FasTor: A faster Tor client

Evangelos Dimitriou

Master of Informatics

School of Informatics
University of Edinburgh

2021

Abstract

While Tor provides a high level of anonymity, its performance is significantly lower compared to other main-stream web browsers due to its architecture philosophy. This paper introduces FasTor, an alternate Tor client scheme which aims to improve average-case performance through the Tor network, and increase its stability. Its main idea is to provide users with individually-tailored relay performance characteristics, along with live congestion information, to enable a better-informed circuit selection process. It further gives users the choice of personalizing their performance/anonymity trade-off, for an extra performance boost to those whose anonymity requirements are more lenient. Evaluation of the client performed on the Tor network produced very promising results, achieving maximum performance and stability improvements of 38% and 50% respectively, compared to vanilla Tor.

Table of Contents

Chapter 1

Introduction

The following chapter presents a summary of the work undertaken throughout this project. Firstly, the context and motivation for the proposed Tor client will be introduced, followed by the main contributions and the findings deemed to be useful for real-life applications. This was a strong concern throughout the project since the motivation is based on the ever-evolving privacy needs due to the continuous digitization of modern society. Secondly, the objectives of this paper will be clearly stated, along with a summary of the methods used to develop and evaluate proposed solutions. Finally, a layout of the report's structure will be provided, containing an overview of each chapter in order to familiarize the reader with the upcoming content and facilitate overall understanding.

1.1 Context

This thesis involves research in the field of computer networks. A computer network can be defined as a group of systems linked together for the purpose of exchanging information, with the largest one in existence today being the Internet. Its goal is to seamlessly transfer information between two hosts, which in turn makes everyday activities such as web browsing and instant chatting possible. Here, the focus lies on the Tor network which, in contrast to classical networks, prioritizes the anonymity of its users over other common artefacts such as network throughput and latency. As a result of this philosophy, Tor users commonly struggle with poor network performance, discouraging the network's widespread adoption.

This paper proposes an alternate Tor client scheme which aims to address some of Tor's performance issues.

1.2 Motivation

In conventional routing through the internet, the physical network addresses (IP addresses) of both hosts exchanging information are available to not only themselves,

but also all the intermediate internet routers that network packages travel through. This, along with technologies employed by web-browsers (such as user cookies) enable many online privacy leaks such as:

1. Websites can collect and sell user information, such as search histories, for advertising purposes [1].
2. Organizations, ISPs or even governments can employ firewalls and censor online content [2].

In response to this ever-increasing need for privacy and anonymity online, the Tor network was created. Its main idea is to route internet packages through a large network of relays, using many layers of encryption, such that the destination host does not know the real IP address of the sender, removing the websites' ability to track and collect private user data. Furthermore, intermediate internet routers (e.g. ISP routers and switches) cannot link the source of a network transaction with its destination, allowing users to circumvent content censorship.

As of January 2021, there are an average of 2.2 million directly connecting Tor users at any specified point in time [3]. However, when comparing this number to the most popular web browser's (Google Chrome) 2 billion active users [4], we can clearly see that despite the benefits Tor provides, its adoption is not nearly as widespread as other popular browsers.

While there are many possible reasons behind this massive adoption gap, a defining factor is thought to be the low performance Tor offers. Interactive connections, which account for over 90% of connections in the Tor network [5], need low latency and high bandwidth to provide a smooth and pleasant experience for the user. Due to its architecture however, network packets must travel longer-than-normal distances to and from their destination. This increases the data travel time and decreases the effective throughput of a connection. Nevertheless, according to its developers, the Tor network's core performance problem actually is its performance variance which is particularly bad for user experience. People can get accustomed to predictable and stable performance levels but can get very frustrated if their connection speeds are sometimes bad [6].

The main reasons behind Tor's performance instability are thought to be; its lack of congestion control and network load balancing. Both issues are topics of undergoing research since their mitigation is likely to hugely improve the Tor browser's user experience and enlarge its client base.

1.3 Objectives

1. The main objective of this project is to address Tor's performance issue by proposing and evaluating an alternate client scheme. FasTor aims to improve Tor user experience by prioritizing network performance while maintaining user

anonymity, as much as possible, during online transactions. A change to the Tor client protocol is very easy to evaluate and deploy, since it is completely decoupled from Authority and Relay software infrastructures. As a result, a new scheme can be seamlessly deployed to the client software and used to route connections through Tor.

2. The proposed client should also provide users with the choice to further improve performance by lowering their anonymity. Tor provides a very high anonymity assurance, but at the cost of day-to-day internet usability [7]. Since different users have various requirements, an adjustable client scheme would be better suited for wider use. For example, users who simply wish to circumvent censorship in order to access their favourite websites, may prefer a more enjoyable user experience rather than the highest anonymity assurance.

3. Lastly, FasTor also targets to improve congestion control issues associated with the vanilla Tor client protocol, by employing opportunistic relay congestion measurements to avoid highly congested routes, and thus alleviate pressure from heavily used relays. This would theoretically improve performance stability through Tor in an average use case.

1.4 Methodology

1. FasTor is implemented in Python3.7 leveraging Tor's control port functionality through the open-source package Stem [8], instead of being directly implemented inside Tor source code. This design choice was made because Python provides much faster development times when compared to languages such as C [9], and the main objective of this paper is to tune and evaluate the FasTor scheme rather than develop it directly for end-users.

2. FasTor users are given the option to tune two client parameters, a PERFORMANCE_ANONYMITY (PA) parameter and a RELAY_POOL_SIZE parameter, both changing the client behaviour to give more emphasis on performance or anonymity. Their effects will be described in detail in later chapters.

3. The performance of FasTor is evaluated against the vanilla Tor client scheme by performing TTLB (Time To Last Byte) measurements when sending requests to the 500 most visited websites in the world [10]. These tests are performed over prolonged periods of time, so that general performance patterns can be more visible. Furthermore, for these tests to be fair, the Tor vanilla client scheme is also implemented using Python3.7 and Stem, and used for comparisons.

1.5 Structure of Document

This section includes a break-down of the paper, giving a quick overview for each chapter.

Chapter 2 - Background: This chapter introduces Tor's basic functionality, along

with the reasons behind its low performance. It then mentions previous contributions on this topic, some of which provided inspiration for the development of the proposed scheme.

Chapter 3 - FasTor Scheme: In this chapter, the theory behind FasTor's functionality is introduced, and reasons behind design choices are explored in depth.

Chapter 4 - Methodology: The Methodology chapter walks the reader through general implementation details for the software required to realize the proposed scheme. Example usage of the libraries developed are also provided.

Chapter 5 - Scheme Evaluation: Chapter 5 describes the tests performed for evaluating FasTor's performance against that of vanilla Tor. It then presents and analyses the results along with discussing their real-world implications on user-experience and anonymity.

Chapter 6 - Conclusion: The last chapter proposes future investigation necessary for the proposed scheme's deployment, while also suggesting possible improvements on the protocol itself.

Chapter 2

Background

Having a basic understanding of the field we are discussing is important to properly follow the steps described in this paper and their implications. This chapters provides an overview of Tor's functionality, describes the performance issues it is currently facing in detail, and also introduces previous work aimed to tackle them.

2.1 Tor Basics

Tor routes user internet traffic through a network of nearly 7000 volunteer-operated relays, using multiple layers of encryption [11]. These act as intermediary routers which hide some of the sender's details (e.g. IP address) from the target web server, essentially anonymizing online transactions.

Since the Tor network is based on a volunteer scheme, its protocol must treat all relays as not-trusted parties, in case some are operated by malicious users who attempt to collect Tor user information. The only trusted parties in the network are a few Directory Authority servers, whose task is to collect relay information (such as fingerprints and IP addresses) and publish a consensus which is made available to all Tor clients. The clients are tasked with using the information provided by the latest published consensus, to choose a set of relays and construct a Tor 'circuit' through them. These circuits act as network paths which carry network packets from the sender, through the chosen relays and finally to the destination server.

By default, every Tor circuit consists of 3 relays:

1. The first relay in every circuit is called the guard. This acts as the clients' entry point to the tor network and is responsible for forwarding network packets from the user to the next node in the 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 next relay in the circuit is termed as the middle, acting as a 'middleman', and is responsible for receiving traffic from the guard node and forwarding it to

the exit node. The only available information to this server is the address of the guard and exit nodes.

3. Finally, the exit relay forwards packets received from middle nodes to their final destination (for example twitter.com). Following the pattern, the only IP addresses this Tor relay is aware of are those of the middle node and the packet's target.

As stated before, clients are responsible to choose 3 relays for the construction of their Tor circuits, which are changed every 10 minutes by default [12]. Once a guard node is chosen, it has a primary lifetime of 120 days [13], meaning the client will reuse the same guard for all future circuits, whereas the middle and exit relays are chosen at random just before circuit creation.

This protocol achieves a very high degree of anonymity for Tor users, since no other party is aware or able to predict the relays composing a client's circuit.

2.2 Performance Issues

This section discusses the reasons behind Tor browser's network performance being significantly lower than the one expected from a typical browser.

In conventional internet routing, network packets are routed through a very large and complicated network of internet switches and routers before they reach their destination [14]. However, on top of this conventional routing, Tor redirects packets through its own series of relays, in addition to applying many layers of encryption. This naturally adds an overhead in the travel time between sender and destination, but it is not what is hurting the browser's user experience. According to its developers, Tor's core performance problem is actually performance variance [6].

Tor depends on volunteer-run relays for its anonymizing functionality, and hence there is no guarantee these relays will offer high banadwidth or low latency, two traits that are very important for a pleasant and smooth user experience. Today, a Tor client can experience very decent network performance when it chooses a circuit consisting of fast and uncongested relays. However, since middle and exit relays are chosen at random, there is always a probability that a slower or congested relay will be chosen, effectively bottlenecking performance. Since variance in performance has a massive negative impact on user experience [6], this is one of the main factors which slows Tor's wider adoption.

The vanilla Tor client scheme has attempted to mitigate this issue by introducing a non-uniform probability to relay selection. Bandwidth Authority servers are responsible for performing bandwidth measurements on all Tor relays and report the results inside the consensus, which is then used by clients to weigh their choice probability accordingly. Utilizing this scheme, the clients are more likely to choose and use faster relays than

slower ones, improving the average-case performance through the network. However, Tor's core performance problem still exists, and some further contributors are stated below:

1. Lack of knowledge about live congestion levels does not allow clients to avoid congested relays when creating circuits.
2. Relays with a high bandwidth are not necessarily better for use by all clients, since their geolocation can have a big impact on latency and effective throughput [5].
3. Load balancing around Tor is not very efficient since Bandwidth Authorities do not fully account for the effects of geographic diversity [6].

2.3 Relevant Work

2.3.1 LASTor

Tor's standard client scheme adds a significant overhead latency in all user connections since it routes packets through an increased physical distance before they arrive at their destination. If one of the circuit relays is located in a different continent, this issue is exacerbated. Akhoondi et al. [5] proposed LASTor, a Low-Latency AS-Aware Client which aims to improve network latency and reduce the risk of traffic correlation attacks. It does this by accounting for the geolocations of relays when performing relay selection, but in practice, it reduces the number of possible circuits that can be constructed, thus lowering anonymity. It was however shown that it offers an improvement in performance in comparison to the default Tor client by reducing median latencies by 25%.

2.3.2 CAR

Wang et al. [15] proposed CAR, a Congestion Aware Routing client, which intelligently selects Tor circuits with low levels of congestion. The default Tor client does not take into account relay congestion levels when performing path selection, since this information is not made available. Hence, this results to drops in performance when clients randomly choose a congested path through the network. CAR solves this issue by sampling round-trip times (RTTs) when performing connections or building Tor circuits and uses these measurements in relay-selection to choose the fastest out of three random circuits. Furthermore, any active circuit whose mean RTT is over a prespecified threshold is dropped, alleviating pressure from overloaded nodes.

An important disadvantage of CAR is the lower anonymity assurance it offers. RTT measurements can technically be manipulated by malicious nodes, hence affecting the clients path selection. Geddes et. al [16] suggested that the use of these measurements for latency improvements results in an increase in the effectiveness of latency-based attacks.

2.3.3 PredicTor

Barton et al. [17] proposed a path selection scheme which uses a Random Forest classifier to predict performance of randomly chosen Tor circuits and enables clients to drop slow Tor circuits before they are actually used. PredicTor achieved this by measuring many download times (TTLB) of a file from a common server, across many different Tor circuits, and classifying them as 'slow' or 'fast' according to a set time threshold. Using this measurement data, Barton et al. were able to train an ML classifier to predict whether any given path through Tor would be fast or slow.

The classifier model is based on the assumption of a relationship between download times and the consensus bandwidth as well as the network location of each relay. Using the consensus bandwidth and the location information as relay training features, they utilized a Random Forest classifier to predict speeds of circuits chosen as per standard Tor relay selection. Their protocol dictates: If predictions are over a predefined threshold, the circuits in question are to be used by clients, otherwise they are dropped, and the testing loop repeats. This proposed scheme, when simulated in Shadow [18], achieved as much as 23% and 13% median performance improvement compared to Vanilla and CAR clients respectively.

Chapter 3

FasTor Scheme

3.1 Proposal

The main objective of the FasTor client is to decrease the performance variance present in vanilla Tor and improve user experience. To achieve this, the scheme aims to lower the probability of clients constructing circuits which contain under-performing relays, by extending the Vanilla Tor client functionality and adding an extra step between relay-selection and circuit construction.

3.1.1 Relay Pool

When picking a new circuit, FasTor utilizes Tor's standard protocol of relay selection using consensus-published bandwidths as probability weights and chooses two pools of N middle and exit nodes respectively. Continuing, all chosen relays are assigned a score which reflects their client-perceived performance levels, and the highest rated ones are selected from both groups to be the middle and exit relays in the new circuit. This process is visualized in figure 3.1, with an example pool size of 3. This pool-based selection scheme reduces the probability of clients using under-performing relays, since they are likely to be discarded before circuit construction, and it effectively smooths the general-case performance curve through Tor's network.

The size of the pool is a tunable parameter. Using larger sizes would provide clients with a higher number of fast relays and thus magnify FasTor's effect on performance. However, this would also potentially decrease the anonymity assurance provided to users. For example, assuming the client-assigned relay scores are directly proportional to their published bandwidths, the probability of FasTor clients choosing low-bandwidth nodes would further decrease. This effectively lower the clients' choice entropy, making it easier for malicious third parties to predict the relays a target will use.

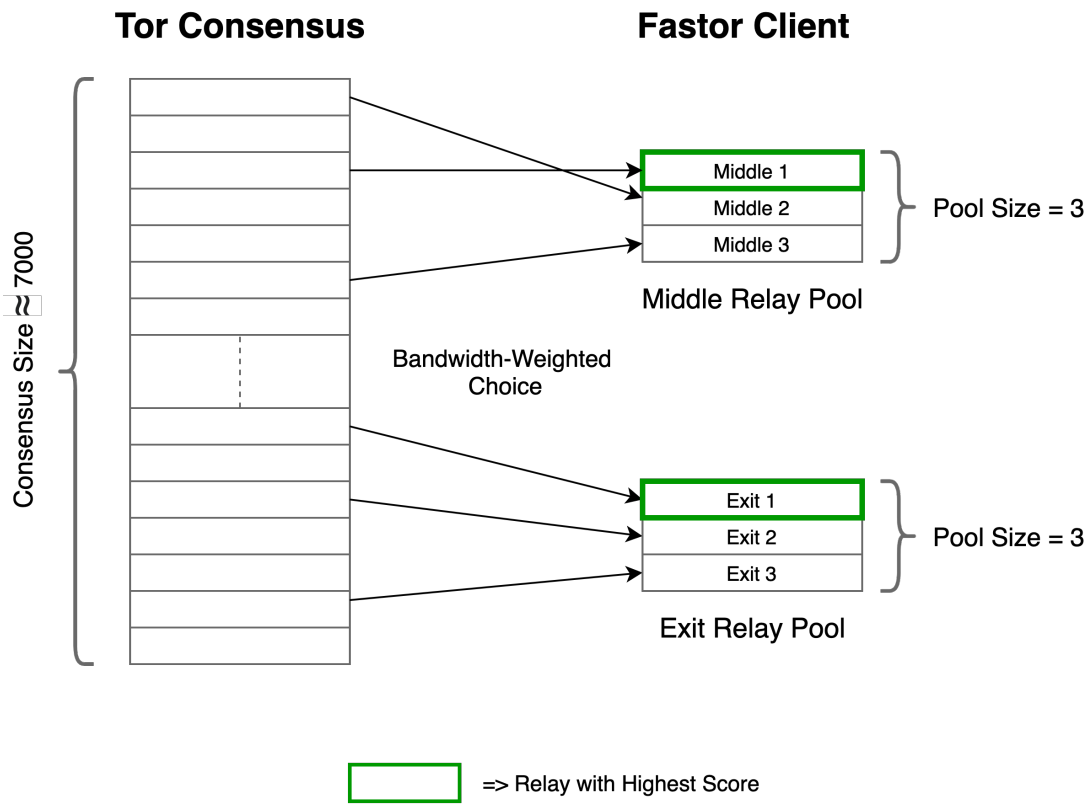

Figure 3.1: FasTor Relay Pool Selection Visualization

3.1.2 Score Metric

The network's hourly published consensus provides bandwidth information for all relays using active measurements by central servers, a technique which gives rise to a handful of issues [6]. Firstly, there is no account for users' geographic diversity, leading to location-based bias. Secondly, such a system cannot handle rapid changes in capacity or load, leading to some nodes being over-congested, and others being underused.

FasTor's score metric aims to mitigate the aforementioned issues. For the relay pool scheme to improve performance, the rating of relays must provide valuable information about their current performance state in respect to the client. This should take into account both previously known throughput/stability data and live congestion levels, information that is not available in Vanilla Tor. Therefore, the score metric is calculated from two data sources collected directly by individual clients.

3.1.2.1 Historical Score

As mentioned above, publicly available relay bandwidth levels do not accurately reflect their performance provided to all users around the world. To solve this issue, FasTor collects user-specific performance data from the network in the form of TTLB measurements every time the utilized guard node is changed. Since clients use specific guards for long periods of time, the data collection process does not require frequent

updates and no dramatic increase in network traffic.

To isolate performance information for a specific relay, FasTor constructs a 2-hop circuit through the client guard and the relay under investigation, and consecutively downloads a prespecified file from a target server while recording TTLB. This way, recorded throughput and stabilities of Tor nodes do not contain location bias, since all measurements are performed by the client using similar paths through the network. Once FasTor completes measurements for all published network descriptors, the mean and spread of TTLBs are used as relay-specific scores. Mean TTLB provides the effective throughput a user is expected to experience, while its deviation gives an insight on a node's general stability (or jitter).

While the TTLB mean can be calculated from the collected data to provide a relay's expected throughput, there are various statistical measures that can be used to provide a throughput stability rating, such as variance, mean absolute deviation and median absolute deviation. For testing purposes, all mentioned metrics were used as relay-scores and evaluated against Vanilla Tor.

3.1.2.2 Live Score

Historically collected relay data provide general information about expected user-perceived performance, however in practice this will greatly vary according to levels of live congestion. In addition to using historically determined scores to perform relay-selection, FasTor performs opportunistic measurements on all nodes inside the chosen relay pool in order to infer their congestion levels.

When pool selection has been completed, FasTor constructs 2-hop circuits through the active guard and all relays under consideration. It then downloads the file used for historical data collection from the same target server, and records TTLB through all circuits. This, when compared to past measurement statistics, provides live congestion information on individual relays and allows clients to choose the highest performing relay at specific points in time.

3.1.3 Circuit Selection

When a new circuit must be constructed and the relay pool is populated, FasTor averages historical and live relay scores to calculate final ratings and order the selected nodes by their expected performance. Finally, the best middle and exit nodes, along with the current guard, are used to construct the next circuit.

Vanilla Tor reuses the same circuit for TCP streams for a span of 10 minutes, as long as it does not fail. At the end of its lifetime, it is replaced by repeating the same circuit selection process. FasTor follows the same pattern. However, it also allows clients to carry forward previously highest rated relays into subsequent relay selection pools, asserting that the performance of future circuits is not likely to drop. Of course, this is

not 100% certain as future congestion could alter the score of a previously highly rated relay, nevertheless it further mitigates the performance variance Tor clients face at the moment.

3.1.3.1 PA Parameter

This model of circuit selection could potentially provide optimum client performance through Tor, given the state of the network. However, it negatively impacts user anonymity levels, since clients could potentially end up using the same "best" relays for prolonged periods of time, making it easier for malicious third parties to de-anonymize their traffic. To mitigate this issue, FasTor introduces a Performance-Anonymity (PA) parameter.

The PA parameter allows the user to choose the extent at which FasTor prioritizes performance over anonymity. Its value lies withing the range [0,1], and it represents the probability of the client carrying forward the previous highest rated middle and exit relays on subsequent circuit selection processes.

- A value of 0 signifies that FasTor will completely replace the relay pool every time a new circuit must be chosen. As a result, minimum performance cannot be guaranteed across different circuits, since there is a significant probability of all selected relays in the pool being 'slow'. However, at the same time, this guarantees higher anonymity assurance, since using the same relays consecutively is highly unlikely.
- On the other hand, a value of 1 guarantees the persistence of the highest rated relay across consecutive circuit selections, setting a minimum performance level. Theoretically (not taking congestion into account), this would provide purely increasing performance levels as time progresses. Nevertheless, this lowers anonymity.

The probability of both middle and exit relays being carried forward, and possibly be re-selected is p^2, where p is the value of the PA parameter.

Chapter 4

Methodology

4.1 Client Development

The FasTor client was developed from scratch in Python3.7, using Stem [8] to communicate with the background Tor instance, through its ControlPort. To enable fairer performance comparison tests between the standard and proposed clients, the vanilla scheme was also implemented from scratch using the same technologies. Both clients are included inside a standalone Python package named *FasTor*, with the vanilla client following Tor's path specification [12], and FasTor extending it.

An event-driven multithreaded system was developed to achieve Tor's functionality and perform parallel actions such as:

- Retrieving the latest published consensus from Tor's Directory Authority servers every 60 minutes.
- Updating Tor client circuits after their lifetime has concluded.
- Accepting user calls to perform web queries through the currently utilized circuit.

The implementation of this component proved very difficult, since there were many bugs induced by the parallel architecture. For example, when two circuit construction calls happened in parallel, the system produced an internal error and only one was constructed. This was overcome by the use of locks on event calls, which allowed previous event calls to terminate before the subsequent ones ran.

Due to performance concerns, the *FasTor* Python package operates utilizing only two threads.

- The main thread attaches the Python client to the background running Tor instance upon initialization, and subsequently executes network requests through Tor using the currently available circuit.

- The second thread is active only while the client is attached to Tor. It continuously checks state conditions depending on the active client scheme, (eg. validity of consensus, or life remaining in the current circuit) and if necessary, performs actions to update the state of the client (eg. replaces the previously used consensus with a newly downloaded one).

The Python client uses *pycurl* [19] and its PROXY option, to create web requests and route them through Tor.

4.1.1 Limitations

Due to limitations of STEM's functionality, the process of collecting live relay congestion information was not truly parallelized. If the proposed scheme was to be developed for production, the relay selection process (when switching circuits) would have to run in the background while the user is browsing the web with the previously constructed circuit. This would provide users with a more pleasant experience, making the process of switching circuits seamless.

However, STEM does not allow the use of two different Tor circuits in parallel, meaning that web requests must terminate before subsequent ones can take place using other circuits, causing problems when attempting to perform relay selection and requiring background congestion measurements. This problem was solved by introducing of an event-queue system, where each thread-unsafe action performed by the client is added in a task-queue and performed in series. This does not affect the evaluation of FasTor, since queries (and their measurements) are only performed when the circuit selection process has finished and the new circuit has been constructed successfully.

4.1.2 Example Usage

The FasTor Python package was designed to be easy-to-use, so as to make the process of developing testing software easier, and to enable reproducibility of results. This section will provide some short Python code snippets that initialize and use the developed client with different setups. To run this code, the FasTor package, along with its dependencies, must be installed on the system, and a Tor process must be running in the Background. This was tested on Debian Linux distributions.

4.1.2.1 1. Vanilla Scheme Usage

Listing 1: Retrieval and Usage of Vanilla Tor Client

```
1 from FasTor.client import getClient
2 from FasTor.client.client import ClientException
3
4 client = getClient('vanilla')  # Get vanilla Tor client
5 client.attach()               # Attach client to background
  ↪ Tor instance
6
```

```
# Send Web Query to youtube.com
try:
    result = client.query('youtube.com')
except ClientException:
    print("Could perform query")

contents = result.getData().      # Web page contents
time_lapsed = result.timeTaken()  # TTLB (seconds)
client.detach()                   # Detach client
```

4.1.2.2 2. FasTor Scheme Usage

Listing 2: Retrieval and Usage of FasTor Default Client

```
client = getClient('FasTor')  # Get Vanilla Tor client
client.attach()  # Attach client to background Tor instance
# Send Queries
# ...
client.detach()                   # Detach client
```

4.1.2.3 3. Modified Parameters

Listing 3: FasTor Usage with Modified Parameters

```
# Get FasTor client with custom parameters.
# 1. Set pool size to 3 (default=5)
# 2. Set Performance Anonymity Parameter to 0.5
 ↪  (default=0)
# 3. Set Score Metric to variance (default=mean)
client = getClient('FasTor',
                   pool_size=3,
                   pa_parameter=0.5,
                   score_metric='variance')

client.attach() # Attach client to background Tor instance
# Send Queries
# ...
client.detach() # Detach client
```

4.2 Historical Data Collection

This section talks about the collection of relay-specific data for the calculation of their individual score ratings. The tests collect TTLB download times of a specific 1kb file from a target web server, using 2-hop Tor circuits consisting of the relay under investigation and a common client guard. Performing this for every node in the network provides indications of their performance as experienced by the user. To ensure higher precision, TTLB was measured 10 consecutive times using each 2-hop circuit and

their mean along with other statistics, such as variance and median absolute deviation, were calculated. A file size of 1kb was selected due to concerns about increased traffic generation if the client scheme was to to scale to a large number of users. Furthermore, this small file size makes path latency the main contributing TTLB factor, which is beneficial due to network latency being essential for user experience in interactive web applications.

4.2.1 Collection Software

The software collecting relay-specific TTLBs was deployed on a Raspberry Pi 3b+ for a period of 24 hours on 27th of February 2021.

- The relay testing software was written in Python3.7, utilizing Stem for communication with the background-running Tor instance and the creation of custom circuits.
- The TTLB target file of size 1kb was hosted on a custom Apache2 webserver running on the same hardware.

The process starts by retrieving the latest consensus from Tor's Authority servers and collecting all relay fingerprints in a list. For each relay entry, the program constructs a 2-hop circuit using a constant guard node. When a path is built successfully, the 1kb file hosted on the Raspberry Pi is downloaded through it, and its TTLB is recorded. Once all relays in the consensus have been tested, this process is repeated for a total of 24 hours which resulted in each relay being measured roughly 12 times. ($\approx 84,000$ datapoints)

4.2.2 Result Presentation

This section showcases examples of TTLB statistics collected from Tor relays using the software introduced in the previous chapter.

Figure 4.1 visualizes 5 measurements from randomly chosen relays of varying performance. The plot on the left displays the raw, consecutively measured TTLB download times, while the second one summarizes their results in terms of mean and deviation, shown as a black markers and coloured error bars respectively. While all nodes achieved sub-second download times, there are two clear sub-groups with different performance characteristics. Circuits through relays 1 and 4 provided TTLBs of roughly 0.65 seconds. This is considerably more than relays 2, 3 and 5, which downloaded the target file consistently under 0.4 seconds. From these results, it is also clear that TTLB jitter (shown as data deviation) is not directly related to node throughput. Even though relays 1 and 2 achieved very different TTLBs when requesting the same file, their jitter is very similar. Same can be said for relays 4 and 5.

As mentioned earlier, Tor users are not only negatively affected by low speeds, but also by performance variance. This begs the question: Which characteristic is more useful when rating relays in terms of user-performance? A node with a low TTLB mean can

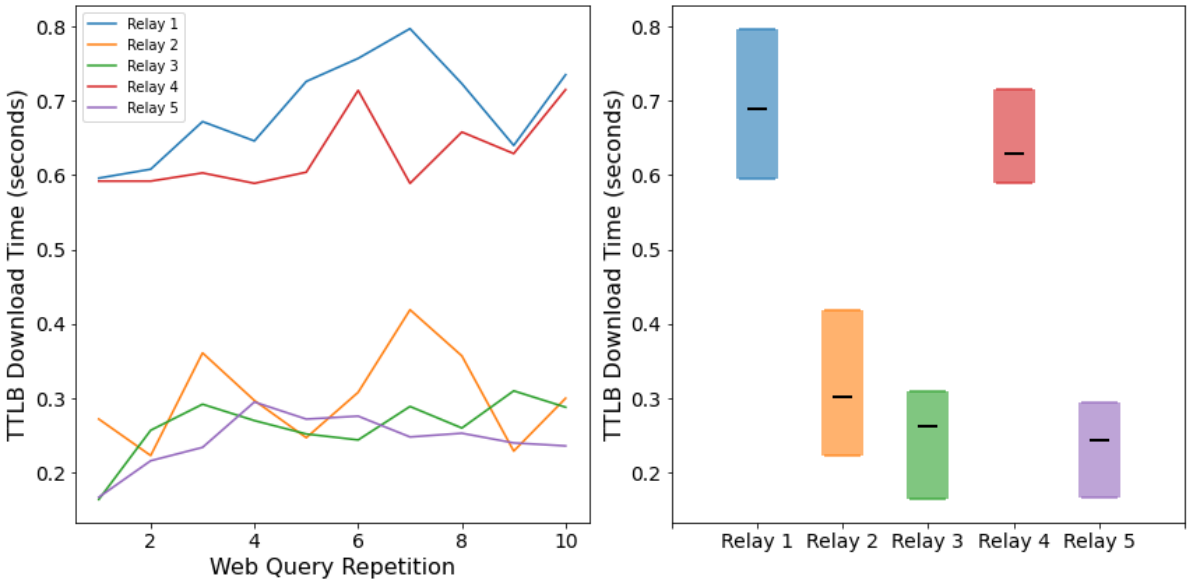

Figure 4.1: TTLB Results for 5 Different Relays

provide faster average download times (eg. Figure 4.1 Relay 2), but on the other hand, nodes with low deviation provide a more stable user experience (eg. Relay 4). To solve this dilemma, FasTor was built to support various relay statistics as score metrics.

4.2.3 Relay Statistics Discussion

Tor Authorities publish all nodes' measured bandwidths inside the consensus, which are used by clients as weights during relay selection. FasTor seeks to gain an advantage by providing users with extra relay characteristics, specifically suited to them. However, if these new statistics do not provide useful information (that is not already known), then FasTor's benefits are nullified. This section focuses on the relationship between the client-collected statistics and their publicly available bandwidths.

Figure 4.2 visualizes this relationship in four separate plots, each using a different score-metric. As expected, the general pattern for all graphs shows negative correlation, that is, relays with higher bandwidths tend to provide lower TTLBs, with less jitter and vice versa. However this is clearly not always the case, as there are many instances for all statistics where higher-bandwidth nodes provide worse TTLB results. The statistic with the highest correlation to bandwidth is variance, while the one with the least is the mean.

Since consensus bandwidths are measured by central authorities, they do not accurately reflect relays' true performance for all users around the world. FasTor's measured characteristic provide each user with performance information specifically suited to them. Analysing the results above, the mean TTLB score seems to provide the most useful information about relays, having the lowest correlation. Furthermore, two node performance-groups can be distinguished at 0.3 and 0.6 seconds respectively, possibly

reflecting their geolocation in terms to the client. Relays that are further away, even though they might have higher bandwidths, tend to provide worse TTLBs for small-sized-webpages.

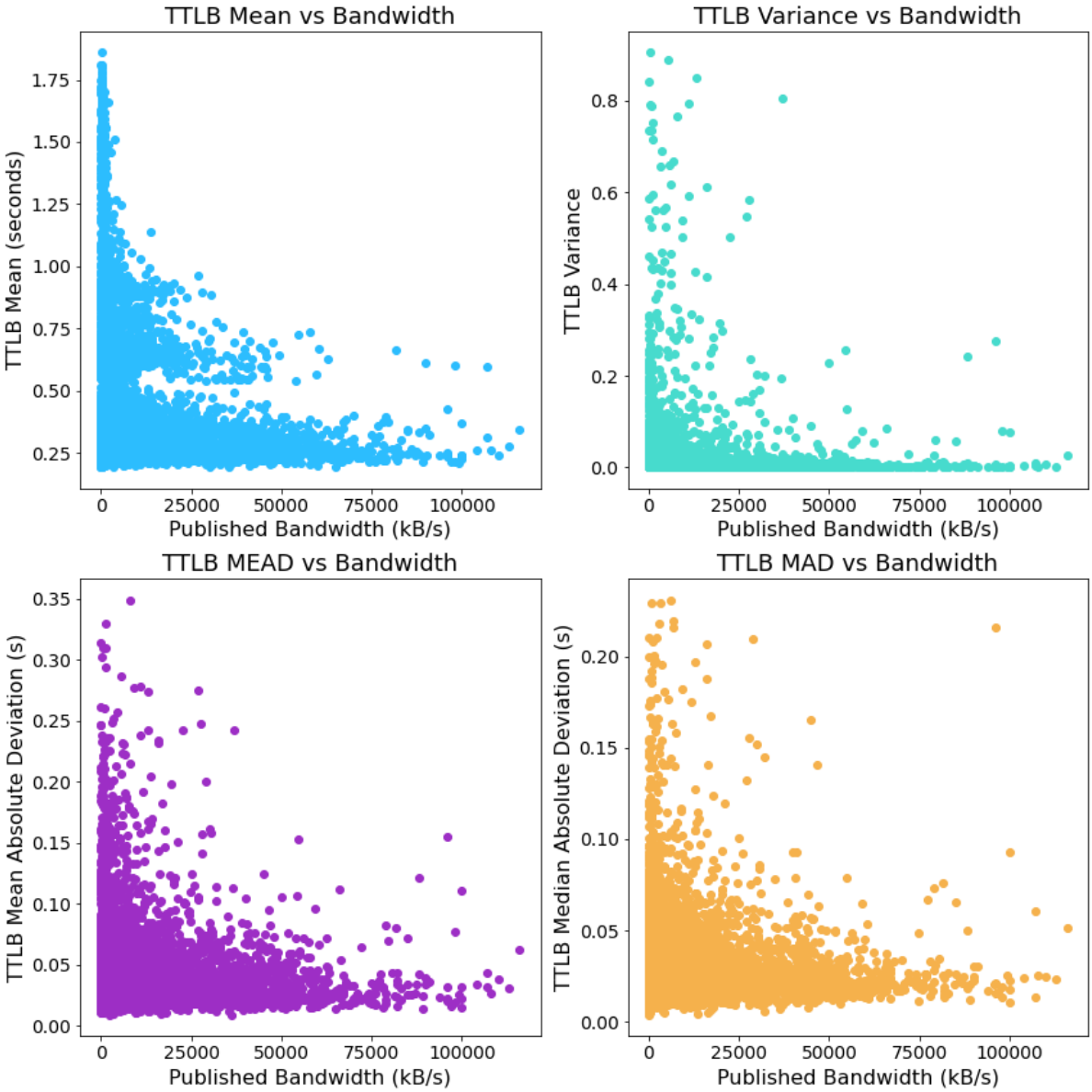

Figure 4.2: Relay Statistics vs Consensus Bandwidth

Chapter 5

Scheme Evaluation

This chapter starts by describing the tests performed to evaluate FasTor's performance when compared to vanilla Tor. Secondly, the results obtained by various scheme setups are presented and analysed to produce meaningful conclusions about the proposed solution's benefits on real-world user performance, while also commenting on its anonymity impact.

5.1 Evaluation Tests

Vanilla Tor, along with FasTor's different score and PA parameter setups, were evaluated by collecting TTLB times when downloading the top 500 most popular websites globally. This traffic simulates one generated by an average browser, and thus provides meaningful information about the performance users would experience if FasTor was deployed in the real world.

Each scheme received a list of the most popular domains and downloaded their web pages in sequence, repeating the process for roughly 12 hours. For the sake of this test, both vanillas's and FasTor's circuit renewal intervals where changed to 1 minute (instead of $\approx$ 10 minutes on Tor browser). This allows the circuit selection process to be performed 10 times more frequently, essentially simulating almost 120 hours of real-life Tor usage. The tests were performed on a Raspberry Pi 3b+ and ran for a total duration of 10 days, since some had to be repeated, on March 15th 2021.

The vanilla client was first evaluated, followed by the various FasTor setups. The list below includes different parameter values used as part of this test:

- Pool Size: 5. *NOTE: Only one pool size was used for performing these tests due to time constraints. Five is a medium pool size which can provide a good indication of the potential benefits of FasTor, without being too traffic-heavy on the client-side*
- Score Metrics: mean, variance, median absolute deviation (mad)
- PA Parameter: 0, 0.5, 1

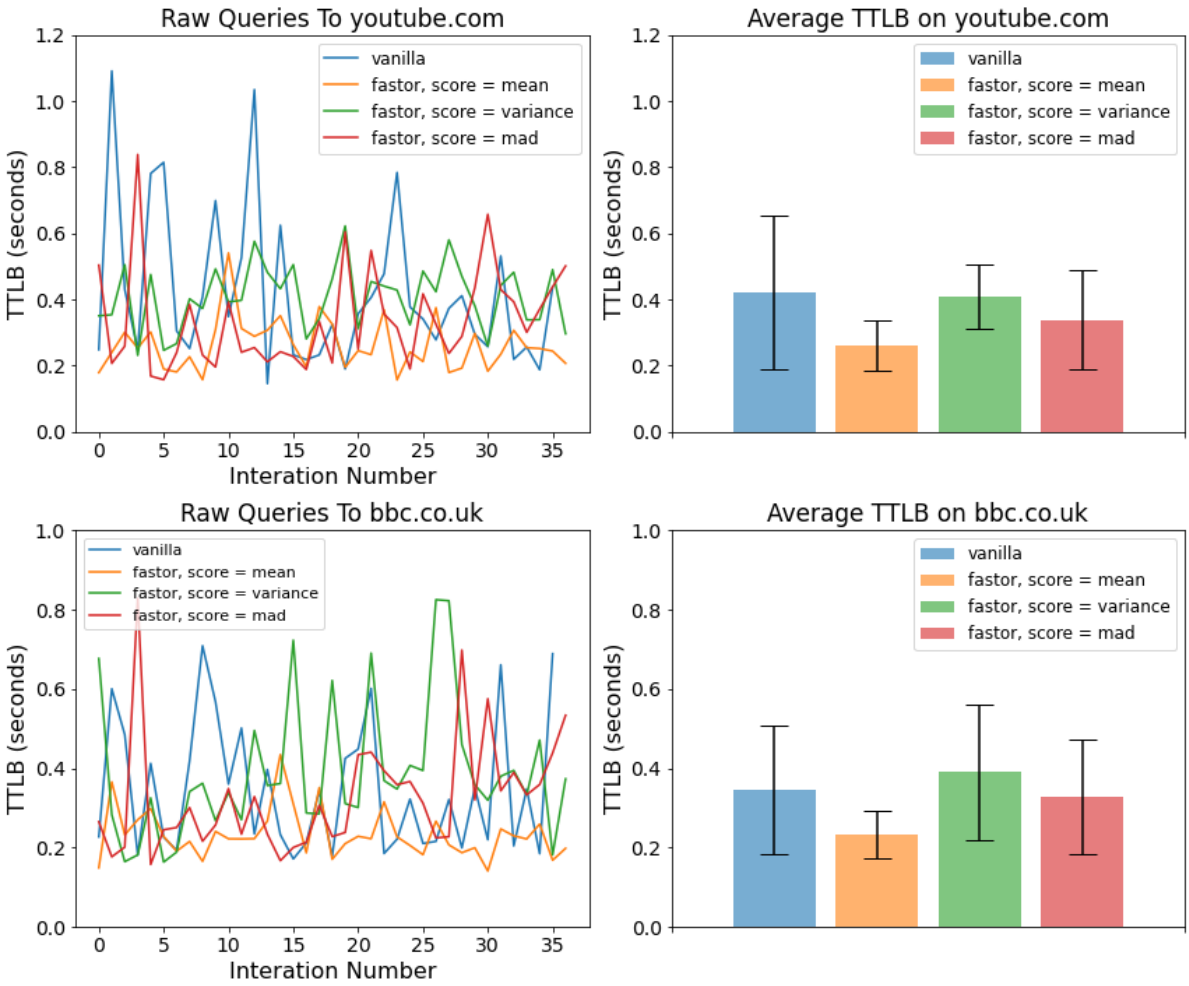

Figure 5.1: Mean TTLB Download Times of youtube.com and bbc.co.uk, for the Vanilla and FasTor Schemes, using different score metrics

5.2 Evaluation of Results

Figure 5.1 visualizes example TTLB results obtained by downloading two webpages at different points in the day, using the vanilla Tor and FasTor clients (including all score metrics). The plot on the left depicts raw query TTLB measurements, while the one on the right shows their mean. Their jitter is visualized by black error bars displaying their standard deviations.

The vanilla Tor client achieved a mean TTLB of 0.4 and 0.35, with a significant jitter, when downloading Youtube and BBC webpages respectively. Variance and Median Absolute Deviation (MAD) relay-scores provide FasTor with performance and jitter similar to vanilla's, when inspecting both pairs of results. The scheme providing lowest TTLB mean and jitter is Fastor, with the use of mean as score metric. In both cases, it achieves nearly 30% lower mean TTLB than vanilla, with less than half of its jitter.

These are promising results, but to further generalize real-world performance, all 500

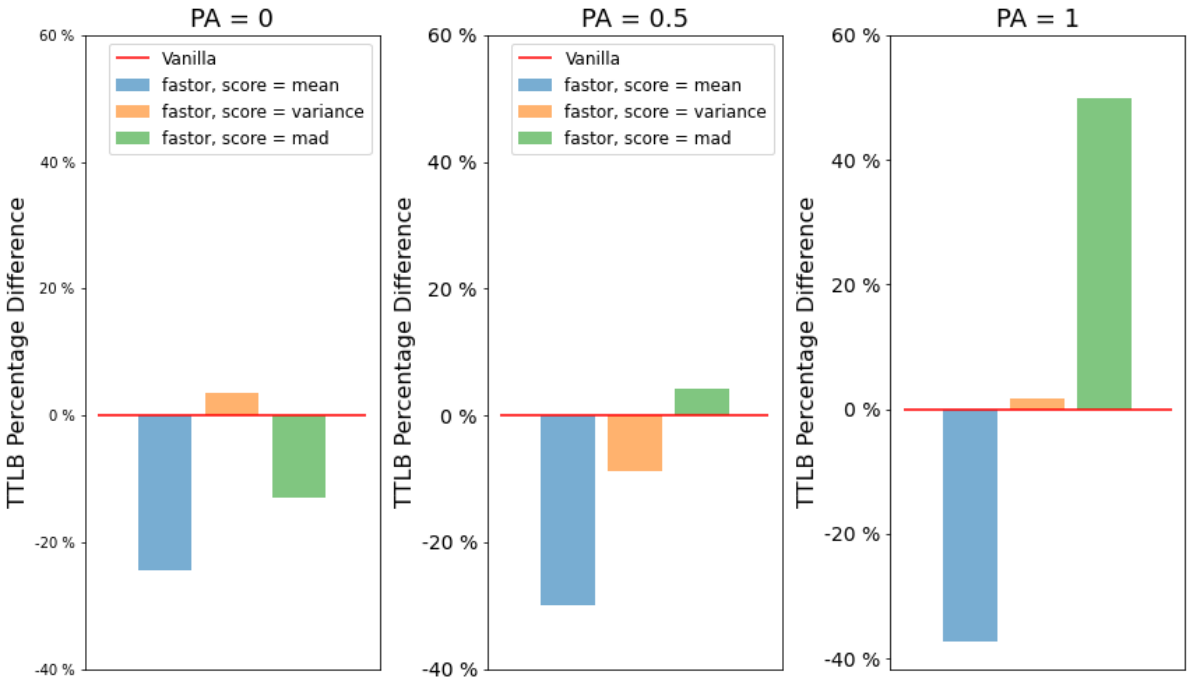

Figure 5.2: Mean Percentage TTLB Difference between Vanilla Tor and FasTor for Different Score Metrics and Performance Anonymity Settings, collected across the world's 500 most popular websites

domains must be taken into account. However, TTLBs measured when querying different domains cannot simply be averaged, since their webpages have varying sizes. Furthermore, the data cannot be normalized according to page size, due to a major download delay factor being path latency (e.g. 1kb and 10kb files could have an almost identical TTLB). To overcome this issue, the different schemes were compared on a domain-basis. For each Fastor setup, the achieved mean TTLB of every website, along with its standard deviation, were directly compared to the respective data obtained from vanilla Tor. The mean calculated percentage differences for every possible FasTor setup are visualized in figure 5.2.

Figure 5.2 shows the mean percentage TTLB difference between FasTor schemes with varying setups, and Tor's vanilla client. Each plot, beginning from the left, displays performance results obtained by all three score metrics for a single PA parameter value. This value increases across graphs. Furthermore, these performance results can be visualized as throughput gain in figure 5.3, for more intuitive comprehension. The performance stability, or jitter percentage difference, produced by each setup is displayed in figure 5.4. A lower jitter means higher performance stability.

- **Mean as Score:** Overall, the best performing scheme is FasTor using TTLB mean as relay score metric, and a PA parameter of 1. This was expected since recorded mean-TTLB relay characteristics were less correlated to previously known bandwidth levels, hence providing more useful information during relay-selection. A PA parameter of 1 means the fastest-picked relay is never discarded,

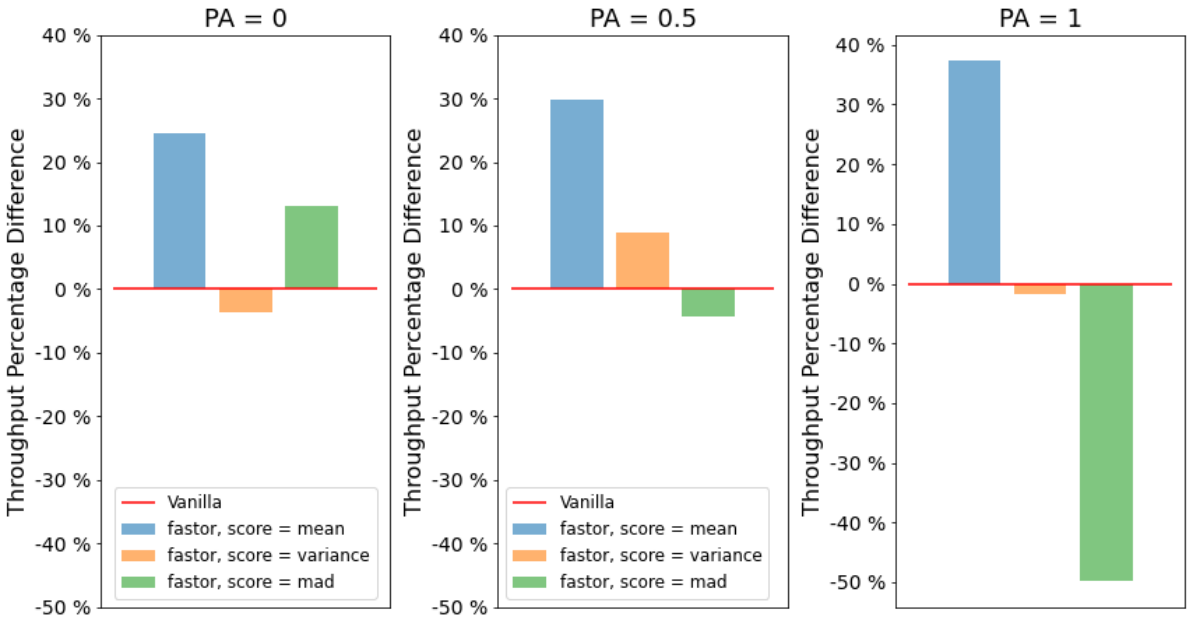

Figure 5.3: Mean Percentage Throughput Difference between Vanilla Tor and FasTor for Different Score Metrics and Performance Anonymity Settings, collected across the world's 500 most popular websites

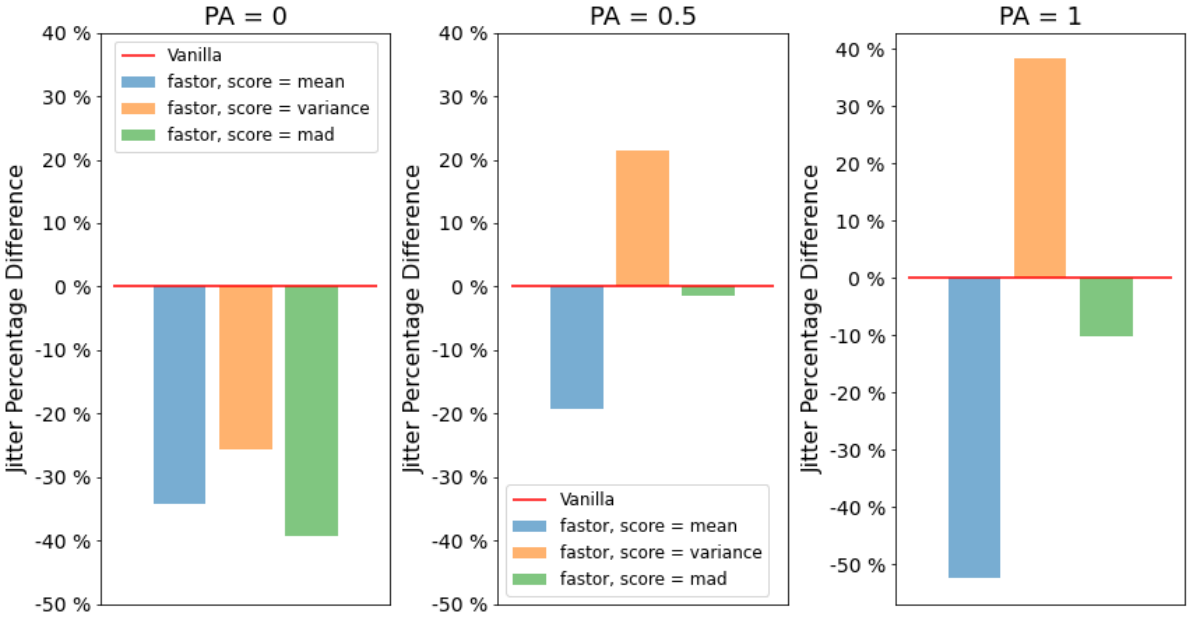

Figure 5.4: Mean Percentage Jitter Difference between Vanilla Tor and FasTor for Different Score Metrics and Performance Anonymity Settings, collected across the world's 500 most popular websites

but can only be replaced by higher performing ones, adding an extra edge over other settings. This setup achieves a maximum **performance gain** of almost **40% over vanilla Tor** with consistently less performance jitter (higher stability), which is a significant boost. With a PA parameter of 0, the highest anonymity setting supported by the proposed client, FasTor offers roughly **25% performance boost**, while for a medium PA setting of 0.5, this rises to around **30%**. All PA settings significantly increase performance stability over vanilla To, reaching a dramatic 50% improvement with PA=1.

- **Variance as Score:** Using variance as relay score for the FasTor scheme seems to provide very similar performance to vanilla Tor, irrespective of the PA parameter. Since the correlation of relay TTLB variance to publicly available bandwidths is very high, the information gain it provides during circuit-selection is minimal, hence not having a large effect on performance. However, the use of higher PA parameters hurts the performance stability achieved by this setup, meaning that short-term relay variance is not a good indication of its long-term stability characteristics.
- **MAD as Score:** The use of Median Absolute Deviation to rate relays produces very interesting performance results. In this setup, the PA parameter seems to have an inverse effect on performance, reaching a dramatic -50% drop in throughput compared to vanilla Tor, when $PA = 1$. This could be due to TTLB median deviation not being a good representation of relays' performance. In addition, higher PA settings nullify the stability gains MAD provided when PA was 0. Both of these facts produce evidence that short-term TTLB MAD is not a good relay performance, or long-term stability indicator.

5.3 Results Discussion

The performance results produced by FasTor when using TTLB mean as score metric are very promising. It can potentially provide individual users with a **25% throughput boost** and a **30% increase in performance stability** over vanilla Tor, without a significant drop to their anonymity. If users choose to put less emphasis on anonymity, setting the PA parameter to 1, they can potentially increase their throughput by **38% and stability to over 50%**. This is significantly higher than the 23% improvement achieved by PredicTor [17] or other previous clients [15] [5].

The performance results however, could have been skewed by a number of different factors. Some of them are recognized and discussed below:

- Since the tests for each scheme were performed over a duration of roughly 12 hours, and different schemes were tested in different time spans, the recorded performance could have been affected by any random changes in Tor's live traffic. These can never be predicted, and their effects would be mitigated by further (extensive) testing.
- The tests only took into account requests that succeeded, and not failed due to circuit incompatibility. Some Tor exit relays might not support certain target

destinations, and if in real-life a selected circuit is used to query such a server, the request will fail hence hurting user performance. However, the vanilla Tor client tests also dropped such query results, providing a fairer performance comparison.

- Client tests were performed by a client located in Edinburgh, Scotland. There is no guarantee that the use of client-side relay TTLB data would provide all users with similar performance gains, independent of client location.

5.4 Evaluation of Anonymity

It has been made clear that FasTor focuses on user performance through the Tor network. Its gains over vanilla Tor however, come hand-to-hand with a drop to its user anonymity assurance. While the default FasTor setup (PA=0) attempts to maintain as much anonymity as possible, maximum performance boosts are achieved by focusing purely on selecting the best performing Tor relays, and not making an effort to randomize the selection process. In this section, the focus is on evaluating the anonymity FasTor's standard setup provides.

Instead of choosing relays individually using their bandwidths as probability weights, like the standard Tor scheme currently does, FasTor uses the same technique to select a variable sized relay pool. It then goes on to select the best-performing relays in terms of their pre-recorded TTLB scores, further reducing the probability of clients picking slower relays. As a result, users' relay-choice entropy is lowered making it easier for malicious third parties to predict their future paths through Tor, hence increasing the risk of de-anonymization. Furthermore, this decrease in anonymity ranges according to the relay-pool's chosen size. Smaller pools may provide less performance gains, but will not decrease the user's choice entropy by a large factor, while larger pool achieve the opposite effects.

Another source of anonymity loss could arise due to throttling attacks, introduced by Geddes et. al. [16]. Malicious third-party Tor relays can induce these attacks to exploit performance enhancing mechanisms, such as FasTor's. If a node is able to recognise an incoming stream as a "performance-measurement", it could artificially throttle the circuit in order to affect the client's relay-selection process. As a result, users could be tricked into using these malicious relays.

5.5 Deployment Concerns

Even though FasTor can potentially dramatically increase client performance and stability, as seen by its comparison to Vanilla Tor, there is an issue that could nullify its benefits, when used by all Tor users around the world. The scheme's performance boost stems from its more up-to-date and individually-tailored user relay information. This is collected on the client-side, instead of centrally controlled such as bandwidth

measurements. As a result, every client would have to generate increased levels of traffic, both when collecting relay statistics, and when performing circuit selection.

If this scheme was to be widely-used by Tor users, it would significantly increase the base network traffic of Tor. All 2 million active customers would have to perform congestion measurements when selecting circuits, on top of measuring node-specific statistics for every one of the ≈ 7000 Tor relays every time their assigned guard is updated. This can potentially decrease average-case performance through Tor due to the higher congestion relays would face.

Chapter 6

Conclusion

6.1 Future Work

While FasTor's performance results look very promising, there are more steps required to validate and deploy such a scheme to users around the world. This section mentions some of these necessary future additions, along with introducing possible extensions to the protocol.

- In this paper, FasTor was evaluated using a single pool size of 5 due to time constraints. While it produces a good estimation of the potential gains the scheme can provide Tor users, the effects a varying size has on performance are not fully investigated. In theory, larger sizes should provide higher performance boosts and lower anonymity, whereas smaller ones should have the opposite effect. As a result, a future investigation which focuses on finding the optimum size for FasTor's performance-anonymity trade-off would be most useful for practical applications.
- As mentioned in earlier sections, this scheme could prove problematic if widely-used due to its increased base network traffic. Further investigation is required to assess whether this traffic increase would nullify the its performance benefits. Future work can involve FasTor's implementation inside Tor's codebase, which will allow large-scale Shadow simulation testing. These results can provide a useful insight on the scheme's performance gains/losses when deployed around the world.
- A future extension of this project could be the investigation of a relay-score metric which combines many relay characteristics. Having obtained performance results using TTLB mean, short-term variance and MAD, the mean statistic was found to be the most successful in rating relays both in terms of performance and their stability. However, a combination of these (or other) collected features could potentially provide a better rating, indicating relay-performance with higher accuracy.
- Another possible FasTor extension would be to individually tailor the relay-statistic collection process to each user. Different users have various web-browsing

habits. Some perform more data-downloading tasks while others prefer interactive web-pages. In this project, for evaluation purposes, a 1kb file was downloaded through all relays to determine their TTLB characteristics. While this file size is useful when largely focusing on path latency, a larger file would be required to better determine physical bandwidths. Consequentially, users could have the option of individually selecting a file size when determining relay characteristics.

6.2 Remarks

While Tor users enjoy a high level of anonymity, their performance is significantly lower compared to using other web browsers. Even though, Tor's priority is not user experience, its usability would greatly benefit if it provided a more stable and enjoyable web-browsing environment. At the end of the day, a larger user-base would inherently provide more anonymity, since it would make some Tor de-anonymization attacks (such as correlation attacks) harder to perform. FasTor introduces client-side relay-characteristic collection along with a form of congestion control, which aim to improve user perceived performance while minimizing the cost to anonymity. Results collected on the live Tor network from a single client seem very promising, providing a maximum of 38% throughput and 50% stability improvements over vanilla Tor, when querying the 500 most popular websites globally. While this provides compelling evidence to suggest FasTor can improve user-experience in the real-world, further analysis must take place before it can be stated for certain.

Bibliography

[1] D. S. Evans, "The online advertising industry: Economics, evolution, and privacy," *Journal of Economic Perspectives*, vol. 23, pp. 37–60, September 2009.

[2] B. Warf, "Geographies of global internet censorship," Nov 2010.

[3] T. T. Project, "Users," 2020.

[4] F. Lardinois, "Google says there are now 2 billion active chrome installs," Nov 2016.

[5] M. Akhoondi, C. Yu, and H. V. Madhyastha, "Lastor: A low-latency as-aware tor client," in *2012 IEEE Symposium on Security and Privacy*, pp. 476–490, 2012.

[6] T. . J. 22, "Tor's open research topics: 2018 edition," Jul 2018.

[7] "Tor review: Is it safe? can you be tracked? everything you need to know."

[8] "Stem documentation."

[9] J. K. Ousterhout, "Scripting: higher level programming for the 21st century," *Computer*, vol. 31, no. 3, pp. 23–30, 1998.

[10] "The top 500 sites on the web."

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

[12] T. T. Project, "Tor specifications," 2020.

[13] Whonix, "Tor entry guards," Mar 2021.

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

[15] T. Wang, "Congestion-aware path selection for tor."

[16] J. Geddes, R. Jansen, and N. Hopper, "How low can you go: Balancing performance with anonymity in tor," in *Privacy Enhancing Technologies* (E. De Cristofaro and M. Wright, eds.), (Berlin, Heidelberg), pp. 164–184, Springer Berlin Heidelberg, 2013.

[17] A. Barton, M. Wright, J. Ming, and M. Imani, "Towards predicting efficient and anonymous tor circuits," in *27th USENIX Security Symposium (USENIX Security 18)*, (Baltimore, MD), pp. 429–444, USENIX Association, Aug. 2018.

[18] “Shadow: Tor simulator.”

[19] “pycurl.”