Final Project Report - Methodically Modeling Tor

Amalee Wilson *
amalee@stanford.edu

Neil Perry
naperry@stanford.edu

ABSTRACT

This report documents the reproduction of Methodically Modeling the Tor Network [9] for CS 244. While the techniques for generating Tor network models have evolved since 2012, the model presented in the original paper and the reported performance results in Shadow were successfully replicated. We found that the model remains fairly accurate when generated using modern data, and once generated, the model can continue to accurately simulate Tor network performance for up to a year. After reproducing the initial results, we experiment with a much larger download size and find that larger downloads are less accurate than smaller downloads using their model. We then extend their work to account for the increased percentage of video traffic observed on the Internet. By replacing half of the default clients in the simulated network with video clients, we improve model accuracy for larger downloads with little to no adverse effect on smaller downloads.

1 INTRODUCTION

Tor seeks to enable anonymous communication by directing clients' traffic through an overlay network of volunteer relays. Tor clients encrypt messages multiple times before they are source-routed through a circuit of relays. At each relay, part of the client's message is decrypted before being forwarded to either the next relay or the destination. Tor preserves anonymity because no single node on this communication path is aware of both the sender and the receiver.

Due to its distributed architecture, emphasis on privacy, and the fact that Tor is actively used, experimenting on live Tor networks is challenging. At the time that the original paper was written, there were several testbeds, emulators, and simulators available to use for Tor experimentation. This prior work made assumptions about the Tor network in order to simplify the problem of modeling the network. However, Methodically Modeling the Tor Network [9] sought to create a model that was tractable, transparent, justified, and well-explained, which could ideally be used by different tools.

In the paper, Jansen et al present two algorithms which together allow them to generate a scaled-down simulated network representative of the real Tor network. They test this model with two tools: Shadow and ExperimenTor. The results of their experiments show that the model performed well in both simulations when compared to the real Tor network.

In this reproduction effort, we target Shadow because it can be easily run on a single, well-provisioned machine, while ExperimenTor requires more resources and does not appear to be actively maintained. For example, current links on the ExperimenTor website have Ubuntu 11.04 VMs. Shadow, on the other hand, is still being developed 8 years later with an active community. Additionally, Rob Jansen et al. improved on the modeling capabilities of Shadow and its family of related tools in 2018 [12] by incorporating data collected from live Tor into the default Markov clients created using TorNetGen. TorNetGen generates a network based on the algorithms presented in the 2012 paper [9].

The goals of this reproduction are to recreate the previous results in the 2012 paper with modern data using Shadow and compare these results to performance clients on the live Tor network. We also explore how the model performs across time and with much larger downloads. After reproducing the original results, we extend the model to explicitly account for video traffic on the network and compare these new results to live Tor performance clients.

Our findings show that the model presented in the original paper and the Shadow ecosystem of tools does, in fact, produce an accurate model of the performance and load characteristics of the real Tor network. Additionally, the models are robust in that they consistently give accurate results when compared to Torperf data from a different month. Thus researchers may create a model of the Tor network that will be good for several months instead of needing to remake the model each month. For larger downloads, the model does not perform as well. However, our modified model that explicitly accounts for video traffic results in significant gains to model accuracy for larger downloads without adversely affecting performance of smaller downloads. Our video client approximations are not perfect and there is room for improvement. We hope that this approach of explicitly modeling video traffic in the Tor network shows a promising direction for future work.

2 RELATED WORK

Drawing on the work of Rob Jansen et al. for the purpose of reproducing their work to show the validity of their findings, we look to their related works sections for background knowledge. To this end, this related works section uses the same citations and ideas as [9, 12]. Previous work has been

*Each author contributed equally to this project, and there is no significance to author ordering in this document.

done on measuring and modeling the Tor network in order to test applications and improvements without actual deployment; this includes straight forward approaches like logging and analyzing traffic on the actual Tor network (McCoy et al. [14] and Chaabane et al. [2]), modeling [1, 15], and simulation [10, 15]. The first method is not capable of testing new ideas at scale and has many potential privacy violations [13] which led to proposals of guidelines to follow to protect users on these networks [18]. This further led to systems like PrivCount [11] and later differential privacy versions [4] being developed to collect privacy preserving aggregated information from the relays. This however has not stopped some researchers from continuing to use direct methods, ignoring privacy concerns [17]. Many of these concerns can be avoided by modeling and the use of simulations such as through a tool like Shadow [8] which simulates other characteristics of networks like packet loss and jitter. Also, Shadow-Plugin-Tor allows Shadow to run any locally installed version of Tor on simulated clients [5]. The original data gathered by McCoy et al. [14] and Chaabane et al. [2] have also been used to inform the design of many models used in simulators [8, 9] and have been further enhanced by Wacek et al. [19]. Finally, VoIP performance on anonymous networks like Tor has been examined [16] but we are unaware of any work modeling streaming video over Tor.

3 METHODS

Due to resource constraints, we chose to use only Shadow as opposed to comparing Shadow to ExperimenTor with the presented model. ExperimenTor was difficult if not impossible to run on a single machine, whereas Shadow is capable of running on a well-provisioned PC or VM.

We chose to use an existing ecosystem of tools rather than re-implementing model generation from scratch. Several factors contributed to this decision. Foremost is the fact that all of these tools are open source. TorNetGen is a tool produced by the authors of the original paper that uses actual Tor data to generate configurations for a Tor network modeled after the real-world version but at a configurable scale [6]. TorNetGen uses the same (albeit slightly updated) algorithm presented in the original paper which we determined by inspecting the code. Additionally, this tool outputs the model in a format that is compatible with the specific (and somewhat cumbersome) format requirements of Shadow. However, in each case fresh data was used to generate new models: Tor data from metrics.torproject.org for January 2019 was used to generate the models presented in this report. Data from Tor Metrics will be referred to as Torperf data, and it is used both for model generation and for comparison. Also, TorNetGen uses some of the Tor traffic model measurements from a more recent 2018 paper [12] by Rob Jansen et al. to produce Markov models for non-performance-measuring hosts in the network.

For the simulation, we used Shadow and Shadow-Plugin-Tor (formerly named Scallion, as in the original paper) to run Tor on simulated hosts in Shadow. Additionally, we used TGen, a traffic generator [7], to generate traffic for both the modeled network and the actual network. Onionperf is a tool that can be used to measure (real) Tor performance, and it uses the same TGen style traffic graphs that Shadow uses [3]. This compatibility between Shadow and Onionperf allowed us to write TGen graphs and run them on both the simulated network and the actual Tor network, allowing for direct comparisons. For the case of Onionperf, however, we had to modify its model-generating script to produce the TGen graphs we wanted. Lastly, OnionTrace traces Tor's circuit building and stream assignment and is used by Shadow and TGen.

Most of these tools produce some form of useful logging and often come with tools to parse the log files. Many of these analyses and visualization tools were not quite suitable for our purposes, so we either modified them or wrote new scripts. We have reproduced this work using both a native Linux installation and a VM. The native Linux machine was used to run all experiments presented in this report because it has more memory than the machine with the VM, and Shadow is a memory-hungry application. It maintains the full state throughout the duration of the simulation, so even with 64 GB of memory, the "small" network was killed after about 40 simulated minutes (several wall clock hours).

For model generation, we downloaded Tor metrics data from January 2019, including relay descriptors, user statistics, and Torperf data. Torperf is a set of utilities that tracks Tor network performance, and Tor Metrics runs performance tracking clients on a near-constant basis. This data is important for both model generation and the evaluation of the performance of the model. We also downloaded Torperf data from March 2020 to compare with our January 2019 model's performance. To collect data for 300 MiB downloads on the real network, we used Onionperf to measure successive 300 MiB downloads through Tor. For 300 MiB downloads on simulated Tor, we wrote TGen models for modified performance clients.

Since much of the traffic on the non-Tor Internet traffic is video traffic, it is a reasonable assumption that this distribution of traffic holds true for the Tor network too. From this assumption we chose to replace 50% of the clients in the model with video clients. To approximate video client behavior, we used Wireshark and Python to capture and analyze traffic while streaming a video over the Tor browser. From this, we wrote a new client in the TGen format to send and receive data at approximately the rates as the observed video client. TGen uses a directed graph to generate traffic. For

the video client, we specify a root node that pauses for one microsecond, sends 66 bytes of data to the server, and then receives data with the following probabilities: 2962 bytes with probability $\frac{33}{86}$, 1514 bytes w.p. $\frac{21}{86}$, 5858 bytes w.p. $\frac{14}{86}$, 4410 bytes w.p. $\frac{13}{86}$, or 7306 bytes w.p. $\frac{5}{86}$. These numbers were chosen based on the observed real video client behavior over the Tor network: 87% of the traffic from client to server was 66 bytes, and 86% of the data from the server to the client was 2962, 1514, 5858, 4410, or 7306 bytes, distributed as in the probabilities that were just mentioned. This is a rough approximation of real video client behavior that could certainly be improved. We then replaced 50% of the standard Markov model clients in the small model network with our video clients and measured the performance of 320 Kib, 5 MiB, and 300 MiB downloads on performance clients in Shadow.

4 EXPERIMENTAL SETUP

All experiments reported in this document were run on a desktop computer with an AMD Ryzen 9 3950X 3.5 GHz 16-Core Processor and 64 GB of DDR4-3600 memory.

The first experiment was to try to reproduce the work in the original paper. The goal of this experiment is to confirm the claims made in the paper. To achieve this, we took the following steps: produce a model with one month of Tor Metrics Torperf data, run a simulation in Shadow using this model, and compare the data from the simulated performance clients (dedicated clients that run to collect Tor metrics) with the observed data of real performance clients from that month. The January 2019 Tor Metrics data was downloaded and then we used TorNetGen to generate the network model described in the original paper at 2% of the scale of the real Tor network (same fraction as the "small" network in the paper). We then modified some of the performance clients to perform 320 KiB downloads and ran the simulation. Our custom data processing script was used to parse and plot this data against the observed data. The plots are shown in Figure 2.

The second experiment compares the results of the previously generated model from the first experiment to the Tor Metrics data from March 2020. The goal of this experiment is to understand how well the model simulates the real Tor network as it changes with time. To test the temporal robustness of the model, we simply downloaded the March 2020 data from Tor Metrics and re-ran the analysis script from the first experiment using the March data instead of the January data. These results are shown in Figure 3.

The third experiment was designed to test how the model from the original paper could be improved. The model made from TorNetGen contains performance clients, relays, servers, and Markov clients. The Markov clients send and receive packets to and from the servers based on a later paper's dynamically learned Markov models [12] of flow, stream, and packet generation. We replaced half of these models with our custom, approximate video clients to see what effect this change had on performance for performance clients with 320 KiB downloads and 5 MiB downloads. Results of this experiment are shown in Figure 4.

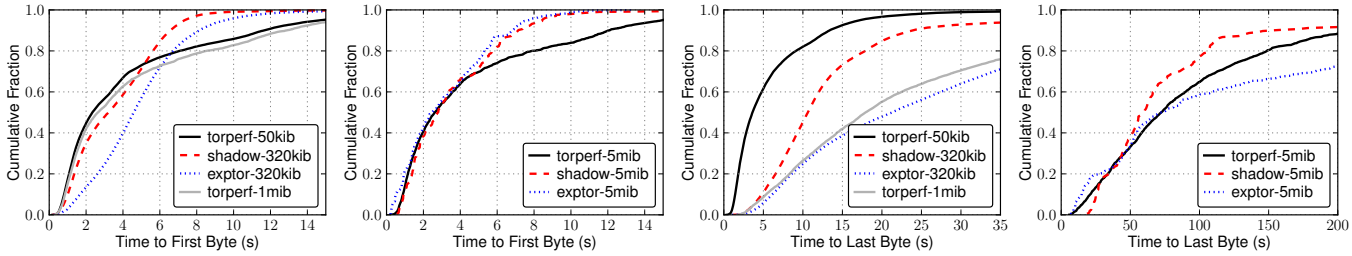
The last experiment shows how each version of the model – the version from TorNetGen with January 2019 data and the version with 50% video clients – performs for larger downloads. In the original paper, the largest file size that is downloaded by performance clients is 5 MiB. Furthermore, they comment in the original paper that "bulk downloads [of 5 MiB] complete slightly faster in Shadow than in Tor."[9] This experiment was designed to determine whether this known issue was exacerbated by a much larger file download and whether the addition of video clients reduced the difference between Shadow's performance with the original model and the observed download times for 300 MiB. Results for this experiment are shown in Figure 5

5 RESULTS

Each graph shows the cumulative fraction (y axis) of clients with a download time of the x coordinate. Shadow results are always displayed in blue, and graphs are labeled with which model Shadow uses as well as which Torperf or Onionperf data Shadow results are compared to.

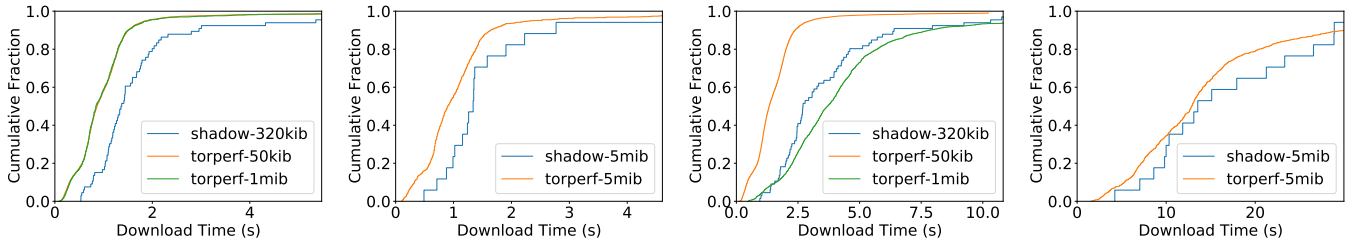
Figure 2 shows the reproduction result for Figure 3 in the original paper, which is shown in Figure 1. Our results are similar: the Shadow results follow the measured data's curve closely in these graphs. The times to first byte and last for both 5 MiB clients are very close to the observed data from Torperf. The time to last byte for the 320 KiB clients runs between the observed 50 KiB and 1 MiB observed data, which is expected and similar to the original graphs. There are some differences in the x-axes and shapes of the curves, likely due to performance improvements in the actual network over the past 8 years as well as slight modification of the model, such as including Markov clients. Otherwise, though, the main figure of the original paper has been reproduced. Overall the model presented in the paper produces fairly accurate results when run in Shadow.

Figure 3 shows a result similar to Figures 2 and 1, but the model results are compared to observed data from March 2020 instead of January 2019. Note that despite 14 months between the collected data use to build the model and the real data used for comparison, the model is still performing well. This result suggests that the Tor network may have been relatively static, except in terms of performance improvements.



(a) 320 KiB clients (b) 5 MiB clients (c) 320 KiB clients (d) 5 MiB clients

Figure 1: Jansen et al Figure 3 for their small network.



(a) First byte, 320 KiB clients. **(b) First byte, 5 MiB clients.** **(c) Last byte, 320 KiB clients.** **(d) Last byte, 5MiB clients.**

Figure 2: Time to first and last byte for 320 KiB and 5 MiB clients in Shadow using the recreated small network, made from January 2019 data, vs Torperf data for the same month.

Figure 4 shows the performance of 320 KiB and 5 MiB downloads in terms of the time to last byte for the associated performance clients in the network model that is 50% video clients. Like the other experiments, time to first byte graphs were also created, but due to space, we are omitting them. Time to first byte performance is not significantly different from previous results. Additionally, the legend in Figure 4a had to be abbreviated due to space constraints. Note that the 320 KiB download tends more toward the middle of the area between the 50 KiB and 1 MiB observed data, instead of hugging the 1 MiB Torperf results as in Figures 2c and 3c. Also note that in Figure 4b, the simulated 5 MiB performance is intersecting with the Torperf data much more often than in Figures 2d and 3d.

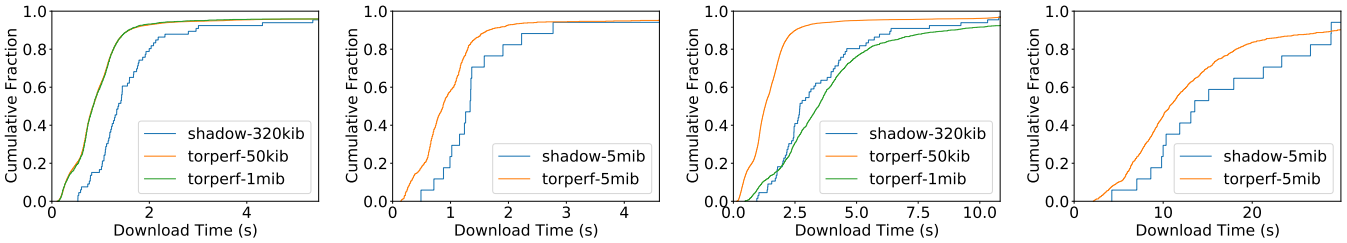
Figure 5 shows graphs of the time to last byte for 300 MiB clients in Shadow using both the standard January 2019 model and the model with 50% video clients. In Figure 5a the Shadow model performs pretty well, but tends to complete much faster than the Onionperf data measured from the real Tor network. When 50% of the Markov clients in the original model are replaced with video clients, however, Shadow's performance is much closer to the measured performance, as shown in Figure 5b.

6 DISCUSSION

The results, especially figure 2 in comparison to Figure 1, show that the model used in the original paper and the performance results that they presented for Shadow using this model are reproducible. Figure 3 shows that this model is relatively robust across time. Even though the model was built with data from January 2019, it was used to successfully simulate the behavior of the real Tor network in March 2020. However, this observation that the model performs as well on "future" networks may not, in fact, be entirely due to the model's robustness. It's likely that the Tor network is relatively slow-changing, which would partially explain the relatively accurate results across time.

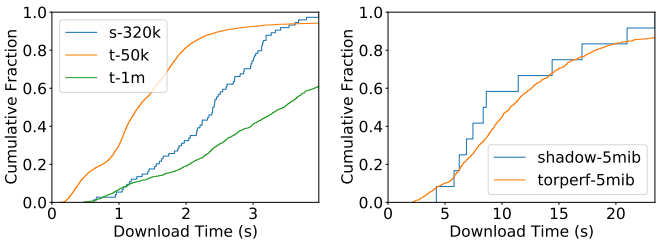
Figure 4 shows that the addition of explicit video traffic clients may improve the model introduced in the paper, and it importantly shows that performance for small downloads are not adversely affected with addition of video clients. In fact, the results in Figures 4a and 4b look a little better than the results in Figures 3c and 3d. Thus, the addition of even crudely-approximated video clients seems to improve the model.

With Figure 5a, we show that the issue mentioned in the original paper where large downloads complete faster in Shadow than on Tor is exacerbated by larger download size. The original paper tested up to only 5 MiB downloads, and Figure 5a shows the results for a 300 MiB download. However,



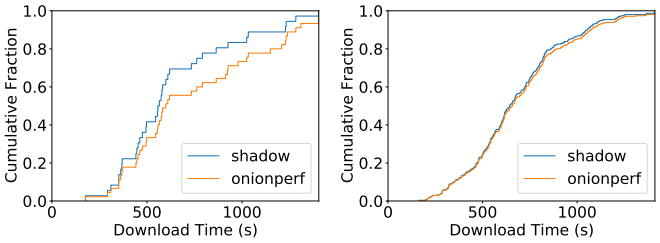
(a) First byte, 320 KiB clients. **(b) First byte, 5 MiB clients.** **(c) Last byte, 320 KiB clients.** **(d) Last byte, 5 MiB clients.**

Figure 3: Time to first and last byte for 320 KiB and 5 MiB clients in Shadow using the recreated small network, made from January 2019 data, vs Torperf data for March 2020.



(a) 320 KiB download time. **(b) 5 MiB download time.**

Figure 4: Time to last byte on the modeled network with 50% video clients vs March 2020 Torperf data, where s- is used for shadow and t- is used for torperf.



(a) 300 MiB download time with standard model. **(b) 300 MiB download time with 50% video clients.**

Figure 5: Time to last byte on the modeled network with 50% video clients vs Onionperf data collected in June 2020.

this large download performance is greatly improved by the addition of video clients, as shown in Figure 5b.

Collectively these results show that the original paper is reproducible. Furthermore, the results from the video client model show that explicitly modeling video traffic in these models is a promising future research direction for modeling Tor network behavior.

7 LIMITATIONS

The clearest limitation in our reproduction work was the lack of ExperimenTor results. The original paper used another tool besides Shadow which we decided not to use due to it not being maintained and the fact that it needed more resources than those which were available to us. As for our extension efforts, our model of video traffic is not perfectly accurate. We tried several iterations of different models, but none of the tools that we were working with allowed for complex state to be recorded or measured. In other words, it is impossible to keep track of a video buffer or to estimate network capacity from within hosts in the Shadow simulation with traffic generated by Tgen. Therefore, it is impossible to fully implement existing implementations of ABR or BBR in this context. Since these limited tools are used to model video traffic in the simulator, their lack of support for buffer state and network capacity estimation reduce the accuracy of the video client models. We carefully tried to tune video clients to make their behavior as realistic as possible by carefully calibrating them based on real video streaming measurements, but they are not perfect models of video streaming behavior.

8 CONCLUSION AND FUTURE WORK

Measurement and modeling work has continued since Methodically Modeling the Tor Network [9] was published in 2012. Internet usage has also changed over time, with more traffic being video content. For example, Netflix makes up over 50% of US internet traffic during peak hours [20]. It seems reasonable that Tor users would also be more likely to want to use the service for video as well. More work needs to be done generating models that are realistic for this type of traffic. The video models presented in our extensions are relatively simple and don't capture key features like buffer state that are needed to implement BBR and ABR algorithms. Shadow and Tgen could be extended to allow for more complicated traffic patterns using this state. This would allow for more realistic direct comparisons between models being run in Shadow and the actual Tor network's behavior for video

performance. On the measurement side, more work can be done logging the endpoints performance while streaming video both over Tor and the standard Internet in order to understand how much of a performance hit streaming takes over Tor. It would also be useful to do measurement work in understanding how much traffic on Tor consists of video. This information would be valuable in understanding what improvements need to be made to Tor to allow for a better experience for users or even new video streaming techniques that could be developed specifically for video over Tor. More accurate ratios of video "background noise" could be used in simulations to improve accuracy as well.

REFERENCES

[1] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2020. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. (06 2020).

[2] A. Chaabane, P. Manils, and M. A. Kaafar. 2010. Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *2010 Fourth International Conference on Network and System Security*. 167–174.

[3] Tor contributors. 2020. Onionperf. https://github.com/torproject/onionperf. (2020).

[4] Tariq Elahi, George Danezis, and Ian Goldberg. 2014. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. *Proceedings of the ACM Conference on Computer and Communications Security* (11 2014), 1068–1079. https://doi.org/10.1145/2660267.2660280

[5] Rob Jansen et al. 2020. Shadow-Plugin-Tor. https://github.com/shadow/shadow-plugin-tor. (2020).

[6] Rob Jansen et al. 2020. TorNetGen. https://github.com/shadow/tornetgen. (2020).

[7] Rob Jansen et al. 2020. TorNetGen. https://github.com/shadow/tgen. (2020).

[8] Rob Jansen. [n. d.]. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. ([n. d.]).

[9] Rob Jansen, Kevin S. Bauer, Nicholas Hopper, and Roger Dingledine. 2012. Methodically Modeling the Tor Network. In *CSET*.

[10] Rob Jansen, Nicholas Hopper, and Yongdae Kim. 2010. Recruiting new tor relays with BRAIDS. *Proceedings of the ACM Conference on Computer and Communications Security*, 319–328. https://doi.org/10.1145/1866307.1866344

[11] Rob Jansen and Aaron Johnson. 2016. Safely Measuring Tor. 1553–1567. https://doi.org/10.1145/2976749.2978310

[12] Rob Jansen, Matthew Traudt, and Nicholas Hopper. 2018. Privacy-Preserving Dynamic Learning of Tor Network Traffic. In *25th ACM Conference on Computer and Communications Security (CCS)*. See also https://tmodel-ccs2018.github.io.

[13] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. [n. d.]. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. ([n. d.]).

[14] Damon Mccoy, Tadayoshi Kohno, and Douglas Sicker. 2008. Shining light in dark places: Understanding the Tor network. In *In Proceedings of the 8th Privacy Enhancing Technologies Symposium*.

[15] Steven Murdoch and Robert Watson. 2008. Metrics for Security and Performance in Low-Latency Anonymity Systems, Vol. 5134. 115–132. https://doi.org/10.1007/978-3-540-70630-4_8

[16] Maimun Rizal. 2014. A Study of VoIP Performance in Anonymous Network - The Onion Routing (TOR).

[17] Nick Savage and Gareth Owen. 2015. Empirical analysis of Tor Hidden Services. *IET Information Security* 10 (10 2015). https://doi.org/10.1049/iet-ifs.2015.0121

[18] Christopher Soghoian. 2012. Enforced Community Standards for Research on Users of the Tor Anonymity Network. In *Financial Cryptography and Data Security*, George Danezis, Sven Dietrich, and Kazue Sako (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 146–153.

[19] Chris Wacek, Henry Tan, Micah Sherr, and Kevin Bauer. [n. d.]. An Empirical Evaluation of Relay Selection in Tor. ([n. d.]).

[20] Te yuan Huang, Ramesh Johari, Nick Mckeown, Matthew Trunnell, and Mark Watson. [n. d.]. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. ([n. d.]).