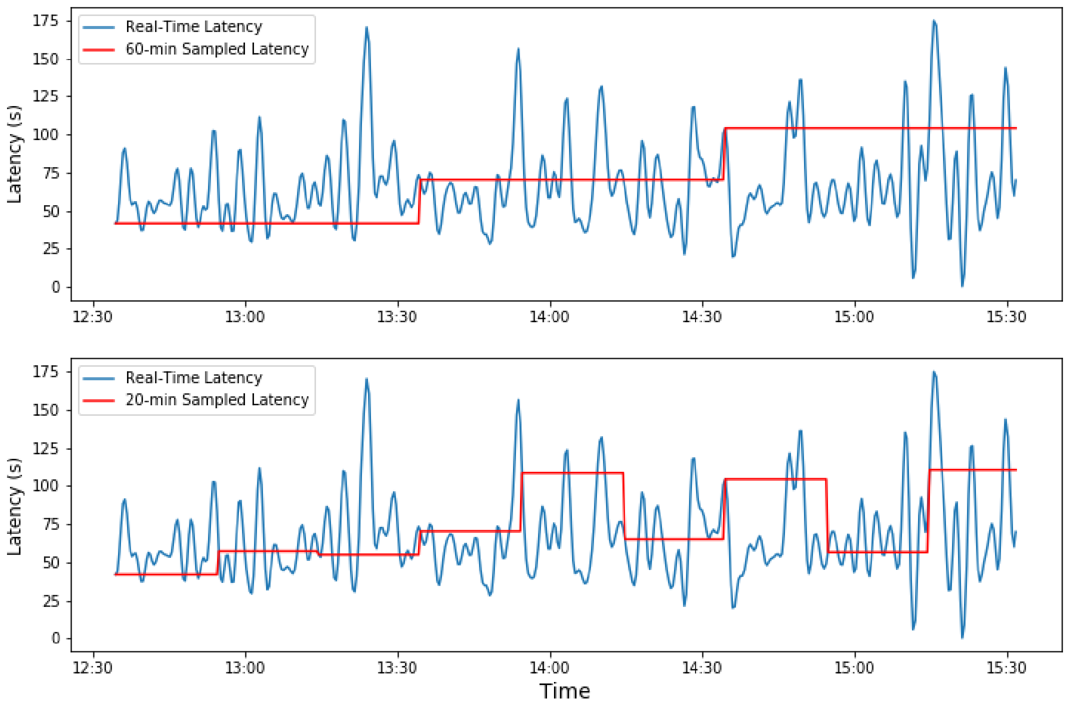


Figure 4.4: *Live latency measurements of a relay sampled using a 60-minute(top) and 20-minute(bottom) interval*

ing this publishing interval could give a more accurate picture of the network to Tor clients and thus improve performance. By sampling the recorded relay latencies using different frequencies we can compare the variation between the real-time and published bandwidth levels, for various simulated publishing intervals.

Given the real-time signal and a sampling interval, we can calculate the sampled signal as shown in figure 4.4, where the same latency data has been sampled using 60 and 20 minute intervals respectively. The top diagram is a good representation of a relay's available bandwidth known to Tor clients during a 60 minute window, before the authorities update the consensus with newer values. Its red sampled levels are mostly static and rarely coincide with the real-time ones due to the non-constant load caused by network traffic. This means that at any point, Tor users' network information does not accurately represent the network snapshot. On the bottom graph, where a 20 minute interval is used to sample the latency data, the red curve seems to follow the blue one more closely thus suggesting a negative relationship between the sampling interval and the variation between real-time and sampled latencies.

The ultimate goal for using different sampling intervals is to directly compare their resulting variance from the real-time bandwidth levels. A big factor in the similarity between the sampled and continuous bandwidth is the starting point, or offset, used in the sampling process. Each starting point generates

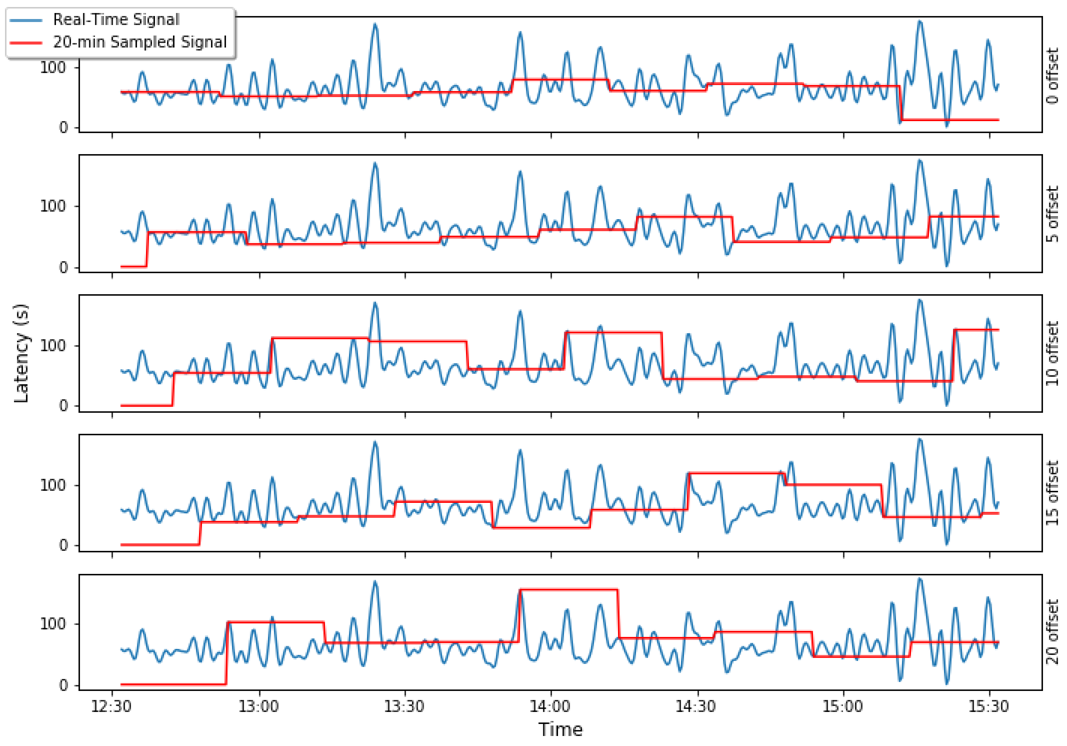


Figure 4.5: *Sampling latency data using the same interval but different starting offsets*

a completely different signal as seen in figure 4.5. While the same signal and sampling interval were used in all 5 plots, the non-similar starting points used for the sampling process cause the red curves to have different levels of variation from their original counterparts. In this case, which initial offset best describes the resulting variation for a publishing interval of 20 minutes? Since all of them represent a possible scenario in the real Tor network, and in order to provide a valid comparison between publishing intervals, all different offsets for each signal must be taken into account.

It is also worth noting that the number of possible offsets depends on the publishing period. The longer the interval, the more possible starting points there are for the sampling process.

Chapter 5

Analysis and Evaluation

Here we will talk about how we can evaluate our findings

5.1 Variance Metric

Tor clients' performance is largely dependent on the accuracy of the consensus bandwidth information since its overestimation leads to slower connection speeds through the network. To evaluate a specific bandwidth sampling interval, in terms of its effects on user performance, the accuracy of the resulting network information at any point in time must be measured. The goal for this section of the project is to produce an appropriate metric that will represent how close a sampled signal is to the real-time collected values. In order for it to fully represent this difference, several factors have to be addressed - how often the sampled signal varies from the original, and the magnitude of that difference. This comparison is important as the further away the sampled signal is from the original, the worse the user's performance becomes. This is because Tor clients choose relay circuits depending on available bandwidth, and so if these measurements are wrong or outdated, the selection of relay circuits will not be optimal and performance will be diminished.

5.1.1 Area Between Curves

An appropriate metric for measuring the variance between two signals, taking into account the previously described factors, is the total area between them during a specific time frame. Since the data is discrete, we can describe this more accurately in the form of the following sum, where t stands for time, t_{start} and t_{end} define the time window, and x and y represent the real-time and sampled data respectively.

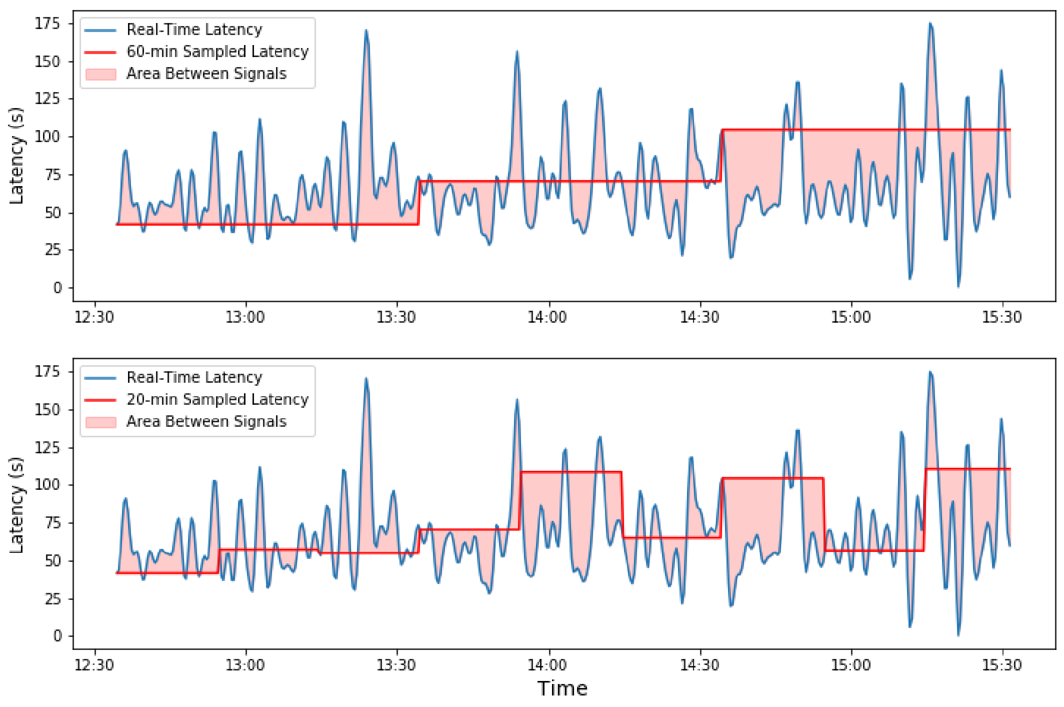


Figure 5.1: *Visualising variance as area between real-time and sampled signals using 60 and 20 minute sampling intervals*

$$\sum_{i=t_{start}}^{t_{end}} |x_i - y_i|$$

This will account for not only the magnitude difference between the signals but also the time for which the difference is present, and therefore affecting user performance. We can visualise this in figure 5.1. Here the

However, the area metric presents a significant problem when using it in this scenario. As mentioned in chapter 4.1.1, the measurements recorded using Ting were not all taken in equal time periods. They range from 20 to 40 seconds each, due to the non-constant Tor circuit construction time. Changing network traffic levels and different relays can cause this inequality. Consequentially, the relays' latency recording sessions, each one lasting 3 hours, do not all have an equal number of data points, ranging from 380-600. The fact that the total sum of the area between two curves, as defined in the formula above, is biased towards biggest datasets, may skew the comparisons. If a session containing 400 datapoints is compared to one with 600, the area between the curves will tend to be higher in the bigger set due to the summing factor. This will be true independently of the actual variance between the sampled and real-time curves. Therefore, we must change this metric in order to take into account the dataset sizes.

5.1.2 MAE vs RMSE

To solve this, we can explore two well established measures of model accuracy - the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) - both of which take into account the size of the dataset under comparison [16]. RMSE is defined as the standard deviation of the residuals between a datapoint and a line of best fit, while MAE is a measure of errors between paired observations expressing the same phenomenon. Both of these metrics give a good measure of 'similarity' between two sets of data, so to evaluate them, we must explore the specific properties which differentiate them. Their formulas are shown bellow.

$$MAE = \frac{1}{N} \sum_{i=0}^{N} |x_i - y_i|$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=0}^{N} (x_i - y_i)^2}$$

Where N is the size of the datasets x and y.

Both metrics express average model error in the same units as the variable under test, they focus on error magnitude (rather than direction) and are negative oriented, meaning a lower score provides evidence of higher similarity. The main difference is the use of squaring by RMSE to cancel negative errors between the two datasets being compared. Since the differences are squared before they are averaged, a relatively high weight is given towards large errors. As a result, RMSE can prove more useful when error magnitude plays a particularly important role. This is very useful in terms of this project, since a big difference between the published and actual available bandwidth of a Tor node has a big negative effect on the performance of the network. So we will be using RMSE as a metric of variance between a real-time and a sampled latency signal.

5.2 Data Analysis

The RMSE was calculated between the continuous and sampled latencies of every recorded relay, using intervals in the range of 1 minute to 1 hour. As described in 4.4, all possible offsets must be taken into account for each sampling frequency tested, so that outlier results are minimized. Achieving this, means calculating the mean and standard deviation of RMSE results produced using

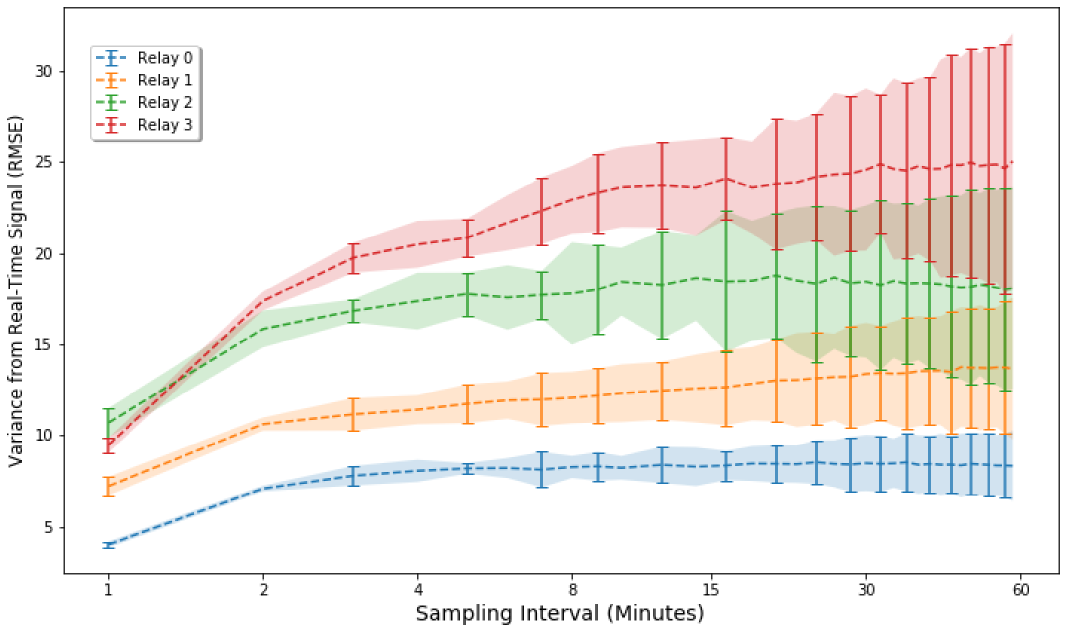


Figure 5.2: *Mean and Standard Deviation of RMSEs calculated between continuous and sampled latencies across a range of sampling intervals for 4 Tor relays.*

all possible offsets of each interval. This will help us better comprehend the effects of shortening consensus publishing periods on the real Tor network.

5.2.1 Individual Relays

The following section will be depicting RMSEs produced by data collected from 4 specific relays. These nodes were chosen with a goal of representing various latency levels, hence helping us visualise possible patterns across nodes, as well as giving us an idea of what the overall effects of increasing sampling intervals are.

The 4 relays in figure 5.2 are numbered in increasing order of resulting variance. The dotted lines depict the mean RMSE calculated for each sampling interval, and the error bars along with the shaded regions represent the spread (standard deviation) of the results across all sampling offsets. A higher RMSE translates to a greater variance between a sampled signal and its original counterpart, suggesting lower performance. On the other hand, a lower error indicates that the resulting sampled latency is a more accurate representation of the real-time values. An of 0 would mean the two curves are identical.

In the sampling interval range of 1 to 20 minutes, all nodes on the graph seem to follow a similar trend. The magnitude of the error between sampled and continuous latencies, and the size of the interval have a positive correlation. This was expected since a decreasing sampling period provides by default more information about the continuous signal, leading to a smaller error. Furthermore, the rate of increase of the error slows down for sampling intervals greater than 2 minutes, for all 4 relays. Even though all RMSEs seem to converge to static values as the sampling periods grow, it is worth noting the errors for nodes 0 and 2 start slowly decreasing when the sampling interval becomes greater than 25 minutes, while for nodes 1 and 3 they do not.

We can also notice a recurring pattern in the standard deviation of the RMSEs across the sampling intervals tested. The spread tends to become greater as the period grows for all measured relays, which leads us to believe that longer periods lead to higher instability in the sampling process.

5.2.2 Overall Trend

In this section we will present the overall trends and patterns captured from all data collected. All calculated relay RMSE means must be merged into one variable so we can directly inspect it. To achieve this, certain assumptions have to be made about the data. Firstly, the interval-specific RMSE means follow a normal distribution, and secondly, they all are independent from each other. In this case, we can prove both assumptions to be valid. According to statistical theory, the core element of the Assumption of Normality asserts that the distribution of sample means (across independent samples) is normal [17]. Furthermore, since RMSE values depend on specific relay latencies which were never measured concurrently, they must be independent of each other. Consequentially we can use a well-established method of finding normal distribution parameters [18] shown below;

$$X_s \sim \mathcal{N}(\frac{1}{R}\sum_{i=0}^{R}\mu_{si},\ \frac{1}{R-1}\sum_{i=0}^{R}(\bar{\mu_s}-\mu_{si})^2)$$

Where X_s is the variable representing the distribution of all RMSE means for a sampling interval s, R is the number of relays, μ_{si} is the RMSE mean of interval s for the i^{th} relay and $\bar{\mu_s}$ is the average of all sample means.

Graphing the means and their standard error (68% confidence interval) for each sampling interval tested, produces the curve seen in figure 5.3. Overall, its pattern looks similar to those of the curves in figure 5.2. The error between the sampled and continuous latencies increases along with the sampling period, in the range of 1 to 20 minutes. At the interval of 20 minutes, the RMSE value is the curve's global maxima. For periods from 30 to 60 minutes it follows a small negative trend, but mostly remains constant. Another important feature that we can visually inspect, is the standard error constantly increasing along

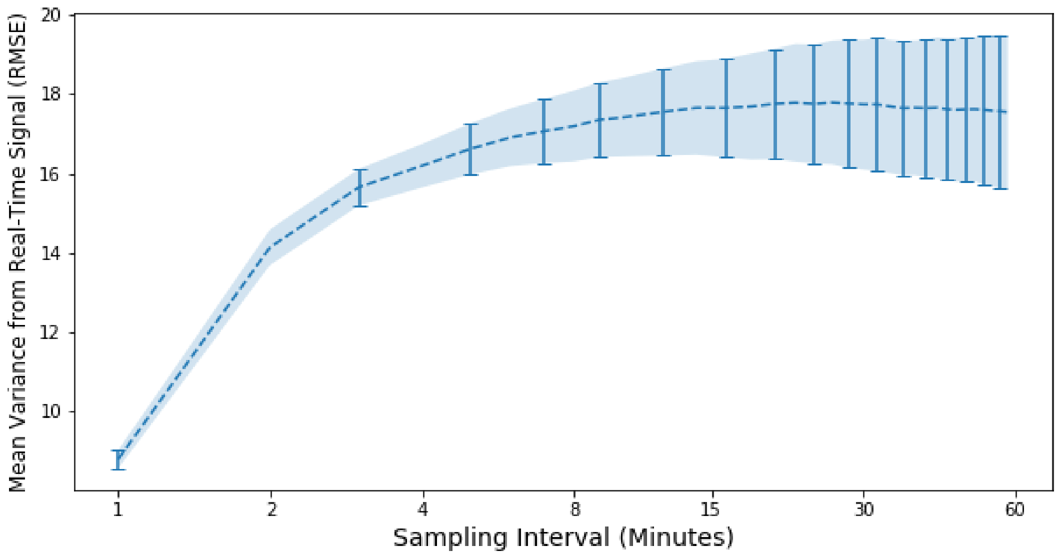


Figure 5.3: *Mean and Standard Error of RMSE calculated using data from all relays.*

with the interval. This means that long sampling periods increase the degree of error variation, thus providing lower levels of precision.

5.2.3 Impact on Tor Network

In order to use the results presented above and infer effects of changing consensus-publishing intervals on the Tor network, the following assumptions must be true:

1. The live latency data collected from each relay is a good representation of Tor relays' temporarily available bandwidth.
2. The sampled latency signal is a good representation of the relays' bandwidth information available to Tor clients at any given point in time.

Both assumptions are known to be true since the latency of a server's response is directly proportional to its available TCP throughput (equal to user perceived bandwidth), as shown in 3.2.1.

Visually inspecting the results obtained in figure 5.3, we can see that a sampling interval of 1 minute, produces the lowest divergence between the continuous and sampled latency signals. This was expected, since sampling using a shorter interval, generally provides more information about the original signal. However, measuring the bandwidth of all Tor relays every 1 minute, is not an easy task. There are less than 10 Tor directory authorities and not all of them publish bandwidths. Thus, the current infrastructure does not support accurately performing measurements using such low periods.

Furthermore, there is almost no change to the error magnitude in the interval range from 10 to 60 minutes. A significant improvement to the accuracy of bandwidth information available to Tor clients, would mean that the publishing period must be reduced below 4 minutes. However, such high publishing frequency is difficult to achieve using the current Tor bandwidth authority infrastructure. On top of this, the increase of network load due to more frequent updates would also negatively impact clients' performance, adding an extra consideration factor when choosing a shorter interval.

By looking at the error improvement rate in figure 5.3, we can be certain that a small reduction in the Tor publishing interval would not improve the bandwidth information accuracy of clients. On the other hand, a large reduction has the potential to do that. However, taking into account factors stated above along with physical limitations present, the resulting overall user performance may be negatively affected. Knowing this, we cannot claim that any reduction to the 60-minute interval used by Tor, would impact users positively.

However, in order to draw definite conclusions, a mathematical formula which provides the relationship between user performance and publishing frequency is needed. This must take all factors into account.

5.3 Performance Evaluation

5.3.1 Relating RMSE to Performance

Throughout this project, we have successfully constructed an error function which explains the variance between live and published bandwidths in the Tor network, given the publishing interval. This is useful as it provides a vital part of the information required to fully calculate the optimum bandwidth publishing frequency, **in terms of client performance**. To evaluate our findings, user performance must be firstly defined within this project's domain.

The Tor network's main objective is achieving user anonymity, as described in chapter 2, and the relay-picking algorithm used by clients reflects this. Since this algorithm biases relay probabilities based on their available bandwidth, for users to achieve stable (optimum) performance they must have access to real-time relay bandwidth values. Any error in the information available to them, would destabilise their connection throughput through the network. This happens because, in case of error, available relay bandwidth could be both overestimated and underestimated. In case of overestimation, client throughput would be lower than the optimal level, whereas in case of underestimation it would be higher. Therefore, we can describe Tor client performance P as a range defined by the bandwidth information error E, depending on chosen publishing interval s. It must be taken into account that all the relationship formulations showed in the next sections are the author's own work unless otherwise stated. This is shown below.

$$P_0[1 - f(E(s))] < P < P_0[1 + f(E(s))]$$

Where P_0 is the optimum performance, and f is a function which converts the RMSE to a percentage factor, adding the error effect on user perceived performance. If there is no error, ie. $E(s) = 0$, user performance P would be at an optimum stable level, ie. $P = P_0$. On the other hand, if there is a difference between the client known bandwidths and the real-time ones, ie $E(s) \neq 0$, then P could take any value in the specified range, making it more unstable.

This project's main objective has been achieved, as the analysis performed has provided the relationship between the absolute bandwidth information error and the publishing interval used; $E(s)$. However f, the function converting the absolute RMSE to performance percentage is unknown, and without it, the publishing interval which maximizes stability of P cannot be calculated. But why? We already know that minimizing the publishing period reduces the error, which in turn results in more stable Tor client performance. Given this information, why is the smallest interval not considered the optimal one? To answer this we must explore its other effects on the base network traffic.

5.3.2 Interval Effects on Tor Base Traffic

Tor bandwidth authorities measure relay throughput and publish results every 60 minutes, as part of the consensus document. This process creates its own constant level of traffic which is always present in the network. If the interval which dictates the frequency of these measurements changes, the level of base traffic induced by the bandwidth authorities will change proportionally. For example, if each relay in the network is tested twice as often, ie. the interval becomes 30 minutes, the base level of Tor traffic will be doubled. Since client traffic is the major cause of fluctuations in the available bandwidth of relays, it is very likely that increased Tor base traffic will also affect client performance.

The level of network traffic induced by bandwidth authorities depends on the amount of data they download, to calculate the throughput of a given relay. Such a measurement consists of creating a circuit through 2 relays, the one under test and a faster random one, and recording the time taken to download a file through them [19]. However, not all tests are performed using a specific file size, and thus there is no standard impact on the network. To be able to approximate the network footprint of a bandwidth measurement, we must generalise its induced traffic. Let T be the total network traffic, in B/s, induced by one bandwidth authority performing a measurement on a relay. For example, if every 60 minutes a 1 MB file is downloaded using a circuit through a specific relay, the overall relay usage would be;

$$T = 2 * \frac{1\,MB}{60\,minutes} = 556\,B/s$$

Here we multiply by 2, since there are 2 relays used for each measurement and the traffic affects both of them equally. Continuing, if we let the average data size used by Tor bandwidth authorities be F (in bytes), then we can re-write this equation;

$$T = \frac{2F}{3600}\,B/s$$

We now have an expression for the base network load caused by measuring a relay every hour. However, multiple bandwidth authorities must measure the same relays at least once per interval, in order to improve accuracy and sustain anonymity. Using this information, we can now construct an expression that gives the load on the Tor network, taking into account the number of bandwidth authorities A measuring the same node and the sampling interval s (in minutes).

$$T(s) = \frac{FA}{30s}$$

Having constructed a formula that describes the Tor network base load in terms of the publishing interval, we can approximate its relationship to client performance. We know that increased base network traffic would only hurt the available throughput provided by relays across Tor, since their load would constantly be higher. Therefore, the required relationship follows an expression such as the following one.

$$P = P_0 * [1 - g(T(s))]$$

Where g is a function with an output range of [0,1). It describes the effect of bandwidth-measuring traffic, on the performance perceived by Tor users. If its output is 0, the traffic $T(s)$ does not have an effect on P, ie. $P = P_0$. Otherwise, the user perceived performance will be a fraction of its optimum level. To approximate g, experimentation would have to be performed on a

Tor-like private network, which provides control over the traffic induced from bandwidth authorities. This however, is out of the scope of this project.

5.3.3 Interval Optimization

Having formulated an expression relating performance with base network traffic, we can now use it to extend our original formula in 5.3.1. The newly calculated range for user performance (P) is shown in the equation below.

$$P_0 * [1 - f(E(s))] * [1 - g(T(s))] < P < P_0 * [1 + f(E(s))] * [1 - g(T(s))]$$

In order to maximise user performance and its stability, there are two factors to consider. Firstly, the error $E(s)$ must be minimized to increase the stability and precision of P. We have proven in the previous chapters that shortening the sampling interval decreases this error. However, this approach also increases Tor base network load ($T(s)$), which creates an optimization problem. The goal is to accomplish the smallest possible range for the value of P (performance stability) while maintaining a high value for P as $T(s)$ increases. Since both sides of the range for P are symmetrical, depending on the magnitude of $E(s)$, optimizing one side will result in the optimal values of all the parameters. The optimization problem is therefore defined as follows:

$$max_s[P_0(1 - f(s))(1 - g(s))]$$

Where $f(E(s))$ has been shortened to $f(s)$ and $g(T(s))$ has been shortened to $g(s)$. Since P_0 is a known constant, we can simplify this equation further by expanding the brackets and minimising the negative term. The optimization problem is transformed to the expression below.

$$min_s[f(s) + g(s) - f(s)g(s)]$$

Through the data analysis described in the previous chapter, the relationship between s and $E(s)$ was approximated successfully, achieving the main goal of the project. Along with a formulation for $T(s)$, two out of the four necessary relations are known. The functions $f(s)$ and $g(s)$ are still unknown and further experimentation is required in order to find their approximations, which is currently outside the scope of this project. However, the formulation of this optimization problem can be used as an outline for future research into the relationship between user performance and publishing intervals.

5.4 Strengths and Limitations

The following is an assessment of the strengths of the solution provided to achieve the project's objectives.

1. One of the main objectives for this project was exploring Tor relays responses to live network traffic and study its effects on the available bandwidth. It is important to view this bandwidth from the perspective of a Tor user, rather than inspecting the specific physical properties of the relay being tested. The use of Ting allowed the research to do this by measuring the latency indirectly through the use of Tor software. This is in contrast to the other available choice, Ping, which would have measured the direct physical link between hosts and servers instead.

2. The use of real Tor data provided a more accurate and reliable representation of relay responses to changing network traffic. Simulated data would have offered a bigger data set to analyse since it would have been less error prone than depending on real Tor nodes, which can go offline at any time. However, this would defeat the purpose of identifying the real trends in relay behaviour.

The following is an assessment of the weaknesses of the solution provided to achieve the project's objectives.

1. The error analysis performed in this research depends on the assumption that, the latency data collected from the chosen relays accurately represent the behavior of the whole Tor network. Even though 32 relays were chosen as an initial sample, only 19 of them provided useful data without errors that could be used for further analysis. Since Tor consists of nearly 7000 nodes, it would be ideal to record relay behaviour from a sample size larger than 19. However, due to time and hardware restrictions, this was not possible. Consequentially, the validity of this paper's findings depends on the chosen sample of relays accurately representing the typical Tor behaviour.

2. Another possible source of error was the use of Ting for recording response latencies of specific relays. Ting, as explained in 3.2.2, measures the path latency between two Tor nodes. Thus, a high-bandwidth node was chosen to serve as an 'anchor' for all measurements, assuming it would not be the bottleneck for the path latency with other relays. This way, any recorded data would be dependent on the relay under test, and not the 'anchor'. However, this assumption cannot be proved and there

is a possibility that the latencies recorded were a skewed representation of the relays' available bandwidth.

Chapter 6

Conclusion

6.1 Future Work

In section 5.3.3, the optimization problem for calculating the optimum publishing interval, in terms of clients performance, was formally stated. It included factors that are thought to have the greatest impact such as the information error $E(s)$, and the resulting network load $T(s)$. However, for the optimization to be possible, there are two further relationships that must be found:

- Function f describes the effect of the clients' bandwidth information error, on their performance.
- Function g describes the effect of the of bandwidth-measuring network traffic, on the performance perceived by Tor users.

These can be estimated by further experimentation. However, using the real Tor network for data collection would not suffice, since control over its authorities is required. The experiments must involve a private Tor network, using simulation software which provides control over many of its properties. The latency data collected as part of this project would be extremely beneficial, as they can be directly applied to the virtual nodes as part of the simulation, providing a more realistic test bench environment. This way, a client's throughput can be calculated using different levels of network load (thus estimating function g), as well as different bandwidth information intervals (allowing estimation of function f).

Since this dissertation is part of a 2-year master's project, these experiments will be performed during the second chapter, allowing the calculation of an optimum Tor publishing interval using the formally stated problem in 5.3.3. The results can be directly used for improving the Tor network's popularity and allowing its users to experience faster (and more stable) connection speeds.

Furthermore, the data collected and analysis performed can also aid in any project involving the study of live traffic effects on network behaviour, in terms of available server bandwidth.

6.2 Concluding Remarks

This project successfully estimated the relationship between the Tor bandwidths publishing interval and the information error present on clients. That error is directly correlated to user performance, and for it to decrease substantially, a publishing interval shorter than 4 minutes is needed. Since such a high publishing frequency is difficult to achieve with the current Tor infrastructure, we can be fairly certain that decreasing the publishing interval would not yield positive results. However, for definite conclusions to be drawn, all factors affecting client performance must be empirically estimated.

Bibliography

[1] A. K. W. R. James F. Kurose, *COMPUTER NETWORKING A Top-Down Approach*, 6th ed. Pearson, 2013.

[2] UN, "Human rights," *Global Issues*, 2006, reference to basic human rights as declared by the UN.

[3] H. V. M. Masoud Akhoondi, Curtis Yu, "Lastor: A low-latency as-aware tor client," Department of Computer Science and Engineering University of California, Riverside, Tech. Rep., 2012.

[4] I. The Tor Project, "History," 2018.

[5] T. T. Project, "Users," 2020. [Online]. Available: https://metrics.torproject.org/userstats-relay-country.html

[6] I. The Tor Project, "Servers," 2020. [Online]. Available: https://metrics.torproject.org/networksize.html

[7] J. M. M. W. Armon Barton, Mohsen Imani, "Towards predicting efficient and anonymous tor circuits," University of Texas at Arlington, Rochester Institute of Technology, Tech. Rep., 2018.

[8] T. T. Project, "Tor specifications," 2020. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

[9] N. S. Frank Cangialosi, Dave Levin, "Ting: Measuring and exploiting latencies between all tor nodes," University of Maryland, Tech. Rep., 2015.

[10] E. B. Rachel A. Orlowski, Frost Hubbard and D. Zahs, "Data processing and statistical adjustment," 2016. [Online]. Available: https://www.ccsg.isr.umich.edu/images/PDFs/CCSG_Data_Processing_and_Statistical_Adjustment.pdf

[11] A. Kirk, *Data Visualisation: A Handbook for Data Driven Design*, 1st ed. SAGE, 2016.

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] M. F. Gobbi, M. Chamecki, and N. L. Dias, "Application of digital filtering for minimizing aliasing effects in atmospheric turbulent surface layer spectra," vol. 42, no. 3, 2006. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2005WR004374

[14] N. R. Center, "Signal filtering," 2008, nondestructive Testing (NDT) is a very broad, interdisciplinary field that plays a critical role in assuring structural components. [Online]. Available: https://www.nde-ed.org/Resources/resources.htm

[15] M. Dodson, "Shannon's sampling theorem," *Current Science*, vol. 63, no. 5, pp. 253–260, 1992. [Online]. Available: http://www.jstor.org/stable/24095490

[16] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature," no. 5, 2014. [Online]. Available: https://www.geosci-model-dev.net/7/1247/2014/gmd-7-1247-2014.pdf

[17] J. T. Mordkoff, "The assumption(s) of normality," 2016. [Online]. Available: http://www2.psychology.uiowa.edu/faculty/mordkoff/GradStats/part%201/I.07%20normal.pdf

[18] J. Jacod and P. Protter, *Gaussian Random Variables (The Normal and the Multivariate Normal Distributions)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 125–139. [Online]. Available: https://doi.org/10.1007/978-3-642-55682-1_16

[19] I. The Tor Project, "How bandwidth scanners monitor the tor network," 2019. [Online]. Available: https://blog.torproject.org/how-bandwidth-scanners-monitor-tor-network

Appendix A

List of Abbreviations

IP	Internet Protocol
ISP	Internet Service Provider
TOR	The Onion Routing
NRL	U.S. Naval Research Lab
TTD	Time To Download
RTT	Round Trip Time
TCP	Transport Control Protocol
UDP	User Datagram Protocol
ACK	Acknowledgement Packet
AS	Autonomous System
ICMP	Internet Control Message Protocol
JSON	JavaScript Object Notation
CSV	Comma Separated Values
RMSE	Root Mean Square Error
MAE	Mean Absolute Error