NavigaTor: Finding Faster Paths to Anonymity

Robert Annessi
Institute of Telecommunications, TU Wien
Gusshausstr. 25/E389, 1040 Vienna, Austria
robert.annessi@nt.tuwien.ac.at

Martin Schmiedecker
SBA Research
Favoritenstraße 16, 1040 Vienna, Austria
mschmiedecker@sba-research.org

***Abstract*—The Tor network is currently by far the most popular system for providing anonymity on the Internet. Even though both latency and throughput have been significantly improved in recent years, Tor users still experience variable delays on connecting to servers. Such delays have been shown to be especially harmful for browsing the web and prevent altogether the use of protocols where a certain quality of service is indispensable.**

In this paper[1] we propose and evaluate methods to measure and improve performance in the Tor network[2]. To estimate the quality of circuits for future traffic, we use active Round-Trip-Time (RTT) measurements and a-priori information of the distribution of RTT values. In this way, slow circuits can be discarded before having negative impact on user experience. Using NavigaTor, our high performance measurement software which includes a custom Tor path generator, we are the first to conduct large-scale performance measurements on the live Tor network, building millions of circuits within days, without stressing the anonymity network. As part of our study, we conduct several experiments from PlanetLab on the live Tor network to analyze the trade-off between the quality of protection and the quality of service. We compare our Circuit-RTT method to the current state-of-the-art method Circuit Build Time (CBT) and the more recently proposed congestion-aware scheme, finding that the congestion-aware scheme in its original design achieves only minor improvements on the current Tor network and that Circuit-RTT improves latency and throughput more effectively than CBT.

***Index Terms*—Tor, Quality of Service, RTT, Anonymity**

1. Introduction

With the tremendous increases in communication over the Internet, security and privacy issues have become more and more important. Much research has been done to develop anonymous communication protocols that would allow people to communicate without revealing potentially identifying information, such as their computers' IP addresses. These protocols became the technical basis for promoting freedom of speech, achieving privacy, and overcoming censorship on the Internet.

Presently, only very few systems exist that are of practical relevance for providing anonymity on the Internet. The most widespread and well researched is Tor [1], a low-latency anonymity system that is loosely based on the onion routing [2] principle. Since it gained public notice in 2004, it has become the most popular low-latency anonymity network, with over two million daily users [3] from over 120 different countries [4], and the platform for research in anonymous communications. Like other anonymity designs, Tor seeks to hide the relationships between communicating parties both from network observers and from the anonymization infrastructure itself. At the same time, Tor developers strive to achieve a reasonable balance between the conflicting demands of performance and anonymity, to encourage the use of the network [5].

Since Tor intentionally bounces traffic around the world several times, end-to-end latency is by design higher than on regular Internet connections. Even though both latency and throughput have been significantly improved [4] in recent years, Tor users still experience variable delays [6, 7, 8] on connecting to servers. Such delays have been shown to be especially harmful for browsing the web, which makes up the vast majority of connections [9, 10] through the Tor network, and prevent altogether the use of protocols where a certain quality of service is indispensable. Because anonymity systems hide users among users, keeping the performance overhead associated with the anonymity system as low as possible, enlarges the user base and thus ultimately enhances the anonymity the system offers [5].

2. Background

The Tor network consists of volunteer-operated nodes that forward traffic on behalf of users running Tor clients. To establish an anonymous communication channel to the user's intended destination, a client first selects a path through the Tor network by choosing suitable nodes among all active ones. A path typically consists of three nodes, called, respectively, the entry, middle, and exit node. A key aspect of achieving anonymity is that each node in a

1. This is a preprint of the paper to appear at IEEE European Symposium on Security and Privacy (Euro S&P) 2016.

2. Our source code, obtained measurement data and our Tor path generator are available online under https://naviga-tor.github.io

path knows only its predecessor and successor, but not the identities of both communicating parties.

Path Selection. Tor's path selection algorithm selects nodes according to certain constraints. For example, two nodes whose IP addresses are in the same /16 network must not be used in the same path. To increase performance while still providing a reasonable level of anonymity, nodes are additionally weighed by their bandwidth, giving a higher selection probability to nodes with more bandwidth [11, 12].

Circuit Building. As soon as a client has selected a path, it starts negotiating session keys with all involved nodes. The resulting encrypted connection through the Tor network is called circuit. Circuits are created one hop at a time; clients negotiate symmetric keys using Diffie-Hellman key exchange, where each partially created circuit is used to communicate with the next node. Constructing a circuit involves computationally expensive public-key cryptography and multiple packet round trip times and thus can take several seconds. To avoid burdening users with such high delays, clients maintain preemptively built circuits. The time required to establish a circuit does not, therefore, harm user experience, as long as a suitable circuit is available on request.

Cells. All packets are transmitted in fixed-size 512-byte cells, to mitigate both website fingerprinting and traffic analysis attacks that try to correlate packets entering and leaving the Tor network based on the packets' sizes. Building circuits is clearly computationally more expensive than relaying cells, since the former involves public-key cryptography whereas the latter uses symmetric key encryption.

Guard Nodes. Instead of selecting a new entry node for each path, every client restricts its choice of potential entry nodes to a small, semi-persistent set of Guard nodes. The concept of Guard nodes was introduced to mitigate both the efficiency of the predecessor attack [13] and the threat of end-to-end timing correlation attacks, in which an adversary observes users' traffic entering and exiting the anonymity network. The duration for which a client keeps its set of Guard Nodes was increased from 30 to 60 days in Tor version 0.2.4.12-alpha, and might soon change to one Guard per client for up to 9 months [14]. The specifics, however, have not yet been fully implemented at the time of writing [15].

Token Bucket Rate Limiting. In order to allow node operators to specify the bandwidth they are willing to provide, nodes use the token bucket algorithm for controlling the rate of traffic. Briefly, each node starts with a fixed number of tokens and decrements its token count as data are sent or received. When a node's token count reaches zero, it must wait until its tokens are refilled. The refill interval was reduced from 1000 to 100 ms in Tor version 0.2.3.5-alpha, in order to improve performance.

3. Related Work

Various methods have been proposed to measure time and improve latency and throughput in the Tor network; this is still an active field of research [16, 17]. In this section, we present the work related to this paper and compare them to our proposed Circuit-RTT method.

3.1. Circuit Build-Time

Circuit Build Time (CBT) is a client-side method that aims to avoid congested nodes and, thus, slow circuits. The basic idea is that the time it takes to establish a circuit gives a hint as to how well that circuit will perform for future traffic, given that some circuits are established within a fraction of a second, whereas other circuits take over one minute to build [6].

By comparing a circuit's build-time to a timeout value, clients discard circuits that take too long to build. Clients calculate and continuously adapt this timeout value by tracking the circuits' build-times and using a priori information about their statistical distribution. Empirically, it was found that build-times fit well to a Fréchet distribution. However, estimators for this distribution converge slowly and are difficult to calculate. For this reason, the tail of the Fréchet distribution is approximated with a Pareto distribution, for which estimators can be calculated quickly. The particular timeout value is calculated using the percentile function such that 80% of the mass of the distribution is below the timeout value. According to this method, it is expected that clients will accept the fastest 80% of all circuits on the network [12].

3.2. Link-RTT

Panchenko and Renner [18] proposed a method to obtain the Round-Trip-Time (RTT) between any two nodes. Their measurement method intentionally violates the exit node's exit policy, using localhost as a dummy-destination. For security reasons, every node refuses such connections and returns an error message to the client without contacting any further host. This ensures that the measured time is very close to the RTT of the circuit, including hardly any additional delay.

In their analysis, the authors measured latency between every possible pair of nodes in the network. They changed Tor's path selection so that the probability of a path to be selected is increased as the sum of the path's link-RTTs decreases and showed that latency and throughput is improved using this scheme, although anonymity is much lower compared to Tor's default path selection algorithm. The scalability of this approach is fairly limited, however, since in a network of n hosts every client has to make n^2 measurements.

We will build upon this basic method of measuring RTT in the Tor network, but instead of inferring the link-wise RTT between nodes, we measure the RTT of circuits. Furthermore, in contrast to Link-RTT, Circuit-RTT scales very well, since the number of measurements grows only linearly with the number of circuits a client uses and is independent of the number of nodes in the network.

3.3. Congestion-Awareness

To reduce congestion and improve load balancing in the Tor network, Wang et al. [8, 19] proposed a congestion-aware scheme by which clients can isolate nodes' congestion delays from other delays, such as that from propagation. To estimate the congestion of involved nodes, clients measure the RTT of a circuit five times immediately after circuit construction. For these active RTT measurements, a technique was used which is conceptually very similar to that previously introduced by Panchenko and Renner. The basic assumption in the paper is that congestion is a property of a node that can be measured in seconds.

Their scheme consists of three different techniques:

1) Clients select the circuit with the lowest congestion delay from three preemptively built circuits.
2) While using a circuit, clients opportunistically measure its congestion and switch to another circuit, if the current one becomes too congested.
3) Clients use a modified path selection algorithm which considers the nodes' congestion delays.

Their experiments on the live Tor network showed that their scheme improves performance; this was also noted by Wacek et al. [20], who simulated several schemes and concluded that the congestion-aware scheme[3] is the most effective in terms of latency, throughput, and anonymity. Thus, we will compare our Circuit-RTT approach not only to CBT but also to the congestion-aware scheme. For evaluating the congestion-aware scheme, we will stick to selecting circuits according to congestion delays and omit both the switching of circuits and the modified path selection algorithm, since the paper's authors themselves found their path selection algorithm to have less impact than the other techniques.

Like the congestion-aware scheme our Circuit-RTT approach uses active RTT measurements; it does not, however, aim to deduce nodes' congestion. Our approach requires measuring the RTT only once on each circuit, rather than five times, and requires constant memory, instead of having memory requirements that grow linearly with the number of nodes. Furthermore, it is difficult to analyze what exact percentage of the network is excluded by the congestion-aware scheme, since it dynamically calculates congestion delay for nodes and disadvantages an unspecified number of highly congested nodes.

3.4. Node-RTT

Panchenko, Lanze, and Engel [21] proposed a new metric for path selection, aiming to enhance performance and anonymity. According to their method, a client measures the RTT of every node through a one-hop circuit. This technique of RTT measurement is again based on the concept previously introduced by Panchenko and Renner [18]. In terms of latency, the proposed method not only improved performance, but also increased anonymity compared to Tor's bandwidth description method. However, the practical applicability of this approach is comprised by the fact that each client needs to perform RTT measurements for every node.

3. For the congestion-aware scheme, Wacek et al. implemented the first and second technique, but not the modified path selection algorithm.

3.5. Delays

Dhungel et al. [7] performed a detailed study of delays in the Tor network. According to their study, the delay of a circuit can be decomposed into link latencies between nodes, and queuing and processing delays within nodes. They observed that delays of circuits are most often the result of queuing and processing delays within nodes. Furthermore, their measurements revealed huge differences in delays introduced by particular nodes, finding that a large fraction of nodes have delays that dramatically fluctuate over time. Therefore, simply measuring all-pairs RTTs centrally and sending that information to clients would not be particular beneficial. With this in mind, we will focus on a method that allows clients to avoid nodes, which are slow at a particular point in time.

4. Contributions

Specifically, we make the following contributions:

- We build a path generator that is identical to Tor's default path selection algorithm, including bandwidth weighting and specific constraints on the selection of paths. Applying only small, non-intrusive changes to Tor's source code ensures high confidence in the logical equivalence of our path generator to Tor's path selection algorithm. In the future, this path generator could also be used to validate whether other path simulators are selecting paths as Tor would.
- Since conducting large-scale performance measurements on the live Tor network is a non-trivial task, previous research examined anywhere from a few hundred to fifteen thousand circuits. We were the first to conduct large-scale measurements on the live Tor network, using NavigaTor [22], our high performance measurement software released under a free license; one of NavigaTor's core features is that it runs (nearly) all code concurrently that involves network I/O. In this way, it became feasible to build and measure millions of circuits within days, without stressing the Tor network.
- We used the NavigaTor software to validate our Circuit-RTT scheme, in which the RTT of a circuit and a priori information of the distribution of RTTs is used to identify and discard slow circuits. We find Circuit-RTT to serve as a better estimator for the quality of performance that circuits will provide for future traffic. Clearly, we observe a significantly stronger correlation with end-to-end network latency using RTTs than using build-times or using congestion delays. We find that

Circuit-RTT improves end-to-end network latency and throughput, compared to the current state-of-the-art method CBT. Moreover, average latency over a circuit's lifetime of 10 minutes (as used by the Tor software) as well as its standard deviation are improved compared to CBT. We deployed NavigaTor on several hosts from PlanetLab [23], allowing us to infer that improvements to the path selection algorithm hold independently of the client's location. Gathered measurement data [22] are provided under a free license.

- We explore whether a combination of the CBT and the Circuit-RTT method could further improve latency, finding improvements of Circuit-RTT alone to be greater than that of a combination of both methods.
- We discover that the congestion-aware scheme in its original design achieves only minor performance improvements in the current Tor network; especially considering that it drastically reduces entropy. Modifying the congestion-aware scheme in such a way that the same statistical approximation is used as in CBT and Circuit-RTT, there is a larger performance gain whereas entropy is less reduced. Nevertheless, the modified congestion-aware scheme is still clearly outperformed both by CBT and by Circuit-RTT.

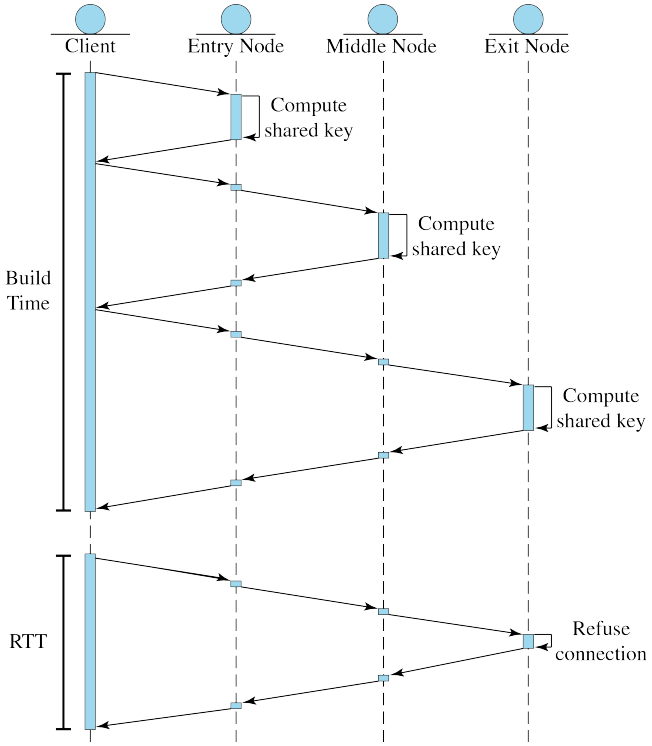

Figure 1. A circuit's build-time and its RTT.

5. RTT Measurements

In order to improve latency, the most important metric from the users' perspective [24], throughput, and thus the anonymity set [25] and overall anonymity in the Tor network, we implement active RTT measurements of circuits (i.e., the time it takes for cells to go from the client through a circuit and back). This active measurement approach intentionally violates the exit node's exit policy; in this way, it is ensured that the RTT is indeed very close to that of the circuit, including hardly any additional delay.

Contrary to previous approaches that aimed to use RTT measurements to improve performance, we infer neither congestion delays of nodes nor the RTT between a client and every single node nor between every possible pair of nodes in the network. Instead, we measure the RTT of a circuit and use a priori information of the distribution of RTTs, using a circuit's RTT as estimator for the quality that circuit will provide for future traffic. In this way, slow circuits can be discarded before having a negative impact on user experience.

Presently, clients use the CBT method to decide whether a circuit will be used to transport traffic after it has been established. Fig. 1 shows the timing associated with building a circuit and measuring its RTT. While a circuit is being established, data pass involved nodes several times. Thus, a circuit's build-time consists of multiple times the latencies between the involved nodes and multiple times the queuing delays within nodes. It additionally includes processing delays, the time spent by nodes both on forwarding packets and on computing encryption keys; these computations can be on the order of seconds if a node is computationally overloaded.

After a circuit has been successfully built, a client using the Circuit-RTT method measures the RTT of that circuit by asking the exit node to open a TCP connection to an IP address in the $127.0.0.0/8$ network. Previous approaches used a static address from that network; using dynamic addresses allows NavigaTor to evaluate multiple circuits at the same time. Since all nodes refuse connections to addresses within the $127.0.0.0/8$ network for security reasons, the exit node sends an error message back through the circuit without contacting any further host. We define the RTT of a circuit as the measured time interval between sending such request and receiving the corresponding reply. The RTT includes the latencies between the involved nodes plus the queuing and processing delays within nodes. In contrast to CBT, the processing delay is negligibly small, since forwarding cells requires only symmetric-key cryptography which is computationally inexpensive compared to building circuits, which involves public-key cryptography.

A potential disadvantage of this approach is that a malicious exit node could identify the measurement and use this knowledge to influence the result. The CBT approach, however, already allows every node in a circuit to easily identify and artificially alter circuit creation, effectively biasing the clients circuit selection. Therefore, we argue that no additional attack vector is added. Furthermore, it should be noted that both approaches do trade security for

performance.

6. Evaluation

Studying the trade-off between the quality of protection and the quality of service, we validate our assumption that a circuit's RTT is a better estimator for the quality that circuits provide for future traffic.
To this end, we examine:

- how RTTs of individual circuits change over time.
- the correlation between
  - a circuit's build-time and the time until the first byte is received from a webserver (Time-To-First-Byte (TTFB)),
  - a circuit's RTT and the TTFB,
  - a circuit's congestion delay and the TTFB.
- whether RTTs can be approximated by a Fréchet (or any other) distribution well.
- the impact of the CBT, Circuit-RTT, and congestion-aware methods on the TTFB, throughput, anonymity, and user-experienced latency changes over a duration of 10 minutes.
- how changes to the rate of accepted circuits influence TTFB, throughput, anonymity, and user-experienced latency over time.

6.1. Path Generator

Since building a circuit takes anywhere from tenths of a second to two minutes, solely establishing millions of circuits one after the other would take several weeks – even without making any measurement. When circuits are built and probed, most of the time is spent waiting for network I/O. To reduce the time required for running experiments, NavigaTor first queries the path generator and then builds and measures circuits concurrently instead of sequentially. In this way, the time required for running experiments is reduced tremendously.

First of all, a path generator identical to Tor's default path selection is required; i.e., given the consensus data, the generator should produce the same paths as Tor would. Existing path generation tools, such as TorCtl [26], COGS [27], or TorPS [28], are unsuitable in this regard for individual reasons. For example, since the authors' primary goal was to simplify the modification of path selection, Tor's path selection algorithm is entirely reimplemented in TorPS; such reimplementation leaves some uncertainty whether it generates paths the same way as Tor would. Instead of reimplementing Tor's path selection algorithm, we make small, non-intrusive changes to Tor's source code so that generating paths without actually building circuits is made possible. This ensures high confidence in the equivalence to Tor's default path selection algorithm, while also keeping our changes portable to other versions of Tor. In the future, our path generator could also be used to validate whether other path simulators are selecting paths as Tor would.

To facilitate the separation of path selection from circuit building, we extend the Tor control protocol, a text-based protocol, which allows programs to communicate with a Tor process, with two new commands: DUMPGUARDS and FINDPATH.

DUMPGUARDS. Since we intend to probe potentially all nodes, different entry nodes for each path are required. However, the probability for middle and exit nodes to be selected depends on whether Guard nodes are enabled. Simply disabling the use of Guard nodes is not an option, because we also require that the selection be identical to Tor's default path selection algorithm, which involves Guard nodes. To explicitly alter a Tor client's selection of Guard nodes during runtime, we implement a new controller command DUMPGUARDS, which makes the client clear its internal list of selected Guard nodes and choose new ones.[4]

FINDPATH. We implement another control command, FINDPATH, that makes the client choose a path and return the particular nodes involved. Before that, there was no way to make a client choose a path without actually building a circuit.

6.2. Measurement Procedure

Algorithm 1 on the next page sketches the process how NavigaTor gathers data. The main thread handles the selection of paths and coordinates all worker threads. It ensures that the client's list of Guard nodes is empty by executing the DUMPGUARDS control command. Subsequently, using the FINDPATH command, the associated Tor client selects a path and returns it. In this way, the path selection algorithm runs entirely in the Tor client process. The main thread can request another path from the Tor client immediately after having spawned a worker thread, because it runs asynchronously with the worker threads. Each of the worker threads receives only a single path from which it builds a circuit and then measures the build-time, RTT, TTFB, and throughput of that circuit.

We attempt to avoid interfering with the live Tor network functionality, since the Tor network can be extremely fragile when stressed with too many circuit creation attempts at once. In order to minimize the effect on the Tor network, NavigaTor keeps track of all nodes that are being probed at any given point in time, and queues new paths if they involve any of those nodes. In this way, a particular node is not probed more than once at a time, which is important considering that our approach would scale up to 1000 concurrent worker threads without any additional changes to Tor's source code.

6.3. Measurement Methods

TTFB Measurement. End-to-end network latency as experienced by users is approximated with the time elapsed

4. This patch was merged into Tor version 0.2.5.1-alpha, but the name of the command changed to DROPGUARDS.

Algorithm 1 NavigaTor's Measurement Procedure

```
1: procedure WORKER(path)
2:     circuit ← BUILD CIRCUIT(path)
3:     CBT ← MEASURE BUILD-TIME(circuit)
4:     RTT ← MEASURE RTT(circuit)
5:     TTFB ← MEASURE TTFB(circuit)
6:     BW ← MEASURE THROUGHPUT(circuit)
7:     return (CBT, RTT, TTFB, BW)
8: end procedure

9: procedure MAIN( )
10:    CONFIGURE TOR CLIENT( )
11:    while NOT FINISHED( ) do
12:        DUMPGUARDS( )
13:        path ← FINDPATH( )
14:        WORKER(path)              ▷ runs asyncronously
15:    end while
16: end procedure
```

between sending a request using the HTTP HEAD method[5] to google.com and receiving the first byte of the response, the TTFB. Since google.com is served from multiple data centers, the website responds quickly to requests independent of the exit node's location.

Throughput Measurement. We registered the domain torrtt.info, and set up an HTTP server to serve a static HTML page. Throughput is measured by the time it takes to download that file, whose size is exactly $5\ \mathrm{MB}$ in accordance with Tor's bandwidth scanners [29]. To reduce the bias a single webserver serving the web page introduces, the page is delivered by CloudFlare's Content Delivery Network (CDN). This reduces the distance and, thus, potentially the latency between the exit node of a particular circuit and the destination webserver.

6.4. Anonymity Metrics

Before any performance-optimizing modification can be integrated into Tor, it is important to study its effect on the anonymity the system provides. In low-latency systems, anonymity is mostly quantified by the probability that an adversary occupies both the entry and the exit node of a user's circuit. We will use two different metrics to quantify anonymity: normalized Shannon-Entropy and an adopted Gini coefficient.

Shannon Entropy. Serjantov and Danezis [30] and Diaz et al. [31] independently proposed evaluation frameworks that use Shannon entropy as a metric for quantifying anonymity. Entropy over all entry and exit node combinations Ψ is defined as

$$E(X) = -\sum_{i=1}^{|\Psi|} p_i * ld(p_i)$$

where i refers to a particular combination of entry and exit nodes and p_i to the probability that this combination is used. The entropy is then normalized relative to the maximal entropy $ld(|\Psi|)$ to ensure that $E_{norm} \in [0, 1]$. $E_{norm} = 1$ implies that nodes are selected uniformly at random, and $E_{norm} < 1$ implies a bias.

Gini Coefficient. Another metric for quantifying anonymity involves the calculation of the skew in node selection. For any other scheme than the selection uniformly at random, the number of nodes an attacker must compromise is below the maximum. To quantify such a disproportionality of node selection, Snader and Borisov [32] adopted the Gini coefficient, an equality metric commonly used in economics. It is defined as

$$G = \frac{1}{\mu} \int_0^\infty CDF(x)(1 - CDF(x))\,\mathrm{dx}$$

where CDF is the observed Cumulative Distribution Function, describing how often a node was selected, and μ is its mean. A Gini coefficient $G = 0$ represents perfectly uniform node selection, and $G = 1$ implies perfect inequality, i.e., the same node is always chosen.

6.5. Experimental Environment

To examine the influence of the client's location on the circuits' RTTs, we deployed NavigaTor[6] on several hosts from PlanetLab [23], a testbed for computer networking research. We restricted the number of measuring hosts, since we want to avoid stressing the live Tor network with too many concurrent measurements. We selected 19 nodes from PlanetLab Europe, biasing towards diversity in terms of geographical distribution, upstream ISP, and country. Additionally, we aimed for nodes that provide sufficient hardware resources.

7. Results

As part of our study on the live Tor network, we observed and analyzed the influence of the CBT, the congestion-aware, and the Circuit-RTT methods on the quality of circuits with regard to latency, throughput, and anonymity. To that end, we conducted several experiments.

7.1. Continuous RTT Measurements

In the first experiment, we examined the stability of RTT measurements on individual circuits over time. From a single host computer, the RTT of each circuit was measured $100\,000$ times. We observed that the RTTs fluctuate dramatically over time on the majority of circuits, confirming the observations [7] mentioned in Section 3.5 on page 3. Occasionally, we observed spikes at $100\ \mathrm{ms}$ and $1000\ \mathrm{ms}$ intervals, which correspond to the refilling intervals of the token bucket algorithm.

5. This method is similar to the HTTP GET method, but requests headers only without the content body.

6. We used Tor version 0.2.3.25 as basis, which was the latest stable version when we started our experiments.

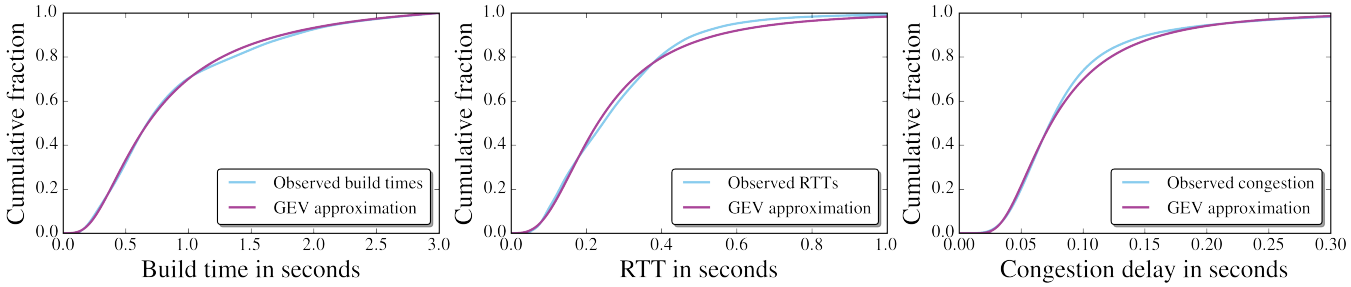

Figure 2. Cumulative Distribution Functions (CDFs) of build-times, RTTs, and congestion delays, and the corresponding GEV approximations.

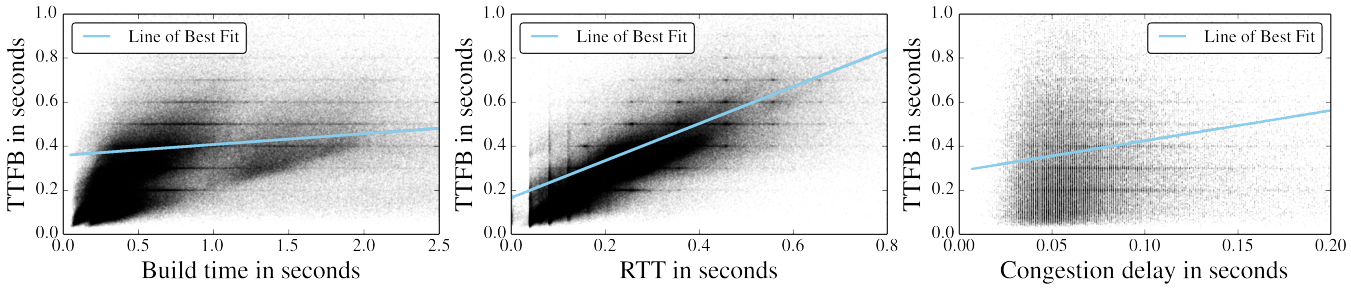

Figure 3. Scatterplots visualizing the relation of TTFB to build-times, to RTTs, and to congestion delays.

If every client were to measure the RTT of each circuit continuously, the Tor network would not be able to handle this additional load well; thus, the number of measurements has to be within a reasonable limit. In an early evaluation, we encountered that the accuracy of assessing a circuit's RTT is only slightly improved when increasing the number of measurements within such reasonable limit. Therefore, we focus on just a single RTT measurement per circuit, considering the overall distribution alone.

7.2. Statistical Analysis

The next experiment was conducted to examine the overall distribution of build-times, congestion delays, and RTTs. On each of the 19 PlanetLab hosts one million circuits were built to measure the build-time and RTT once on each circuit, assuming that they fit the same distribution on any client – apart from the distribution's particular parameters. Congestion delays were measured in a later experiment on a single host which built 100 000 circuits.

Distribution. Fig. 2 depicts the CDFs of build-times, congestion delays, and RTTs, and the corresponding Generalized Extreme Value (GEV) distribution with estimated parameters. GEV is a superset of extreme value distributions which includes the Fréchet distribution. It is equivalent to Fréchet, as long as its parameter ξ is greater than 0; which it is in this case. We verified that the shape of the CDFs across any host can be approximated closely by a GEV distribution, using the χ^2, the Anderson–Darling, and the Kolmogorov-Smirnov test. The results support our assumption that the probability distributions of build-times and RTTs are identical across different clients, ignoring the differences resulting from the distribution's particular parameters. Thus, every client should be able to calculate its individual timeout value.

Correlation. To determine which of the examined schemes is more suitable to improve end-to-end network latency, we examine the correlation both between build-time and TTFB, between congestion delay and TTFB, and between RTT and TTFB. The scatterplots in Fig. 3 suggest a positive correlation between all of them, but the correlation to be strongest between RTT and TTFB. Since many outliers do exist, a correlation method that is robust against outliers is required. To quantify the correlation, we use Spearman's ρ, which essentially is Pearson's r on ranked rather than on observed values and, thus, is more robust to outliers. The correlation coefficient ρ can range from -1 (negative correlation) to 1 (positive correlation).

Using a p-value of 0.0000 for the hypothesis test, we find a positive correlation with TTFB for all three schemes; however, the correlation coefficient ρ is clearly higher for RTTs (0.825) than both that for CBTs (0.588) and that for congestion delays (0.191). Although this result may appear intuitive, we are the first to empirically show, not least because no tool for large-scale live Tor network measurements existed before NavigaTor. In any case, this evidence for a higher correlation between RTT and TTFB strengthens our assumption that the Circuit-RTT method can be used to

TABLE 1. MEDIAN AND 90^{th} PERCENTILE TTFB.

PlanetLab Site	Overall	CBT 80%	Circuit-RTT 80%	CBT 50%	Circuit-RTT 50%
cs.uit.no	340 ms / 614 ms	307 ms / 539 ms	299 ms / 497 ms	258 ms / 471 ms	214 ms / 384 ms
ece.upatras.gr	342 ms / 613 ms	307 ms / 536 ms	300 ms / 495 ms	257 ms / 468 ms	215 ms / 381 ms
upf.edu	332 ms / 605 ms	301 ms / 529 ms	294 ms / 488 ms	249 ms / 455 ms	209 ms / 373 ms
ait.ie	316 ms / 596 ms	289 ms / 513 ms	278 ms / 472 ms	228 ms / 431 ms	196 ms / 357 ms
cs.vu.nl	300 ms / 577 ms	269 ms / 500 ms	262 ms / 458 ms	204 ms / 406 ms	177 ms / 340 ms
virtues.fi	331 ms / 606 ms	300 ms / 529 ms	292 ms / 489 ms	244 ms / 453 ms	204 ms / 370 ms
mta.ac.il	411 ms / 714 ms	383 ms / 639 ms	372 ms / 592 ms	341 ms / 582 ms	289 ms / 469 ms
csg.uzh.ch	312 ms / 594 ms	283 ms / 511 ms	274 ms / 471 ms	219 ms / 423 ms	190 ms / 353 ms
diku.dk	307 ms / 583 ms	278 ms / 500 ms	270 ms / 460 ms	213 ms / 410 ms	186 ms / 342 ms
ple.silweb.pl	329 ms / 611 ms	299 ms / 534 ms	292 ms / 494 ms	241 ms / 453 ms	204 ms / 377 ms
eecs.qmul.ac.uk	299 ms / 565 ms	267 ms / 491 ms	259 ms / 447 ms	202 ms / 400 ms	176 ms / 333 ms
lkn.ei.tum.de	306 ms / 588 ms	276 ms / 503 ms	266 ms / 462 ms	208 ms / 416 ms	180 ms / 346 ms
fc.univie.ac.at	308 ms / 589 ms	278 ms / 505 ms	268 ms / 464 ms	214 ms / 415 ms	186 ms / 348 ms
dis.unina.it	345 ms / 623 ms	310 ms / 548 ms	306 ms / 517 ms	262 ms / 473 ms	234 ms / 419 ms
ifi.uio.no	322 ms / 600 ms	294 ms / 518 ms	284 ms / 478 ms	232 ms / 439 ms	199 ms / 365 ms
jcp-consult.net	348 ms / 632 ms	316 ms / 559 ms	305 ms / 507 ms	268 ms / 485 ms	222 ms / 398 ms
s3.kth.se	322 ms / 598 ms	298 ms / 526 ms	283 ms / 478 ms	240 ms / 447 ms	200 ms / 367 ms
iscte.pt	336 ms / 605 ms	304 ms / 529 ms	300 ms / 496 ms	254 ms / 463 ms	214 ms / 394 ms
ait.ac.th	539 ms / 815 ms	521 ms / 771 ms	505 ms / 707 ms	507 ms / 754 ms	444 ms / 604 ms
Average	339 ms / 617 ms	309 ms / 541 ms	300 ms / 499 ms	255 ms / 465 ms	218 ms / 385 ms

improve end-to-end network latency in the Tor network.

7.3. Latency Evaluation

To test our hypothesis that Circuit-RTT can be used to improve latency in the Tor network as experienced by users, we explore the effect on TTFB of different acceptance rates of circuits, using CBT, Circuit-RTT, and a combination of both methods.

Since it seems reasonable and increases the comparability of our results, we use 1000 values for the estimating the parameters of the particular GEV distribution; this number is currently also used for CBT in Tor. The timeout value is updated every time a new measurement value is gathered; either a circuit build-time, a RTT, or, in later experiments, a congestion-delay. Algorithm 2 sketches that process.

We verified that only the particular timeout values vary between PlanetLab hosts, but the overall percentage of discarded circuits does not. Thus, we find the statistical prediction to work well for both CBT and Circuit-RTT.

Results. Using the fastest 80% of circuits, the median TTFB is reduced from 339 ms by 8.8% to 309 ms with CBT and by 11.4% to 300 ms with Circuit-RTT. The 90th percentile TTFB is reduced from 617 ms by 12.3% to 541 ms with CBT and by 19.2% to 499 ms with Circuit-RTT. Reducing the percentage of accepted circuits to 50%, the median TTFB is improved by 24.9% to 255 ms with CBT and by 35.8% to 218 ms with Circuit-RTT. The 90th percentile TTFB is reduced by 24.6% to 465 ms with CBT and by 37.6% to 385 ms with Circuit-RTT. Table 1 shows the median and the 90^{th} percentile TTFB for all circuits and using the fastest 80% of circuits according to, respectively the CBT and Circuit-RTT methods for all hosts.

Algorithm 2 Calculation of the timeout value

```
1:  procedure CALCULATE_TIMEOUT(values, new_val)
2:      timeout ← ∞
3:      VALUES.APPEND(new_val)
4:      if length(values) ≥ 1000 then
5:          params ← ESTIMATE GEV PARAMS(values)
6:          pdf ← CALCULATE GEV PDF(params)
7:          timeout ← CALC 80TH PERCENTILE(pdf)
8:          DELETE(values[0])
9:      end if
10:     return timeout
11: end procedure
```

As assumed, end-to-end network latency is improved when the percentage of accepted circuits is reduced, according to any of both methods. We find that Circuit-RTT has more of an effect on the TTFB than CBT, for any accepted percentage of circuits except very low ones, at which hardly any circuit would be used anymore.

Fig. 4 on the next page depicts the CDFs of TTFB for all circuits and for CBT and Circuit-RTT with the fastest 80% and 50% of circuits respectively. It shows the performance improvement of Circuit-RTT over CBT, especially when reducing the percentage of accepted circuits.

Due to the sheer number of measurements, the difference in achieved TTFB with Circuit-RTT intuitively seems significantly different to that achieved with CBT. To clearly assess the significance, however, we use the non-parametric Kruskal-Wallis test. The Kruskal-Wallis test does not assume normality of the distribution for the sample populations, as the one-way ANOVA does; it does assume, however, that the populations have the same distribution,

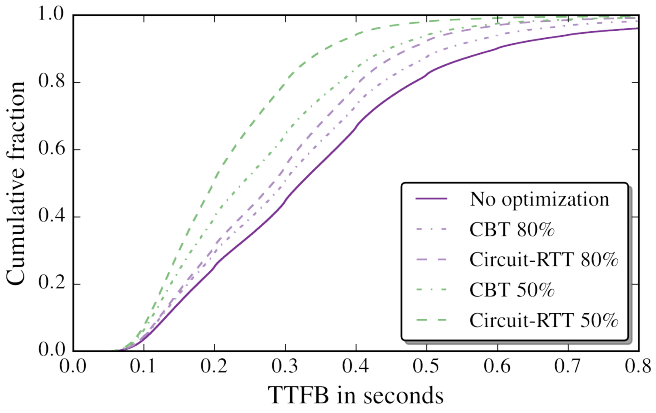

Figure 4. CDFs of TTFB for the fastest percentages of circuits.

which they do in our case. The lowest calculated value of the Kruskal-Wallis test from all hosts (220.81) is greater than the critical value for χ^2 (68.76) and, thus, the TTFB achieved with Circuit-RTT is significantly different to that achieved with CBT.[7] This difference can, therefore, not be attributed to random influences, such as jitter.

Combination of CBT and Circuit-RTT. To explore whether a combination of both methods could improve latency further, we considered only circuits whose build-times and RTTs were below the corresponding timeout values, and then calculated the effective acceptance rate. We find that Circuit-RTT improves the TTFB more than the combination of both methods does. Hence, we focus on Circuit-RTT as a replacement for CBT, instead of using it as an additional method.

7.4. Anonymity Evaluation

To estimate the strength of protection that would be achieved for end-users, we use normalized Shannon entropy over the observed combinations of entry and exit nodes. We additionally use the Gini coefficient to quantify the skew in the selection of entry and exit nodes.

When using the fastest 80% of circuits, the Gini coefficient is on average increased from 0.222 overall by 4.8% to 0.232 with CBT and by 7.8% to 0.239 with Circuit-RTT. We observe a slightly higher skew towards certain nodes with Circuit-RTT than with CBT at the default acceptance rate. This bias increases when reducing the acceptance rate. Using the fastest 50% of circuits, the Gini coefficient is increased by 16.8% to 0.259 with CBT and by 25.8% to 0.279 with Circuit-RTT.

The normalized Shannon entropy is decreased from 0.842 overall by 20.1% to 0.672 with CBT but only by 17.2% to 0.697 with Circuit-RTT when using the fastest 80% of circuits. Table 2 shows the normalized Shannon entropy values for all circuits and using the fastest 80% of circuits according to CBT and Circuit-RTT on all hosts.

7. For every host we get a p-value $< 2.2e^{-16}$.

TABLE 2. NORMALIZED SHANNON ENTROPY

PlanetLab Site	No opt.	CBT 80%	Circuit-RTT 80%
cs.uit.no	0.829	0.661	0.682
ece.upatras.gr	0.836	0.666	0.688
upf.edu	0.829	0.661	0.683
ait.ie	0.829	0.658	0.683
cs.vu.nl	0.834	0.659	0.691
virtues.fi	0.829	0.657	0.680
mta.ac.il	0.834	0.668	0.689
csg.uzh.ch	0.839	0.664	0.693
diku.dk	0.870	0.690	0.720
ple.silweb.pl	0.834	0.661	0.689
nrl.eecs.qmul.ac.uk	0.871	0.694	0.722
lkn.ei.tum.de	0.829	0.654	0.680
fc.univie.ac.at	0.829	0.653	0.681
dis.unina.it	0.861	0.683	0.722
ifi.uio.no	0.834	0.662	0.688
jcp-consult.net	0.849	0.694	0.712
s3.kth.se	0.829	0.659	0.683
iscte.pt	0.901	0.728	0.753
ait.ac.th	0.829	0.705	0.701
Average	0.842	0.672	0.697

TABLE 3. MEDIAN THROUGHPUT IN Mbit/s

PL Site	No opt.	CBT 80%	CBT 50%	Circuit-RTT 80%	Circuit-RTT 50%
cs.uit.no	2.62	3.10	4.00	3.14	4.45
ece.upatras.gr	3.21	3.93	4.91	4.00	5.31
virtues.fi	2.88	3.49	4.70	3.58	5.29
csg.uzh.ch	3.21	3.92	5.69	3.98	6.34
lkn.ei.tum.de	3.13	3.88	5.77	3.94	6.32
s3.kth.se	2.71	3.07	3.61	3.11	3.80
ait.ac.th	1.28	1.31	1.40	1.32	1.39
Average	2.72	3.24	4.30	3.30	4.70

Using the fastest 50% of circuits, normalized entropy is decreased by 46.3% to 0.452 with CBT and by 48.2% to 0.436 with Circuit-RTT. Unlike the Gini coefficient, entropy is slightly improved by Circuit-RTT compared to the values for CBT at the default acceptance rate.

Comparing Circuit-RTT to CBT, we note a minor skew in selecting certain nodes, but an improved selection probability over all combinations of entry and exit nodes at the current default acceptance rate of 80%. When lowering the acceptance rate, we observe that entropy varies only slightly between the CBT and the Circuit-RTT method, but the Gini coefficient increases in favor of CBT.

7.5. Throughput Evaluation

In addition to examining the effect the use of CBT or Circuit-RTT have on latency and on anonymity, we study the effect those methods have on achievable throughput, which is also an important property for users. To this end, we conducted another experiment on seven hosts from Planet-

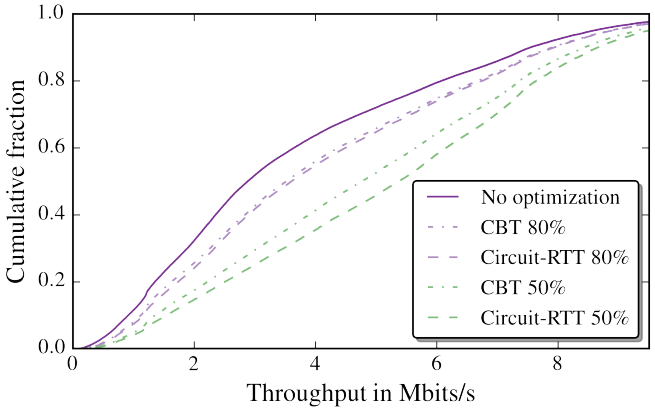

Figure 5. CDFs of throughput for all circuits and for CBT and Circuit-RTT with the fastest 50% and 80% of circuits respectively.

Lab, in which each host built 100 000 circuits to measure throughput.

Table 3 on the previous page shows the median throughput achieved on every host for all circuits and using the fastest 80% of circuits according to CBT and Circuit-RTT. When using the fastest 80% of circuits, median throughput increases from 2.72 Mbit/s by 19.1% to 3.24 Mbit/s with CBT and by 21.1% to 3.30 Mbit/s with Circuit-RTT. The 90th percentile improves by 20.0% from 0.92 Mbit/s to 1.10 Mbit/s with CBT, and by 24.9% to 1.14 Mbit/s with Circuit-RTT. Using the fastest 50% of circuits, median throughput increases by 58.0% to 4.30 Mbit/s with CBT and by 72.8% to 4.70 Mbit/s with Circuit-RTT.

Fig. 5 depicts the CDFs of throughput for all circuits and for CBT and Circuit-RTT with the fastest 80% and 50% of circuits respectively. It shows a slight improvement of Circuit-RTT over CBT at the default acceptance rate of 80% and a clear improvement when reducing the percentage of accepted circuits to 50%.

7.6. Congestion-Aware Scheme

To compare the CBT and the Circuit-RTT methods to the congestion-aware scheme in terms of latency, throughput, and anonymity, we conducted another experiment. To that end, we built 100 000 circuits on a single host and measured the RTT of each circuit 5 times to deduce the nodes' congestion delays. We create sets of three (preemptively) built circuits and select the circuit with the lowest congestion-delay from each of these sets, assuming that this scheme will improve performance significantly.

Results differ, however, as shown in Table 4; entropy drops drastically by 67.9%, due to the low acceptance rate, and median TTFB is improved only by 7.1%. In comparison, Circuit-RTT achieves an improvement in TTFB of 14.9% at the default acceptance rate so that entropy is reduced only by 17.6%. Improvements in throughput by the congestion-aware scheme are similar to those by Circuit-RTT, but, considering the drastically reduced entropy, the congestion-aware scheme is clearly outperformed, nevertheless.

TABLE 4. COMPARISON OF CIRCUIT-RTT TO THE ORIGINAL AND THE MODIFIED CONGESTION-AWARE SCHEMES.

Scheme	TTFB in ms	Throughput	Entropy
No optimization	295 ±0%	4.51 ±0%	0.941 ±0%
Circuit-RTT 80%	251 -14.9%	5.08 +12.6%	0.776 -17.6%
Cong.-Aware orig	274 -7.1%	5.04 +11.6%	0.302 -67.9%
Cong.-Aware 33%	259 -12.2%	5.27 +16.9%	0.380 -59.6%

TABLE 5. LATENCY OVER TIME.

Scheme	Average latency	Standard deviation
No optimization	352 ms (±0%)	142 ms (±0%)
CBT 80%	315 ms (-10.6%)	128 ms (-9.5%)
Circuit-RTT 80%	307 ms (-12.9%)	127 ms (-10.4%)
Congestion-Aware 80%	339 ms (-3.9%)	125 ms (-11.9%)
CBT 50%	271 ms (-23.2%)	116 ms (-18.3%)
Circuit-RTT 50%	252 ms (-28.4%)	116 ms (-18.2%)
Congestion-Aware 50%	326 ms (-7.6%)	113 ms (-20.1%)

The discrepancy between our results and those of previous research can be explained as follows: Wang et al. [8] conducted their measurements on the live Tor network when the token bucket refill intervall still was at 1000 ms instead of 100 ms and, thus, delays due to congestion were significantly larger. We think that this is the reason for us to observe very similar median congestion delays (4 to 5 ms vs. 3 to 5 ms), but a 90th percentile (37 to 46 ms) which is clearly below that observed by the congestion-aware authors (>800 ms). Additionally, it may be the case that nodes are not that congested anymore for various reasons. Wacek et al. [20] ran their simulations also on a Tor version with refill interval of 1000 ms.

As shown previously in the beginning of this section, we found that congestion delays, too, can be approximated by a GEV distribution. We find that using this approximation to select the fastest percentage of circuits according to their congestion delays achieves better results than the original congestion-aware scheme, in which one out of three preemptively built circuits is selected. As shown in Table 4, the modified congestion-aware scheme using the fastest 33% of circuits achieves better improvements in TTFB and throughput, but sacrifices less entropy than the original congestion-aware scheme. We, therefore, stick to the modified congestion-aware scheme for detailed comparison.

7.7. Latency over Time

In this last experiment, we evaluate the influence of the CBT, the Circuit-RTT, and the (modified) congestion-aware schemes also on latency over time. To this end, we additionally measured the TTFB 600 times over a duration of about 10 minutes on each circuit, equivalent to the default lifetime of a circuit in Tor. At the default acceptance rate of 80%, Circuit-RTT improves the average latency most (by

12.9% from 352 to 307 ms), compared to CBT (by 10.6% to 315 ms) and the modified congestion-aware scheme (by 3.9% to 339 ms). Although the gap is smaller, Circuit-RTT also improves throughput more than the other schemes. The standard deviation of latency, however, is least for the congestion-aware scheme. When reducing the percentage of accepted circuits to 50%, all schemes show improvements but the gap between them widens, showing that Circuit-RTT works best for discarding slow circuits. Table 5 on the previous page lists the results in detail.

8. Discussion

As shown, our Circuit-RTT scheme can be used to improve latency and throughput of Tor circuits. In this section we will briefly discuss the limitations of our approach as well as future work.

Latency Attacks. Geddes, Jansen, and Hopper [33] have found that algorithms which improve throughput or responsiveness of circuits also increase the effectiveness of latency-based attacks; especially those attacks either attempting to identify possible Guards nodes or trying to reduce the set of possible clients. Thus, we concede that the latency improvements resulting from the use of Circuit-RTT would also increase the effectiveness of latency-based attacks, but this issue is inherent to any method that improves latency.

Simulation. Since we conducted experiments on the live Tor network, using at most 19 clients concurrently, we cannot predict with certainty the changes the network would undergo if Tor adopted Circuit-RTT and every client were to use it. Nevertheless, we maintain that clients picking better-performance circuits would not cause the overall performance to decrease but that the gain would actually be seen by all users, because Circuit-RTT allows clients to react quickly to changing conditions by assessing the performance of a circuit prior use. In this way, slow circuits can be discarded before they have a negative impact on user experience. However, to come to any definitive conclusion on the entire Tor population of users, whole-network simulation with ExperimenTor [34] or with Shadow [35] would be required.

Network Adversaries. Previous research has shown that adversaries which are able to observe Autonomous Systems (ASs) or Internet eXchange Points (IXPs) pose a threat to users' anonymity. [36, 37, 38, 39, 28] Unfortunately, the classic entropy model we used to quantify anonymity does not cover this type of network adversaries. A path selection algorithm that aims to protect against one type of network adversary can potentially weaken defenses against other threats. It becomes even more complicated when the country or AS of both the client and the final destination are to be considered as well. This is still an area of ongoing research.

Improving Quality of Service. Although not generally applicable due to the increased load on the network, extensive measurements might allow increasing quality of service properties with regard to latency and throughput. This improved quality of service could facilitate the use of certain applications such as VoIP.

When conducting the experiment involving numerous RTT measurements on individual circuits, we occasionally encountered spikes at 100 ms and 1000 ms intervals. Since these values correspond to the token bucket refill intervals, it is a strong indication that one or more nodes in the circuit is being asked to forward more traffic than it can handle. For this reason, we assume that RTT measurements can be used to detect circuits containing nodes which are impeded by their bandwidth limiter.

9. Conclusion

One of the most important goals of Tor's design is to provide a low-latency and high-throughput transport service that supports latency-sensitive applications, such as web browsers, which currently make up the vast majority of connections in the Tor network. However, users still experience variable delays on connecting to servers which has been shown to be especially harmful for browsing the web. Since the queuing and processing delays within many nodes fluctuate dramatically over time, it is vital that clients can react quickly to changing conditions.

The basic idea of our Circuit-RTT approach is that clients create a local view of observed RTTs by actively measuring the circuits' RTTs. After a circuit has been established, the client measures the circuit's RTT and discards that circuit if its RTT is above a client-specific, variable timeout value, which is calculated using a priori knowledge about the statistical distribution of RTTs. RTT measurements are very lightweight, since they require only a single cell to be sent from the client to the exit node and back. Hence, the additional load on the Tor network can be assumed to be very small.

We were the first to conduct large-scale measurements on the live Tor network. To this end, we used our high performance measurement software NavigaTor [22], which enables building and measuring millions of circuits within days, without stressing the Tor network. As part of our study, we conducted several experiments on the live Tor network and analyzed the influence of the current state-of-the-art method CBT, the more recently proposed congestion-aware scheme, and our Circuit-RTT method on the quality of circuits with regard to latency, throughput, and anonymity. Clearly, we observed a significantly stronger correlation with end-to-end network latency using RTTs than using build-times or using congestion delays. Furthermore, we discovered that the congestion-aware scheme in its original design achieves only minor performance improvements in the current Tor network and drastically reduces entropy. For the purpose of comparing CBT to the Circuit-RTT method, we deployed NavigaTor on several hosts from PlanetLab, which allows us to infer that our measurement results hold independently of the client's location. We studied the trade-off between the quality of protection and the quality of service, showing that anonymity is decreased using these

schemes, but that both latency and throughput are improved significantly. Our measurements show that, with the current default acceptance rate of 80%, the use of the Circuit-RTT method achieves improvements in latency and throughput compared both to the current state-of-the-art method CBT and to the more recently proposed congestion-aware scheme. Decreasing the accepted percentage rate clearly improves latency and throughput further, but with adverse effects on anonymity.

In conclusion, we emphasize that keeping the performance costs associated with an anonymity system as low as possible makes it more attractive to users and, thus, ultimately enhances the system's anonymity properties.

Acknowledgment

We would like to thank Mike Perry for his many comments and invaluable support during early development stages, Tanja Zseby for her encouragement to publish this paper, and Caroline Durlacher for proofreading an early version of this paper. Furthermore, we would like to thank the anonymous reviewers for their feedback. This research was funded by COMET K1, FFG - Austrian Research Promotion Agency.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson. "Tor: The Second-Generation Onion Router". In: *Proceedings of the 13th USENIX Security Symposium*. Aug. 2004.

[2] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. "Hiding Routing Information". In: *Proceedings of Information Hiding: First International Workshop*. Ed. by R. Anderson. Springer-Verlag, LNCS 1174, May 1996, pp. 137–150.

[3] The Tor Project, Inc. *Tor Metrics Portal: Users*. https://metrics.torproject.org/userstats-relay-country.html. [Online; accessed 03-December-2015]. 2015.

[4] K. Bauer. "Improving Security and Performance in Low Latency Anonymity Networks". PhD thesis. University of Colorado, May 2011.

[5] R. Dingledine and N. Mathewson. "Anonymity Loves Company: Usability and the Network Effect". In: *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*. Ed. by R. Anderson. Cambridge, UK, June 2006.

[6] R. Dingledine and S. J. Murdoch. *Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it*. https://www.torproject.org/press/presskit/2009-03-11-performance.pdf. [Online; accessed 05-January-2016]. 2009.

[7] P. Dhungel et al. "Waiting for Anonymity: Understanding Delays in the Tor Overlay". In: *Peer-to-Peer Computing*. 2010, pp. 1–4.

[8] T. Wang et al. "Congestion-aware Path Selection for Tor". In: *Proceedings of Financial Cryptography and Data Security (FC'12)*. Feb. 2012.

[9] D. McCoy et al. "Shining Light in Dark Places: Understanding the Tor Network". In: *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*. Ed. by N. Borisov and I. Goldberg. Leuven, Belgium: Springer, July 2008, pp. 63–76.

[10] M. Huber, M. Mulazzani, and E. Weippl. "Tor HTTP Usage and Information Leakage". In: *Proceedings of the 11th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS 2010)*. Vol. 6109. LNCS. Linz, Austria: Springer, May 2010, pp. 245–255.

[11] R. Dingledine et al. *Tor: The Second-Generation Onion Router (2012 DRAFT*. https://people.torproject.org/~karsten/tor-design-2012-11-14.pdf. [Online; accessed 05-January-2016]. Nov. 2012.

[12] R. Dingledine and N. Mathewson. *Tor Path Specification*. https://gitweb.torproject.org/torspec.git/tree/path-spec.txt. [Online; accessed 05-January-2016]. 2015.

[13] M. Wright et al. "The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems". In: *ACM Transactions on Information and System Security (TISSEC)* 4.7 (Nov. 2004), pp. 489–522.

[14] R. Dingledine et al. "One fast guard for life (or 9 months)". In: *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*. 2014.

[15] G. Kadianakis and N. Hopper. *Implement the single guard node proposal*. https://trac.torproject.org/projects/tor/ticket/11480. [Online; accessed 05-January-2016]. 2015.

[16] M. Backes et al. "(Nothing else) MATor (s): Monitoring the Anonymity of Tor's Path Selection". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 513–524.

[17] F. Cangialosi, D. Levin, and N. Spring. "Ting: Measuring and Exploiting Latencies Between All Tor Nodes". In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. ACM. 2015, pp. 289–302.

[18] A. Panchenko and J. Renner. "Path Selection Metrics for Performance-Improved Onion Routing". In: *Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet*. SAINT '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 114–120. ISBN: 978-0-7695-3700-9. DOI: 10.1109/SAINT.2009.26. URL: http://dx.doi.org/10.1109/SAINT.2009.26.

[19] T. Wang et al. *Congestion-aware Path Selection for Tor*. [Accessed 05-January-2016]. 2011. URL: http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-20.pdf.

[20] C. Wacek et al. “An Empirical Evaluation of Relay Selection in Tor”. In: *Proceedings of the Network and Distributed System Security Symposium - NDSS’13*. Internet Society, Feb. 2013.

[21] A. Panchenko, F. Lanze, and T. Engel. “Improving Performance and Anonymity in the Tor Network”. In: *Proceedings of the 31st IEEE International Performance Computing and Communications Conference (IPCCC 2012)*. Dec. 2012.

[22] Robert Annessi and Martin Schmiedecker. *NavigaTor*. https://naviga-tor.github.io/. [Online; accessed 06-January-2016].

[23] B. Chun et al. “Planetlab: An overlay testbed for broad-coverage services”. In: *ACM SIGCOMM Computer Communication Review* (2003), pp. 3–12.

[24] S. Köpsell. “Low Latency Anonymous Communication – How Long Are Users Willing to Wait?” In: *Emerging Trends in Information and Communication Security*. Ed. by G. Müller. Vol. 3995. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 221–237. DOI: 10.1007/11766155_16. URL: http://dx.doi.org/10.1007/11766155_16.

[25] A. Pfitzmann and M. Köhntopp. “Anonymity, unobservability, and pseudonymity—a proposal for terminology”. In: *Designing privacy enhancing technologies*. Springer. 2001, pp. 1–9.

[26] N. Mathewson, R. Dingledine and M. Perry. *TorCtl Python Bindings*. https : / / gitweb . torproject . org / pytorctl . git. [Online; accessed 05-January-2016]. 2015.

[27] T. Elahi et al. “Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor”. In: *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2012)*. Raleigh, NC, USA: ACM, Oct. 2012.

[28] A. Johnson et al. “Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries”. In: *Proceedings of the 20th ACM conference on Computer and Communications Security (CCS 2013)*. Nov. 2013.

[29] K. Loesing, M. Perry, and A. Gibson. *Bandwidth Scanner specification*. https://gitweb.torproject.org/torflow . git / plain / NetworkScanners / BwAuthority / README . spec . txt. [Online; accessed 05-January-2016]. 2013.

[30] A. Serjantov and G. Danezis. “Towards an Information Theoretic Metric for Anonymity”. In: *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Ed. by R. Dingledine and P. Syverson. Springer-Verlag, LNCS 2482, Apr. 2002.

[31] C. Diaz et al. “Towards measuring anonymity”. In: *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Ed. by R. Dingledine and P. Syverson. Springer-Verlag, LNCS 2482, Apr. 2002.

[32] R. Snader and N. Borisov. “A Tune-up for Tor: Improving Security and Performance in the Tor Network”. In: *Proceedings of the Network and Distributed Security Symposium - NDSS ’08*. Internet Society, Feb. 2008.

[33] J. Geddes, R. Jansen, and N. Hopper. “How Low Can You Go: Balancing Performance with Anonymity in Tor”. In: *Proceedings of the 13th Privacy Enhancing Technologies Symposium (PETS 2013)*. July 2013.

[34] K. Bauer et al. “ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation”. In: *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET 2011)*. Aug. 2011.

[35] R. Jansen and N. Hopper. “Shadow: Running Tor in a Box for Accurate and Efficient Experimentation”. In: *Proceedings of the Network and Distributed System Security Symposium - NDSS’12*. Internet Society, Feb. 2012.

[36] N. Feamster and R. Dingledine. “Location Diversity in Anonymity Networks”. In: *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*. Washington, DC, USA, Oct. 2004.

[37] S. J. Murdoch and P. Zieliński. “Sampled Traffic Analysis by Internet-Exchange-Level Adversaries”. In: *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*. Ed. by N. Borisov and P. Golle. Ottawa, Canada: Springer, June 2007.

[38] M. Edman and P. F. Syverson. “AS-awareness in Tor path selection”. In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*. Ed. by E. Al-Shaer, S. Jha, and A. D. Keromytis. Chicago, Illinois, USA: ACM, Nov. 2009, pp. 380–389. ISBN: 978-1-60558-894-0.

[39] M. Akhoondi, C. Yu, and H. V. Madhyastha. “LASTor: A Low-Latency AS-Aware Tor Client”. In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. May 2012.