An Empirical Evaluation of Relay Selection in Tor

Chris Wacek
Georgetown University

Henry Tan
Georgetown University

Kevin Bauer
University of Waterloo

Micah Sherr
Georgetown University

Abstract

While Tor is the most popular low-latency anonymity network in use today, Tor suffers from a variety of performance problems that continue to inhibit its wide scale adoption. One reason why Tor is slow is due to the manner in which clients select Tor relays. There have been a number of recent proposals for modifying Tor's relay selection algorithm, often to achieve improved bandwidth, latency, and/or anonymity. This paper explores the anonymity and performance trade-offs of the proposed relay selection techniques using highly accurate topological models that capture the actual Tor network's autonomous system (AS) boundaries, points-of-presence, inter-relay latencies, and relay performance characteristics.

Using realistic network models, we conduct a whole-network evaluation with varying traffic workloads to understand the potential performance benefits of a comprehensive set of relay selection proposals from the Tor literature. We also quantify the anonymity properties of each approach using our network model in combination with simulations fueled by data from the live Tor network.

1. Introduction

The Tor [11] anonymity network is used by hundreds of thousands of daily users to improve the privacy of their communication [25]. Recently, significant effort has been expended to improve Tor's performance, which as the network's operators have pointed out [12], suffers from both high congestion and latency. Such efforts have focused on improving Tor's circuit processing [3, 46], transport mechanism [28, 39], relay recruitment [22, 31], and—the subject of this paper—relay selection [2, 42, 43, 49].

Tor exhibits high latencies partly due to the manner in which clients select relays for their anonymous *circuits* (a path of three Tor relays, selected in proportion to their bandwidth). For example, a large fraction of Tor's volunteer-operated relays are located in the United States or Germany [30, 47], requiring a typical client's traffic to make at least one transoceanic trip.

Existing work has proposed methods of creating lower latency anonymous circuits by carefully selecting relays to reduce either link latencies [41, 42] or the geographic distance covered by anonymous paths [2]. Other work has proposed that Tor clients be given the ability to *tune* the selection of relays in a manner that allows clients to achieve greater performance (by weighting more heavily toward high-bandwidth routers) or greater anonymity (by weighting selection more uniformly at random) [43]. Still additional work has attempted to decrease latency by avoiding circuits that have high levels of congestion [49].

The Need for Realistic Tor Network Modeling. Despite this growing body of research on path selection techniques for Tor, none of the existing proposals have been evaluated under conditions that accurately reflect those that would be found on a live anonymity network. To illustrate, existing studies [43, 46] have shown that performance gains achieved under modeling and simulation are not present when the system is tested under more realistic conditions [21, 32]. Although a particular algorithm may show advantageous effects – even in the live network – when adopted by a small number of clients and/or relays, the technique may have unexpected negative consequences on the network when adopted en masse.

There is thus a need to move beyond the consideration of solely local effects ("*if I adopt this algorithm, will it improve my performance and anonymity?*") and consider potential impacts on the network in toto ("*what are the effects if a large number of clients/relays adopts this strategy?*"). In short, we desire new methods and tools that (1) more realistically model the Tor network and (2) enable more comprehensive performance and anonymity analyses.

Whole-network Tor Experimentation. Through whole-network experimentation using a state-of-the-art Tor emulation framework armed with highly realistic network models, this paper aims to better understand the potential performance benefits and anonymity risks of path selection algorithms.

The primary contribution of this paper is a framework for measuring performance and anonymity under realistic conditions (Section 3). Our experimental methodology uses the recently proposed ExperimenTor [4] emulator, running native Tor binaries on a synthetic network topology. To maximize realism, we construct topologies that capture Tor's bandwidth distribution, distribution of configurations, geographic diversity (both of relays and the most common clients and destinations), AS and point-of-presence paths, and pairwise latencies. The ability to execute unmodified Tor code and operate on (emulated) networks that share many characteristics with the live Tor network permits us to more accurately predict how a proposed relay selection strategy will behave on the live network. Importantly, our framework allows us to measure multidimensional aspects of both performance (i.e., throughput, latency, time-to-first-byte) and anonymity (i.e., frequency of relay selection, AS path diversity).

Using this framework, we evaluate a set of recently proposed relay selection techniques that we have integrated into Tor (Sections 4 through 6). Our findings indicate that several previously proposed methods are unlikely to produce desirable performance if widely adopted. We additionally evaluate hybrid relay selection techniques that combine aspects of existing approaches. Our results show that a hybrid strategy in which selection is biased in favor of bandwidth and away from congested circuits provides the best performance.

Contributions. In summary, this work offers the following contributions to the field of anonymous communications.

- We introduce a methodology for modeling the Tor network, and present our resulting Tor network model that includes the real Tor network's AS boundaries, link latencies, bandwidths, and relay configurations. We instantiate this model within a state-of-the-art Tor network emulation platform to realize highly realistic Tor network experimentation.

- We implement a comprehensive set of relay selection algorithms from the Tor literature in the Tor source code and conduct an exhaustive performance analysis to identify which path selection algorithm offers the best performance under dynamic traffic loads.

- We also simulate each router selection algorithm and use real data from the live Tor network to evaluate the anonymity implications of each respective approach.

- We find that a combination of Tor's bandwidth-weighted relay selection and techniques that avoid congested circuits results in the greatest improvement in throughput and latency relative to Tor's current design. Augmenting Tor with virtual-coordinate based selection also suggests promising performance improvements. Furthermore, we find that neither approach significantly impacts anonymity.

2. Background

Relay Selection in Tor. Tor is the third-generation onion routing network and provides anonymous communication to TCP-based applications [11]. Tor clients select a source-routed *circuit* of precisely three Tor *routers* (sometimes called relays) by querying any one of several authoritative *directories*. After constructing a circuit, clients forward traffic through their circuits using a layered encryption scheme based on onion routing [17]. Upon receipt of a fixed-size unit of transmission (a *cell*), each router along a circuit adds or removes a layer of encryption, depending on the cell's direction.

While early onion routing systems initially specified that clients should select routers uniformly at random [45], it became necessary to attempt to balance Tor's traffic load over the available router bandwidth as the anonymity network's popularity increased. Tor performs load balancing by weighting router selection in proportion to each router's perceived bandwidth capacity. Tor currently utilizes a set of trusted Bandwidth Authorities which are responsible for actively probing the Tor routers and estimating each router's capacity [36]. Additional constraints are placed on router selection, including the use of entry guards [35] for the first hop to defend against the predecessor attack [50] and exit policies that specify the destination addresses and ports allowed by an exit router's operator. Recent work has also suggested selection algorithms that incorporate users' trust over various parts of the network [23].

Despite Tor's popularity with several hundreds of thousands of daily Tor users [19], one of the primary roadblocks to wide-scale Tor adoption continues to be its poor performance. Prior work [12] has examined a number of factors that contribute to Tor's performance problems, including undesirable inter-circuit interference due to TCP's congestion control [39], suboptimal flow control at the application layer [3], and imperfect load balancing which causes lower bandwidth routers to handle too much traffic.

Snader and Borisov offer refinements to Tor's router selection policy that allow senders to *tune* the performance of their anonymous paths by defining the degree to which relay selection is biased in favor of bandwidth [43]. However, follow-up work by Murdoch and Watson have found that Tor's default router selection algorithm offers a good trade-off between performance and anonymity [32].

Other work has noted that Tor's use of three often geographically distributed routers introduces yet another source of high latency due to cell propagation over circuitous routes; latency-informed router selection [42] using virtual coordinate systems [8, 41] and minimal geographical distances between Tor routers [2] have been proposed as methods to optimize Tor circuits for low latency. However, to date, no whole-network evaluation of either proposal has been performed; thus, it is unclear how each path selection algorithm affects performance when deployed at scale.

In this work, we seek to provide a unifying framework for reasoning about the many different methods for router selection that have been proposed in the literature and an understanding of what techniques are effective.

Tor Evaluation and Modeling. A large volume of existing work attempts to approximate the live network's behavior, often as a means to evaluate a refinement to the network's protocols or configuration. At a high-level, efforts at modeling Tor can roughly be organized into three categories: *analytic*, *simulation*, and *emulation*.

Analytic methods [5, 32] allow researchers to evaluate refinements to Tor's protocols. However, accurately and analytically modeling complex network effects (e.g., congestion, jitter, etc.) remains an open problem. Indeed, existing analytic approaches often ignore network effects entirely.

There are also a number of available Tor simulators [22, 32–34]. As with the analytic methods, existing simulators fail to fully capture Tor's complexities. (To highlight this complexity, we note that recent versions of Tor have more than 200 configuration options.) In general, it is difficult to assess how well simulated behavior predicts the behavior of Tor in an actual deployment.

More recently, Moore *et al.* [31] and AlSabah *et al.* [3] use the ExperimenTor emulator [4] to respectively examine alternative rate limiting and congestion/flow control policies for Tor. However, their topologies do not model the live Tor network's latency, geography, or AS distributions. Similarly, Jansen and Hopper [21] introduce the Shadow framework for executing (slightly modified) Tor code on a synthetic network. They sample the live network's bandwidth distribution, but configure geographic locations and latencies only according to a sample of several PlanetLab nodes.

Jansen *et al.* [20] extend their model to construct a network graph that clusters hosts into geographical regions, assigns upstream and downstream bandwidths and loss rates to hosts using measurements from Ookla Net Index, and utilizes link latency and jitter data obtained from iPlane [27]. An algorithm for obtaining a "best fit" approximation of real Tor network's relay bandwidth distribution is used to downsample and produce experimental Tor networks with 50 and 100 relays. One important limitation of this work, however, is that it does not attempt to model the Tor network at the AS-level; as such, it cannot be used to conduct an anonymity and security analysis of an AS-level adversary.

To the best of our knowledge, ours is the first work that attempts to model the distributions of latencies, bandwidths, relay types, AS assignments, and geographies found on the live Tor network. We apply our models both under simulation (for scalability) and emulation (for realism).

3. Modeling the Tor Network

One of the difficulties inherent in testing new protocols and technologies on the Tor network is the network's size: Tor has over 2500 relays and nearly 1000 bridges distributed globally. Certain types of research either cannot or should not be tested on the live Tor network because they either require large scale modifications to core protocols (and hence are impractical to implement globally) or because they could unwittingly compromise the anonymity of people who depend upon the network for their safety [44].

Technologies that make changes to the core Tor network therefore require an experimental platform that can emulate or simulate the characteristics of the live Tor network. There are a number of critical characteristics that should be considered when evaluating the performance and anonymity of an experimental technology: the latency between Tor relays; the bandwidth of individual Tor relays; and the distribution of clients, relays, and destinations, both across geographies as well as autonomous systems (ASes). In what follows, we describe how we construct (Section 3.1) and verify (Section 3.2) scaled-down models of Tor that faithfully represent the live network's distributions of latencies, bandwidths, geographies, and AS memberships.

3.1 Topology Construction

Our goal is to create a reduced map of the Internet that includes the network locations of Tor relays as well as clients and destinations, and supplies latency measurements between hosts.

A number of methods have been proposed for estimating the latencies between arbitrary Internet end-hosts, most notably IDMaps [15], the King method [18], and iPlane [26, 27]. We evaluated the feasibility of these existing options and concluded that they were insufficient for our purposes. IDMaps estimates latencies between arbitrary hosts by performing triangulation from strategically placed *Tracer* nodes; however, no such service is currently in place on the Internet. The King method relies upon recursive DNS queries, which the majority of DNS servers currently disable. iPlane has helpfully built and maintained a map of the Internet based on autonomous system data. Using iPlane data, we were able to construct AS level network topologies, but this unfortunately did not provide sufficiently accurate latencies: we found that in many cases, real world latencies between ASes are often small, while those across ASes can be relatively large — the iPlane data did not capture this characteristic, resulting in unrealistically small latencies within ASes. We additionally examined the iPlane "point of presence" data to construct our own AS graphs, but were unable to build a suitably connected graph using their data.

Accurately modeling the Tor network. We desire a "scale" model of the actual Tor network that accurately reflects the network's bandwidths and latencies, as well as the locations of its clients, destinations, and relays. Since no such model of the Tor network exists, we constructed a compact network graph suitable for our emulation experiments (see Section 6) using data from multiple available datasets. We constructed our model of the Tor network at the granularity of a *point-of-presence* (PoP), where a PoP is roughly intended to represent an access point on the Internet. We built our model of the Tor network as follows:

1. *Latency normalization.* To construct our model, we utilize traceroute data from CAIDA [7].[1] CAIDA collects and makes available traceroute measurements to most of the Internet's /24 prefixes from geographically and topologically diverse vantage points. We normalize the traceroutes by removing all *negative* latency hops; these occur when the traceroute data indicate that the latency required for reaching a node b in a path sequence $a \rightarrow b \rightarrow c$ is *greater* than that required to reach node c in the same path.[2] We normalize such occurrences by setting the time required to reach b to be the average of the times required to reach a and c.
2. *PoP grouping.* Using the cleansed CAIDA data, we group IP addresses into PoPs using a simple *nearness heuristic*: IPs within 2.5 ms of each other, within the same /24 network, and belonging to the same AS are assigned to a single point-of-presence (PoP). These grouping rules preserve the AS paths in our topology and reduce its size significantly while still maintaining meaningful inter-PoP latencies.
   The nearness heuristic may result in multiple "edges" between two PoPs. This occurs when the CAIDA datasets contain traceroute measurements for multiple $(src, sink)$ pairs, where src and $sink$ are IP addresses belonging to the two respective PoPs. To ensure that only one "edge" exists between PoPs in our model, we assign the latency of each PoP-level link to be the median latency over all the $(src, sink)$ links.
3. *Tor relays attachment.* We then identify the Tor relays on the live Tor network whose IP addresses are in the same /24 network as some PoP in our model. We add the matching relays to our model, and mark the corresponding PoPs as a "point of interest" (PoI). The use of PoIs is explained below.
4. *Attachment of clients and destinations.* Prior research [13] has identified popular Tor client and destination ASes. We add clients and destinations to our model at the PoPs that belong to the popular client and destination (resp.) ASes, and mark their PoPs as PoIs. (Note that based on our grouping heuristic, a PoP belongs to exactly one AS.)
5. *Graph pruning and compaction.* To reduce the size of our model and make it practical for experimentation, we prune unimportant nodes and edges. First, we perform All-Pairs-Shortest-Paths over the PoIs (i.e., clients, destinations, and relays) and retain only the nodes and edges that appear on the shortest paths. Conceptually, this removes the portion of the Internet from our model that does not "participate" in the live Tor network. Second, we iteratively replace all segments $a \leftrightarrow b \leftrightarrow c$, *where* b *has degree two*; if $w(a \leftrightarrow b)$ and $w(b \leftrightarrow c)$ are the respective costs of links $a \leftrightarrow b$ and $b \leftrightarrow c$, we remove b from our model and insert a new edge $a \leftrightarrow c$ with cost $w(a \leftrightarrow c) = w(a \leftrightarrow b) + w(b \leftrightarrow c)$.

The resulting model represents a reduced map of the Internet built directly from traceroute data that contains Tor relays, clients, and destinations (see Figure 1). We are able to effectively model 1524 relays in our full topology, which constitutes a large proportion of the Tor network. While we were unable to model the full Tor network (since we lacked the necessary traceroute information), it is worth noting that the 1524 Tor relays in our graph handle 71.3% of all traffic on the live network.

Tor bandwidths and rate limits. To generate a scaled-down topology that is faithful to the bandwidth distribution of the live Tor network, we sample router bandwidths from the live Tor network as follows. We first take a list of all routers in a current Tor consensus and sort the list by the routers' observed bandwidths, as reported in each router's descriptor. We sample routers uniformly from this sorted list to select precisely the desired number of routers.[3]

Since Tor allows router operators to configure rate limits using a token bucket rate-limiting mechanism, we also sample each router's rate-limiting configurations (i.e.,

[1]iPlane [27] provides a similar traceroute dataset; we used CAIDA because, as of this writing, its data was drawn from a wider distribution of sources and destinations.

[2]Such inconsistencies likely occur due to jitter and other transient network effects that take place during successive ICMP echo requests belonging to the same traceroute query.

[3]While we note that this procedure allows us to approximate the bandwidth distribution of the live Tor network, it may slightly underestimate the bandwidths since the live network's measurements are affected by latencies (in addition to relays' actual bandwidths).

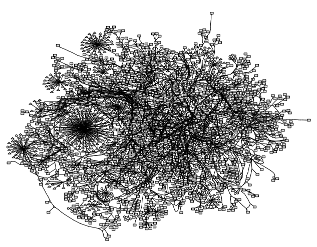


Figure 1. 1524-relay model of the Tor network.

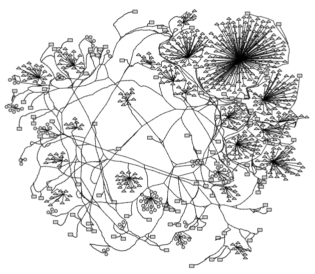


Figure 2. 100-relay model of the Tor network. 50 of the 100 relays are active during experimentation.

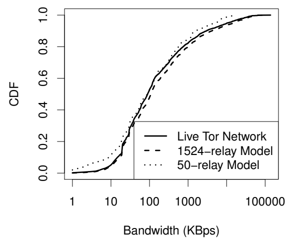


Figure 3. CDF of bandwidths as reported by the actual Tor network, and our 1524- and 50-relay models.

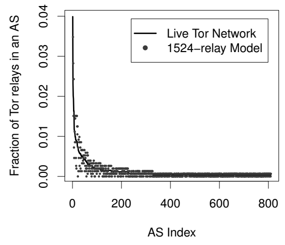


Figure 4. Histogram of relays' AS membership for the live Tor network and our 1524-relay model.

the `BandwidthRate` and `BandwidthBurst` options), which are also advertised within each router's descriptor.

Lastly, it is necessary to configure the directory authorities in the emulated network to advertise the correct bandwidth weights for each router. These weights ensure that clients select routers in the proper proportions. As described in Section 2, the live Tor network uses a set of Bandwidth Authorities to measure and compute these bandwidth weights. In our emulated network, we take a more simple approach: Each router is configured with an estimated bandwidth capacity according to the observed bandwidth value given in its live descriptor. The emulated directories then use these observed bandwidth values to compute a set of bandwidth weights to be used by clients for router selection in our subsequent experiments.

Client and server configurations. We assign unlimited bandwidths to the clients and server PoIs in our models so that they do not create bottlenecks. Although this may be slightly unrealistic, we note that except for very bandwidth-limited clients, performance bottlenecks occur in the Tor network itself, not at the sender or receiver. The "last-mile" latencies for servers and clients are assigned to be the median latency of the links within the PoP they are attached to, if available. If not, the latency is set to 10 ms.

We run a single Tor client for each client PoI within our topology. Each Tor client uses different configuration options depending on the selection strategy being evaluated; however, there are a number of standard configuration options that we apply for our emulation experiments. We disable the use of entry guards in emulation[4] due to the scaled down nature of the evaluation environment, the use of entry guards would impose unrealistic levels of congestion, since all paths would pass through only a small number of guards. Since guards fix the first hop, disabling guards increases the available paths for each selection strategy similarly. Entry guards are enabled in simulation. We also use the `MaxCircuitDirtiness` and `LearnCircuitBuildTimeout` parameters to increase the frequency with which new circuits are requested and to prevent historical data from being used to choose circuits.

Destinations are handled by a single server listening on all designated destination PoI IP addresses.

Routing. We use shortest path routing to compute the latency between any two points on our constructed network graph. Existing work has demonstrated that the Internet generally obeys shortest path routing policies, with some notable exceptions [15]. Using the CAIDA AS Relationship dataset [6], we validate that the resultant routes obey the valley-free property [16], i.e., that routes do not traverse from a provider AS down to a customer AS and back again. This property holds for 80% of the sequences in most routes; for the remaining sequences, no AS relationship data are available in the CAIDA dataset and we are consequently unable to verify whether or not these sequences are valley-free. However, since we found no cases in which our routing heuristic violated the valley-free property using the available data, and all links were constructed using traceroute data (i.e., actual Internet paths), we believe our routes are largely valley-free.

We produce two models using the above techniques: one with 1524 relays and another with 100 relays (Figures 1

[4]As discussed in Section 6.1, each client node in the graph may conceptually represent multiple clients on the live Tor network who share the same AS. Somewhat counterintuitively, enabling the use of entry guards thus *adds* unrealism by modeling the improbable scenario in which the clients who share the same AS and are represented in the graph by a single client node *all* select the identical set of guards.

and 2, respectively).[5] The 1524-relay topology contains every relay that could be mapped from the live Tor network. The 100-relay model was constructed by down-sampling from the 1524 model, while preserving the bandwidth profiles and relay type (i.e., guard, exit, etc.) distributions from the larger model. We use the larger model for simulations of circuit building events (Section 5), and the smaller model in our emulation environment (which requires a more manageable topology size). Our experimental emulation (Section 6) uses 50 of the 100 possible relays to increase the ratio of clients-to-relays and better approximate the performance offered by the live Tor network. As such, we will refer to it as the 50-relay model.

3.2 Verifying our Topology

We next verify our models by demonstrating that they share important characteristics with the live Tor network.

Relay types. Tor biases relay selection in part based on relay type (e.g., guard, exit, etc.). Since relay selection affects both the performance and anonymity properties of Tor circuits, to properly evaluate performance and anonymity, we desire models that reflect the same proportions of relay types as the live Tor network. Table 1 shows that our topologies reflect the numeric distribution of non-exit guards, exit guards, middle relay, and non-guard exits that occur on the live Tor network. We reasonably approximate the bandwidth handled by those classes of relays in our 1524-relay model, but see a modest shift in bandwidth capacity from Exit Guards to Non-Exit Guards in our 50-relay model.

Bandwidth distributions. Figure 3 plots the cumulative distributions of bandwidth capacities for relays in the live Tor network as well as our 1524- and 50-relay models. Applying the two sample Kolmogorov-Smirnov test (a statistical measure for comparing the similarity between two empirical distributions), we find a Kolmogorov-Smirnov (K-S) statistic of 0.050 between the 1524-relay model and the live network, and a K-S statistic of 0.065 between the 50-relay model and the live network. This strongly indicates that the bandwidth distributions of our models closely match that of the live Tor network.

AS distribution. Tor's anonymity is affected by the network's AS topology [13]. An AS that exists both on the ingress path – between a client and the first relay – and the egress path – between the exit relay and the destination – can apply known timing attacks [24] to link the two segments and discover the identities of both the sender and receiver. To accurately assess the anonymity offered by various relay selection policies, our models should therefore exhibit AS distributions that closely match that of the live Tor network.

A histogram of AS memberships for the live network and our 1524-relay topology is shown in Figure 4. For ease of presentation, AS numbers have been replaced with indexes, sorted by the count of constituent relays for the live Tor network. As can be seen from the figure, our model accurately reflects the live Tor network's distribution of ASes. Comparing against the live Tor network, the K-S statistic for the 1524-relay model is 0.046.

While the AS distribution in the 1524-relay model closely resembles that of the live Tor network, the 50-relay model is not particularly representative (here, the K-S statistic is 0.153). This "loss in fidelity" results from the small size of our 50-relay sample, relative to the number of relays on the live network. We discuss this limitation in more detail in Section 3.4. However, we note that our *security results* (in which we investigate how often an AS appears on both a circuit's ingress and egress segments) are based on simulations over the larger 1524-relay topology (Section 5). Our *performance* analyses, which are less dependent on AS topologies, are conducted using emulation over the 50-relay model (Section 6).

Geographic diversity. Figures 5, 6, and 7 respectively show the global distribution of Tor relays for the full Tor network, our 1524-relay model, and our 50-relay model. We use the GeoIP [29] service to map relays to geographic locations based on their IP addresses. Our down-sampled set of 1524-relays maintains similar geographic characteristics to the full set of Tor relays. The 50-relay model used for emulation unavoidably loses some fidelity due to down-sampling, but still retains a diverse geographic distribution that covers sixteen countries.

3.3 Client Behavior and Workloads

To reflect realistic workloads, we model two types of Tor clients. *Interactive* (also called *web*) clients repeat a fetch-sleep cycle where they access content for five minutes and sleep for up to one minute. While in the fetch stage of this cycle, clients request files (i.e., "web pages") between 100 KB and 500 KB in size, which approximates the average web page size (320 KB) as reported by Google [38]. Between each fetch, clients wait for up to 11 seconds to simulate the behavior of someone browsing the web (i.e., they do not click links continuously, but pause to decide where to navigate next).

In contrast, *bulk* clients download continuously, and request files between 1 MB and 5 MB in size. Bulk clients roughly approximate the behavior of file sharers on the Tor network. To match existing studies [30] of behavior on the live Tor network, 3% of the clients are configured to be bulk; the remaining 97% are interactive.

[5]Our topologies are available in GraphML format for download at https://security.cs.georgetown.edu/lib/ndss2013/topos.tar.gz.

Network/Model	Relays	Non-Exit Guards	Exit Guards	Middle	Non-Guard Exits
Live Tor network	2642	579 [22%, 40%]	239 [9%, 31%]	1208 [46%, 18%]	616 [23%, 10%]
1524-relay model	1524	389 [26%, 43%]	154 [10%, 31%]	650 [43%, 18%]	332 [22%, 9%]
50-relay model	50	13 [26%, 64%]	4 [8%, 17%]	22 [44%, 15%]	11 [22%, 4%]

Table 1. Distribution of relays in the live Tor network and our 1524- and 50-relay models, by count. The percentage of the network by count and the percentage of the network by bandwidth are respectively indicated in brackets.

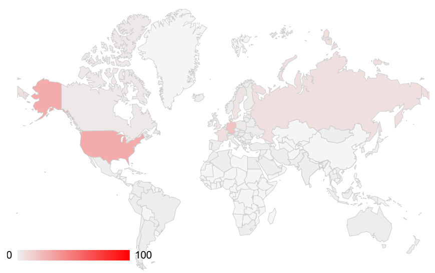


Figure 5. Geographic distribution of Tor relays in the full Tor network.

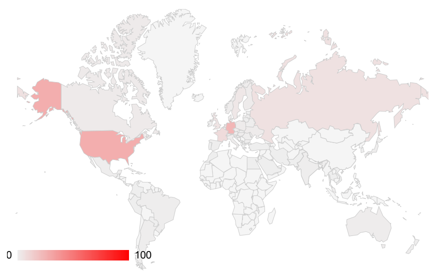


Figure 6. Geographic distribution of Tor relays in our 1524-relay topology.

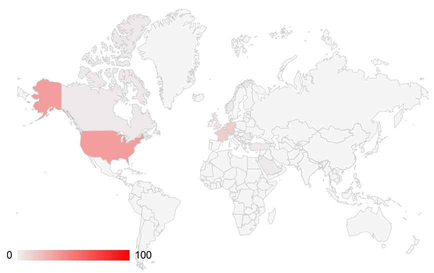


Figure 7. Geographic distribution of the 50 relays we select for emulation.

To create workloads that capture the latency of Tor connections, each client additionally runs a low-bandwidth "echo" client that sends a single Tor cell once a second through the Tor network.

Our models also include *destination* nodes, which are the targets of anonymous communication. They serve HTTP requests and respond to "echo" messages.

3.4 Limitations

As described above, our goal was to construct models that accurately represent the live Tor network's bandwidth, relay type, geographic, and AS distributions. However, due to the inherent loss of fidelity due to down-sampling and the inability to perfectly represent client behavior, our technique has some limitations:

Client behavior. By design, Tor makes it difficult to capture the behavior of the network's users. Existing studies [13, 30] of client behavior rely on sampled data from specially instrumented Tor guard and exit relays that record usage statistics. We utilize the results of these studies to place clients and destinations in our topology (see Section 3.1). Unfortunately, these studies are becoming somewhat dated [13, 30]. We chose not to repeat the experiments described in the studies due to privacy concerns; as others have noted [44], recording client behavior on the live Tor network runs contrary to the network's principles and has the potential to put the network's users at risk. Although our datasets may not perfectly match current behavior, our placement of clients conforms to high-level statistics reported by the Tor Metrics Portal [47].

Scaled-down emulation. To maximize realism, we use the ExperimenTor [4] emulator in which unmodified Tor binaries communicate over a virtual network topology. However, the ability to scale our emulation is limited by our CPU and bandwidth capacities. Since we cannot emulate the hundreds of thousands of users who are estimated to use Tor [25], we instead opt to capture the level of congestion that occurs on the live Tor network. To do this, we adjust the number of Tor clients, and tune their behavior by changing how often they request pages. In doing so, we are able to approximate the performance characteristics of the Tor network with a reduced number of clients (see Section 6).

4. Relay Selection

Using our constructed models, we evaluate the performance and anonymity of various relay selection strategies *under realistic network conditions*. In what follows, we enumerate existing and novel relay selection algorithms (Section 4.1), describe how we integrate the relay selection techniques into Tor (Section 4.2), and present metrics for measuring anonymity and performance (Section 4.3).

4.1 Relay Selection Algorithms

We consider the following relay selection algorithms:

- **Tor.** Conceptually, Tor's relay selection algorithm weighs relays proportionally according to their bandwidth [10, 11]. Murdoch and Watson have found that such a strategy offers good load balancing properties while providing reasonable anonymity [32]. In practice, however, Tor uses a slightly more complex weighting strategy that de-emphasizes unstable and/or new relays in favor of more longstanding routers. Additionally, Tor biases against selecting guard relays except as entry points, and exit relays except at egress locations.

- **Snader/Borisov.** Snader and Borisov [43] propose a refinement to Tor's algorithm that allows the sender to "tune" the degree to which selection is biased in favor of bandwidth. They introduce a family of functions

$$f_s(x) = \begin{cases} \frac{1-2^{sx}}{1-2^s} & \text{if } s \neq 0 \\ x & \text{if } s = 0 \end{cases}$$

  where s is a parameter that trades off between anonymity (selecting relays uniformly at random) and performance (biasing more heavily in favor of bandwidth). Given a list of relays sorted by their bandwidths, the Snader/Borisov (SB) algorithm selects the relay at index $\lfloor n \cdot f_s(x) \rfloor$, where x is chosen uniformly at random from $[0, 1)$ and n is the number of relays. In the remainder of this paper, we denote the SB strategy with some fixed value of s as *SB-s*.

- **Unweighted Tor.** As a point of comparison, we include an *Unweighted Tor* selection strategy where clients build paths by choosing relays uniformly at random without replacement from the set of available relays provided by Tor. Clients using *Unweighted Tor* will only choose paths terminating at relays with accepting exit policies; they also are subject to any other constraints imposed by Tor.

- **Coordinate.** Sherr *et al.* [41, 42] propose latency-aware *link-based* relay selection strategies. In their approach, relays participate in a virtual coordinate embedding system [8]. (To avoid potential anonymity attacks, neither clients nor destinations participate in the coordinate system.) The Euclidean distance between any two relays' virtual coordinates serves as an indicator of the latency between the pair. By summing the virtual distances between relays' advertised coordinates, clients can estimate the latencies of potential anonymous circuits *before they are instantiated.*

  We implement two variants of coordinate-based routing. In the *Coordinates* strategy, clients select — but do not instantiate — k candidate paths where the relays in each path are selected using the *Unweighted Tor* methodology. Clients compute the expected latency of each of their k candidates paths, and select the path with the lowest estimated latency.

  We also introduce a hybrid *Tor+Coordinates* strategy. Here, clients select k candidate paths *using Tor's default bandwidth-weighted relay selection strategy*. Clients then compute the expected latencies of the k candidate paths and instantiate the path with the lowest expected latency.

  An evaluation of several potential values showed that setting $k = 3$ offered the best trade-off between increased performance and the time spent identifying the best path.

- **LASTor.** The recently proposed *LASTor* [2] system selects relays in a manner that (1) reduces the probability that an autonomous system will appear on both sides of the anonymous circuit and (2) reduces path latencies by using geographic distance as an estimate for latency. (*LASTor* uses the GeoIP service to map network addresses to physical locations.) All possible candidate paths between a client and a destination are weighted based on their great circle distance (i.e., the distance measured over a spherical representation of Earth), and a path is selected that seeks to minimize that weight. To make this computationally tractable, relays are clustered into gridsquares based on latitude and longitude, and paths are calculated through these gridsquares. In addition, *LASTor* makes use of iPlane datasets [26, 27] to avoid selecting paths where there exists an AS at both ends that could correlate traffic across the anonymous path.

- **Congestion-aware selection.** This technique, recently proposed by Wang *et al.* [49], seeks to intelligently select Tor circuits with the lowest levels of congestion. Congestion measurements for a given circuit are obtained by opportunistically sampling roundtrip times across that circuit and subtracting the lowest recorded roundtrip time. Both circuit building events and application connections are used to measure circuits with little additional overhead.

  Based on these congestion measurements, Wang *et al.* propose two *immediate* and one *long-term* path selection techniques. We applied the two immediate techniques together but omit the long-term algorithm entirely, as it was found to have negligible impact by the paper's authors [49]. The immediate techniques are as follows: (a) when choosing a circuit to use, randomly choose three of the available circuits, then select the one with the lowest measured congestion time, and

(b) if at any point, the mean of the last five measured congestion times on a given circuit is more than 0.5 seconds, switch to another circuit.

4.2 Integrating Selection Algorithms into Tor

We implement the described selection algorithms within Tor version 0.2.3.0-alpha. the set of For the *Coordinates* and *Tor+Coordinates* algorithms, we implement additional Tor cell types to support `ping` messages between Tor instances. Ping targets are selected uniformly at random from the list of running relays once every three seconds. A TLS connection is established with that target, and ping requests and responses are exchanged. The initiating relay uses the minimum ping response received to update its coordinate using the distributed Vivaldi algorithm [8]. We also modify Tor to include coordinate information in relay descriptors, enabling clients to collect the necessary information to estimate the latencies of potential circuits.

Our implementations of the *LASTor* and *Congestion-aware* protocols use the Python TorCtl controller interface to select and instantiate paths according to the specifications outlined by Akhoondi *et al.* [2] and Wang *et al.* [49], respectively. For *LASTor*, we statically designate the latitude and longitude of our 50-emulated relays based on their real-world locations, obtained through IP-geolocation with the GeoIP City database, resulting in 38 geographic clusters. We do not implement the AS avoidance portion of *LASTor* since it relies upon iPlane Nano data [26] for BGP routing policies which may not map accurately to the routing on our experimental topology. Akhoondi *et al.* [2] showed that AS awareness *increased* the latency of selected paths, and hence we expect our *LASTor* performance results to be slightly optimistic.

Our *Congestion-aware* implementation obtains opportunistic measurements from three sources: the time taken to extend the circuit to the third relay; the time taken for application connection requests and acknowledgment; and a special `PINGED` cell sent once to measure the roundtrip time immediately after circuit construction.

4.3 Metrics

Our goal is to understand the implications of running the above relay selection strategies on the actual Tor network.

Our *performance* metrics are: **throughput**; **time-to-first-byte** (TTFB) (the time required for clients to fetch the first byte of a document); and **average ping time** (P-RTT) (the median roundtrip-time of sixty 100-byte pings).

Our *anonymity* metrics include: (1) the fraction of instantiated anonymous paths in which the same AS appears on both sides of the path, as proposed by Edman and Syverson [13]; (2) the Shannon entropy over the distribution of relays in the entry and exit positions of paths [9, 40]; and (3) the Gini coefficient over those relays, as proposed by Snader and Borisov [43]. The Gini coefficient is a measure of equality (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., only one router is always chosen). Note that because each path requires that a distinct entry and exit relay be selected, a Gini coefficient of 1 is effectively impossible to attain; the ceiling is 0.98 for our 50 relay emulation environment.

We compute Shannon entropy and the Gini coefficient over only the first and last relays in a given path. Tor instantiates paths containing three relays; the middle relay communicates only with the entry and exit relays using TLS encryption. An adversary observing the middle relay obtains little information of value, while one who observes both the entry and exit does not need to see the middle relay to break anonymity. Thus the distribution of entries and exits is most critical to anonymity. We desire relay selection strategies that produce high entropy and low Gini coefficients, as this prevents a subset of relays from observing a disproportionate amount of the network's traffic.

Using our models of the live Tor network, we next evaluate the anonymity (Section 5) and performance (Section 6) properties of proposed relay selection strategies.

5. Simulation-Based Anonymity Analysis

We simulate path selection on our 1524-relay model of the live Tor network. The simulator is based on actual Tor code (version 0.2.2.33) and uses Tor's relay selection functions. Our simulator implements only Tor's relay selection logic and does not simulate the actual construction of paths, the transmission of data, or network effects such as congestion.[6] The performance of various relay selection policies, which is heavily dependent on network effects, is studied under full-network emulation in Section 6. Here, we focus our simulation experiments on measuring the AS diversity of paths as well as the distribution of selected relays.

We modify Tor's relay selection logic to support the *SB-s* and *LASTor* strategies. Since we do not simulate network conditions, we do not consider the *Coordinates*, *Tor+Coordinates*, or *Congestion-aware* strategies.

For each tested strategy, we simulate 5 million paths. Using the client and destination AS distributions reported by Edman and Syverson [13], we assign clients and destinations to each AS in the distribution, for each of the 5 M paths. We then check whether the same AS appears both on the path from the client to the guard relay as well as on the

[6]Our simulator is based on the framework described by Elahi *et al.* [14].

path from the exit relay to the destination.If so, we weigh the result by the probability that this client and destination pair would be chosen (again, using Edman and Syverson's AS distribution). Effectively, this method yields the percentage of vulnerable paths, assuming clients and destinations are distributed as they were in Edman and Syverson's study. As discussed in 4.2, our *LASTor* implementation does not include its AS avoidance strategy, resulting in a higher percentage of vulnerable paths than is likely to occur in a deployed implementation. The entropy and Gini coefficient results for *LASTor* are unaffected.

Table 2 shows the percentage of vulnerable paths for the various relay selection strategies, as well as the Gini coefficient and entropy over the distribution of selected relays. These metrics should not be taken as direct indicators of the strength of a particular anonymity technique, but rather as a mechanism for comparing the security properties of different strategies. The *Unweighted Tor* strategy offers the smallest percentage of vulnerable paths. This is unsurprising, since randomly selecting relays increases the diversity of paths. For the *Snader-Borisov* paths, increasing the value of s (i.e., biasing more heavily in favor of performance) increases the percentage of vulnerable paths, as the distribution of selected relays becomes less uniform. This effect is best captured by the increase in the Gini coefficient (indicating increasingly uneven distributions) as s increases.

Overall, in all cases, the relay selection strategy did not significantly increase the percentage of vulnerable paths. However, we note that the prevalence of a small number of ASes on both sides of the anonymous circuits cause approximately one quarter of the circuits to be vulnerable. This is slightly higher than the value reported in Edman and Syverson's study [13] (approximately 18% for Tor's default strategy). The potential increase in vulnerability may be due to topological changes on the Internet since their study was conducted in 2009. Additionally, we note that while our Internet model is based on empirical traceroute data, Edman and Syverson estimate AS paths using Qiu's inference algorithm [37] applied to RouteViews [1] data. In Figure 8, we report the ASes that most commonly occurred on either side of a path, and the rate at which that occurred for each selection strategy.

Relay selection strategy	% vulnerable paths	Gini coef.	Entropy
Unweighted Tor	24.34	0.530	9.65
Tor	27.39	0.891	7.68
SB-3	24.66	0.662	9.32
SB-6	24.99	0.776	8.53
SB-9	26.01	0.841	7.58
SB-12	26.84	0.878	6.68
SB-15	27.42	0.900	5.95
LASTor	24.94	0.644	9.38

Table 2. Percentage of vulnerable paths, Shannon entropy and Gini coefficient for various relay selection strategies, under simulation using the 1524-relay model.

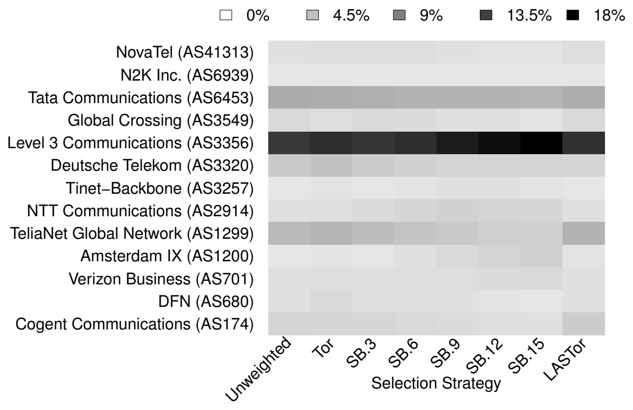


Figure 8. The ASes that most often appeared on both sides of anonymous paths, for each selection strategy. Shading indicates the percentage of paths on which the AS appeared.

6. Full-Network Emulation Study

To measure the performance of relay selection, we run a modified version of Tor in an emulated network. We use ExperimenTor [4], executing all Tor instances on a 12-core 2.8 GHz Xeon X5660 machine with 64 GB of memory, running Ubuntu 11.10 with the 2.6.38 Linux kernel. ExperimenTor's ModelNet [48] virtual network backend runs inside of a FreeBSD 6.3 virtual machine connected to the host emulator with a direct 10 GbE link. Our experimental configuration uses the 50-relay model described in Section 3. We carefully monitor the CPU, memory, and network utilization of both machines to ensure that resource constraints do not introduce any artifacts into our experiments.

We run each experiment for 2.5 hours to allow the system to stabilize and record results only from the last 90 minutes of each experiment. This allows the coordinates to stabilize for the *Coordinates* and *Tor+Coordinates* strategies, and the bandwidth weightings to properly adjust for *SB-s* and *Tor*.

6.1 Establishing Traffic Levels

As discussed in Section 3.4, establishing an appropriate level of traffic is difficult when modeling the Tor network. Our goal is to select a level of congestion that is on par with that found on the live Tor network.

We alter the number of clients to tune the level of congestion in our emulated Tor network. To select a level of congestion that matches Tor, we compare the throughput

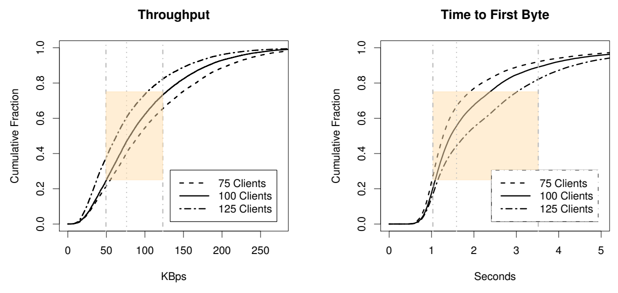


Figure 9. Cumulative distribution of throughput (left) and time-to-first-byte (right) of paths in our emulated environment with 75, 100, and 125 clients. The yellow highlighted boxes depict the interquartile ranges of performance of the live Tor network, as reported by the Tor Metrics Portal. To match the live network, the performance curve should intersect the lower-left and upper-right corners of the highlighted region, and intersect the dashed vertical line (the median of the live Tor network) when the cumulative fraction is 0.5.

and time-to-first-byte experienced in our emulation to performance data collected from the live network in March 2012 [47]. As shown in Figure 9, with 100 active clients (three of which are *bulk* clients), our emulated throughput matches that of the Tor network almost exactly, while our time-to-first-byte is 32% faster at the third quartile, but only 6% faster at the median and 5% slower at the first quartile.

While our use of 100 active clients approximates the performance of the current live Tor network, we also evaluate relay selection strategies under both less and more severe congestion conditions. Specifically, we emulate 25 and 175 clients in the "low" and "high" congestion configurations, respectively, to evaluate performance for possible future conditions on the Tor network.

6.2 Performance Results

Homogeneous networks. We first consider a homogeneous network in which all clients adopt identical relay selection strategies. Figure 10 shows the cumulative distribution of measured client throughput, time-to-first-byte, and P-RTT for a network experiencing a medium level of congestion. (For readability, we adopt the convention of listing the labels in the figures' keys in order of the corresponding curves' median values, while maintaining the same line types between figures.) There is a distinct difference in performance between the selection strategies that use bandwidth to influence their selection and those that do not. The *Tor+Coordinates* and *Congestion-aware* strategies achieve a median throughput of 85 KBps, outperforming all other selection strategies, although *Tor* (81 KBps) is only 4.7% worse. Strategies that apply either little or no weight to bandwidth perform poorly: *Unweighted Tor* nets a median throughput of just 22 KBps, while *LASTor* has a median throughput of 24 KBps.

The latency metrics follow a similar trend. Here *Congestion-aware* performs the best with a median time-to-first-byte of 1.321 seconds, 8.5% better than *Tor*, and 10.7% better than *SB-9*. Notably, while *LASTor* remains poor at time-to-first-byte, it actually performs reasonably in P-RTT; its median P-RTT of 1.41 seconds is only 12% worse than *SB-9* and much better than that of *Unweighted Tor*, *Coordinates* and *SB-3*. This is unusual considering both metrics measure latency. We suggest that the difference is due to a distinction that strongly affects *LASTor*: time-to-first-byte incorporates TCP connect time, while P-RTT does not begin measuring until a connection is established. Since *LASTor* optimizes the latency across the full path between client and destination, the destination must be known *before a Tor circuit can be established*. Time-to-first-byte captures the time required for *LASTor* to pick and extend an appropriate path each time a new application request is received. By contrast, Tor normally maintains a set of established circuits and simply routes new traffic over one of them.

In Figures 11 and 12, we explore the performance of the relay selection strategies in networks with resp. high and low congestion. In the highly congested environment, throughput and time-to-first-byte suffer across the board. *Tor*, *SB-9* and *Tor+Coordinates* perform similarly, with median throughput of 46 KBps, 47 KBps, and 44 KBps respectively. Somewhat surprisingly, *Coordinates* also does reasonably well, with a median throughput of 43 KBps. The other less bandwidth-focused strategies produce median throughput of less than 33 KBps. *Congestion-aware*, by virtue of its focus on avoiding congestion, performs the best by a considerable margin with a median throughput of 53.8 KBps.

The *Congestion-aware* strategy continues to be effective when performance is measured in time-to-first-byte. *Congestion-aware* outperforms all other strategies with a median time-to-first-byte of 1.87 seconds, 14%, 21%, and 27% faster than *SB-9*, *Tor*, and *Tor+Coordinates* respectively. Similarly, *LASTor* continues to have impressive P-RTT times (although not throughput or time-to-first-byte): under high congestion, its 1.31 seconds are the second lowest at the median, behind only *Congestion-aware*.

In a low-congestion environment (Figure 12), throughput is considerably higher, and time-to-first-byte lower, due to reduced traffic and congestion. Bandwidth remains an important factor in path selection, with a clear delineation between strategies that weight heavily for bandwidth and

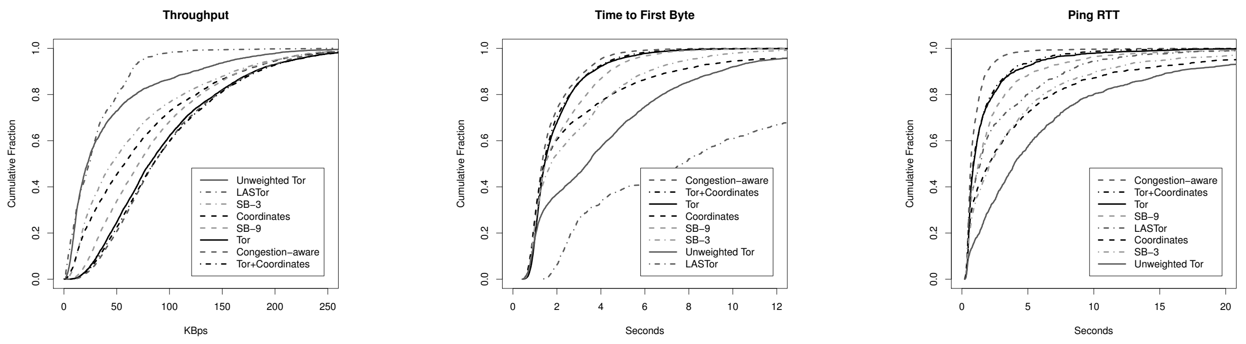


Figure 10. Cumulative distribution of measured client throughput (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with medium congestion.

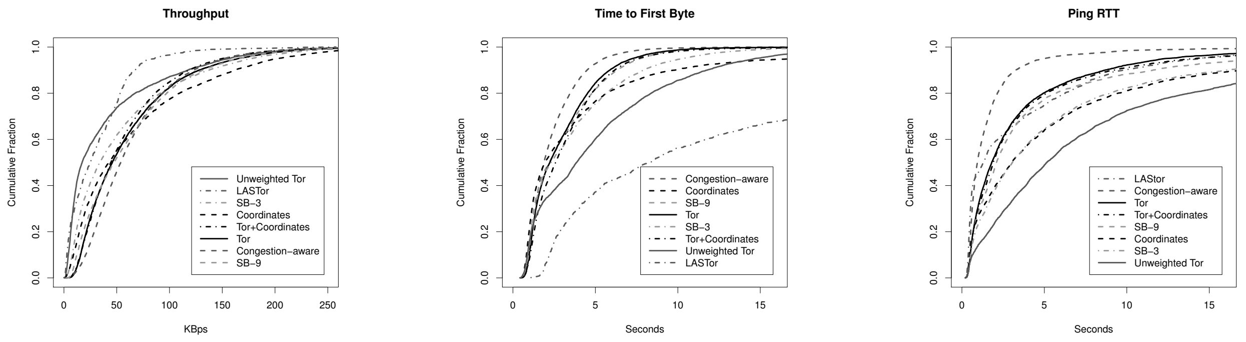


Figure 11. Cumulative distribution of measured client throughput (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with high congestion.

those that do not. The former category is led by *SB-9* which exhibits 45% better throughput than *Tor*. With the lower level of congestion, our *Tor+Coordinates* selection strategy is also able to improve upon *Tor* by 22%. Similar to the results for medium and high congestion networks, strategies that do not focus heavily on bandwidth do not perform particularly well.

SB-9 and *Tor+Coordinates* have the lowest time-to-first-byte, both with median times under one second. At their medians, they respectively perform 14% and 17% better than default *Tor*.

We see that even under low congestion, *Unweighted Tor* and *LASTor* do not appear to be effective at selecting paths through Tor, and experience the worst throughput and time-to-first-byte. It should be noted that *LASTor* again performs reasonably when performance is measured in terms of P-RTT. *Congestion-aware* continues to perform well but with less distinction, likely due to low levels of congestion.

Heterogeneous networks. We also explore the effects of relay selection when some clients elect to run a different relay selection strategy. This scenario is likely in incremental deployments, and additionally models *application-tunable anonymity* [42] in which clients select a relay selection policy to meet their underlying applications' communication requirements. When evaluating heterogeneous selection, 20% of clients use a specific selection method, while the remaining 80% use *Tor*. We present the median, 10th-, and 90th-percentiles of throughput, time-to-first-byte, and P-RTT for this heterogeneous selection under medium congestion in Table 3. We also show (1) the performance of the 80% of the clients that use the vanilla Tor client, and (2) the percentage improvement of the non-Tor strategy over *Tor*.

SB-9, *Tor+Coordinates*, and *Congestion-aware* respectively provide performance improvements between 9% and 12% in throughput and time-to-first-byte, while other selection strategies generally under-perform compared to *Tor*.

Our results indicate that even in a heterogeneous environment, a selection strategy that does not use bandwidth weighting (e.g., *Coordinates*) performs poorly relative to the majority of clients who use *Tor*. Even small numbers

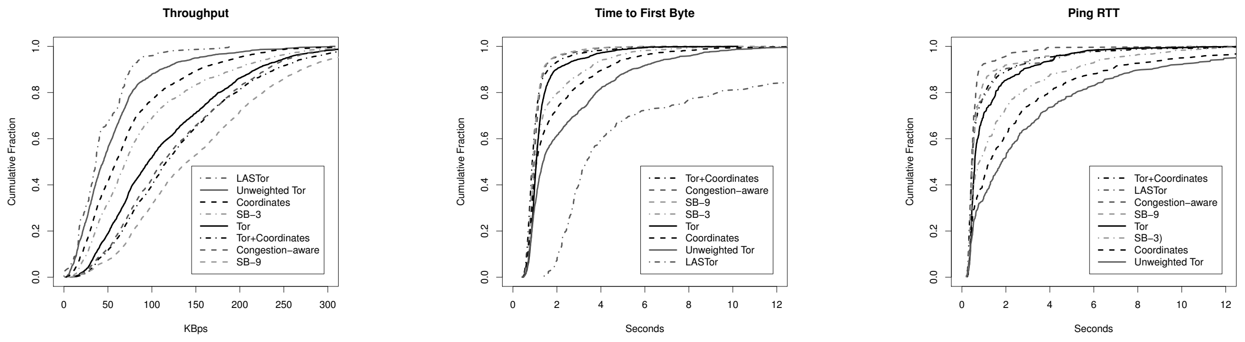


Figure 12. Cumulative distribution of measured client bandwidth (left), time-to-first-byte (center), and average ping time (right) for various relay selection policies in a network with low congestion.

Relay selection strategy	Gini coef.	Entropy
SB-9	0.79	3.63
Tor+Coordinates	0.77	3.87
Tor	0.71	4.21
Congestion-aware	0.61	4.11
SB-3	0.63	4.57
Coordinates	0.56	4.76
Unweighted Tor	0.53	4.80
LASTor	0.50	5.00

Table 4. Gini coefficient and entropy for various relay selection strategies, under emulation.

of clients using specialized strategies must weight for bandwidth to obtain reasonable performance.

6.3 Security Analysis

The entropy and Gini coefficients for the various relay selection strategies in the medium congested network are presented in Table 4. *Tor+Coordinates*, *SB-9*, and *Tor* have the highest Gini coefficients and the lowest entropy, indicating that the set of relays used by those selection strategies is smaller and less diverse than that used by *Unweighted Tor*, *SB-3*, or *Coordinates*. There is a strong correlation between better performance and a more selective relay selection strategy, confirming that within the context of anonymity systems, performance is a commodity that requires a trade-off with anonymity. *Congestion-aware* impressively shows high performance while giving up less anonymity than *Tor*.

Comparing Tables 2 and 4, we find that our results generally agree between our simulation and emulation experiments. The main outlier is *LASTor*, which exhibits an entropy greater than, and Gini coefficient less than, *Unweighted Tor*. We believe that this is due to *LASTor*'s entry node distribution, where the entry node is the first hop in a path (not necessarily a guard, see Section 3.1), which when added to its exit node distribution, results in a more entropic overall relay selection distribution.

6.4 Summary

Our emulation results demonstrate that congestion aware routing offers an improvement in anonymity and performance, especially under heavily congested conditions. *Tor+Coordinates* also shows potential for improvement over Tor's standard relay selection, posting modest benefits in throughput over Tor's default bandwidth-weighted strategy.

7. Discussion

One clear result of our performance evaluation is the critical importance of bandwidth to any effective relay selection strategy. The live Tor network is heavily oversubscribed and most network performance characteristics become irrelevant when bandwidth is the constraining factor. Our results show that strategies that weight heavily for bandwidth perform better than those that weight only lightly, and much better than those that do not do so at all. In particular, under our medium congestion level, the median throughput achieved by *Tor*, *Congestion-aware*, *SB-9* and *Tor+Coordinates* were all at least 70 KBps, while *Unweighted Tor* and *LASTor* — the two strategies that ignore bandwidth — both produced median throughput of less than 25 KBps.

One outlier is the *Coordinates* strategy, which often achieved throughput similar to *SB-3* without weighting on bandwidth. A likely explanation is that there is an inherent correlation between bandwidth and latency *when empirically measured*. While the cost of pings in the coordinate system is not large, a low-bandwidth relay will be slower to respond than a high-bandwidth one. Additionally, there

		throughput (KBps)		time-to-first-byte (seconds)		P-RTT (milliseconds)	
SB-9	*SB-9*	83.96 [30.19, 191.05]	**11.0%**	1.27 [0.83, 3.70]	**12.4%**	894 [401, 4152]	**9.7%**
	Default *Tor*	75.7 [29.77, 176.05]		1.45 [0.88, 3.95]		981 [427, 4423]	
SB-3	*SB-3*	46.66 [16.02, 133.93]	**-36.4%**	1.80 [0.87, 5.38]	**-33.3%**	1728 [436, 7821]	**-55.7%**
	Default *Tor*	73.34 [24.75, 177.20]		1.35 [0.86, 4.06]		1110 [422, 6063]	
Unweighted Tor	*Unweighted Tor*	29.21 [11.69, 75.43]	**-66.1%**	2.80 [0.95, 7.13]	**-101.4%**	2963 [536, 10235]	**-235.8%**
	Default *Tor*	86.13 [27.69, 197.43]		1.39 [0.89, 3.88]		885 [441, 4730]	
Tor+Coordinates	*Tor+Coordinates*	83.92 [32.84, 188.63]	**10.6%**	1.32 [0.87, 4.06]	**12.6%**	797 [396, 3655]	**17.4%**
	Default *Tor*	75.81 [29.80, 173.31]		1.51 [0.91, 3.84]		965 [436, 3973]	
Coordinates	*Coordinates*	61.53 [19.34, 148.37]	**-28.6%**	1.41 [0.85, 5.99]	**-2.9%**	1723 [422, 8051]	**-88.7%**
	Default *Tor*	86.15 [31.68, 191.12]		1.37 [0.89, 3.62]		913 [437, 4365]	
LASTor	*LASTor*	28.26 [8.11, 56.38]	**-70.0%**	6.78 [2.36, 63.91]	**-425.6%**	1409 [435, 6440]	**-82.3%**
	Default *Tor*	94.43 [35.20, 205.96]		1.29 [0.86, 3.03]		773 [418, 3840]	
Congestion-aware	*Congestion-aware*	86.38 [33.66, 186.10]	**9.5%**	1.31 [0.86, 3.11]	**10.3%**	648 [395, 1884]	**32.1%**
	Default *Tor*	78.86 [30.37, 176.99]		1.46 [0.90, 3.88]		954 [439, 6410]	
Homogeneous *Tor*	*Tor*	81.17 [31.51, 180.55]		1.44 [0.90, 3.64]		901 [447, 4009]	

Table 3. Performance metrics at the "median [10th percentile, 90th percentile]" for various relay selection strategies applied heterogeneously under medium congestion. Also shown in bold is the percentage improvement relative to default *Tor* at the median for each strategy and metric. Note that improvement means higher numbers for throughput, and lower numbers for time-to-first-byte and P-RTT.

is an indirect relationship between latency and bandwidth when using TCP (as Tor does). Thus *Coordinates* actually incorporates an (admittedly loose) proxy for bandwidth in its selection.

Our results also indicate that the recently proposed *LASTor* relay selection strategy is unlikely to be effective on the live Tor network. It is likely that the evaluation method used by Akhoondi *et al.* of making HTTP HEAD requests over *LASTor* paths did not realistically exercise the performance of those paths: a HEAD request is generally less than 1 KB, while typical web pages are two orders of magnitude larger [38]. Since *LASTor* does not take the bandwidth of relays into account (we discuss proposed refinements to *LASTor* that consider bandwidth next), its performance degrades drastically when a realistic traffic load is applied. We note that our clean room implementation of *LASTor* does not include the AS awareness portion of the algorithm, although Akhoondi *et al.* showed that AS awareness increased the latency of instantiated paths so we believe it unlikely that its inclusion would have *improved* performance.

Akhoondi *et al.* suggest refinements to their basic *LASTor* strategy that omit relays whose bandwidth is less than 100 KBps from consideration. We evaluated this refinement, and showed that while this provides a significant improvement in performance, *LASTor* still lags other strategies. We present the details of this evaluation in the Appendix.

Finally, our results suggest that the addition of a coordinate-based system to Tor may provide some incremental advantage to relay selection. *Tor+Coordinates* obtained a slight performance improvement over *Tor* under most conditions, and rarely performed worse. Similarly the performance of the *Congestion-aware* strategy suggests that similar techniques provide a window of opportunity for incrementally improving Tor. While bandwidth remains the most significant indicator of performance, a layered approach which seeks to optimize latency and congestion as well as throughput might be a beneficial addition to Tor. Since these techniques are largely orthogonal, future work could explore the benefits of further combination. Additional areas of future work include the development of defenses to protect coordinate systems from manipulation.

8. Conclusion

We have shown the feasibility of building a topology from Internet routing information that reasonably represents the live Tor network and is reducible to a size viable for emulation and other environments. Our models of the Tor network enable whole-network comparative evaluation of a set of relay selection methods. Our results indicate that recently proposed algorithms that do not consider bandwidth during relay selection result in poor performance. The *Congestion-aware* strategy shows significant promise, especially in congested networks, and has little anonymity impact. Our *Tor+Coordinates* selection method also offers a modest improvement in both bandwidth and latency over Tor's default strategy. Additionally, we show the importance of full network evaluation when considering new relay selection strategies, as performance in toto can differ significantly from the experience of a single client.

We anticipate that as emulation frameworks such as Shadow and ExperimenTor mature and begin to support larger topologies, the limitations of the 50-relay model can be ameliorated by using larger models (such as our 1524-relay model) for performance evaluation.

Acknowledgments

We would like to thank Harsha Madhyastha for helping us obtain Internet measurement data, and Aaron Johnson, Rob Jansen, and Paul Syverson for their insights and helpful discussions regarding this paper. We are also grateful to Matthew Edman and Paul Syverson for providing us with additional AS data.

This work is partially supported by NFS CAREER CNS-1149832. This material is based upon work supported by the Defense Advanced Research Project Agency (DARPA) and Space and Naval Warfare Systems Center Pacific under Contract No. N66001-11-C-4020. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Project Agency and Space and Naval Warfare Systems Center Pacific.

References

[1] Advanced Network Technology Center. University of Oregon Route Views Project. `http://www.routeviews.org/`.

[2] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE Symposium on Security and Privacy (Oakland)*, 2012.

[3] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker. DefenestraTor: Throwing out Windows in Tor. In *Privacy Enhancing Technologies Symposium (PETS)*, 2011.

[4] K. Bauer, M. Sherr, D. McCoy, and D. Grunwald. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2011.

[5] 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 (CCS)*, 2007.

[6] The CAIDA AS Relationships Dataset, Jan, 2009. `http://www.caida.org/data/active/as-relationships/`.

[7] The CAIDA UCSD IPv4 Routed /24 Topology Dataset - Jan 7-9, 2012. `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. *SIGCOMM Comput. Commun. Rev.*, 34(4):15–26, 2004.

[9] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[10] R. Dingledine and N. Mathewson. Tor Path Specification. `http://www.torproject.org/svn/trunk/doc/spec/path-spec.txt`, January 2008.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, 2004.

[12] R. Dingledine and S. Murdoch. Performance Improvements on Tor, or, Why Tor is Slow and What We're Going to Do About It. `https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf`, March 2009.

[13] M. Edman and P. Syverson. AS-Awareness in Tor Path Selection. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[14] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*. ACM, October 2012.

[15] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.

[16] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6):733–745, 2001.

[17] D. Goldschlag, M. Reed, and P. Syverson. Hiding Routing Information. In *Workshop on Information Hiding (IH)*, 1996.

[18] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency Between Arbitrary Internet End Hosts. In *ACM SIGCOMM Workshop on Internet Measurment (IMW)*, 2002.

[19] S. Hahn and K. Loesing. Privacy-preserving Ways to Estimate the Number of Tor Users, November 2010. Available at `https://metrics.torproject.org/papers/countingusers-2010-11-30.pdf`.

[20] R. Jansen, K. Bauer, N. Hopper, and R. Dingledine. Methodically Modeling the Tor Network. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2012.

[21] R. Jansen and N. Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[22] R. Jansen, N. Hopper, and Y. Kim. Recruiting New Tor Relays with BRAIDS. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[23] A. M. Johnson, P. Syverson, R. Dingledine, and N. Mathewson. Trust-based Anonymous Communication: Adversary Models and Routing Algorithms. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[24] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright. Timing Attacks in Low-latency Mix Systems. In *Financial Cryptography*, 2004.

[25] K. Loesing. Measuring the Tor Network: Evaluation of Client Requests to the Directories. Technical report, Tor Project, June 2009.

[26] H. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[27] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2006.

[28] N. Mathewson. Evaluating SCTP for Tor. `http://archives.seul.org/or/dev/Sep-2004/msg00002.html`, September 2004. Listserv posting.

[29] MaxMind. GeoIP. `http://www.maxmind.com/app/ip-location`.

[30] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[31] B. Moore, C. Wacek, and M. Sherr. Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race With Tortoise. In *Annual Computer Security Applications Conference (ACSAC)*, December 2011.

[32] S. J. Murdoch and R. N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[33] T.-W. J. Ngan, R. Dingledine, and D. Wallach. Building Incentives into Tor. In *Financial Cryptography and Data Security*, 2010.

[34] G. O'Gorman and S. Blott. Large Scale Simulation of Tor: Modelling a Global Passive Adversary. In *Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security (ASIAN)*, 2007.

[35] L. Øverlier and P. Syverson. Locating Hidden Servers. In *IEEE Symposium on Security and Privacy (Oakland)*, 2006.

[36] M. Perry. Torflow: Tor network analysis. HotPETS, 2009.

[37] J. Qiu and L. Gao. AS Path Inference by Exploiting Known AS Paths. In *IEEE Global Communications Conference (GLOBECOM)*, 2006.

[38] S. Ramachandran. Web Metrics: Size and Number of Resources, May 2010. Blog post, available at `https://developers.google.com/speed/articles/web-metrics`. Retrieved May 1, 2012.

[39] J. Reardon and I. Goldberg. Improving Tor using a TCP-over-DTLS Tunnel. In *USENIX Security Symposium (USENIX)*, 2009.

[40] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[41] M. Sherr, M. Blaze, and B. T. Loo. Scalable Link-Based Relay Selection for Anonymous Routing. In *Privacy Enhancing Technologies Symposium (PETS)*, August 2009.

[42] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. T. Loo, and M. Blaze. A3: An Extensible Platform for Application-Aware Anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2010.

[43] R. Snader and N. Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

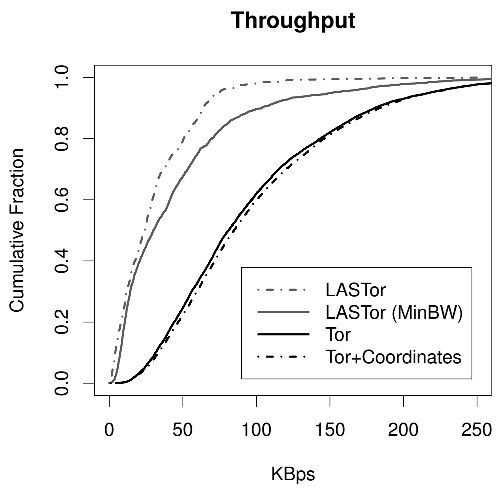
[44] C. Soghoian. Enforced Community Standards For Research on Users of the Tor Anonymity Network. In *Workshop on Ethics in Computer Security Research (WECSR)*, 2011.

[45] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[46] C. Tang and I. Goldberg. An Improved Algorithm for Tor Circuit Scheduling. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[47] Tor Project, Inc. Tor Metrics Portal. `https://metrics.torproject.org/`.

[48] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-scale Network Emulator. *SIGOPS Oper. Syst. Rev.*, 36:271–284, December 2002.



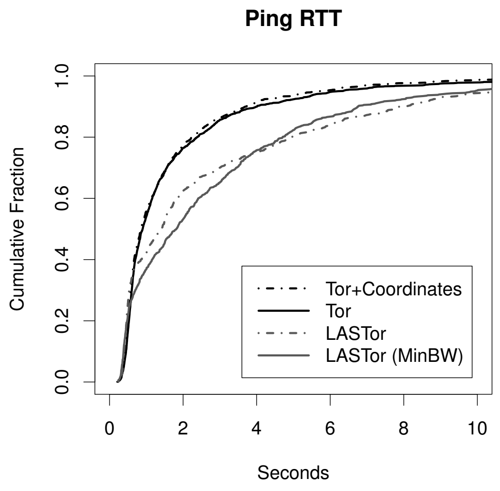


Figure 13. The throughput (top) and P-RTT (bottom) of *LASTor* **with the proposed refinement shown alongside** *Tor+Coordinates*, *Tor*, **and the standard** *LASTor*.

[49] T. Wang, K. Bauer, C. Forero, and I. Goldberg. Congestion-aware Path Selection for Tor. In *Financial Cryptography and Data Security (FC)*, 2012.

[50] M. Wright, M. Adler, B. N. Levine, and C. Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004.

A LASTor proposed bandwidth enhancement

We evaluated a variant of the bandwidth enhancement proposed by Akhoondi *et al.* [2]. They refine their basic *LASTor* strategy by restricting the relays chosen within each gridsquare to only those relays that have a reported bandwidth greater than 100 KBps. We implemented this refinement with one modification due to the potentially small number of relays per gridsquare in our emulation environment: we limit the set of paths to those in which a relay meeting the 100 KBps bandwidth restriction is guaranteed at each step.

Figure 13 shows the throughput and P-RTT of the modified version of *LASTor*, under medium congestion with homogeneous clients. The proposed refinement significantly increases the performance of *LASTor* (relative to the version without the bandwidth constraint), but still results in much worse performance than *Tor* and *Tor+Coordinates*.