On the Accuracy of Tor Bandwidth Estimation

Rob Jansen and Aaron Johnson

U.S. Naval Research Laboratory
{rob.g.jansen,aaron.m.johnson}@nrl.navy.mil

Abstract. The Tor network estimates its relays' bandwidths using relay self-measurements of client traffic speeds. These estimates largely determine how existing traffic load is balanced across relays, and they are used to evaluate the network's capacity to handle future traffic load increases. Thus, their accuracy is important to optimize Tor's performance and strategize for growth. However, their accuracy has never been measured. We investigate the accuracy of Tor's capacity estimation with an analysis of public network data and an active experiment run over the entire live network. Our results suggest that the bandwidth estimates underestimate the total network capacity by at least 50% and that the errors are larger for high-bandwidth and low-uptime relays. Our work suggests that improving Tor's bandwidth measurement system could improve the network's performance and better inform plans to handle future growth.

1 Introduction

Tor [12] is an anonymous communication overlay network with thousands of *relays* that forward over 200 Gbit/s of traffic for millions of daily *clients* [3,24] in order to provide unlinkability between the source and destination of traffic flows.

In order to balance client traffic across the relays, Tor relies on TorFlow to estimate of the speed at which relays can forward traffic through the network [29], and these *forwarding capacity* estimates are essential to both the performance and security of the network [5, 20, 21, 30]. A relay's capacity estimate is derived from a self-measurement called the *observed bandwidth*: the highest throughput it has sustained over any ten second period over the last five days (see § 2). This measure is imprecise and may be inaccurate in many realistic cases: (i) a *new* relay will not have forwarded any traffic and thus will be estimated to have a low capacity regardless of its available resources; (ii) a relay that is used *inconsistently* may not sustain a high throughput long enough to result in an accurate capacity estimate; and (iii) a relay that is *underutilized* will underestimate its capacity. TorFlow uses relays' capacity estimates as the basis for its relay selection algorithm that drives more user traffic load to higher-capacity relays [29]. Therefore, inaccurate capacity estimates could result in sub-optimal load balancing which would degrade user-perceived network performance and security [20].

Inaccurate capacity estimates also make it more difficult to understand how to prioritize research and development effort in order to plan future network improvements [31]. For example, obtaining funding to improve Tor scalability is

more challenging without understanding the current limits of the network [26]. Improper network management also complicates relay recruitment and retention, and may dissuade the development of incentive schemes [13, 14, 17–19, 25, 27].

In this paper, we explore the inconsistency in Tor's estimated relay capacities using: (i) passive measurements collected by relays and published by Tor metrics [3]; and (ii) an active relay speed test measurement experiment. In § 3 we study variability in relay capacity estimates, which we use as an indication of inaccurate estimation. We find significant variation in relays' advertised bandwidths: the capacity estimates of 25% of relays vary by more than 41%, the capacity estimates of 10% of relays vary by 71% or more, and some relays' capacity estimates vary by more than 200%. We find that higher variation is associated with lower capacity relays and with relays that are online less frequently. In § 4 we present an active speed test experiment, through which we find that: (i) Tor underestimates its total capacity by about 50%; (ii) most relays increased their capacity estimate following our experiment (some by a 10× or greater factor); and (iii) larger error is associated with high-capacity relays, exit relays, and relays with lower uptimes than with other types of relays. Our results suggest that indeed relay underutilization is a cause of significant error in capacity estimates.

Our work provides the first systematic exploration of the error in Tor's capacity estimation technique, and our results suggest that improvements to capacity estimates could significantly improve load balancing and network performance.

2 Background and Related Work

The Tor Network: The Tor network consists of thousands of *relays* forwarding traffic for millions of *clients* [3, 24]. To assist in balancing client traffic load across relays, Tor assigns a *weight* to each relay according to an estimate of the relay's available bandwidth and publishes relay information (including addresses, weights, and various other *flags*) in a *network consensus* document [2, § 3.4.1]. To use the network, a Tor client downloads the consensus and computes *selection probabilities* from the weights. The client builds a *circuit* through a series of typically three relays, using the selection probabilities to choose a relay for each *position*; relays with the Exit flag typically serve in the *exit* position, relays with the Guard flag (but not the Exit flag) typically serve in the *entry* position, and relays with neither flag serve in the *middle* position [11]. The client tunnels application data (e.g., web requests) through the constructed circuit, rotating to new circuits every 10 minutes (or when they browse to new websites). Although circuits rotate frequently, clients generally use long-term entry Guard relays [9] to help prevent predecessor attacks [33]. To be a Guard, Tor requires that a relay maintain high *uptime*: the percentage of hours during which it is online.

Relay Bandwidth: A relay's *forwarding capacity* is the maximum sustainable rate at which it can forward traffic through the network and is useful for balancing traffic load across relays. Relay operators do not directly report the true forwarding capacity of their relays, so Tor uses a heuristic to estimate it. Each relay calculates its *observed bandwidth* by tracking the highest throughput that it was

able to sustain for any 10 second period during each of the last 5 days [10, § 2.1.1]. To bootstrap the observed bandwidth calculation, a relay conducts a bandwidth self-test when it starts by creating four circuits through Tor and sending 125 KiB over each circuit; if this process completes within 10 seconds, the relay will start with an observed bandwidth of $4 \cdot 125/10 = 50$ KiB/s (≈410 Kbits/s) [8]. Additional remote measurements are conducted by TorFlow [29] (discussed below), and the observed bandwidth is updated over time as a relay forwards client traffic. Relay operators may limit the amount of bandwidth a relay consumes by configuring *average bandwidth* and *burst bandwidth* options, which control the refill rate and size of an internal token bucket rate limiter. Every 18 hours, relays publish a *server descriptor* file [10, § 2.1.1] which contains their latest observed, average, and burst bandwidth values. A relay's *advertised bandwidth* is the minimum of the observed and average bandwidths published by the relay, and is used as a basis for load balancing.

Load Balancing: Tor uses a measurement tool called TorFlow [29] to assist in balancing client traffic across relays. TorFlow measures relay performance by creating two-hop circuits through each relay and downloading files ranging in size from 16 KiB to 64 MiB from a known server through the circuit. TorFlow produces relay weights by: (i) computing the ratio of the measurement speed of each relay to the mean measurement speed of all relays; and (ii) multiplying each relay's ratio by its advertised bandwidth. The relay weights are published in the consensus and used to compute relay selection probabilities as described above.

Related Work: Previous work has established that TorFlow is insecure and vulnerable to manipulation, in part because a relay can detect when it is being measured [5,20,21,32]. Several alternative bandwidth measurement systems that produce relay weights have been proposed. SmarTor [4] and Simple Bandwidth Scanner [23] are similar in measurement design to TorFlow and suffer from similar limitations. EigenSpeed proposes that relays conduct peer measurement, and produces per-flow throughput estimates rather than estimates of relay forwarding capacity [30]. PeerFlow is a passive peer measurement system that proposes a secure aggregation inference technique to produce relay capacity estimates from multiple peers observations [21]. TightRope proposes a centralized approach for optimally balancing load given a set of accurate capacity weights [7], and Ting focuses on measuring latencies between relays [6].

Dingledine outlines the lifecycle of a new relay and explains that it can take three days for a relay to be measured by TorFlow, several weeks for a relay to obtain the Guard flag, and even longer to reach steady state [8]. Dingledine motivates the need for further analysis of Tor metrics data to better understand relay operations in the real world. Using both passive and active measurements, our work provides the first systematic exploration of the error in Tor's capacity estimation technique. More recently, Greubel et al. analyze load distribution in Tor and find that relays with more forwarding capacity are associated with larger relay weights [15]. Although we are focused on measuring the accuracy of forwarding capacity estimates rather than relay weights, the association established by Greubel et al. will aid in explaining some of our results.

3 Analysis of Tor Metrics Data

To better understand the accuracy of Tor's capacity-estimation heuristic, we analyze publicly available Tor metrics data [3]. Relays passively measure throughput over time and publish bandwidth information in their server descriptors [10, § 2.1.1], while the load-balancing weights that TorFlow derives from the bandwidth information are published in network consensus files [2, § 3.4.1]. The Tor Project has collected these documents for over a decade [3], and we analyze the data published throughout the 52 week period starting on `2018-08-01`.

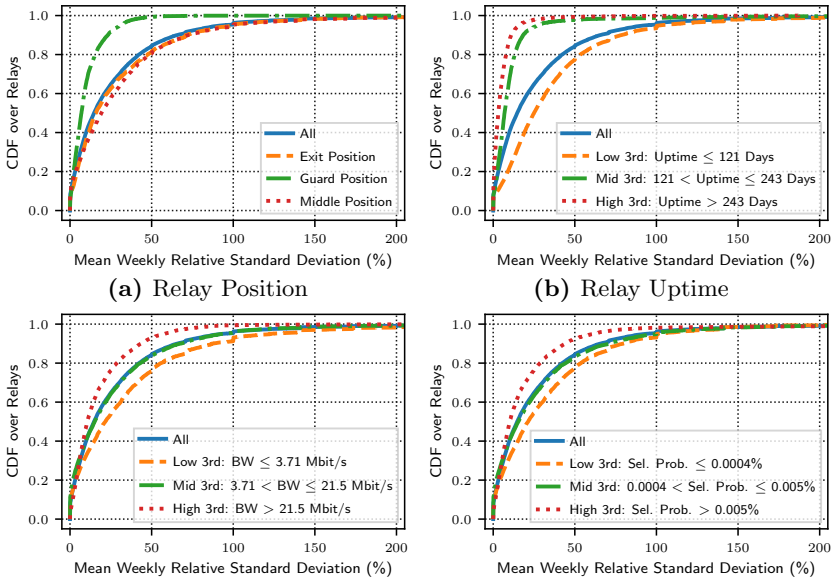
Relay Capacity Variation: A relay with a perfect capacity estimation algorithm would consistently report the same advertised bandwidth; thus, *variation* in advertised bandwidths indicates *inaccurate* capacity estimation. Let $A(r, w)$ be the sequence of advertised bandwidths published by relay r during week w. We quantify the variability in $A(r, w)$ by computing the *relative standard deviation* (RSD) as

$$\text{RSD}(A(r, w)) = \text{stdev}(A(r, w))/\text{mean}(A(r, w)) \tag{1}$$

where stdev() and mean() compute the standard deviation and mean, respectively. Higher RSDs are associated with more fluctuation of the capacity estimate around the expected capacity and indicate error in the estimation.

We summarize the variability in the estimated relay capacity for relay r by computing the mean of $\text{RSD}(A(r, w))$ over all $1 \leq w \leq n$ weeks in which r published at least one *valid* server descriptor. We remove potential sources of bias by considering a server descriptor for r valid unless: (i) it was published before r was measured (i.e., before r appeared in a consensus without the `unmeasured` flag); or (ii) it was published during a week in which a change in r's average or burst bandwidth options caused a reduction in the advertised bandwidth. We call $\text{mean}(\text{RSD}(A(r, 1)), \ldots, \text{RSD}(A(r, n)))$ the *mean weekly RSD* for relay r. We compute mean weekly RSDs for only those relays that were not flagged as `unmeasured` in at least one consensus (to avoid potential bias from bootstrapping new relays). Although we suppose that the true forwarding capacity of each relay does not often change (i.e., relays do not often upgrade to faster network access links), computing the RSD on a weekly basis ensures that any upgrades that do occur during one of the weeks in our analysis period are likely to only affect a small fraction of the n total weeks that we consider (and thus have a small effect on the mean weekly RSD summary statistic).

Analysis Results: We compute mean weekly RSDs for relays over $n = 52$ weeks, where $w = 1$ includes the seven days starting on `2018-08-01` and $w = 52$ includes the seven days starting on `2019-07-24`. During this analysis period, 34,850 unique relays appeared across 8,736 consensus files (many more than are online at any given time due to churn). Of these, 11,296 (32%) were never measured (i.e., never appeared in any consensus without the `unmeasured` flag), and 1,503 (4.3%) did not publish a valid descriptor (as explained above). We consider the remaining 22,051 relays (63%) as valid for the purposes of this analysis, and we compute the mean weekly RSDs for these valid relays.



(a) Relay Position **(b)** Relay Uptime

(c) Relay Advertised Bandwidth **(d)** Relay Selection Probability

Fig. 1: The distribution of the mean weekly RSD over all valid relays. For each valid relay r, we compute the RSD for each week that r published a valid server descriptor, and then compute the mean over all such weeks to get the mean weekly RSD for r.

Figure 1 compares the distribution of the mean weekly RSD over all such valid relays and over distinct subsets that are separated by common relay characteristics (position, uptime, advertised bandwidth, and selection probability). Over all relays (the solid line in each subfigure), we find that the reported advertised bandwidths exhibit significant variation. The mean over all relays of the mean weekly RSD is 27%, while 25% and 10% of the relays have a mean weekly RSD of 35% and 66% or more, respectively (and a non-trivial fraction of relays have RSDs of 100% or greater). Such variation is larger than expected when the true capacity does not change. We also find that the largest RSDs are associated with lower capacity relays and relays that are online less frequently, as we explain next.

Position: A relay's position is that in which it serves most frequently throughout the year. We compare mean weekly RSDs across relays of different positions in Figure 1a. We observe that guard relays exhibited significantly lower variation in their advertised bandwidths than did exits and middles: compared to exits, guards' RSDs dropped from 16% to 7.0% at P50 and from 71% to 23% at P90. Tor requires that relays must be stable with high uptime to receive the Guard flag, which may help explain this result.

Uptime: A relay's uptime is the percentage of hours during which it was online over the entire year. We compare mean weekly RSDs across relays with different ranges of uptime in Figure 1b. We observe that relays with lower uptime were correlated with larger mean weekly RSDs: 25% of the lowest uptime (≤ 121 days)

and highest uptime (>243 days) relays had mean weekly RSDs of 46% or more and 6.7% or more, respectively. This result suggests that relays that are less consistently available are underutilized by Tor clients and are thus unable to observe enough traffic to reach their true capacity.

Advertised Bandwidth: We compare mean weekly RSDs across relays with different ranges of mean advertised bandwidths (here, the mean is computed over the entire year) in Figure 1c. We find that relays with lower mean advertised bandwidths were associated with higher variation, with the one-third of the relays advertising less than 3.71 Mbit/s accounting for the highest variance. The same *absolute* change in throughput (such as that caused by a single client) could result in a larger *relative* change in advertised bandwidth (and thus the RSD) for slower relays than for faster relays, which could help explain this result.

Selection Probability: We compare weekly RSDs across relays with different ranges of selection probabilities (the mean normalized weight from all consensuses in which it appeared throughout the year) in Figure 1d. Relays with the lowest one-third of selection probability were correlated with higher mean weekly RSDs, while relays with the highest one-third of selection probability were correlated with lower variation. Since selection probability is directly associated with the amount of traffic a relay will observe, it follows that relays that are chosen most consistently report advertised bandwidths with the least variation.

Overall, we find significant variation in relays' advertised bandwidth, and that lower capacity and lower uptime relays are correlated with higher variation. However, we are unable to deduce the *true* causes of the observed associations because *correlation does not imply causation*. Next we conduct an active measurement experiment to help us further understand error in capacity estimates.

4 Tor Relay Speed Test Experiment

Our analysis of variation in advertised bandwidths suggests that there is significant error in Tor's system for determining relay capacities. However, without more information, it seems difficult to tell why and to what extent these errors are made. Based on our understanding of the TorFlow system, though, we can hypothesize that the predominant error is to *underestimate* the true capacity of Tor relays. This hypothesis seems plausible because the observed bandwidth is a self-measurement that mostly is limited by how much client traffic is sent through a relay, and it has been observed that there is a slow feedback process in which some client traffic is attracted, the observed bandwidth increases and causes the relay weight to increase, and then more client traffic is attracted [8].

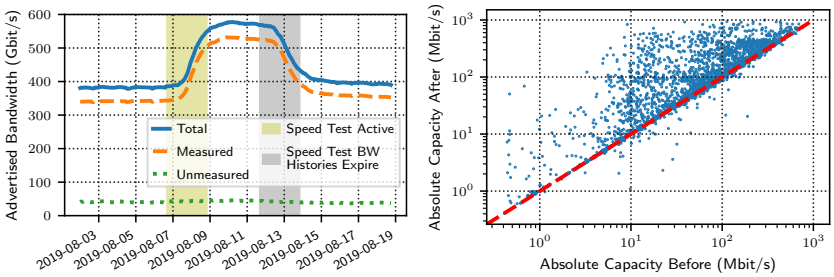
To test this hypothesis, we perform a speed test on the live Tor network by actively attempting to send 1 Gbit/s of Tor traffic through each relay. If a relay is not already receiving sufficient client traffic to reach its true capacity (at least for 10 seconds every 5 days), the extra traffic we add should increase its observed bandwidth, as reported in its server descriptors. The resulting observed bandwidths should increase our overall estimate of Tor's capacity and give us a more accurate estimate of how much total client traffic it could forward.

Moreover, as suggested by our capacity variation analysis, we may be able to identify differences in the amount of underestimation depending on the relays' positional flags (e.g. Guard and Exit), advertised bandwidth, and uptime. For example, our previous results may lead us to hypothesize that relays with lower uptime will have have a larger increase in observed bandwidth due to the speed test (i.e., their current observed bandwidths are larger underestimates of their forwarding capacity). Such non-uniform errors would imply that Tor's load balancing is suboptimal, where relays with higher degrees of capacity underestimation receive too little traffic and relays with lower degrees receive too much.
Setup: We added 487 lines of code to Tor v0.3.5.7 in support of our speed test experiment. Our changes include the addition of a new `SPEEDTEST` cell; when a `SPEEDTEST` cell that was sent by a client running our version of Tor is received by a relay running our version of Tor, the relay will simply return the cell back to the client over the same circuit. When creating a circuit that starts and ends with a client and relay running our version of Tor, the `SPEEDTEST` cell allows us to send a burst of traffic in both directions through the circuit. We also added Tor client controller commands to enable us to instruct a client (through the control port) to build speed test measurement circuits through a path of relays, to start and stop sending `SPEEDTEST` cells through a measurement circuit, and to extract information about each measurement result.

We conduct our speed test experiment from a single dedicated machine with 32 GiB of RAM, 8 CPU cores, and a 1 Gbit/s symmetric network link. We set up 10 Tor clients ($C_1, \ldots, C_{10}$) and 10 Tor relays ($R_1, \ldots, R_{10}$) on this machine that each run our enhanced version of Tor. We connect our relays to the Tor network so they function as regular Tor relays; we set the `MaxAdvertisedBandwidth` Tor option to the minimum allowed value (<500 Kbits/s) to ensure that our relays do not receive a large weight and are seldom used by Tor clients that we do not control. The speed test experiment proceeds sequentially as follows:

1. We download the latest list of relays from a Tor directory mirror;
2. We randomly choose an *unmeasured* target relay T from the list;
3. For $i \in [1, 10]$, we command C_i to build a circuit $C_i \rightleftarrows T \rightleftarrows R_i$;
4. For $i \in [1, 10]$, we command C_i to send `SPEEDTEST` cells to R_i through the circuit for 20 seconds as fast as Tor (and TCP) will allow;
5. Upon receiving the `SPEEDTEST` cells from T, R_i sends them back to T;
6. T simply forwards the cells in each direction as it would on any other circuit;
7. When the 20 second measurement is complete, we close the measurement circuits, mark T as *measured*, and continue from 1.

By using 10 circuits in parallel (20 sockets in parallel on T), we increase the traffic rate through T while mitigating any potential rate limits imposed by Tor's stream and circuit flow control or by TCP congestion control. Our measurement has the potential to send a burst of traffic at an aggregate rate of 1 Gbit/s through each target T. The measurement effect will be reflected in the following server descriptor that T publishes, in which it will report its observed bandwidth (the highest throughput that it was able to sustain for any 10 second period).



(a) Advertised Bandwidth over Time **(b)** Relay Capacity Estimates

Fig. 2: The effects of the speed test on Tor relays (≈200 Gbit/s of capacity discovered).

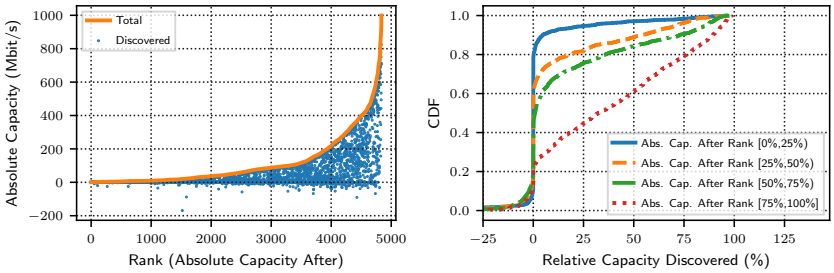
Our experiment is designed to minimize Tor network relay overhead. We add load to only one remote target relay at a time and only for a short period. We submitted our experimental design and plans to the Tor Research Safety Board [1] for feedback. We received encouraging feedback and a "no objections" decision. We also explained our plans to the Tor community through a post to the public *tor-relays* mailing list [16]. We gave instructions on how to opt out and allowed one week to collect feedback. Finally, we served a web page containing a link to the mailing list post on the IP addresses used in the experiment.

Results: Our speed test experiment ran for just over 2 days (51 hours) starting on `2019-08-06`. We plot in Figure 2a the sum of the most-recently published advertised bandwidths of all online relays over time. The first green region shows the period during which the speed test was active, and the second gray region shows the period during which the effects of the speed test expired. Note that the delay in the increase and decrease in advertised bandwidth relative to our experiment is caused by: (i) the 18 hour server descriptor publishing interval; and (ii) the observed bandwidth algorithm which stores history for each of the last 5 days. We successfully *measured* 4,867 relays, while 2,132 relays were *unmeasured* due to circuit building timeouts. On average, the measured relays represent 341/382 Gbit/s (89%) and 525/570 Gbit/s (92%) of the total advertised bandwidth before and after the speed test took effect, respectively, whereas the unmeasured relays represent 41/382 Gbit/s (11%) and 45/570 Gbit/s (8%).

In the remainder of our analysis, we consider only those 4,867 relays that we successfully measured. We take the relay capacity before the test to be the maximum advertised bandwidth over the period from `2019-08-01` until the speed test starts on `2019-08-06`, and we take the relay capacity afterwards to be the maximum advertised bandwidth from the speed test start until `2019-08-12`.

Relay Results: Figure 2b shows the per-relay capacities before and after the speed test: we observe that many relays increased their capacity estimates, some by a 10× or greater factor. We do see some relays with slightly reduced capacity estimates, which could be due to reasons such as reduced bandwidth rates (i.e., average bandwidths) or increased background traffic from other applications.

Network Results: We find that the estimated network capacity (the sum of relay capacities) increases by about 50% after our speed tests push relays into reporting higher observed bandwidths. Specifically, the network increases from



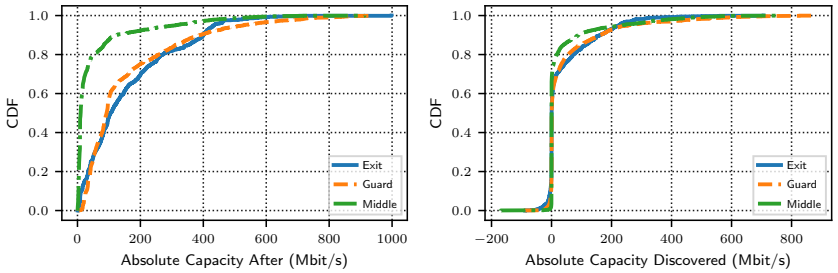
(a) Discovered and Total Capacity **(b)** Relative Capacity Discovered

Fig. 3: Rank is by the capacity after the speed test. Discovered capacity is after−before, whereas relative discovered is (after − before)/after. Summary of relay capacities after the speed test (in Mbit/s): min=0.262, Q1=12.4, med=53.7, Q3=135, max=998.

360 Gbit/s before the experiment to 550 Gbit/s afterwards, which gives a 52.9% increase in estimated total capacity. The capacity increase among exit relays (i.e., with the Exit flag) is 31.8 Gbit/s (33.9%), the increase among guard relays (i.e., with the Guard flag but not the Exit flag) is 96.5 Gbit/s (42.4%), and the increase among the middle relays (i.e., those remaining) is 62.1 Gbit/s (162%). Because exit bandwidth limits Tor's overall throughput, we therefore could expect that Tor could handle 31.8 Gbit/s (33.9%) more traffic than previously expected. We emphasize that these results may still underestimate the true capacity of the network: our test setup was limited by a 1 Gbps network link and we were unable to measure many relays, so our results should be taken as a lower bound on both Tor's true capacity and on the degree of error in its current capacity estimates.

Effects of Capacity: There are at least a couple of reasons to expect that the capacity of a relay may affect the amount by which its capacity is currently underestimated. First, the variance of client traffic is likely lower on higher-capacity relays, as the number of clients they attract is larger, and so by the law of large numbers we expect the variance in the sum of client traffic to decrease. Because observed bandwidths take the maximum bandwidth over several days, small relays are more likely by chance to attract a large amount of traffic relative to their size. Second, large relays have fewer peers that they can be paired with during TorFlow measurements without the other relay acting as a bottleneck during the measurement. We therefore investigate how the capacity of a relay affects the amount of capacity "discovered" during the speed test, that is, the change in the advertised bandwidth after the speed test.

Figure 3a shows the capacity discovered per relay ranked by the capacity after the speed test. The capacity after the speed test should be closer to the true capacity. We notice that at all capacity ranks, the discovered capacity ranges from none to all of the post-speed-test capacity. To better understand the quantitative relationship between relay capacity and discovered capacity, Figure 3b plots CDFs for relative discovered capacity after ranking relays by capacity afterwards and dividing that list into quartiles. Note that the discovered capacity is calculated relative to the capacity after the speed test, and thus is almost always a value between 0% and 100%.



(a) Capacity After Speed Test **(b)** Capacity Discovered by Speed Test

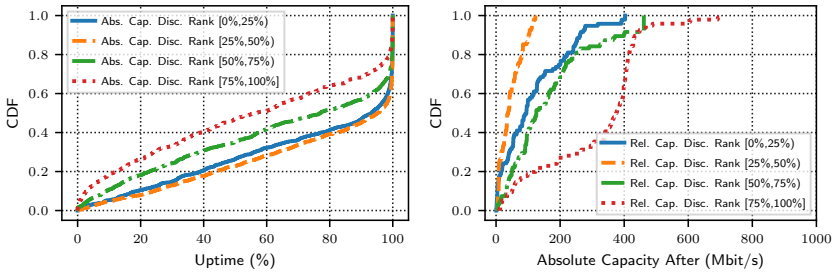
Fig. 4: Capacity after and discovered by speed test by relay position.

We observe that higher-capacity relays have higher discovered capacity, *even relative to their capacity*. The median increase is 0.0% for the quartile with the lowest-capacity relays, 0.0% for the second quartile, 1.0% for the third quartile, and 32.5% for the highest quartile. This result shows that the largest Tor relays have the most inaccurate capacity estimates, on both an absolute and relative basis. It also suggests that the Tor weights may be too low for such relays, reducing load-balancing and thus Tor performance overall. We do notice that for all but the smallest relays, there is a high degree of capacity underestimation: at P90 the relative discovered capacity is 4.6% for the first quartile, 54.9% for the second, 72.7% for the third, and 89.4% for the fourth.

Effects of Position: We might also expect that relays in different positions have different degrees of capacity underestimation. An exit relay, for example, carries more traffic relative to its capacity than other relays because the exit position has the least total bandwidth, and so we may expect that it has a better estimate of its true capacity. Figure 4a shows the distribution of advertised bandwidths after the speed test. We again (and throughout the paper) consider relays with the Exit flag to be exits, relays with the Guard but not Exit flag to be guards, and the remaining relays to be middles. There were 1,582 exits, 4,138 guards, and 3,954 middles. We see that exit and guard relays have similar distributions, with medians of 105 Mbit/s and 92.7 Mbit/s, respectively. The middle relays have significantly smaller capacities, with a median of 10.1 Mbit/s. Figure 4b shows the amount of discovered capacity by position. While the median values are all at or near zero, we discovered a relatively large amount of bandwidth for a significant fraction of relays in each position, with third quartile (Q3) values of 43.7 Mbit/s for exits, 32.8 Mbit/s for guards, and 6.7 Mbit/s for middles. These results show surprisingly that exit relays generally had the most discovered capacity, despite their relatively high traffic load.

Effects of Uptime: To investigate the capacity estimation errors, we next consider how a relay's uptime affects its discovered capacity. We expect that increased uptime will lead to lower discovered capacity because of the slow feedback between increasing the observed bandwidth, which attracts additional client traffic, which then further increases the observed bandwidth [8].

We compute uptime as the fraction of consensuses (i.e., hours) in which the relay was present during the year preceding our speed test (`2018-08-01` to



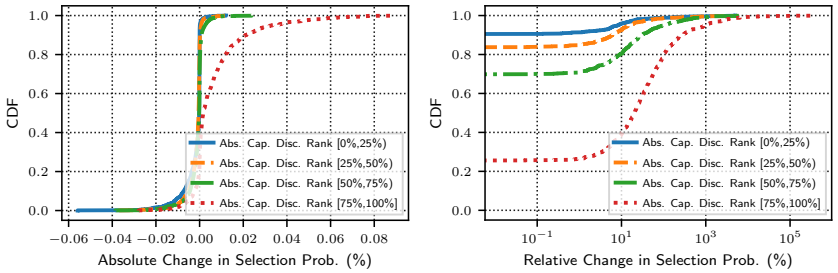
(a) Relay Uptime **(b)** Capacity of Highest Uptime Exits

Fig. 5: The effects of uptime on discovered capacity. (a) Relay uptime, where relays are ranked by absolute discovered capacity (after − before). Absolute discovered capacities summary (in Mbit/s): min=−169, Q1=0.00, med=0.01, Q3=20.8, max=881. (b) Relay capacity after the speed test of exits with ≥ 75% uptime (762 such exits). Relays are ranked by relative discovered capacity ((after − before)/after). Relative discovered capacities summary: min=-96.6%, Q1=0.00%, med=0.00%, Q3=11.1%, max=91.0%.

`2019-07-31`). Figure 5a shows that increased uptime is correlated with decreased discovered capacity. The median annual uptime of the top quartile of relays (i.e., those with the largest discovered capacities) is 56.9%, while the median uptime of the bottom quartile is 93.4%. We note that the bottom two quartiles each have nearly zero discovered capacity, explaining their similar uptime distributions. If we consider the uptimes by position, we observe the same general pattern: guards generally have higher uptime and middles generally have lower. These results support the observed phenomenom that relays' observed capacities increase over time towards the true amounts [8].

We have shown that position, capacity, and uptime separately lead to different amounts of error in the advertised bandwidth. To somewhat disentangle these effects, we consider now the discovered capacity for the exit position (other positions are similar and appear in the Appendix), and we only consider relays with an uptime of at least 75% during the year preceding our experiment. By considering only the relays that were online for many months, we expect to largely remove the slow-increase phase of Tor's measurement system. Moreover, by considering just the exit position, we focus on the position for which the capacity is most limited and the effects of poor load balancing thus most impactful.

We show the results in Figure 5b. Among the high-uptime exits (for which we might have expected little undiscovered bandwidth), there are large relative discovered capacities among the largest quarter by capacity after the speed test, ranging from 11.1% in the third quartile to 91% at maximum. Moreover, larger amounts of capacity are clearly still discovered among the largest exits, with a median of 377 Mbit/s capacity among the relays in the highest quartile of relative discovered capacity, compared to 127 Mbit/s, 41.9 Mbit/s, and 86.4 Mbit/s in the median for the third to first quartiles, respectively. This is despite the fact that relays are ranked by *relative* discovered capacity, which means that not only do the largest exits have the largest total error in capacity measurement, they have the largest *fraction* of capacity error. This is consistent with a hypothesis that



(a) Absolute Change in Select. Prob. **(b)** Relative Change in Select. Prob.

Fig. 6: Change in selection probabilities before and after the speed test. Discovered capacities summary (in Mbit/s): min=−169, Q1=0.00, med=0.01, Q3=20.8, max=881.

the largest Tor relays are unable to attract enough traffic to recognize their true capacity. It shows a consistent *bias* in the Tor bandwidth measurement system against large relays, which consequently is likely to cause the Tor weights to be too low for such relays, reducing Tor performance overall. Note that these results are shown by absolute discovered capacity in the Appendix.

Effects on load balancing: To understand how Tor load balancing is affected by its biased capacity estimation, we analyze the relay selection probabilities before and after the speed test. Our speed test is designed to investigate the advertised bandwidths, and the resulting effects on the weights are complicated both by any changes in the relay population and by the somewhat complex effects of the TorFlow load balancing system. However, Greubel et al. [15] find high correlation between the advertised bandwidths and the Tor weights that determine the selection probabilities, and Tor's load-balancing goal is indeed to choose each relay proportional to its capacity. Therefore, we expect biases in the advertised bandwidths to result in suboptimal selection probabilities.

Figure 6 shows the change in selection probabilities caused by the speed test. Relays are divided into quartiles by the total amount of discovered capacity. We can clearly see that, as expected, the relays with the largest discovered capacity experienced the largest increases in their selection probabilities. For relays ranked in the top quartile, the change in the median selection probability is 0.002% (a 24.3% relative increase), while at P90 we observe an even more extreme weight change of 0.024% (a 325% relative increase).

5 Discussion

Throughout the paper, we have highlighted the performance implications of Tor's capacity estimation errors. We note further that the bandwidth estimation errors we have observed have security implications. A primary security mechanism Tor uses is to make it expensive to run a large fraction of the network by requiring a large amount of bandwidth to observe a large fraction of client traffic. It accomplishes this by making the selection weights highly correlated with (i.e., roughly proportional to) the advertised bandwidths [15]. The errors we have discovered allow an adversary to more cheaply attract and attack client connections (e.g., traffic correlation [22] or website fingerprinting [28]). Our results imply that an

adversary can gain an advantage by maintaining many high-uptime relays each with low capacity. Moreover, we show that the sensitive exit and guard positions are vulnerable to this exploitation.

Thus, an adversary could run a large number of low-bandwidth relays for many weeks as both exits and (eventual) guards. Simply due to the bias of Tor's measurement system, those relays would obtain higher total weight than than the relative cost of running them. Running additional relays simply requires additional IP addresses, due to Tor's limit of two relays per IP address. Therefore, assuming bandwidth is the dominant cost, the adversary would spend less to observe and attack a given amount of client traffic than if the network bandwidth were accurately measured. The adversary could use its relays to deanonymize clients via known attacks.

We further observe that our speed test could be executed by a malicious party to direct more client traffic to *any* subset of the Tor network, by raising the advertised bandwidths of relays in that subset and thus their weights. Easier attacks to inflate malicious relay bandwidth are already known [5,20,21,32]. However, in this attack the adversary need not control the relays to which it directs traffic. For example, a malicious network adversary (e.g., an ISP or nation-state) is able to direct more client traffic to relays on networks it can observe, without running any of those relays. Such an ability again would enable deanonymization attacks on the connections thus directed. This ability also enables denial-of-service by allowing the adversary to artificially increase the weights of a subset of the network, overloading those relays and degrading network performance.

6 Conclusion

Estimates of Tor relays' forwarding capacity are used to balance client traffic load across relays and therefore accurate estimates are vital to the performance and security of the Tor network. We analyzed the accuracy of Tor relay capacity estimation using passive measurements of relay bandwidth that are published by Tor metrics [3]. We found significant variation in relays' advertised bandwidths which indicates inaccurate estimation; higher variation was associated with lower capacity relays and relays that were online less frequently. We further explore the accuracy of Tor capacity estimation techniques through an active speed test experiment on the live Tor network. Through this experiment, we find that Tor underestimates its total capacity by about 50%, and that most relays increased their capacity estimate following our experiment (some by a 10× or greater factor). We also found that higher capacity relays and exit relays discovered more capacity than lower capacity and non-exit relays, respectively, and that relays with lower uptimes were correlated with higher discovered capacity. Our results suggest that improvements to capacity estimates could significantly improve load balancing, which could lead to better network performance and security.

Acknowledgments

This work has been partially supported by the Office of Naval Research (ONR) and the Defense Advanced Research Projects Agency (DARPA) and the National Science Foundation (NSF) under award number CNS-1925497.

References

1. Research Safety Board. https://research.torproject.org/safetyboard (August 2019)
2. Tor directory protocol, version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt (September 2019)
3. Tor Metrics Portal. https://metrics.torproject.org (August 2019)
4. Andre, G., Alexandra, D., Samuel, K.: SmarTor: Smarter Tor with Smart Contracts: Improving resilience of topology distribution in the Tor network. In: Annual Computer Security Applications Conference (ACSAC) (2018)
5. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource Routing Attacks Against Tor. In: Workshop on Privacy in the Electronic Society (WPES) (2007)
6. Cangialosi, F., Levin, D., Spring, N.: Ting: Measuring and Exploiting Latencies Between All Tor Nodes. In: Conference on Internet Measurement (IMC) (2015)
7. Darir, H., Sibai, H., Borisov, N., Dullerud, G., Mitra, S.: TightRope: Towards Optimal Load-balancing of Paths in Anonymous Networks. In: Workshop on Privacy in the Electronic Society (WPES) (2018)
8. Dingledine, R.: The lifecycle of a new relay. Tor Blog Post (September 2013), https://blog.torproject.org/lifecycle-new-relay
9. Dingledine, R., Hopper, N., Kadianakis, G., Mathewson, N.: One Fast Guard for Life (or 9 months). In: Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs) (2014)
10. Dingledine, R., Mathewson, N.: Tor Protocol Specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt (November 2018)
11. Dingledine, R., Mathewson, N.: Tor Path Specification. https://gitweb.torproject.org/torspec.git/tree/path-spec.txt (September 2019)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: USENIX Security Symposium (2004)
13. Dinh, T.N., Rochet, F., Pereira, O., Wallach, D.S.: Scaling Up Anonymous Communication with Efficient Nanopayment Channels. Proceedings on Privacy Enhancing Technologies (PoPETs) **2020**(3) (2020)
14. Ghosh, M., Richardson, M., Ford, B., Jansen, R.: A TorPath to TorCoin: Proof-of-Bandwidth Altcoins for Compensating Relays. In: Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs) (2014)
15. Greubel, A., Pohl, S., Kounev, S.: Quantifying measurement quality and load distribution in Tor. In: Annual Computer Security Applications Conference (ACSAC) (2020)
16. Jansen, R.: Measuring the Accuracy of Tor Relays' Advertised Bandwidths. https://lists.torproject.org/pipermail/tor-relays/2019-July/017535.html (July 2019)
17. Jansen, R., Hopper, N., Kim, Y.: Recruiting New Tor Relays with BRAIDS. In: Conference on Computer and Communications Security (CCS) (2010)
18. Jansen, R., Johnson, A., Syverson, P.: LIRA: Lightweight Incentivized Routing for Anonymity. In: Network and Distributed System Security Symposium (NDSS) (2013)
19. Jansen, R., Miller, A., Syverson, P., Ford, B.: From Onions to Shallots: Rewarding Tor Relays with TEARS. In: Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs) (2014)
20. Jansen, R., Vaidya, T., Sherr, M.: Point Break: A Study of Bandwidth Denial-of-Service Attacks against Tor. In: USENIX Security Symposium (2019)

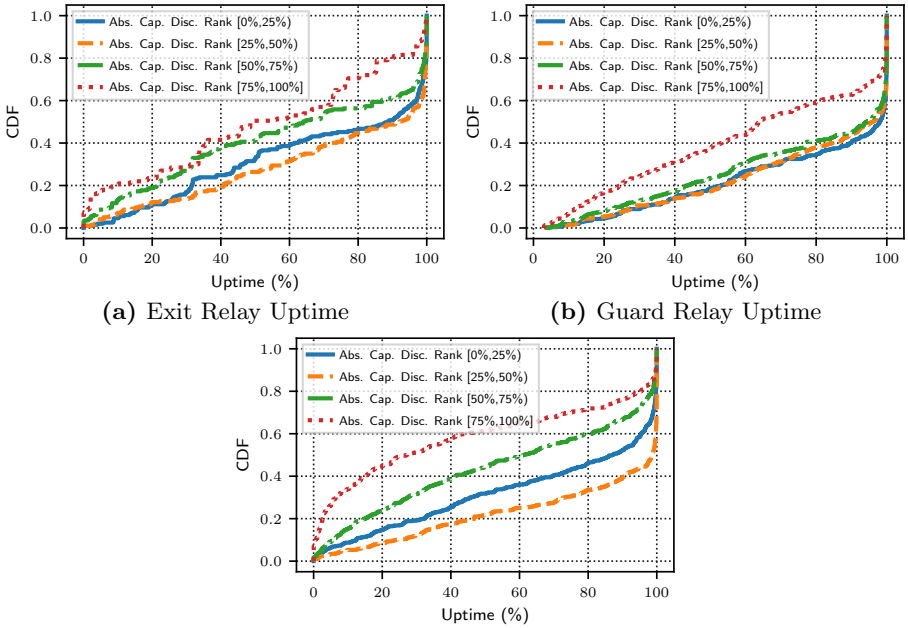
21. Johnson, A., Jansen, R., Hopper, N., Segal, A., Syverson, P.: PeerFlow: Secure Load Balancing in Tor. Proceedings on Privacy Enhancing Technologies (PoPETs) **2017**(2) (2017)
22. Johnson, A., Wacek, C., Jansen, R., Sherr, M., Syverson, P.: Users Get Routed: Traffic Correlation on Tor By Realistic Adversaries. In: Conference on Computer and Communications Security (CCS) (2013)
23. Juga: How Bandwidth Scanners Monitor The Tor Network. Tor Blog Post (April 2019), https://blog.torproject.org/how-bandwidth-scanners-monitor-tor-network
24. Mani, A., Brown, T.W., Jansen, R., Johnson, A., Sherr, M.: Understanding Tor Usage with Privacy-Preserving Measurement. In: Internet Measurement Conference (IMC) (2018)
25. Moore, W.B., Wacek, C., Sherr, M.: Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race with Tortoise. In: Annual Computer Security Applications Conference (ACSAC) (2011)
26. Mozilla: Mozilla Research Grants 2019H1. https://mozilla-research.forms.fm/mozilla-research-grants-2019h1/forms/6510 (2019), call for Proposals
27. Ngan, T.W.J., Dingledine, R., Wallach, D.S.: Building Incentives into Tor. In: Financial Cryptography and Data Security (FC) (2010)
28. Panchenko, A., Lanze, F., Zinnen, A., Henze, M., Pennekamp, J., Wehrle, K., Engel, T.: Website Fingerprinting at Internet Scale. In: Network and Distributed System Security Symposium (NDSS) (2016)
29. Perry, M.: TorFlow: Tor Network Analysis. In: Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs) (2009)
30. Snader, R., Borisov, N.: EigenSpeed: Secure Peer-to-Peer Bandwidth Evaluation. In: International Workshop on Peer-to-Peer Systems (IPTPS) (2009)
31. Mozilla Research Call: Tune up Tor for Integration and Scale. https://blog.torproject.org/mozilla-research-call-tune-tor-integration-and-scale (May 2019), blog Post
32. Thill, F.: Hidden Service Tracking Detection and Bandwidth Cheating in Tor Anonymity Network. Master's thesis, University of Luxembourg (2014)
33. Wright, M., Adler, M., Levine, B.N., Shields, C.: The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. ACM Transactions on Information and System Security (TISSEC) **4**(7), 489–522 (November 2004)

Appendix

Figure 7 shows for each relay position the uptime by absolute discovered capacity quartiles. We observe that for all positions relays with higher discovered capacity have higher uptime, although we notice that guard relays have higher overall uptime (due to the additional uptime and stability requirements to get the `Guard` flag), and middles have a larger number of relays with low uptime.

Figure 8 shows the capacity of guard and middle relays after the speed test by quartiles of relative discovered capacity. It includes only relays with at least 75% uptime. It shows that for high-uptime relays in both positions, most of the discovered capacity is among the largest relays. We can especially see that for middle relays, the low amount of discovered capacity is due to the large number of relays with very little total or discovered capacity.

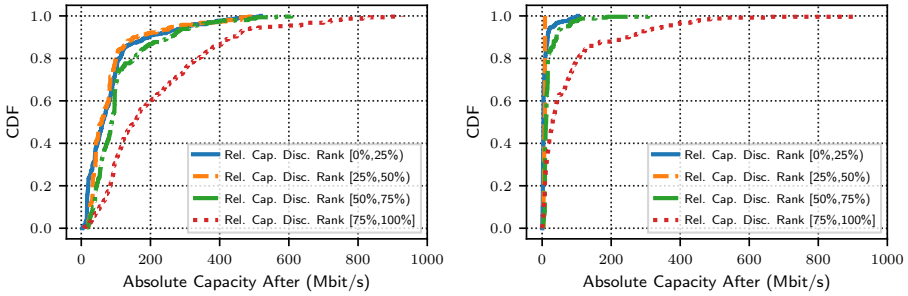
Figure 9 shows the effect of relay capacity on the discovered capacity by position when only relays with at least 75% uptime are considered.



(a) Exit Relay Uptime **(b)** Guard Relay Uptime

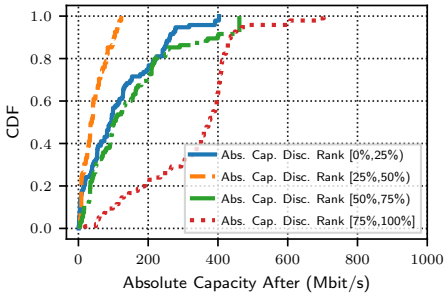
(c) Middle Relay Uptime

Fig. 7: The effect of relay uptime, where relays are ranked by their absolute discovered capacity. (a) Shows exit relays split into 4 sets by rank. Summary of the absolute discovered capacities (in Mbit/s): min=−88.0, Q1 0.00, med=0.192, Q3=42.4, max=707. (b) Shows guard relays split into 4 sets by rank. Summary of the absolute discovered capacities (in Mbit/s): min=−90.3, Q1=0.00, med=0.586, Q3=32.8, max=881. (c) Shows middle relays split into 4 sets by rank. Summary of the absolute discovered capacities (in Mbit/s): min=−169, Q1=0.00, med=0.00, Q3=6.71, max=751.

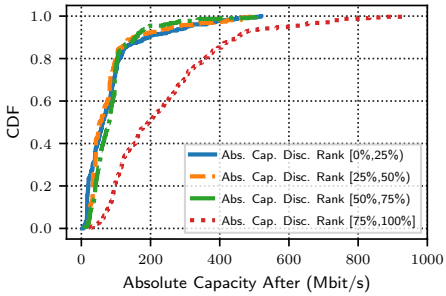


(a) Guard Relay Absolute Capacity **(b)** Middle Relay Absolute Capacity

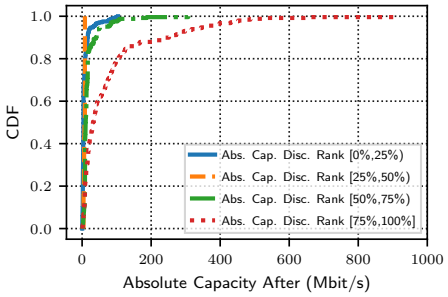
Fig. 8: Absolute capacity of relays after the speed test, where relays are ranked by their relative discovered capacity. Relative discovered capacity is computed as (after − before)/after. Includes only relays with uptime of 75% (273 days) or more during the year preceding the speed test. (a) Shows guard relays split into 4 sets by rank (2,474 guards had at least 75% uptime). (b) Shows middle relays split into 4 sets by rank (1,966 middles had at least 75% uptime).



(a) Exit Relay Absolute Capacity



(b) Guard Relay Absolute Capacity



(c) Middle Relay Absolute Capacity

Fig. 9: Absolute capacity of relays after the speed test, where relays are ranked by their absolute discovered capacity. Absolute discovered capacity is computed as after − before Includes only relays with uptime of 75% (273 days) or more during the year preceding the speed test. (a) Shows exit relays split into 4 sets by rank (762 exits had at least 75% uptime). (b) Shows guard relays split into 4 sets by rank (2,474 guards had at least 75% uptime). (c) Shows middle relays split into 4 sets by rank (1,966 middles had at least 75% uptime).