Improving Security and Performance in the Tor Network through Tunable Path Selection

Robin Snader and Nikita Borisov, *Member, IEEE*

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. Although Tor limits the self-reported bandwidth, it uses a high maximum value, effectively choosing performance over high anonymity for all users. We propose a router selection algorithm that allows users to control the tradeoff between performance and anonymity. We also propose an opportunistic bandwidth measurement algorithm to replace self-reported values that is more sensitive to load and more responsive to changing network conditions. 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 through simulation 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.

Index Terms—Anonymous communication, bandwidth estimation, path selection

1 Introduction

Anonymous communication on the Internet seems finally within reach. Though an initial commercial deployment of Onion Routing [1], The Freedom Network [2], was in the end shut down, a volunteer-run replacement network using the second-generation onion routing design — Tor [3] — has been operational for several years and has nearly two thousand nodes [4] and several hundred thousand users [5] as of late 2009. Tor is used by an increasing variety of parties: reporters communicating with sources, dissidents and embassies hiding their activities from local governments [6], people trying to get around geographic restrictions [7], 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 practical attacks on the system.

Efforts to improve the performance of the Tor network can often decrease the anonymity, and vice versa. To address this problem, we 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.

At the heart of our work is the Tor load-balancing algorithm. Currently, 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 [8]. 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 graph 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 helps reduce the long tail of the transfer time distribution, making the worst case significantly better.

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. This improved flexibility should make Tor palatable to a wider range of users, and thus increase anonymity for everyone due to a larger community [9].

The remainder of this paper is structured as follows:

- *R. Snader is with Shook, Hardy, and Bacon, L.L.P.*
- *N. Borisov is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61822. Email:* `nikita@illinois.edu`

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 [3].

2.1 Tor Design

The Tor network is based on an onion-routing [1] 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 10 MB/s[2]) is enforced.

To defend against the predecessor attack [10], recent versions have introduced *guard nodes*, first described by Wright et al. [11]. 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., nodes with a high uptime that have a bandwidth higher than the median bandwidth reported by all Tor nodes.

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 changed this upper bound to 10 MB/s from its previous value of 1.5 MB/s, increasing network utilization at the cost of increased vulnerability to low-resource routing attacks.

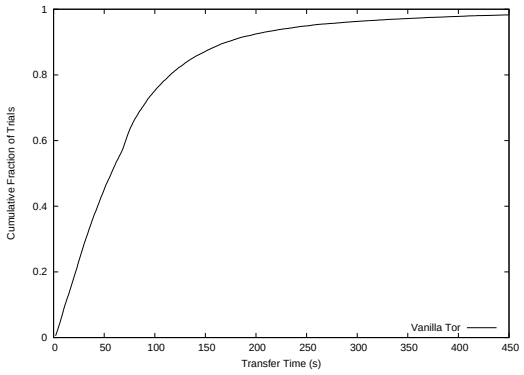


Fig. 1. CDF of the time required to transfer a 1 MB file over the Tor network in July of 2009 (18422 trials).

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. [8] report that a small fraction of attacker nodes can attain the first and last node positions (thus violating anonymity) on nearly half the tunnels, despite using the older (and more secure) cap of 1.5 MB/s.

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. Recent studies of Tor (see Figure 1) show 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 19 KB/s, the 90^{th} percentile bandwidth is less than a third of that, at 6 KB/s, and there is a significant fraction of tunnels that perform worse still. This presents a poor user experience, especially to users who are browsing the web (the majority of connections in Tor [12]), with connections frequently slowing down. Furthermore, comparing these results with those from 2007 in Figure 10 shows that the situation is, if anything, getting worse over time despite the influx of new routers.

2.3 User Heterogeneity

The Tor load balancing algorithm provides a single, static compromise between performance and anonymity. Users who are highly anonymity sensitive (e.g., whistle blowers) might wish to distribute all of the tunnels uniformly across all routers, to prevent (purportedly) 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 using the same path selection algorithm for both of these, the Tor router selection algorithm sacrifices the needs of both 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 performance metric is 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 [14], [15]. 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 [12] — applications that are relatively insensitive to jitter, and where latency can be treated simply as a part of the total transfer time; when low bandwidth makes this transfer time large, latency effects are negligible. Finally, most latency in Tor comes from poor congestion control handling; observed end-to-end latencies significantly exceed the total network latency. Other work [16] attempts to address this problem.

3.1 Router Measurement

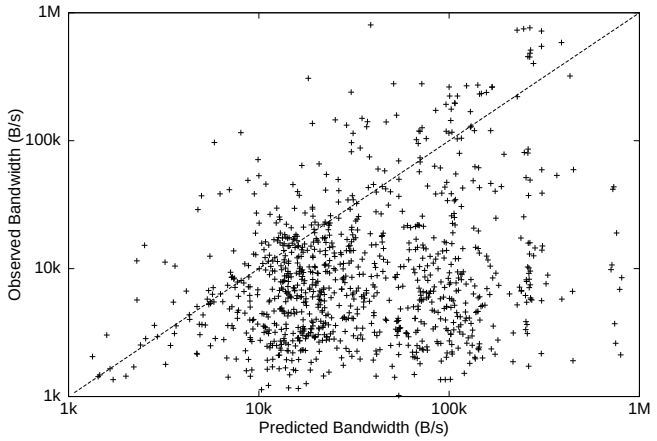
A simple way to measure the available bandwidth at a router is to perform a probe. Though crude, this mechanism is likely to present a reasonably accurate picture of the performance of a node; probing to determine node availability and therefore reliability is done for high-latency anonymous-communication networks by Echolot [17]. Figure 2-(a) shows the correlation between probed router bandwidth and subsequently achieved tunnel bandwidth in the real Tor network when the probed router is the bottleneck router for that tunnel. The probe results are a good predictor of tunnel performance; however, probes themselves use up valuable bandwidth, which is a scarce resource in the Tor network. In particular, probes need to appear identical to real traffic, lest a node give priority to probe traffic to enhance its performance, and thus need to generate significant data transfers. Furthermore, since it is not realistic for all nodes to probe all other nodes, this task must be delegated to a small collection of probing agents, which can act as a point of failure or compromise. For these reasons, we consider bandwidth estimation via active probing to be impractical.

We propose instead that opportunistic monitoring be used to measure bandwidth capacity; that is, each router in the Tor network keeps track the 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. These statistics can then be aggregated by each router to a single observation per peer and then uploaded to the directory server (as the self-reported bandwidth is currently). The directory server can in turn aggregate these N^2 observations into N router evaluations. The process for these aggregations is discussed below in Sections 3.1.1 and 3.1.2. The final evaluations are then distributed by the directory servers as the current self-reported evaluations are. Figure 2-(b) shows the accuracy of bandwidth prediction for a single router in the real Tor network; comparing this Figure 2-(c), we can see that observation by a single client approaches the accuracy of the advertised bandwidth figures employed in the current Tor network; 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.

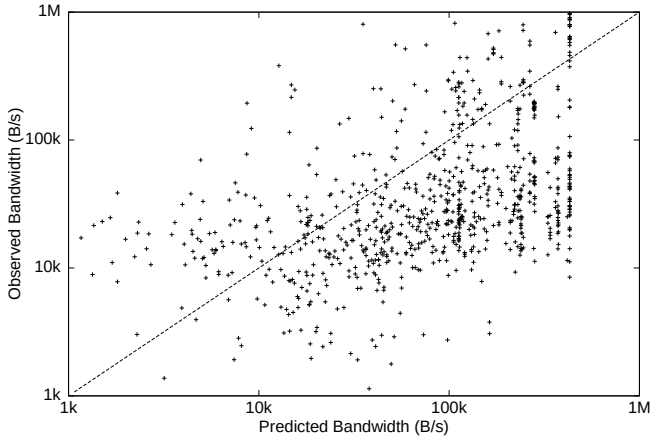
This approach has the additional advantage that, because each peer considers the *individual* bandwidth it receives, an overloaded router will produce lower measurements; whereas the aggregate bandwidth observed by a router will stay the same or even increase in the case of overload. Using frequent updates of opportunistic measurements can thus help balance the load by using overloaded routers less frequently. In the extreme case, such dynamic load balancing can cause route oscillations [18], but an update frequency of about once a minute provides sufficient damping that we do not anticipate that this will be a problem; Figures 3 and 4 support this claim.

We must also consider how to aggregate observations, both multiple observations of a router by a single node, and multiple observations of a router by multiple nodes. The naïve approach is for each node to use the maximum

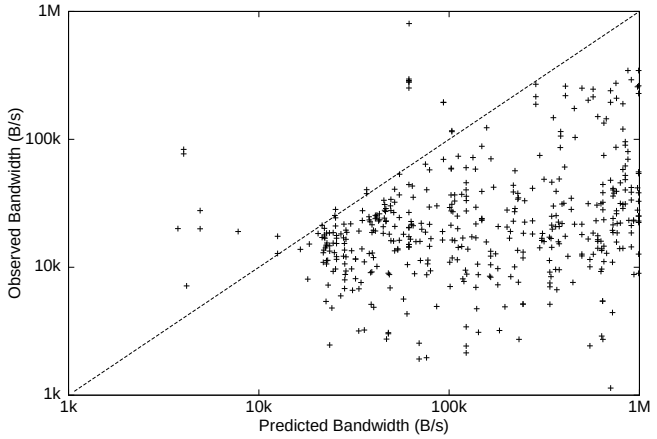
3. In fact, a recent discussion on the `or-dev` mailing list about raising the upper bound of reported bandwidth hit precisely the stumbling block of not being sure which of these two groups to serve [13]; the determination was eventually made to significantly raise the limit.



(a) Accuracy of probing for bandwidth prediction in the real Tor network $(r = 0.629)$.



(b) Accuracy of passive observation for bandwidth prediction in the real Tor network $(r = 0.481)$.



(c) Accuracy of advertised bandwidths for bandwidth prediction in the real Tor network $(r = 0.569)$.

Fig. 2. Accuracy of active probing, passive observation and self reporting for bandwidth estimation in the real Tor network, as measured by the log-log correlation (r). In all cases, the router under test is the middle router, and the same entry and exit routers are used to minimize confounding effects.

of its own observations, but this approach has two flaws. First, each router using its own view of the network creates the possibility of partition attacks (described in [19]), where an attacker focuses all of its bandwidth on nodes of interest. Thus these nodes, and only these nodes, are more likely to select the attacking nodes when creating tunnels. Additionally, aggregating observations via their maximum allows "spotlight attacks" [20], where an attacker focuses all of its bandwidth of one node at a time for a single measurement interval, ignoring all other nodes. Assuming the maximum age of measurements is long enough, the attacker can thus convince the entire network that its bandwidth is many times the actual value.

In the following sections, we discuss observation aggregation and propose two novel methods of evaluating node bandwidth before going on to evaluate their predictive power.

3.1.1 Bandwidth Observation Aggregation

The first question to be addressed is that of aggregating multiple observations of a single router by the same node. It seems clear that we would like our aggregated value to be as close to the actual bandwidth of that peer as possible; taking the maximum observed value over a long interval gives a high probability that there will be a measurement when the measuring node is sole consumer of the router's bandwidth and therefore a fairly accurate value. However, even for much shorter observation lifetimes such as the current Tor value of one day, this approach is vulnerable to spotlight attacks as described above.

Another possibility is using a moving average of recent observations such as an exponentially-weighted moving average (EWMA):

$$B_{\text{new}} = (1 - \alpha)B_{\text{old}} + \alpha B_{\text{obs}}.$$

This way, if an attacker ignores a node for a sufficient period of time, that node's estimation of the attacker will drop and it will be less likely to select that node. However, this approach suffers from lessened accuracy in the face of bandwidth fluctuations: a congested router that offers lower bandwidth to its peers for a period of time will either have its reputation fluctuate rapidly (for small values of α) or drop slowly and recover slowly (for higher values of α). Neither of these scenarios is good for the load balancing of the network.

We propose instead a variant on an EWMA that we call a Min-Max Weighted Moving Average, or MWMA:

$$B_{\text{new}} = (1 - \alpha)\max(B_{\text{old}}, B_{\text{obs}}) + \alpha \min(B_{\text{old}}, B_{\text{obs}})$$

This allows the bandwidth estimation to increase rapidly, but still decay slowly if a router is providing poor service[4], combining the benefits of maximum-based aggregation with those of EWMA-based aggregation.

4. "Quickly" and "slowly" here are, of course, dependent on the parameter α; we find empirically that $\alpha = 0.15$ gives good performance.

3.1.2 Combining Measurements Across Hosts

As discussed previously, having each node rely solely on its own observations to make router selection decisions leads to the possibility of an attacker manipulating that node's view of the network and therefore their tunnel selection. In order to prevent this, nodes can share their observations of their peers. In this section, we present two methods of doing this.

1) For the first, we exploit the fact that, while an attacker might be able to manipulate a node's view of the network, it must know which node to target in order to do so. To prevent this foreknowledge, nodes can ask a random peer for *its* view of the network. Thus an attacker might be able to affect the network view of a node, but it will only affect the router selection of a different, random node. By periodically changing its source for network data, a node can be reasonably certain that an attacker is not intentionally altering its view of the network. However, there remains the possibility that an attacker is blindly presenting bogus data to anybody who asks or to certain targets when they ask. More benignly, there is the possibility that the router queried has recently joined the network and has not yet had time to gather observations for many of its peers. Happily, the solution to both of these problems is the same: instead of querying a single peer, a node queries five[5] and takes the median of the reported values. This substantially reduces the probability of making decisions on poor or malicious data. However, since nodes are making routing decisions based on different data, partition attacks remain theoretically possible. This scheme also has the advantage of being conceptually simple and simple to implement. This is the method proposed in an earlier version of this work [21].
2) A second approach is to use principal component analysis to compute a consensus vector from the $N \times N$ observation matrix using principal component analysis. In our other work, we have designed a secure bandwidth evaluation system that accomplishes this task, called EigenSpeed [22]; we will present a summary of its operation here. In EigenSpeed, a node updates its own observation vector by incorporating the observations of other nodes, weighted by their observed bandwidth. The intuition behind this is that n odes that have higher bandwidth capacity can estimate other nodes' capacities more accurately since they are unlikely to be a bottleneck; furthermore, weighting the observations from nodes that have demonstrated a high bandwidth more highly forces attackers to spend resources to attack the system. This update process is iterated until the process converges (to the principal eigenvector of the observation matrix). We previously showed that EigenSpeed requires a small number of iterations to converge, and incorporated defenses from potential attacks on the PCA algorithm by malicious nodes; see [22] for more details.

In the next sections, we examine how these bandwidth evaluation metrics perform and how they affect the performance of the network as a whole.

3.1.3 Evaluation of Predictive Power

To evaluate the effectiveness of the bandwidth evaluation algorithms, we first consider the correlation between the evaluated bandwidth of a node and the subsequently observed bandwidth of tunnels passing through that node. To minimize confounding effects, we consider the evaluated bandwidth of a tunnel to be the minimum of the evaluated bandwidths of the routers it passes through.

To generate the data in this section, we used a custom-written flow-level simulator of the Tor network. We created a 1 000-node network, with actual bandwidth data obtained from the Tor network [3]. For each tick of the simulator, it simulated 10 000 flows, with intermediate routers chosen at random weighted by their bandwidth as evaluated by each metric[6]. Routers allocated their bandwidth between their flows using fair queuing. For each scenario, the simulation ran for 1 440 ticks, simulating one day's worth of one-minute flows. Each of the graphs in Figure 3 shows the flows for the final tick of the simulation, after each algorithm has a full day to make observations. Note that each router is splitting its bandwidth between 20–30 flows on average, so direct observation of the available bandwidth of a peer is unlikely.

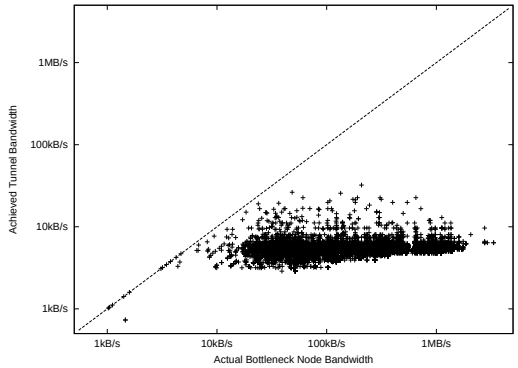
Figure 3-(a) plots the actual bandwidth of the bottleneck node against the bandwidth of the tunnel. We see that, due to the load balancing of the network, the total node bandwidth serves as a poor predictor of tunnel bandwidth; the log-log correlation[7] is 0.223. This is because nodes with more bandwidth are chosen more often, regardless of the number of other tunnels through that node or the bandwidth of the other nodes of the tunnels.

This helps explain Figure 3-(b): since nodes report the total traffic they observe, nodes that are overloaded stay overloaded and even tend to increase their reported bandwidth slightly, but tunnels through them suffer

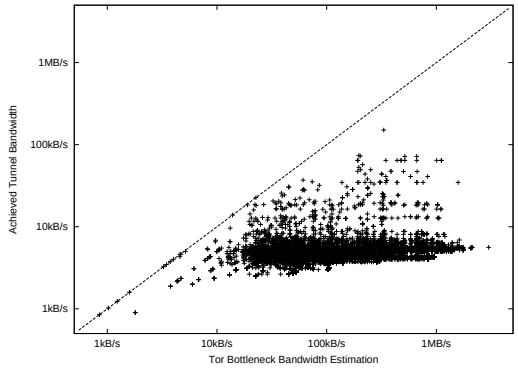
5. The number five was chosen for the same reason it is used for voting elsewhere in Tor: first, at least three of five must have data about the node for that node to be considered evaluated, and then the median of the (at least) three values present is considered the evaluated value.

6. That is, by the Tor router selection algorithm

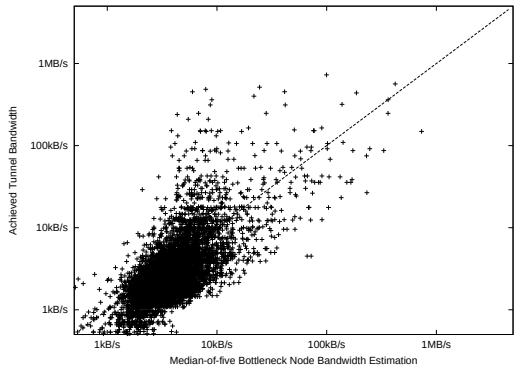
7. The log-log correlation between two sets of values is simple the correlation between their logarithms. We use this metric over the linear correlation for two reasons: first, we argue that bandwidth is inherently perceived logarithmically; that is, the difference between 1 kB/s and 2 kB/s is perceived as more similar to that between 1 MB/s and 2 MB/s than to that between 1 MB/s and 1.001 MB/s. Second, the range of values spans four orders of magnitude; using a linear correlation would dramatically overweight higher x values.



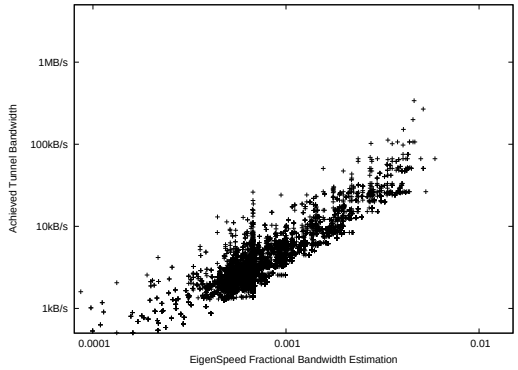
(a) Actual node bandwidth vs. achieved bandwidth ($r = 0.223$)



(b) Current Tor bandwidth measurement vs. achieved bandwidth ($r = 0.176$)



(c) Median-of-five bandwidth measurement vs. achieved bandwidth ($r = 0.680$)



(d) EigenSpeed bandwidth measurement vs. achieved bandwidth ($r = 0.881$)

Fig. 3. The accuracy of various prediction mechanisms for a sample of the trials. The $x = y$ line is included for reference.

from poor performance. Thus the basic shape of Figure 3-(b) resembles that of Figure 3-(a): tunnel bandwidth is only weakly related to node bandwidth (log-log correlation 0.176).

As Figure 3-(c) shows, however, the two methods proposed in the previous section tend to measure *per-flow bandwidth*, rather than *total node bandwidth*. This provides a much stronger link between predicted and observed bandwidth; it also means that routers which become overloaded see their evaluated bandwidth decrease, resulting in being chosen less often and therefore recovering form the overload. The log-log correlation here is considerably stronger, at 0.680.

Finally, Figure 3-(d) shows that EigenSpeed similarly measures an approximation of per-flow bandwidth. Since a node uses measurements from the entire network to estimate peer bandwidths, the estimations are even more accurate than the previous scheme, giving a log-log correlation of 0.881. Note also that the majority of flows are now above $10\,\mathrm{kB/s}$ at the expense of a smaller number falling below 1kB/s. We will exploit this predictable flow heterogeneity in Section 3.2.

3.1.4 Evaluation of Network Utilization

Though the proposed bandwidth metrics increase the predictive ability of bandwidth estimations considerably, it is critical that the choice of metric not adversely affect the overall performance or load balancing of the network. Since this is unclear from Figure 3, we turn to Figure 4, which presents another view of the same data. For each evaluation algorithm, this figure shows the total bandwidth achieved by all flows at each simulator tick, averaged over a ten-minute moving window for clarity. Included are similar curves for the current Tor evaluation algorithm and for no evaluation algorithm (i.e., choosing routers uniformly at random). All the curves in this graph are scaled to the theoretical maximum of a three-hop routing network, *i.e.* one third of the total router bandwidth.

There are several things to observe in this graph. The first is that appropriate load balancing is vital to maximizing network utilization (and therefore average tunnel throughput). With load balancing, network utilization can exceed 95%, whereas choosing routers uniformly at random hovers at approximately half of that. The second is that Tor, as currently implemented, does a fairly good job of this load balancing. This is to be expected, since Tor's router selection was designed for the sole purpose of load balancing. We can also see that EigenSpeed also does an excellent job, while providing the additional predictive power described in the previous section.

The Median-of-five algorithm performs less well, only achieving approximately 80% utilization after a single day. We theorize that this is due to the need for a considerably denser observation matrix to achieve high accuracy; however, lower-bandwidth nodes are less likely to be selected (and therefore observed) for the first time, resulting in their under-utilization. Since currently

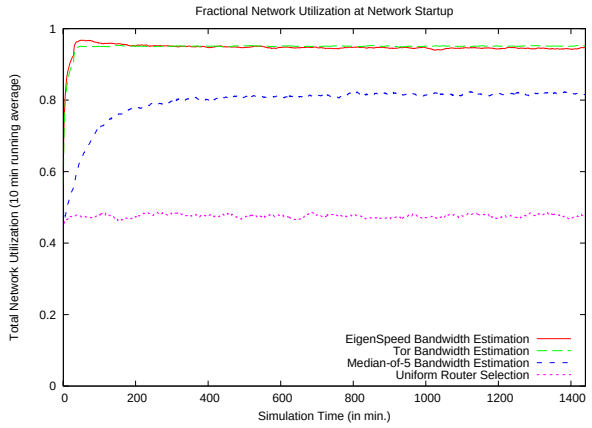


Fig. 4. Fractional network utilization for the bandwidth evaluation algorithms presented. Utilization using the current Tor algorithm and using no bandwidth estimation (i.e., flat uniform router selection) are included for comparison.

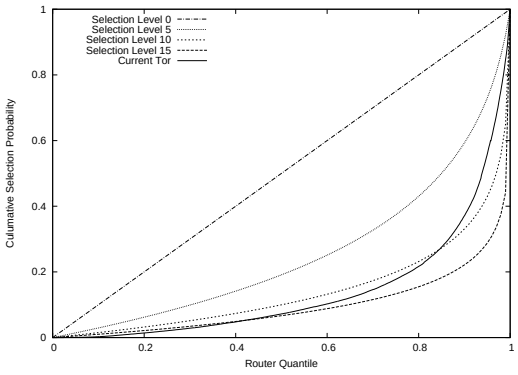


Fig. 5. Cumulative distributions of routers selected by ranking for some selection levels; note that these curves represent $1 - f_s^{-1}(1-x)$. The equivalent curve for the Tor router selection algorithm is included for comparison purposes.

measurement lifetime is on the order of a day, we did not investigate further to determine the time required to reach high utilization[8], as a significant increase in measurement lifetime may increase vulnerability to the spotlight attacks discussed previously. Accordingly, for this reason and those discussed in Section 3.1.2, we conclude that EigenSpeed represents the superior choice for a new bandwidth evaluation algorithm for Tor.

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 its vulnerability to subversion as well as provide a better experience for all classes of users; we call this algorithm Tunable Tor, due to its user-configurability.

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 [9]. 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 common pool of routers, but weighting their selection differently.

We define a family of functions $f_s : [0,1] \to [0,1]$, with parameter s, by:

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

Note that this family of functions is well-defined for all $s \neq 0$. For $s = 0$, we define f_s to be its point-wise limit as $s \to 0$; *i.e.*, $f_0(x) = x$.

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[9], 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 $\lambda = -s$.

The cumulative distribution function of the probability of choosing any given router is shown in Figure 5 for different values of s; a similar CDF of router selection for the current Tor router selection is included for

8. Analytically, the problem is quite complex: the expected time before all queries are expected to return a valid value for an N-router network represents an N-copy, N-coupon coupon-collector's problem with heterogeneous probabilities determined by the router selection algorithm. A cursory search of the literature reveals no good closed-form approximations for this type of problem.

9. And in fact, Sherr et al. [23] do apply this selection function to a variety of other metrics.

comparison.[10] This procedure is then repeated for any other routers to be selected, enforcing the restriction that duplicate selections are not allowed, nor are nodes within the same /16 subnet or node family.

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 for completeness; a value of $s = 15$ implies that the most highly ranked router in a set of $n = 1\,700$ (a typical number of routers available in the Tor network at any given time) will be chosen 23% of the time[11]. It should be stressed, however, that a practical upper bound for s is 10, which results in the most highly ranked router being chosen less than 4% of the time in the above scenario.

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.

3.2.1 Discovering the Selection Level: Single Path

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: such attacks as those described by Murdoch and Danezis [24] become much more feasible with only a few flows to focus on. In this section and the following one, we examine the ability of an attacker to fingerprint the selection level chosen by the user. To do so, we consider two scenarios: fingerprinting a single tunnel, and fingerprinting a correlated series of tunnels.

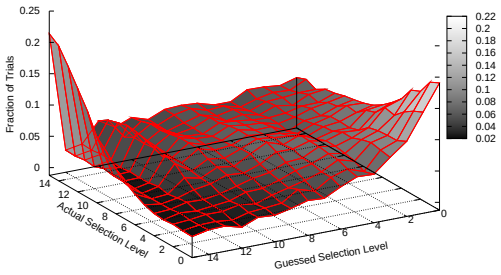


Fig. 6. Actual selection level and most likely selection level according to a naïve Bayesian classifier

For this first scenario, the threat model we address is that of a single malicious router, the controller of which wishes to identify the selection level of tunnels which pass through it. 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[12] of a single tunnel for another data set (again, $n = 100\,000$). The results of this classification are shown in Figure 6: 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 7 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 4.568 for the uniform distribution and 4.567 for the skewed distribution.

Note also that this sort of fingerprinting requires that the attacker be able to determine which routers form the tunnel; the ability to do so implies either a widespread passive adversary, a malicious router on the selected path, or a compromised or malicious destination web site.

10. Tor's CDF is, of course, dependent on the bandwidths of the currently available routers; this CDF is based on a static snapshot of router bandwidths.

11. 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 $\log_2(2^{s-1}/n + 1)/s$.

12. This classification is based on a single router, which could be the intermediate router or the exit router, since the entry guard is not affected by selection level.

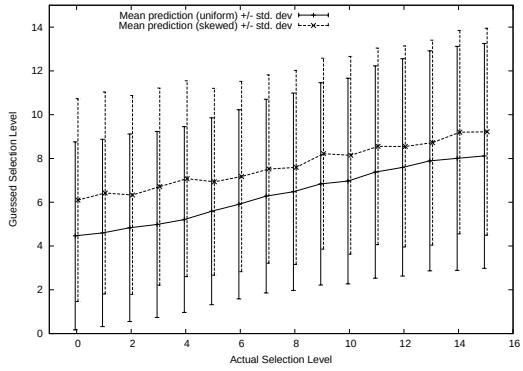


Fig. 7. Mean and standard deviation of guessed level by actual selection level according to a naïve Bayesian classifier, for both a uniform and skewed distribution of selection levels.

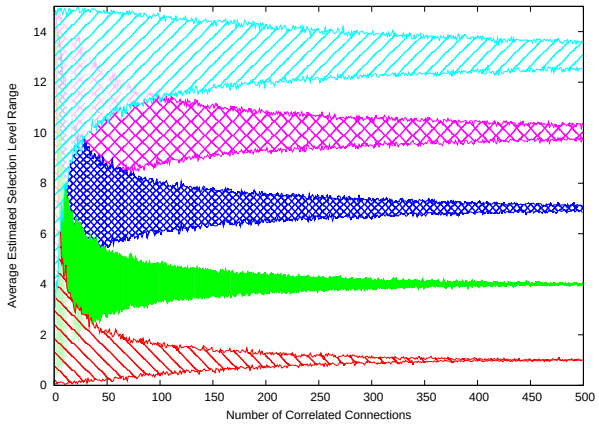


Fig. 8. Average range of possible selection levels according to a known-distribution Kolmogorov-Smirnov test.

3.2.2 *Discovering the Selection Level: Multiple Paths*

For the next scenario we examine, the threat model is an attacker who is able to correlate (potentially many) tunnels at a single selection level. This may be due to a user using a pseudonym for their connection, or by including linkable information, such as cookies[13]. Under this threat model, the attacker slowly accrues correlated exit routers. To identify which selection levels at which these routers were chosen, we employ a known-distribution Kolmogorov-Smirnov (K-S) test[14]; specifically, for each data set we attempt to confirm the null hypothesis that the data could arise from each of the possible selection levels at the 95% level.

For each selection level, 500 exit routers were chosen one at a time; after each choice, the K-S test was performed for the data collected so far, and the range of selection levels passing the test was recorded for each new observation. This experiment was repeated 100 times for each selection level. Figure 8 shows the average range of selection levels passing this test for some representative selection levels as the number of correlated tunnels increases. We can see that, for each data set, the possible range of selection levels is initially very broad, but it decreases with additional observations. It decreases most quickly for the lower selection levels: the possible selection level range is reduced to only three possibilities after only 50 observations at selection level 1, but it takes nearly five times as many observations to reduce the possibilities that far for selection level 13.

13. We emphasize that these possibilities are not weaknesses in Tor, but rather failures in user education similar to that described in [6].

14. Credit is due to Matt Wright for suggesting the use of the Kolmogorov-Smirnov test in this attack.

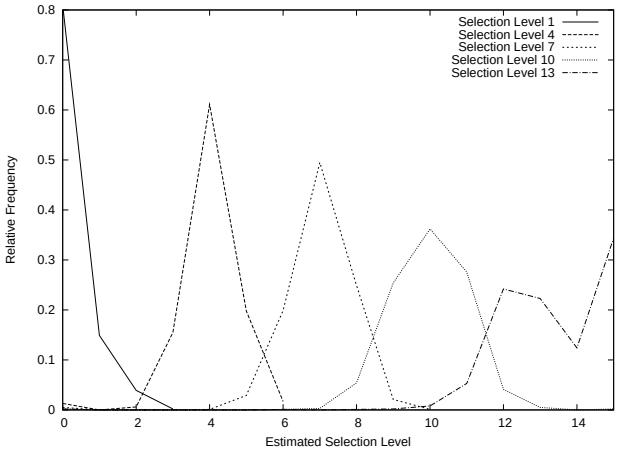


Fig. 9. First unique matching selection level according to a known-distribution Kolmogorov-Smirnov test.

When attempting to identify the selection level, it is not at all obvious when enough observations have been made to make the correct identification. One possibility is to stop as soon as only a single level passes the K-S test. Figure 9 shows the results of this approach; we can see that it works moderately well for intermediate selection levels, identifying the selection level correctly as much as 80% of the time. However, selection levels closer to the extrema are identified much less accurately: selection level 1 is misidentified as selection level 0 nearly 80% of the time. One possible explanation for this phenomenon is that selection levels 0 collects all of the long tail that would fall into negative selection levels; we can see a similar effect happening at the other extreme, with selection level 13 being misidentified as selection level 15 a substantial fraction of the time. Note also that several moderate to high selection levels are occasionally misidentified as selection level 0. This seems to occur when more than one router selected early is ranked poorly.

Another possible application of this attack is to gather correlated pairs of exit routers and websites (rather than pairs of exit routers and users, as above). If enough data were gathered, it might be possible to automatically determine which websites are visited at high anonymity levels and note them for further investigation. Since any website would likely be visited with a variety of selection levels, suspicious sites would be those with a

visitor selection level distribution biased towards higher levels. Determining the *mix* of selection levels visiting a website is more complex and computationally intensive, and we do not investigate it further here.

4 Whole-System Evaluation

In order to evaluate the degree to which the proposed changes meet the dual goals of improving 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

In order to obtain an accurate picture of what the performance of a Tor network using these proposed improvements would look like, we ran tests using a single client modified to use the Tunable Tor algorithms in the real Tor network and a simulator where all clients use the modified algorithms. The relative agreement between the two sets of results argues for their fidelity.

4.1.1 Performance in the real Tor network

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

Figure 10 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 95^{th} 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 11 presents the 90^{th} percentile and median 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 3(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. Though the difference is more dramatic in the 90^{th} percentile, this effect can be seen in the median as well.

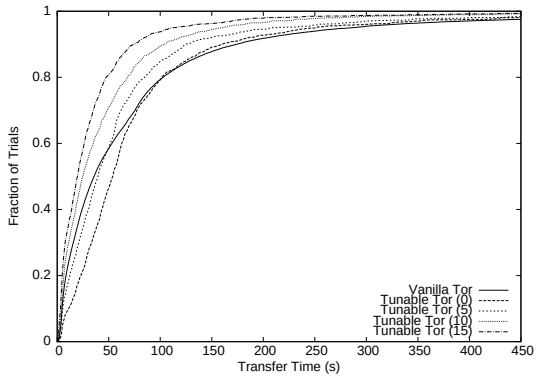


Fig. 10. Cumulative distribution of transfer times for a 1 MB file for vanilla Tor and several selection levels. For Tunable Tor, note that Selection Level 0 corresponds to maximum anonymity, while Selection Level 15 corresponds to maximum performance.

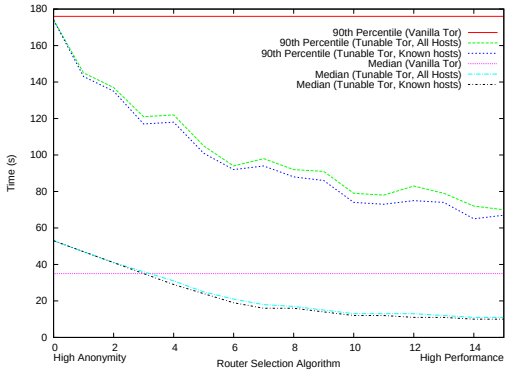


Fig. 11. The 90^{th} percentile and median of transfer times by selection level for known routers and all routers. Lines for vanilla Tor are included for comparison.

Finally, Figure 12 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. As before, vanilla Tor is included for comparison[15]. Again we see the deleterious effect of Tor's long tail: even though, as Figure 10 shows, the median time is only 35 seconds, the mean time is more than twice that, at 84 seconds.

15. Due to the larger data set, the confidence interval is sufficiently small (± 2 seconds) as to be omitted.

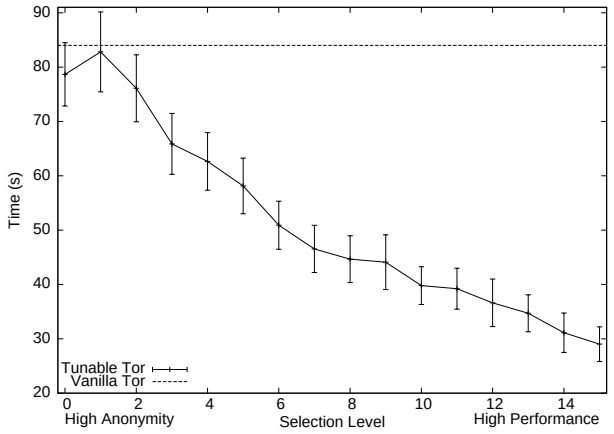


Fig. 12. Mean transfer times with 95% confidence intervals for known routers by selection level; the mean time for vanilla Tor is included for comparison.

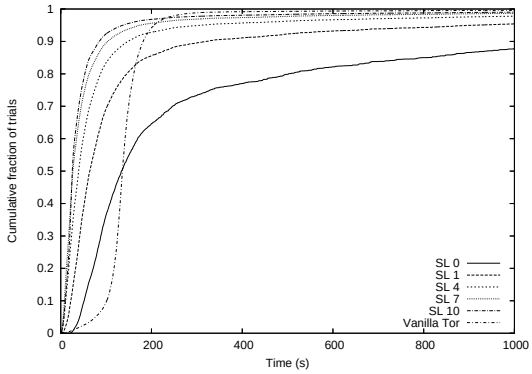


Fig. 13. The cumulative distribution of transfer times for a 1 MB file by selection level as determined using a flow-level simulator. 24% of the clients are using selection level 0, 49% are using selection level 10, and 3% are using each of the remaining levels.

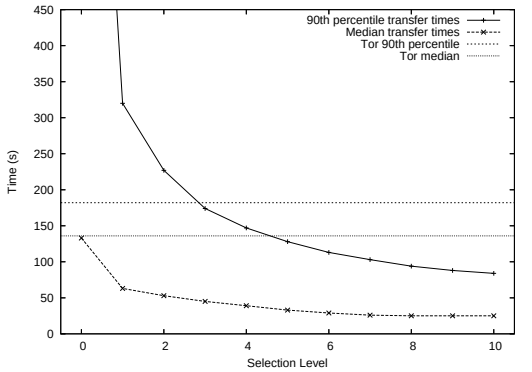


Fig. 14. The 90^{th} percentile and median transfer times for a 1 MB file by selection level as determined using a flow-level simulator. The 90^{th} percentile time for Selection Level 0 is 1243 seconds; the data point is omitted for reasons of scale. The selection mix is that of Figure 13.

For Tunable Tor, the mean time decreases with increasing selection level, as expected.

One important conclusion that can be drawn from our results is that the Tor load balancing algorithm does *not* effectively distribute load among the high- and low-bandwidth nodes: our trials that preferentially choose high-bandwidth nodes see much higher performance and a reduced long tail, showing that these nodes have more spare capacity than the low-bandwidth ones. This, of course, raises a question of how Tor would work if all users were to use our proposed path selection algorithms; we address this question next.

4.1.2 Performance in a Simulated Tor Network

To study the effect of the proposed changes in a network where all clients are choosing paths using the Tunable Tor algorithm and evaluating routers using the EigenSpeed algorithm, we used the flow-level simulator described in Section 3.1.3. The mix of selection levels is based on the assumption that most users will prefer maximum performance, with a smaller fraction preferring maximum anonymity and a much smaller fraction tuning their performance to each of the intermediate selection levels; under this assumption, the results are relatively insensitive to the exact mix of selection levels used.

Figure 13 shows the cumulative distribution of transfer times for representative selection levels. Though the absolute times differ (due to differences in the offered load), the trend is the same as that seen in Figure 10. Selection level 0 (i.e., maximum anonymity) performs noticeably worse than the other selection levels; this may indicate that even a slight bias towards higher performance routers can make a difference in performance out of proportion to its effect on anonymity. Figure 14 supports this: selection level 0 has a 90^{th} percentile transfer time nearly four times that of selection level 1. The median transfer time statistic show a similar though smaller difference between selection level 0 and selection level 1. Section 4.2 shows that the decrease in anonymity between these two levels is relatively small; therefore, it may be the case that selection level 1 is a better default for "high anonymity" than is selection level 0. These experiments also confirm that selection level 10 is a good value for the "high performance" selection level; when a similar mix of selection levels was tested with 15 as the highest selection level, some higher selection levels actually performed more poorly than some intermediate levels due to congestion at the highest-bandwidth routers.

Looking at the mean transfer times in Figure 15, we see a similar pattern: selection level 0 takes, on average, more than twice as long as selection level 1. Increasing selection levels provide diminishing returns, supporting our earlier suggestion that selection level 10 are impractical. (Note that, due to the high number of trials performed, the confidence intervals for these sample means are quite small.)

We also tested the flow-level simulator where all flows

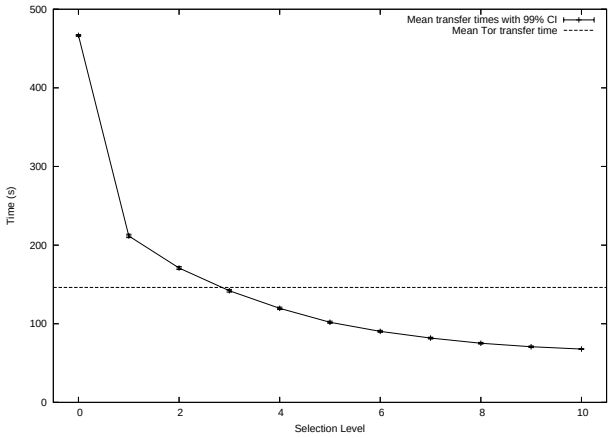


Fig. 15. The mean transfer times for a 1 MB file with 99% confidence intervals, by selection level as determined using a flow-level simulator. The selection level mix is described in Figure 13.

used the Tor selection algorithm. As seen in Figure 13, the long-tail behavior that we observed in the real Tor network has disappeared. This highlights the difference between our simulator and the Tor network: the simulator assumes that all clients send constant-rate flows and that bandwidth is perfectly balanced between them. In the real Tor network, flows are bursty and congestion causes significant performance degradation. [16] This motivates future research on higher-fidelity modeling, perhaps using packet-level simulator that can fully capture the complex behavior of the Tor routers.

That aside, the other points of comparison remain similar between simulation and real-world measurements. Lower selection levels underperform the default Tor algorithm in terms of mean and 90th percentile transfer times; interestingly, the median transfer time is better than Tor even at the most anonymous level 0. Users who choose selection level 3 and above will notice performance better than Tor on all of the metrics, and users at selection level 10 get a drastic performance improvement. We also compared the overall utilization levels of the network between the tunable selection levels and vanilla Tor; at 96.3% and 95.8%, respectively, they were similar.

4.2 Anonymity

We next analyze the effects of tunable path selection on anonymity. One measure of anonymity is how many routers an intelligent attacker must subvert in order to have a high probability of compromising a tunnel. Throughout this section, our threat model is an attacker who can compromise some fraction of the routers in the Tor network, or alternately, eavesdrop on all of their traffic. While these two threats are, for the most part equivalent, compromising routers allows for the "false advertising" attack described below, while eavesdropping does not.[16]

Intuitively, it is clear that if routers are chosen uniformly at random, more routers must be compromised in order to achieve a high probability of tunnel compromise, while skewing the selection towards certain routers requires fewer to be compromised (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 [25] (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). We have also considered using the entropy as a metric, due to its popularity in the study of anonymous systems, but rejected it because entropy measures uncertainty, rather than equality or uniformity[17].

Figure 16 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 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 16 with Figure 12; 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 17. At high selection levels, an attacker who controls a relatively small fraction of the most highly ranked routers can compromise a significant fraction of

16. Another possible threat model is that of an attacker introducing additional routers to the network. For small number of number of added routers, this is equivalent to the threat model given; however, the threat presented by botnets is different: if an attacker can inject sufficient routers to control the majority of the network, these schemes fail to protect the user. This vulnerability to botnets is shared by all networks of this type, and is beyond the scope of this paper.

17. Because the threats to anonymity arise when attackers control some fraction of the routers in the network, maximum anonymity is achieved when the attacker derives no benefit from compromising one router over another; for this reason, equality metrics such as the Gini coefficient are a better metric than those based on self-information.

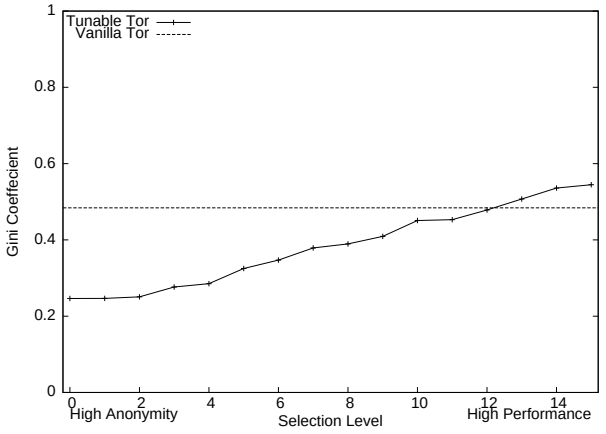


Fig. 16. Gini coefficient of router selection equality by selection level. The Gini coefficient for the router selection equality for vanilla Tor is included for comparison.

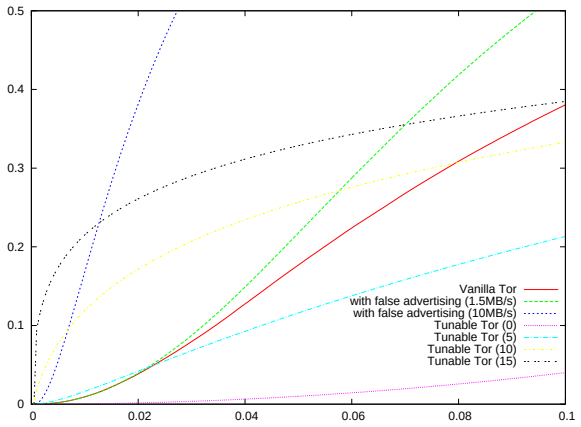


Fig. 17. The fraction of tunnels compromised if an attacker compromises the top given fraction of Tor routers, for vanilla Tor and for various selection levels.

tunnels[18]. 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%.[19] 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 and third curves for vanilla Tor, labeled "false advertising," which reflects an attacker changing the advertised bandwidth for each compromised router to the previous and current maximum believable bandwidth (1.5 MB/s and 10 MB/s, respectively) 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 EigenSpeed has mechanisms to detect colluding groups of malicious nodes reporting false bandwidths and prevent them from impacting the global consensus vector. Our findings are similar to those by Bauer et al. [8].

18. We consider a tunnel compromised here if the attacker controls both endpoints.

19. As the compromised nodes are both fast and likely to be stable, we assume that they are all eligible to be guards; thus they comprise 40% of all guard nodes.

5 Related Work

Whereas our work optimizes tunnel bandwidth, for reasons discussed in Section 3, considerable work has been done studying the use of Tor paths optimized for latency as opposed to bandwidth. Sherr et al. propose the use geographic coordinates to create paths that fall within selected bounds [26] and use the family of functions f_s described in this paper for a link-based router selection algorithm more suited to optimizing latency [23]. Renner developed a controller for Tor to select paths according to criteria such as avoiding ocean crossings and otherwise minimizing latencies [27]. Reardon and Goldberg show that modifying Tor to run DTLS over each router link and use a single end-to-end TCP session can significantly reduce end-to-end latency and queue lengths [16] and can improve throughput as well. In general, the problem of measuring and optimizing for latency, and the security implications of doing so, is a complex one and beyond the scope of this work.

Our results regarding the variability of Tor performance match a comparative study of Tor and AN.ON performance [28], which also showed large standard deviations for bandwidth values provided by Tor. Bauer et al. [8] consider distributed probing, perhaps in the style of anonymous auditing [29], 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.

Extensions such as Torbutton [30], which selectively enables or disables Tor depending on task, and FoxTor [31], which tracks whether Tor is currently enabled, provide users with (among other things) a cruder way to trade off performance and anonymity. The fact that users are toggling the use of Tor often enough to require a convenient way to do so and to monitor the current state indicates that the performance-security tradeoff is a real one which needs to be better addressed. Other work aims to improve Tor performance by speeding up cryptographic operations [32], [33].

One approach to improve overall Tor performance is to use a peer-to-peer design where all users contribute forwarding capacity [34], [35], [36]. Tor design-

ers avoided peer-to-peer approaches due to Sybil attacks [37]; unfortunately, existing peer-to-peer anonymous designs are either insecure [38], [39] or not scalable [35]. New approaches such as ShadowWalker [40], using redundant structured peer-to-peer topologies, show promise but remain untested.

Murdoch and Watson [41] analyzed a version of the router selection algorithm we propose in Section 3.2 from a queuing theory perspective; however, since the queuing theory approach fails to account for the congestion control mechanisms of TCP and the Tor network itself, the results obtained were not a useful measure of the performance of Tunable Tor. Furthermore, they analyzed only two scenarios: one where all users were using selection level 1 and one where all users were using selection level 15. As previously mentioned, selection level 15 is likely too biased towards high-bandwidth routers to be practical and is included in our analysis mainly for completeness; selection level 10 provides a more reasonable upper bound for users who want high performance. They also point out that, in the case of botnets, better anonymity might be achieved by actually choosing routers with *higher* bandwidth. Using tunable Tor, users can adjust their selection strategy based on whether they believe a botnet attack or a compromise of high-bandwidth nodes is a more likely attack.

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 are currently implementing a more detailed, packet-level simulator of the Tor network; this will increase the fidelity of the simulation by including such effects as variable file sizes, variable intervals between requests, and TCP slow-start behavior. 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.

Acknowledgments

We would like to thank Roger Dingledine, Ian Goldberg, and Steven Murdoch for helpful discussions about this work, and Joshua Juen for helping gather Tor data sets. This work was supported in part by a grant from the National Science Foundation, CNS 06–27671.

References

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

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

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

[4] "TorStatus - Tor network status," http://torstatus.kgprog.com/.

[5] K. Loesing, "Measuring the Tor network," https://git.torproject.org/checkout/metrics/master/report/dirreq/directory-requests-2009-06-25.pdf.

[6] D. Goodin, "Tor at heart of embassy passwords leak," *The Register*, Sep. 10, 2007.

[7] 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.

[8] 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)*, Oct. 2007.

[9] 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.

[10] 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*, Feb. 2002.

[11] ——, "Defending anonymous communication against passive logging attacks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[12] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in *Proceedings of the Eighth Privacy Enhancing Technologies Symposium (PETS'08)*, Aug. 2009.

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

[14] 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.

[15] 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.

[16] J. Reardon and I. Goldberg, "Improving Tor using a TCP-over-DTLS tunnel," in *Proceedings of the Eighteenth USENIX Security Symposium*, Aug. 2009.

[17] P. Palfrader, "Echolot," http://www.palfrader.org/echolot/, accessed on 10 August 2009.

[18] 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)*, Apr. 2006.

[19] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III anonymous remailer protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, pp. 2–15.
[20] R. Dingledine, Nov. 2007, Personal Corresponence.
[21] R. Snader and N. Borisov, "A tune-up for Tor: improving security, performance and anonymity in the Tor network," in *Proceedings of the Fifteenth Annual Network and Distributed System Security Symposium (NDSS'08)*, Feb. 2008.
[22] ——, "EigenSpeed: Secure peer-to-peer bandwidth evaluation," in *Proceedings of the Eighth International Peer-To-Peer Systems Workshop (IPTPS'09)*, Apr. 2009.
[23] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Proceedings of the Ninth Privacy Enhancing Technologies Symposium (PETS'09)*, Aug. 2009.
[24] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.
[25] C. Gini, "Measurement of inequality of incomes," *The Economic Journal*, vol. 31, no. 121, 1921.
[26] M. Sherr, B. T. Loo, and M. Blaze, "Towards application-aware anonymous routing," in *Workshop on Hot Topics in Security*, Aug. 2007.
[27] 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.
[28] R. Wendolsky, D. Herrmann, and H. Federrath, "Performance comparison of low-latency anonymisation services from a user perspective," in *Privacy Enhancing Technologies Symposium*, ser. Lecture Notes in Computer Science, N. Borisov and P. Golle, Eds., vol. 4776. Ottawa, Canada: Springer, Jun. 2007.
[29] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *IEEE INFOCOM*, Apr. 2006.
[30] S. Squires and M. Perry, "Torbutton — quickly toggle Firefox's use of the Tor network," https://torbutton.torproject.org/, 2006.
[31] S. Romanosky, "FoxTor: Anonymous web browsing," http://cups.cs.cmu.edu/foxtor/, 2006.
[32] L. Øverlier and P. Syverson, "Improving efficiency and simplicity of Tor circuit establishment and hidden services," in *Privacy Enhancing Technologies Symposium*, ser. Lecture Notes in Computer Science, N. Borisov and P. Golle, Eds., vol. 4776. Ottawa, Canada: Springer, Jun. 2007.
[33] A. Kate, G. Zaverucha, and I. Goldberg, "Pairing-based onion routing," in *Privacy Enhancing Technologies Symposium*, ser. Lecture Notes in Computer Science, N. Borisov and P. Golle, Eds., vol. 4776. Ottawa, Canada: Springer, Jun. 2007.
[34] 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, Nov. 2002.
[35] 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, Nov. 2002.
[36] A. Nambiar and M. Wright, "Salsa: a structured approach to large-scale anonymity," in *ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM Press, 2006, pp. 17–26.
[37] J. Douceur, "The Sybil Attack," in *Proceedings of the 1st International Peer-To-Peer Systems Workshop*, Mar. 2002.
[38] P. Tabriz and N. Borisov, "Breaking the collusion detection mechanism of MorphMix," in *Privacy Enhancing Technologies Workshop*, Jun. 2006.
[39] 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.
[40] P. Mittal and N. Borisov, "ShadowWalker: Peer-to-peer anonymous communication using redundant structured topologies," in *Proceedings of the Sixteenth ACM Conference on Computer and Communications Security (CCS'09)*, Nov. 2009.
[41] S. J. Murdoch and R. N. Watson, "Metrics for security and performance in low-latency anonymity systems," in *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS 2008)*, Jul. 2008.
[42] N. Borisov and P. Golle, Eds., *Privacy Enhancing Technologies Symposium*, ser. Lecture Notes in Computer Science, vol. 4776. Ottawa, Canada: Springer, Jun. 2007.

PLACE PHOTO HERE

Robin Snader Robin Snader received his Ph.D. in computer science from the University of Illinois at Urbana-Champaign in 2009. His research interests are in the field of networking, including wireless communications, and security, including a systems-level approach to private and anonymous communication. He currently works as a Research Analyst and Patent Agent for Shook, Hardy and Bacon, L.L.P.

PLACE PHOTO HERE

Nikita Borisov Nikita Borisov is an assistant professor at the University of Illinois at Urbana-Champaign. His research interests are online privacy, network security, and Internet-scale distributed systems. He is the co-designer of the "off-the-record" (OTR) instant messaging protocol and was responsible for the first public analysis of 802.11 security. He has also served as co-chair of the Privacy Enhancing Technologies Symposium in 2007 and 2008. He received his Ph.D. from the University of California, Berkeley in 2005.