A Tune-up for Tor: Improving Security and Performance in the Tor Network

Robin Snader
Department of Computer Science
University of Illinois at Urbana–Champaign
rsnader2@cs.uiuc.edu

Nikita Borisov
Department of Electrical and Computer Engineering
University of Illinois at Urbana–Champaign
nikita@uiuc.edu

Abstract

The Tor anonymous communication network uses self-reported bandwidth values to select routers for building tunnels. Since tunnels are allocated in proportion to this bandwidth, this allows a malicious router operator to attract tunnels for compromise. Since the metric used is insensitive to relative load, it does not adequately respond to changing conditions and hence produces unreliable performance, driving many users away. We propose an opportunistic bandwidth measurement algorithm to replace self-reported values and address both of these problems. We also propose a mechanisms to let users tune Tor performance to achieve higher performance or higher anonymity. Our mechanism effectively blends the traffic from users of different preferences, making partitioning attacks difficult.

We implemented the opportunistic measurement and tunable performance extensions and examined their performance both analytically and in the real Tor network. Our results show that users can get dramatic increases in either performance or anonymity with little to no sacrifice in the other metric, or a more modest improvement in both. Our mechanisms are also invulnerable to the previously published low-resource attacks on Tor.

1. Introduction

Anonymous communication on the Internet seems finally within reach. Though an initial commercial deployment of Onion Routing [27], The Freedom Network [2], was in the end shut down, a volunteer-run replacement network using the second-generation onion routing design — Tor [8] — has been operational for several years and has about a thousand nodes and a hundred thousand users [20]. Tor is used by an increasing variety of parties: reporters communicating with sources, dissidents and embassies hiding their activities from local governments [14], people trying to get around geographic restrictions [13], and more. However, for the average user, the performance penalty introduced by Tor is still prohibitively high for everyday use. At the same time, the popularity of Tor has lead to development of a number of attacks on the system.

Contributing to both problems is the Tor load-balancing algorithm. Tor routers self-report their bandwidth capabilities, and clients choose them in proportion to their fraction of the overall Tor capacity. This enables a low-resource attack, where routers misreport their bandwidth to be the artificially high and thereby capture a large fraction of tunnels [3]. Additionally, due to constantly changing conditions, self-reported bandwidth is frequently an overestimate of the actual node capacity, leading to unreliable performance delivered to Tor users.

We propose to replace the Tor mechanism with an *opportunistic bandwidth measurement mechanism*. Due to the complete topology of the Tor network, each router will have a chance to interact with most other routers and thus observe their performance empirically. We show through experiments that this mechanism is a suitable replacement for self-reported bandwidth in that it accurately predicts the performance of the routers and is significantly less susceptible to low-resource attacks. Also, since over-utilized routers will show decreased performance, it also greatly reduces the long tail of the transfer time distribution, making the worst case significantly better.

We also propose a *user-tunable mechanism for selecting routers* based on their bandwidth capabilities. Rather than trying to find a compromise that satisfies both those users who desire strong anonymity protection and those for whom performance is more of a priority, as is done in the current

Tor design, we suggest letting users express a preference in the tradeoff between anonymity and performance and make router selections accordingly. We design a mechanism that effectively blends the traffic of users with different preferences, making partitioning attacks difficult.

Our experiments with Tunable Tor show that users can achieve great improvements in performance without sacrificing much anonymity, or significantly increase anonymity protection without any loss in performance. They also allow for moderate improvements in both, which is what we expect to be chosen by most users. This improved flexibility should make Tor palatable to a wider range of users, and thus increase anonymity for everyone due to a larger community [7].

The remainder of this paper is structured as follows: Section 2 examines the current implementation and points out two important weaknesses. Section 3 analyzes these weaknesses and proposes improvements to Tor to address them; it also evaluates these changes in isolation. Section 4 evaluates their performance in the real Tor network. Section 5 discusses some related work. Finally, Section 6 summarizes our conclusions and examines future directions for this work.

2. Weaknesses in the implementation of Tor

We first present a high-level overview of the Tor network design and then highlight two important problems in the load-balancing algorithm. Interested readers can find more details about the Tor protocol in [8].

2.1. Tor design

The Tor network is based on an onion-routing [27] design, where traffic is forwarded through several routers and multiply encrypted, with each router removing one layer of the encryption. The path through the network — a *tunnel* — is constructed in a telescoping fashion, so that each router knows only the previous and the next router in the path. In particular, the first (*entry*) router knows the source of the tunnel, but not its destination, and the last (*exit*) router knows the destination but not the source. However, if both routers cooperate, they can use traffic analysis to link communication over the same tunnel; hence there is little benefit to using long paths and in practice Tor path length is set to 3.[1]

Tor routers are registered with a directory service. Each router reports its IP address, public key, policies about what traffic it will accept, and a bandwidth value that is determined by monitoring the peak bandwidth achieved by the router over a period of time. The directory service also maintains statistics about the uptime of each router. The Tor path construction algorithm, executed by the Tor client, will first select all routers that have an acceptable forwarding policy (*e.g.* many routers are unwilling to serve as exit routers) and then choose a random router out of the list, with the selection weighted by the reported bandwidth. This way, traffic is roughly balanced across Tor nodes in proportion to the bandwidth they have available. To prevent a router from reporting an unreasonably high bandwidth, an upper bound (currently 1.5 MB/s[2]) is enforced.

To defend against the predecessor attack [30], recent versions have introduced guard nodes, first described by Wright et al. [31]. Each client picks a set of three nodes that will be used as entry routers for *all* of its tunnels. Guard nodes are chosen among stable nodes, *i.e.* those with a high uptime, that have a bandwidth higher than the median bandwidth reported by all Tor nodes.

Fundamentally, Tor forms an overlay network for forwarding traffic, and thus needs to address the performance issues associated with this task. It also has an extra requirement of preserving anonymity, making this task that much more difficult. We next examine two shortcomings of the Tor load-balancing scheme that motivate our work.

2.2. Advertised Bandwidth

The bandwidth values used in the load balancing algorithm are self-reported by each node and are not verified in any way. This leaves the door open to attacks where malicious nodes can report a higher-than-actual bandwidth so that a larger fraction of tunnels are routed through them. Despite the enforced upper bound, the attack can be quite successful: Bauer et al. [3] report that a small fraction of attacker nodes can attain the first and last node positions (thus violating anonymity) on nearly half the tunnels.

Even when nodes are honest, the reported values can be a poor predictor of the available bandwidth at a node due to changing network conditions and other factors. This makes Tor performance highly variable. Our studies of Tor (see Figure 5) shows that, although the Tor network provides reasonable bandwidth on most connections, the performance curve has a long tail. In particular, while the median bandwidth is 29 KB/s, the 90th percentile bandwidth is less than a quarter of that, at 6 KB/s, and there is a significant fraction of tunnels which perform still worse. This presents a poor user experience, especially to users who are browsing the web (the majority of connections in Tor [17]), with connections frequently slowing down.

[1] There are some small benefits to using 3 rather than 2, a full discussion of which is beyond the scope of this paper.

[2] On August 30, 2007, the Tor project released a version of Tor that changes this upper bound to 10 MB/s, increasing network utilization at the cost of increased vulnerability to low-resource routing attacks.

2.3. User Heterogeneity

The Tor load balancing algorithm is a compromise between performance and anonymity. Users who are highly anonymity sensitive (*e.g.* dissidents) might wish to distribute all of the tunnels uniformly across all routers, to prevent (reportedly) high-bandwidth routers from having a higher chance of compromising their traffic. Users who are less privacy-sensitive and are using the network for casual web browsing (*e.g.* users who want to hide their browsing activities from their neighbors) might value performance more and would be more willing to use high-bandwidth routers more often. By aiming to achieve a common default, the Tor router selection algorithm sacrifices the needs of both of these classes.[3]

3. Proposed Improvements

To address these issues, the fundamental questions of an overlay network must be readdressed: first, how is the performance of a router measured; and second, given a list of measured routers, how is the route selected. In this work, our goal is to improve the bandwidth available to a Tor tunnel, rather than other performance characteristics such as latency or jitter. Our reason for focusing on bandwidth is threefold. First, bandwidth is already a key factor in Tor design. Second, bandwidth is typically a property of a node rather than a link between two nodes, since the bottleneck is likely to be close to the node rather than in the intermediate network [1, 16]. This makes measurements and optimizations much more feasible than for link properties, since for N nodes there are $O(N^2)$ links. Additionally, a scheme that optimizes latency is bound to leak at least some information about the starting point of a path, whereas it is possible to optimize bandwidth without such information leaks. Finally, the overwhelming majority of Tor traffic, by both data volume and number of connections, is from web and peer-to-peer traffic [17] — applications that are relatively insensitive to latency and jitter.

3.1. Router Measurement

A simple way to measure the available bandwidth at a router is to perform a probe. Though crude, this mechanism is likely to present the most accurate picture of the performance of a node. Of course, it is unrealistic to expect all nodes to probe all routers, since that will generate an unreasonable amount of extra traffic and create a negative impact on overall Tor performance. A single prober, on the other hand, will serve as an unnecessary point of failure. Additionally, if probes can be identified, malicious routers may choose to devote more of their resources to probes to gain a higher rating.

We propose instead that opportunistic monitoring be used to measure bandwidth capacity; that is, each router in the Tor network keeps track the peak bandwidth it has recently seen for each of its peers. In practice, Tor routers communicate with a large set of routers over a short period of time — we observed up to 800 routers contacted within a single day — and thus can accumulate a large set of statistics. By aggregating statistics across multiple Tor routers, a client can obtain an even more accurate picture of the network, and at the same time be less susceptible to attacks. We propose that the client use the median of the collected bandwidth values, since it is significantly harder to influence the median than other aggregation functions, such as mean, maximum, minimum, etc. The directory servers that a client uses to learn about other nodes are good candidates for providing this information, as they are already used to distribute similar information securely.

This approach has the additional advantage of being dynamic: if a router's available bandwidth fluctuates over time (*e.g.* a router located at the university is likely to have more available bandwidth during the summer and on weekends), this will be noticed by its peers and it will be used more or less frequently accordingly. To some extent, dynamic measurements can also help balance load across different routers, since as a router gets overloaded, its measurements will suffer and it will start getting used less frequently. In the extreme case, such dynamic load balancing can cause route oscillations [11], but the Tor bandwidth measurement algorithms use intervals of ten seconds, providing sufficient damping that we do not anticipate that this will be a problem. A full investigation of the load balancing behavior of dynamic router measurements, as well as our other proposed improvements, requires a complex whole-system simulation or analysis and thus is left to future work.

3.1.1 Evaluation of Predictive Power

In order to determine the predictive power of the various methods of bandwidth assessment, we ran a large number of tests (approximately 22 000) where a 1 MB file was fetched over HTTP via the Tor network. In order to eliminate confounding factors, both the entry node and the exit node were fixed hosts near the file host; the middle hop was selected randomly from all Tor active at the time the file fetch began.

The three methods of bandwidth assessment are:

1. Advertised bandwidth: the bandwidth prediction is simply the advertised bandwidth of the host under test. As Figure 1(a) shows, this method of assessment gives a systematic overestimation of the available

[3] In fact, a recent discussion on the `or-dev` mailing list about raising the upper bound of reported bandwidth hit precisely the stumbling block of not being sure which of these two groups to serve [6]; the determination was eventually made to significantly raise the limit.

bandwidth, since the capacity is necessarily shared among all tunnels using that host. The log-log correlation between predicted and actual bandwidth in this case is approximately 0.57.

2. Opportunistic bandwidth measurement: as described above, a Tor router was modified to record the maximum bandwidth seen recently[4], and this becomes the predicted bandwidth. It is to be expected that this method will become more accurate as data from multiple sources in the network is combined. We see in Figure 1(b) that this method has a much smaller systematic bias than that seen in the previous method. It gives a log-log correlation of 0.48.

3. Active bandwidth probing: finally, we tested the predictive power of active bandwidth probing by using all previous tests from a given host to estimate the bandwidth available. Note that this is significantly different from the second method described above: because the exact file size and how long it took to fetch are known in this case, an accurate assessment of bandwidth can be achieved. In the previous method, only a rough estimate of the bytes-per-second sent during an observation window is available; if only a small amount of data was requested, the available bandwidth will be unavoidably underestimated. The predicted bandwidth then becomes the simple mean of past observations. As expected, Figure 1(c) shows that this method has the best predictive power with a log-log correlation of 0.63.

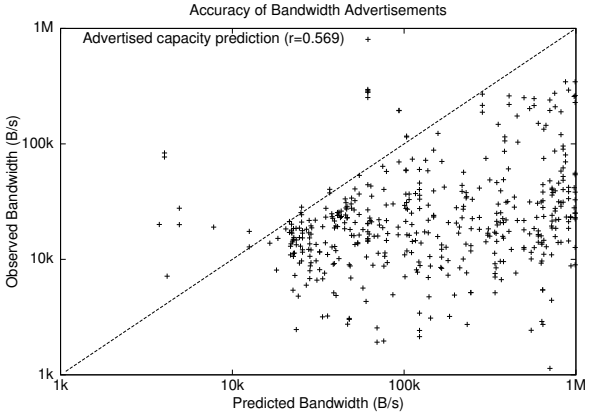
These evaluations show several things: first, as expected, active probing provides the most accurate prediction of available bandwidth, even though it is certainly not practical for other reasons. Second, bandwidth advertisements correlate relatively strongly to available bandwidth, but they exhibit a strong systematic bias toward overestimation. Finally, opportunistic bandwidth measurements provide a strong predictor of actual bandwidth without the unacceptable overhead of active probing or the bias and vulnerability to malicious nodes of using bandwidth advertisements.

In the following section we will examine how this bandwidth measurement can be used to further harden Tor against malicious routers and simultaneously improve the user experience.

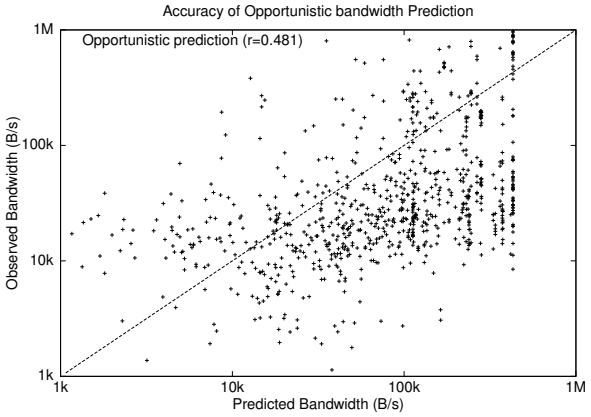
3.2. Variable Router Selection Algorithm

In this section, we propose several modifications to the router selection algorithm used by Tor in order to decrease

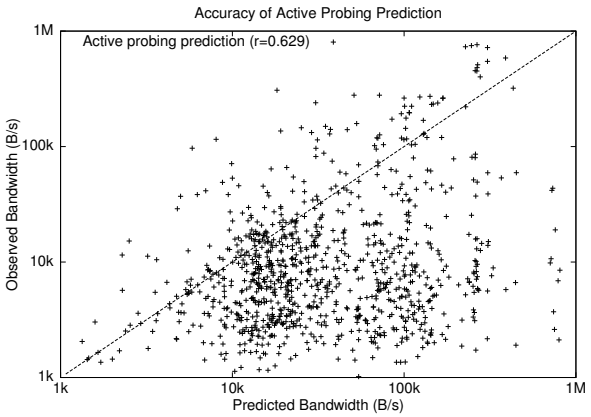
[4]This is measured according to the existing Tor bandwidth tracking functionality; by default this is the smaller of the single-direction maximum bandwidths over (potentially different) 10 second intervals over the past day.



(a) Advertised vs. actual bandwidth ($r = 0.57$)



(b) Opportunistically probed vs. actual bandwidth ($r = 0.48$)



(c) Actively probed vs. actual bandwidth ($r = 0.63$)

Figure 1. The accuracy of various prediction mechanisms for a sample of the trials. The $x = y$ line is included for reference.

its vulnerability to subversion as well as provide a better experience for all classes of users.

As described in Section 2.3, there is a trade-off between selecting routers for optimal performance and providing maximum anonymity protection. Even if the bandwidth measurements are accurate, using high-bandwidth nodes more frequently increases a user's exposure, and some users will wish to pick uniformly from all routers. Others may be willing to expose themselves even more than the current Tor design in order for increased performance. We propose giving users control over this tradeoff by letting them select a point on the anonymity–performance scale either globally (*i.e.* in the Tor configuration file), or depending on the task.

Providing such flexibility not only helps existing Tor users, but attracts new users to the network as well, improving anonymity for all by increasing the anonymity set [7]. However, care must be taken to avoid partitioning attacks. If it is easy to identify what level of privacy a user is aiming for, the anonymity set may be in fact reduced. For example, if only privacy-sensitive users use poorly performing routers, then attackers may wish to focus their efforts on them. Our proposed selection function blends traffic from both privacy-sensitive and privacy-insensitive users by having both sets select from a pool of routers, but weighting their selection differently.

A family of functions $f_s : [0,1] \to [0,1]$, with parameter s, is defined by

$$f_s(x) = \frac{1 - 2^{sx}}{1 - 2^s}.$$

Note that this family of functions is well-defined for all $s \neq 0$. In that case, we define f_s to be its point-wise limit as $s \to 0$; *i.e.* $f_0(x) = x$. Several examples of this family of functions for varying values of s are shown in Figure 2. Note also that $f_s(0) = 0$, $f_s(1) = 1$, and f_s is continuous and monotone increasing for all values of s, so f_s a valid cumulative distribution function for any s.

To choose a router given a selection function f_s, a list of routers and their rankings must first be obtained; while this ranking can be based on any metric, we propose the opportunistically probed available bandwidth metric described in Section 3.1. This list can be of all routers in the Tor network, or only those matching certain criteria (fast, stable, exit, *etc.*). If this list is indexed from 0 to $n-1$, then the router selected is that with the index $\lfloor n \times f_s(x) \rfloor$, where x is selected uniformly at random from $[0,1)$. Note that this procedure is obtaining a value of a random variable from the normalized exponential distribution with parameter $\lambda = -s$. This procedure is then repeated for any other routers to be selected, with the usual caveat that duplicate selections are not allowed.

There are several features to note about this algorithm. First, the chance of a router being selected is based on the ranking of its metric, rather than on the metric itself. This means that an attacker cannot simply add a router with a very large amount of available bandwidth to the network and attract a large fraction of all circuits; instead, many routers must be added, each with enough bandwidth to rank highly. Second, f_s is well defined for all real s. This means that, should a reason arise for preferring routers with *low* bandwidth, a negative s can be used. Also, while there are, in principle, no bounds on the strength of a preference for high bandwidth (*i.e.* how large an s can be chosen), too high a value can result in nearly always choosing the most-highly ranked router. In this paper, values of s from 0 to 15 are examined; a value of $s = 15$ implies that the most highly ranked router in a set of $n = 1000$ (a typical number of routers available in the Tor network at any given time) will be chosen 27% of the time[5]. A practical upper bound for s is 10, which results in the most highly ranked router being chosen only 6% of the time in the above scenario.

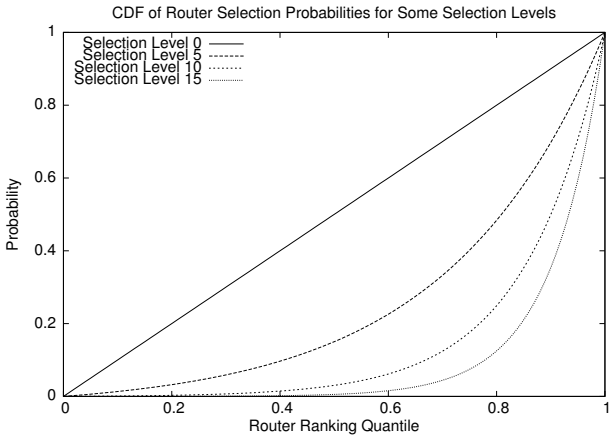


Figure 2. Cumulative Distributions of Routers Selected by Ranking for some Selection Levels

In practice, we observe an additional problem: due to routers frequently joining and leaving the network, a router often lacks any data on the bandwidth of a significant fraction of the total router population. In order to bootstrap data for these routers, we divide the population into those those routers for which we have data (*i.e.* known routers) and those routers for which we do not (*i.e.* new routers) as a first step in choosing a router. A population-weighted coin toss is used to choose between these groups; if the population of new routers is chosen, we choose a router uniformly at random, and if the population of known routers is chosen, we use the algorithm described above. This modified algorithm is the one used for the evaluations in Section 4.

[5]It follows from the definition of f_s that the most highly ranked of a set of n routers will be chosen with a probability of $\log_2(2^{s-1}/n + 1)/s$.

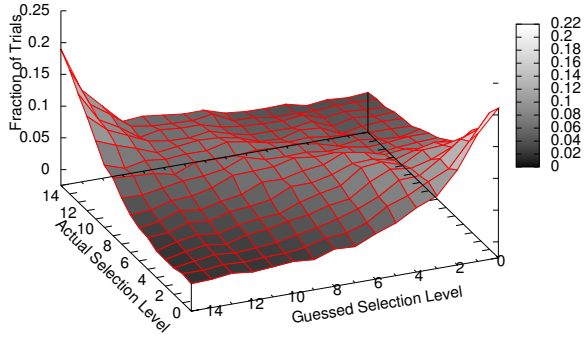


Figure 3. Actual selection level and most likely selection level according to a naïve Bayesian classifier

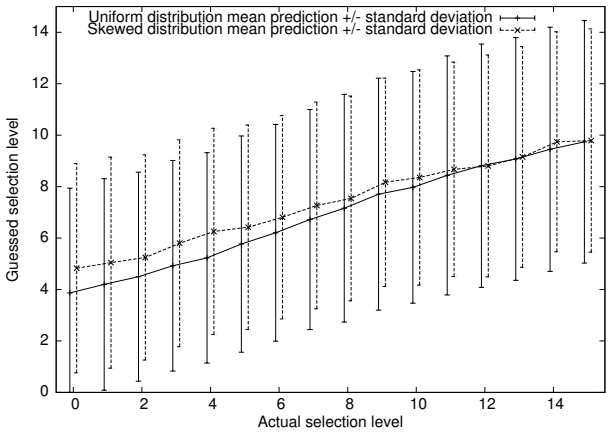


Figure 4. Mean and standard deviation of guessed level by actual selection level according to a naïve Bayesian classifier, for both a uniform and skewed distribution of selection levels.

3.2.1 Evaluation of Router Selection Algorithm

One obvious concern that arises in the context of tuning the router selection algorithm according to the privacy needs of the user is that these needs might be leaked, allowing an attacker to focus on those users who demand the most privacy. In this section, we evaluate the ability of an attacker to fingerprint the selection level chosen by the user.

To perform this evaluation, a large number ($n = 100\,000$) of paths are chosen according to the proposed router selection algorithm, with equally probable selection levels; these paths were then used to train a naïve Bayesian classifier. This classifier then attempted to determine the most likely selection level[6] of a single tunnel for another data set (again, $n = 100\,000$). The results of this classification are shown in Figure 3: the extreme levels (*i.e.* 0 and 15) are most likely to be identified correctly, but even in these cases, the probability of correct identification is no more than 0.21; the intermediate levels are correctly identified much less frequently.

Figure 4 shows the mean guess that was guessed for each selection level over the same data set, along with the standard deviation. For comparison, it also shows the same statistics for a data set where the selection level for both the training set and the test set were chosen from a skewed distribution where level 0 (maximum anonymity) is chosen 20% of the time, level 15 (maximum performance) is chosen 52% of the time, and all other levels are chosen 2% of the time. Over all trials, the average absolute error in the predicted selection level was 3.98 for the uniform distribution and 4.74 for the skewed distribution.

Note also that this represents a best-case scenario for the attacker, as all of the routers were selected from a single, static snapshot of bandwidth data. In actual practice, the data used to choose routers will vary over time, making this sort of classification much more difficult.

[6]This classification is based on the intermediate router and the exit router, since the entry guard is not affected by selection level.

4. Whole-System Evaluation

In order to evaluate the degree to which the proposed changes meet the dual goals of increasing user experience and increasing resistance to subversion, we evaluated them according to two categories of metrics: performance and anonymity. We examine each of these categories below.

4.1. Performance

To evaluate the performance of the proposed modifications to the Tor protocol, a large number of tests were performed over the Tor network; each trial involved downloading a 1 MB file over HTTP using an exit router connected via a high-bandwidth connection to the hosting server; the web server, the exit router, and the client are kept fixed, while the intermediate routers are ranked according to the algorithms described in Section 3.1 and then chosen according to the algorithms described in Section 3.2. The results shown for Tunable Tor are based on approximately 20 000 trials over the period from July 17, 2007 to September 26, 2007; those for vanilla Tor are based on approximately 40 000 trials over the period from January 22, 2007

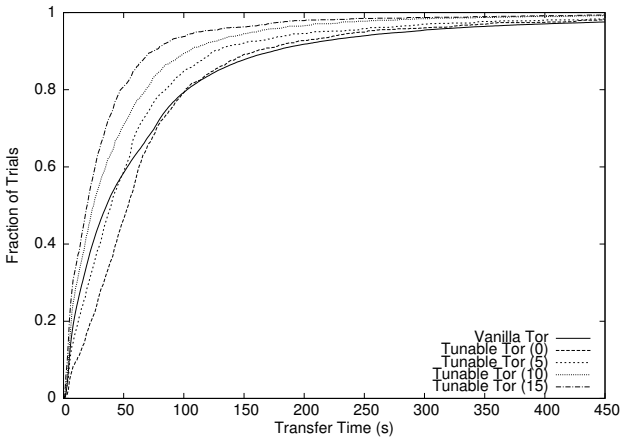


Figure 5. Cumulative distribution of transfer times for a 1 MB file for vanilla Tor and several selection levels. For Tunable Tor, note that Selection Level 0 corresponds to maximum anonymity, while Selection Level 15 corresponds to maximum performance.

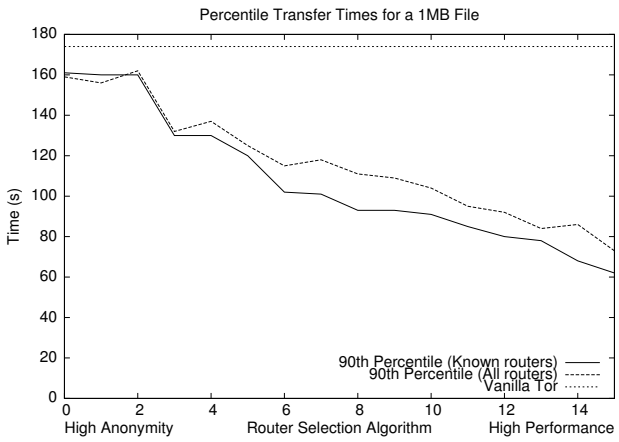


Figure 6. The 90^{th} percentile of transfer times by selection level for known routers and all routers. The 90^{th} percentile for vanilla Tor is included for comparison.

to March 26, 2007. The router selection levels were chosen uniformly at random from the integers between 0 and 15.

Figure 5 shows the CDFs of the file transfer times for vanilla Tor and Tunable Tor at several selection levels. The CDF captures many elements of user experience; since Tor changes tunnels by default every 10 minutes, a user can expect to get the 95th percentile performance several times a day. Note that, as expected, vanilla Tor outperforms Tunable Tor at selection level 0; this is due to Tunable Tor disregarding router performance at that level in the aim of maximizing the equality of router selection. However, at selection level 5, Tunable Tor has a significantly higher fraction (69%) of trials below the one-minute mark than vanilla Tor (62%). At the higher selection levels, Tunable Tor outperforms vanilla Tor across the board; notably, at selection level 15, 85% of trials fall below the one-minute mark.

To further examine the long-tail statistics of the data, Figure 6 presents the 90^{th} percentile of the transfer times for both known routers (those for which we have bandwidth data) and all routers (including those for which we lack bandwidth data) by selection level as well as vanilla Tor. Note that vanilla Tor's lack of relative load information can be seen here in its poor long-tail performance: in times of high load, routers still advertising their full capacity (see Figure 1(a)) become overloaded, resulting in poor performance. Tunable Tor, by comparison, switches away from those routers which tend to become overloaded, resulting in much better performance at high selection levels.

Finally, Figure 7 examines the mean transfer time together with 95% confidence intervals at various selection levels; this corresponds to the user experience for downloading a single, relatively large file. Again, vanilla Tor is included for comparison[7]. Again we see the deleterious effect of Tor's long tail: even though, as Figure 5 shows, the median time is only 35 seconds, the mean time is more than twice that, at 84 seconds. For Tunable Tor, the mean time decreases with increasing selection level, as expected.

4.2. Anonymity

As stated, the other goal of the proposed improvements was resistance to subversion. One measure of this is how many routers an intelligent attacker must subvert in order to have a high probability of compromising a tunnel. Intuitively, it is clear that choosing routers uniformly makes this number increase, while skewing the selection towards certain routers make it lower (because the attacker can choose to compromise the more popular routers). To quantify this intuitive notion, we choose the Gini coefficient. The Gini coefficient is a measure of equality [12] (equality of selection probability, in this case), used frequently in economics. A Gini coefficient of 0 represents perfect selection equality (*i.e* all routers are chosen with equal frequency), while a coefficient of 1 represents perfect inequality (*i.e.* the same router is always chosen). Figure 8 shows the observed Gini coefficient for various selection levels as well as the Gini coefficient of vanilla Tor over a similar sample size. There are several points worth noting: first, the equality is the

[7]Due to the larger data set, the confidence interval is sufficiently small ($\pm$2 seconds) as to be omitted.

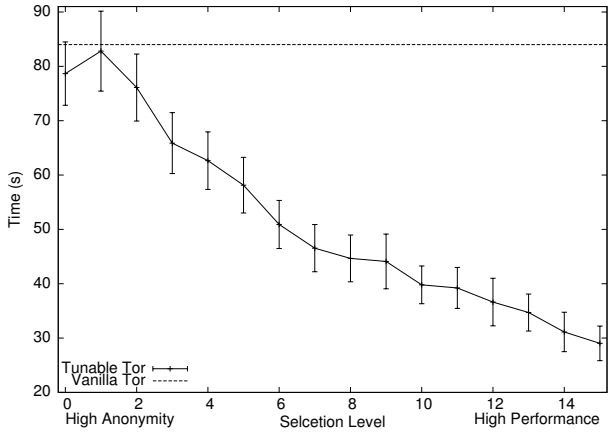


Figure 7. Mean transfer times with 95% confidence intervals for known routers by selection level; the mean time for vanilla Tor is included for comparison.

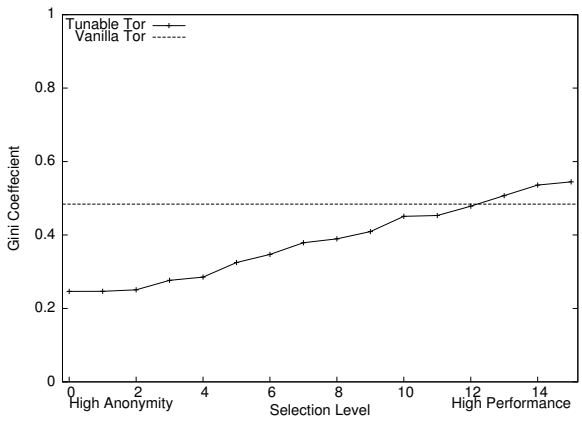


Figure 8. Gini coefficient of router selection equality by selection level. The Gini coefficient for the router selection equality for vanilla Tor is included for comparison.

highest at selection level 0, and lowest at selection level 15, as expected. The reason that selecting uniformly from all routers present at any given time (as selection level 0 does) still gives a coefficient significantly different from 0 is that the router population over time is itself non-uniform; since some routers are present more often than others, they are proportionally more likely to be chosen. Second, using this metric, the bias in Tor's current router selection metric is apparent: all the selection levels below 13 have a more balanced selection than does vanilla Tor. Finally, it is informative to compare Figure 8 with Figure 7; the inherent tradeoff between performance and anonymity becomes quite apparent and the need to allow the user to chose the appropriate point along this continuum for their needs is clear.

To further examine the effects of selection inequality, we consider the success of an attacker who controls a certain fraction of the top performing routers. This can be acquired through either compromising the best routers, or by inserting routers that have high bandwidth (or in the case of vanilla Tor, pretend to). We plot the results in Figure 9. At high selection levels, an attacker who controls a relatively small fraction of the most highly ranked routers can compromise a significant fraction of tunnels[8]. However, at low selection levels, an attacker must control a much higher fraction of Tor routers to compromise even a small fraction of tunnels. At level 0, even when an attacker controls the top 10% of routers, the chance of a compromised tunnel is only 4% (this is higher than $(10\%)^2$ because attackers, with their fast and stable nodes, actually comprise 40% of all guard

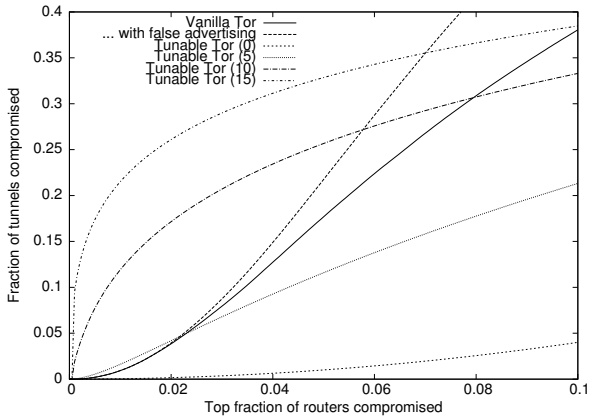


Figure 9. The fraction of tunnels compromised if an attacker compromises the top given fraction of Tor routers, for vanilla Tor and for various selection levels.

[8]We consider a tunnel compromised here if the attacker controls both endpoints.

nodes). It is interesting to note that, even though vanilla Tor does relatively well for a low fraction of compromised nodes, by the time 10% of routers are compromised, it performs comparably to Tunable Tor at its least anonymous setting. Note also that an attacker does not need to compromise the existing best nodes for vanilla Tor, but can compromise arbitrary nodes and have them falsely advertise high bandwidth; this is reflected in the second curve for vanilla Tor, labeled "false advertising," which reflects an attacker changing the advertised bandwidth for each compromised router to the maximum believable bandwidth (1.5 MB/s, by default[9]) after it is compromised. The analogous situation for Tunable Tor, where collaborating nodes report each other as having high bandwidth, is much less effective, since the likelihood that the majority of the reports (*i.e.* 3 of 5) are from compromised routers changes with the cube of the fraction of compromised; this gives a probability of 0.1% for the highest fraction of compromised nodes shown in Figure 9. Our findings are similar to those by Bauer et al. [3].

5. Related Work

Our results regarding the variability of Tor performance match a comparative study of Tor and AN.ON performance [29], which also showed large standard deviations for bandwidth values provided by Tor. Bauer et al. [3] consider distributed probing, perhaps in the style of anonymous auditing [25], as a means of defending from low-resource attacks. They reject it due to the extra load imposed on the system and the ability of malicious nodes to falsely respond to probes. In our case, the distributed measurements are performed opportunistically and thus impose no extra load on the network, and they correspond to real traffic therefore a node seeking to appear as high-bandwidth has to actually provide good performance to real users.

Several projects aim at optimizing latency in anonymous communications. Sherr et al. propose the use o. geographic coordinates to create paths that fall within selected bounds [24]. Renner developed a controller for Tor to select paths according to criteria such as avoiding ocean crossings and otherwise minimizing latencies [21]. Extensions such as the Tor button [26] and FoxTor [23] give users a cruder way to trade off performance and anonymity by selectively enabling or disabling Tor depending on the task. Other work aims to improve Tor performance by speeding up cryptographic operations [19, 15].

One approach to improve overall Tor performance is to use a peer-to-peer design where all users contribute forwarding capacity [22, 10, 18]. Tor designers avoided peer-to-peer approaches due to Sybil attacks [9]; unfortunately, existing peer-to-peer anonymous designs are either insecure [28, 4] or not scalable [10].

[9] As previously described, the Tor project recently increased the maximum believable bandwidth to 10 MB/s. While a cursory analysis of the data gathered before and after this release showed no significant differences, a set of experiments to determine the effect of this update are planned.

6. Conclusions and Future Directions

In this paper, we have proposed improvements to the existing Tor router bandwidth evaluation and router selection algorithms. We examined these changes individually and in combination, showing that they result in a Tor protocol that is both more secure (since it does not use self-reported bandwidth to choose routers for tunnel creation) and performs better, both in terms of observed performance and in terms of achievable anonymity. Additionally, by allowing the user to select their preferred balance of performance and anonymity, these improvements increase the usability, and therefore the potential user base and security of the Tor network.

Evaluations of these changes show that they can result in increasing average throughput by a factor of almost three in exchange for a modest decrease in anonymity, or they can result in drastically improved anonymity while maintaining similar average throughput. We also show that the improvements we propose can reduce or even eliminate the long tail of the transfer time distribution, greatly improving performance as perceived by the users of the network.

We plan to expand on this work in the future in several ways: first, we plan to study the whole-network effects of the changes we propose using a simulation of the Tor network. We would also like to examine the other aspects (such as latency) of the tradeoff between performance and anonymity in anonymous networks of varying types. Additionally, we observed a number of interesting characteristics of the Tor network over the course of this study which could provide insight into the observed behavior of the Tor network, and which we would like to study further.

References

[1] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC03)*, 2003.

[2] A. Back, I. Goldberg, and A. Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[3] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. In *Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES)*, 2007.

[4] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? How attacks on reliability can

compromise anonymity. In *ACM Conference on Computer and Communications Security*, Oct. 2007.

[5] N. Borisov and P. Golle, editors. *Privacy Enhancing Technologies Symposium*, volume 4776 of *Lecture Notes in Computer Science*, Ottawa, Canada, June 2007. Springer.

[6] R. Dingledine. Exit balancing patch. `http://archives.seul.org/or/dev/Jul-2007/msg00022.html`, 2007. Mailing list post to `or-dev`.

[7] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In *Designing Security Systems That People Can Use*. O'Reilly Media, 2005.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium (USENIX Security '04)*, 2004.

[9] J. Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer-To-Peer Systems Workshop*, March 2002.

[10] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.

[11] R. Gao, C. Dovrolis, , and E. W. Zegura. Avoiding oscillations due to intelligent route control systems. In *Proceedings of the 25th IEEE International Conference on Computer Communications. (INFOCOM 2006)*, 2006.

[12] C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121), 1921.

[13] G. Goodell, S. Bradner, and M. Roussopoulos. Building a coreless Internet without ripping out the core. In *Fourth Workshop on Hot Topics in Networks*, College Park, MD, Nov. 2005.

[14] D. Goodin. Tor at heart of embassy passwords leak. *The Register*, Sept. 10, 2007.

[15] A. Kate, G. Zaverucha, and I. Goldberg. Pairing-based onion routing. In Borisov and Golle [5].

[16] K. Lakshminarayanan and V. N. Padmanabhan. Some findings on the network performance of broadband hosts. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC03)*, 2003.

[17] D. McCoy, K. Bauer, D. Grunwald, P. Tabriz, and D. Sicker. Shining light in dark places: A study of anonymous network usage. Technical Report CU-CS-1032-07, University of Colorado at Boulder, 2007.

[18] A. Nambiar and M. Wright. Salsa: a structured approach to large-scale anonymity. In *ACM Conference on Computer and Communications Security*, pages 17–26, New York, NY, USA, 2006. ACM Press.

[19] L. Øverlier and P. Syverson. Improving efficiency and simplicity of Tor circuit establishment and hidden services. In Borisov and Golle [5].

[20] P. Palfrader. Number of running Tor routers. `http://www.noreply.org/tor-running-routers/`.

[21] J. Renner. Implementation and evaluation of path selection algorithms for performance-improved onion routing. `http://code.google.com/soc/2007/eff/appinfo.html?csaid=6AFA998995C47478`, 2007. Google Summer of Code Project.

[22] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[23] S. Romanosky. FoxTor: Anonymous web browsing. `http://cups.cs.cmu.edu/foxtor/`, 2006.

[24] M. Sherr, B. T. Loo, and M. Blaze. Towards application-aware anonymous routing. In *Workshop on Hot Topics in Security*, Aug. 2007.

[25] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *IEEE INFOCOM*, Apr. 2006.

[26] S. Squires and M. Perry. Torbutton — quickly toggle Firefox's use of the Tor network. `https://torbutton.torproject.org/`, 2006.

[27] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[28] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In *Privacy Enhancing Technologies Workshop*, June 2006.

[29] R. Wendolsky, D. Herrmann, and H. Federrath. Performance comparison of low-latency anonymisation services from a user perspective. In Borisov and Golle [5].

[30] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium*. IEEE, February 2002.

[31] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.