Methodically Modeling the Tor Network

Rob Jansen
U.S. Naval Research Laboratory
rob.g.jansen@nrl.navy.mil

Kevin Bauer
University of Waterloo
k4bauer@cs.uwaterloo.ca

Nicholas Hopper
University of Minnesota
hopper@cs.umn.edu

Roger Dingledine
The Tor Project
arma@torproject.org

Abstract

Live Tor network experiments are difficult due to Tor's distributed nature and the privacy requirements of its client base. Alternative experimentation approaches, such as simulation and emulation, must make choices about how to model various aspects of the Internet and Tor that are not possible or not desirable to duplicate or implement directly. This paper methodically models the Tor network by exploring and justifying every modeling choice required to produce accurate Tor experimentation environments. We validate our model using two state-of-the-art Tor experimentation tools and measurements from the live Tor network. We find that our model enables experiments that characterize Tor's load and performance with reasonable accuracy.

1 Introduction

Tor [20] is an anonymizing overlay network consisting of thousands of volunteer *relays* that provide forwarding services used by hundreds of thousands of *clients*. To protect their identity, clients encrypt their messages multiple times before source-routing them through a *circuit* of multiple relays. Each relay decrypts one layer of each message before forwarding it to the next-hop relay or destination *server* specified by the client. Without traffic analysis, the client and server are *unlinkable*: no single node on the communication path can link the messages sent by the client to those received by the server.

Tor is a distributed system containing a handful of authorities that assist in distributing a *consensus* of trusted relay information. This *directory* of relays informs clients about the stability of and resources provided by each relay. Clients use this information to select relays for their circuits: the choice is weighted by the relative difference in the perceived throughput of each relay in an attempt to balance network load. Although Tor's main purpose is to protect a client's communication privacy, it also serves as a tool to resist censorship. Citizens in countries controlled by repressive regimes rely on Tor to mask their intended communication partners, thereby circumventing the block that may otherwise occur at censors' borders. Although several nations have attempted to block Tor, its distributed architecture has thus far proven resilient to long term censorship.

Tor's popularity, distributed architecture, and privacy requirements increase the difficulty in experimenting with new algorithm and protocol designs. New designs require software updates before testing their network effects, which both prolongs and complicates the experimental process. Further, since the live network is not a controlled environment, fluctuations in network conditions may both bias results and make them impossible to replicate. Finally, experiments that require client data collection are generally discouraged due to privacy risks.

The disadvantages to live Tor experimentation have led researchers to explore alternative approaches, including utilization of network testbeds such as PlanetLab [29], simulation [23,24,28], and emulation [4,5,16]. Each of these live Tor experimentation alternatives must make choices about how to model the existing network. A lack of details about and justifications for such choices obscures the level of faithfulness to the live network and decreases confidence that the obtained results provide meaningful information.

We improve the state of cyber security by contributing a novel and complete model of the Tor network that may be used for safe and realistic Tor experiments. In Section 3, we enumerate, explore, and justify each Tor modeling decision through methodical reasoning, using data from real Internet measurements where possible. We provide insight into non-intuitive consequences of alternative modeling strategies while precisely specifying and discussing our modeling techniques.

We validate that our model produces an accurate environment whose performance and load are characteristic of the live Tor network. To this end, we utilize two state-of-the-art Tor experimentation platforms: Shadow [23] and ExperimenTor [16]. We describe the tools and discuss their pros and cons in Section 2, both to show that our model is applicable in multiple testing environments

and to guide future work in selecting the tool most suitable to a given research question. In Section 4, we instantiate our model with both Shadow and ExperimenTor and compare results obtained with each tool to data collected from the live Tor network. We find that both tools produce reasonable Tor load and performance characteristics using networks of various sizes produced with our model. We inform the research community about the lessons we learned in Section 5 while concluding and discussing future work in Section 6.

The following summarizes this paper's contributions:

- Justified, precise specifications of techniques used to create accurate Tor network models
- Validation that our model produces an accurate environment whose performance and load are characteristic of the live Tor network, with multiple experimentation tools
- The first direct comparison between results obtained with Shadow [23] and ExperimenTor [16]—two state-of-the-art Tor experimentation platforms

2 Background and Related Work

While Tor is the most widely used anonymity network today with hundreds of thousands of daily users, Tor is still an active research network on which researchers work to improve its performance and security. To that end, prior Tor research has utilized a wide variety of methodologies which includes: analytical modeling [18,28] and simulation [24,28] of specific aspects of Tor's design; relatively small Tor deployments on PlanetLab [15,33]; and direct experimentation [17] and measurement [27] on the live Tor network.

Analytical modeling, simulations and small-scale Tor deployments on testbeds such as PlanetLab each make certain simplifying and potentially unrealistic assumptions that often leave many open questions about how the results obtained might translate to the live Tor network. Direct measurement and experimentation with the live network are unable to investigate design changes at scale due to software upgrade delays. Further, such well-intentioned research might have a negative impact on real Tor users' quality of service or privacy [25].[1]

In an effort to enhance the realism and safety of Tor experimentation, two designs for whole-Tor network testbeds, Shadow [23] and ExperimenTor [16], have been independently developed and made publicly available for use by the research community. In contrast to prior approaches to Tor research, these testbeds seek to replicate in isolation the important dynamics of the live Tor network at or near scale, complete with directory authorities, Tor routers, Tor clients, applications, and servers. While the details of how these tools model the live Tor network are discussed at length in Section 3, we first overview each tool's distinct approach.[2]

Shadow. To produce high fidelity experiments in a controlled and repeatable manner, Shadow leverages discrete-event simulation of the network layer and runs real, unmodified application software within the virtual network topology. Shadow also simulates the effects of background Internet traffic by introducing non-deterministic jitter and packet loss on links. Shadow offers an extensible plug-in framework through which an investigator can integrate an application or protocol of her choice into the Shadow experimentation environment. A plug-in called Scallion for simulating the Tor network is available. Important advantages of Shadow are that it can simulate large-scale distributed systems (such as Tor) on a single well-provisioned machine, results can be trivially replicated due to its design, and it can scale to arbitrarily sized networks because it runs in virtual time. Furthermore, virtual machines are available for running Shadow in the cloud on Amazon's EC2. See Shadow's webpage for more details [11].

ExperimenTor. Similar to Shadow, ExperimenTor offers the ability to run unmodified Tor software within an isolated environment to conduct experiments that are faithful to dynamics of the live Tor network. In contrast to Shadow's network simulation approach, ExperimenTor is a network emulation-based testbed, built on the mature Modelnet [34] network emulation platform. ExperimenTor uses one machine to emulate a specified network topology and another machine (or possibly several machines) to run unmodified software within the virtual network. Also unlike Shadow, ExperimenTor does not endeavor to account for the effects of unrelated background Internet traffic on experiments. While ExperimenTor cannot easily be run on a single machine, it has an advantage of using the operating system's native network stack, rather than a simulation. More details about ExperimenTor can be found on its webpage [6].

3 Model

Tor experimentation outside of the live network benefits from accurate models of network characteristics and node behaviors. This section details our approach to modeling Tor while discussing alternative approaches and common pitfalls. Our model is not intended to be a complete set of *all* characteristics and behaviors one *could* model, but rather the subset that we found most important and most useful. Although tested with Shadow [23] and ExperimenTor [16], we intend the model to apply to a broad range of research problems.

[1] See Bauer *et al.* [16] for a survey of prior methods for Tor research.

[2] This work describes and uses Shadow version 1.4.0 with Scallion version 1.3.1, and ExperimenTor as of April 2012. Later versions may have new features and capabilities not described here.

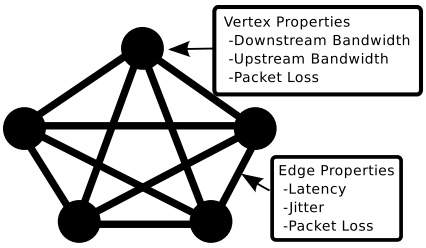


Figure 1: The vertex and edge properties in our modeled topology. The topology forms a complete graph.

3.1 Topology

We first consider the structure of our experimental network. Ideally, our network topology would replicate the Internet architecture, including all autonomous systems (ASes), core, backbone, and edge routers, and all links between them. Such a structure would provide the most accurate view of the Internet to an experimental framework. Unfortunately, the exact structure of the Internet is unknown and inferring it is an open research problem (e.g., [32]). Even if the Internet structure were known, it would be extremely large and too inefficient to replicate for experimental purposes. Therefore, we produce a small-scale, manageable model of the Internet.

Mapping the Internet topology is a major research area that has resulted in the development of multiple tools and techniques [3, 19, 26, 35]. This work utilizes geographic clustering *by country*[3] to scale the Internet down to a manageable topology because Tor similarly reports statistics about its users, allowing for a natural assignment of Tor node properties and placement of Tor nodes in our topology. Further, our approach produces small and efficient complete topologies (see Figure 1): we minimize the number of topology vertices and edges while remaining compatible with Tor's reporting method, and do not require routing algorithms to send packets through the network backbone. Finally, geographical clustering simplifies the process of mapping nodes to vertices, since any desired location (IP address) can be mapped to a cluster using a wide variety of GeoIP tools (e.g. those provided by MaxMind [8]).

Network Vertices. In our clustering approach, we create a *network vertex* for each country, Canadian province, and American state.[4] We take this approach, as opposed to clustering by Autonomous System (AS), because geographical clustering most closely resembles the actual structure of the Internet: end-users and hardware are physically located in clearly defined geographical regions. ASes, however, typically span multiple geographical regions. Further, many properties of network vertices and edges directly correspond to their geographical location, resulting in less variance when aggregating measurements of such properties.

Each vertex is assigned default *upstream bandwidth*, *downstream bandwidth*, and *packet loss* properties obtained from the Ookla Net Index dataset [9]. The dataset provides aggregate statistics collected during bandwidth speed tests [2] and ping tests [10]. Ookla aggregates millions of such tests and provides the rolling mean throughput for each geographic region (vertex in our topology) over thirty day intervals. The cumulative distributions on bandwidths are shown in Figure 2a.

Network Edges. Each vertex in our topology is connected to every other vertex, forming a complete graph. Each of these pairwise connections are represented as a *network edge*. We assign each network edge the following properties: latency (end-to-end packet delay), jitter (the variation in packet delay), and packet loss (the fraction of packets that are dropped). Note that full end-to-end loss rates are computed by combining the loss rates of the source and destination vertices and the connecting edge. Due to the lack of accurate loss rate measurements in the Internet core, our model currently utilizes only vertex loss rates from Ookla [9].

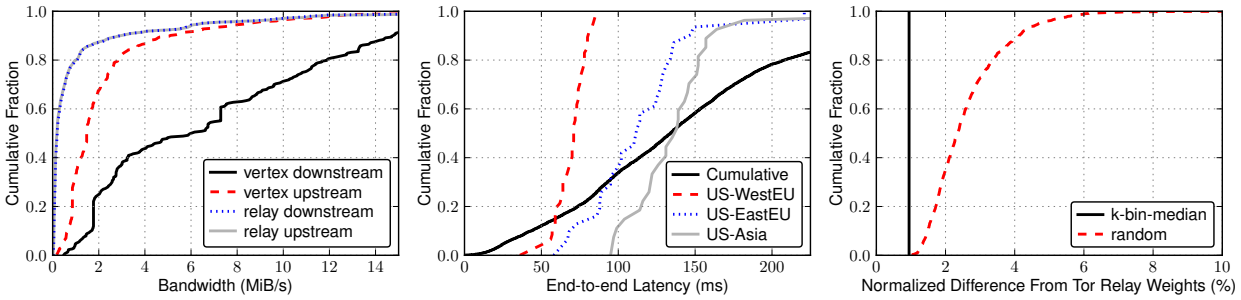
To model edge latency in our topology, we use round trip times (RTTs) measured by the iPlane [26] latency estimation service.[5] iPlane gathers RTTs from several vantage points, including PlanetLab nodes and traceroute servers, on a daily basis [7]. We use $\frac{RTT}{2}$ to approximate latency between every iPlane node.[6] We then use GeoIP lookup to assign each iPlane node to a network vertex, and therefore each estimated latency value corresponds to a network edge. Since there may not be an iPlane node corresponding to every network vertex (because there is not an iPlane node in every country), we create a temporary virtual overlay topology containing only nine "regional" clusters (e.g. US East, US West, EU East, EU West, etc.) and aggregate our latency estimates on the corresponding regional overlay edges. Then, we assign each network edge for which we have no `RTT` measurements the median latency value from the corresponding overlay edge. Figure 2b shows the iPlane latency estimates between common regional overlay edges, and confirms that an increase in physical distance between nodes implies an increase in latency. Finally, we approximate jitter over our network edges as $\frac{IQR}{2}$, where IQR is the edge latency inter-quartile range.

[3]Note that some Tor research questions may require a more detailed model of the Internet topology, a problem future work should consider.

[4]We used Tor's directly-connecting-user country database [12] to form our list of countries, which we supplemented with states and provinces from Net Index [9].

[5]The traceroutes were collected on 2012-03-28.

[6]Although Internet paths may be asymmetric, we found $\frac{RTT}{2}$ a suitable approximation of edge latency after aggregating measurements.



(a) (b) (c)

Figure 2: (a) Topology vertex bandwidths from Net Index, and estimated relay bandwidths from published relay documents. (b) Topology edge latency. (c) Sampling relays for scaled-down Tor experiments. Our sampling algorithm produces the best fit to the original relay bandwidth distribution by minimizing the area between the CDF curves.

3.2 Hosts

Once we have configured a topology, we next configure hosts that operate in that topology. In the context of a Tor network, we are most concerned with Tor relays, Tor clients, Tor authorities, and Internet web/file servers. Although the live Tor network contains thousands of relays and hundreds of thousands of clients, it is often the case that experiments must be scaled down significantly due to hardware limitations. We now explain our approach to scaling down the Tor network for each host type.

Tor Relays. Relays are an important part of a Tor network model, as each of them donates bandwidth and provides the forwarding service upon which the network is built. Recall that when building circuits, clients select relays according to weights published in the consensus. These selection-weights direct clients to relays according to each relay's perceived throughput, and have a dramatic impact on relay and network load and congestion [31]. Therefore when scaling down from the thousands of relays in the Tor consensus to a manageable number for experiments, it is important that the distribution of selection-weights in the scaled network is as close as possible to that of the live network. Past work has sampled uniformly at random from the existing set of relays [24] when choosing relays for experiments. Unfortunately, the distribution that results from randomly selecting relays may not fit the original selection-weight distribution well. We now describe an algorithm that produces the *best fit sample* of the original distribution while quantifying its improvement over random selection.

To scale the number of relays down to $\mathcal{K}$ of $\mathcal{N}$, we split a sorted list of $\mathcal{N}$ relay selection-weights into $\mathcal{K}$ bins and choose the median weight from each bin. The resulting weight distribution *best fits* that of the original relay population: any non-median weight value would only increase the distance between the distributions. This approach is detailed in Algorithm 1. To quantify our algorithm's effectiveness, we compare it to random selection

Algorithm 1: Sample relay bandwidths to produce a distribution that best fits that of the original relay population

Input: sorted list $\mathcal{L}$ of $\mathcal{N}$ relay bandwidths, sample size $\mathcal{K} \leq \mathcal{N}$
Output: sorted list of sampled bandwidths $\mathcal{S}$

```
n ← floor(N/K);
r ← K − n;
i ← 0;
for k ← 0 to K − 1 do
    j ← i + n;
    if k < r then j ← j + 1;
    bin ← L.slice(i, j);          // range [i, j)
    S.add(median(bin));
    i ← j
```

using the difference from the original relay weight distribution as a metric. This is calculated as the integral of the absolute value of the difference between the sampled CDF $s(x)$ and the Tor relay selection-weight CDF $f(x)$: $\int_0^\infty \mid f(x) - s(x) \mid dx$. The result is then normalized. Figure 2c compares the distribution of this closeness metric for 1000 samples of $\mathcal{K}$ relays using our algorithm and random sampling. (We found insignificant variance in the sample distributions when choosing $\mathcal{K} \in [50, 1000]$.) While our algorithm always produces the best fit result for each sample (hence the vertical line in Figure 2c), random sampling produces weight distributions as far as ten percent from optimal.

We draw two samples of relays from those listed in the consensus: one for exit relays (discussed below) and one for non-exit relays. We then consider several relay properties. First, we assign each relay to the network vertex in our topology corresponding to its geographic location (found by GeoIP lookup of the relay's IP address). This allows communication between relays while also resulting in latencies between relays that correlate with physical distances. Next we compute relay rate limits[7]

[7] A relay operator may limit the amount of bandwidth its relay con-

Algorithm 2: Estimate relay upstream and downstream capacities using data published in the consensus, server descriptors, and extra infos

Input: consensus weights $\mathcal{C}$, max bw bursts $\mathcal{B}$, max read and write bw histories $\mathcal{R}$ and $\mathcal{W}$
Output: capacity up $\mathcal{U}$ and down $\mathcal{D}$

for $i \leftarrow 0$ **to** $getRelayCount() - 1$ **do**
  if $\mathcal{B}[i] > 0$ **then**
    if $\mathcal{R}[i] > 0$ **and** $\mathcal{W}[i] > 0$ **then**
      $ratio \leftarrow \frac{\mathcal{R}[i]}{\mathcal{W}[i]}$;
      if $ratio > 1$ **then**
        $\mathcal{U}[i] \leftarrow \mathcal{B}[i]$;
        $\mathcal{D}[i] \leftarrow (\mathcal{B}[i] \cdot ratio)$;
      else
        $\mathcal{D}[i] \leftarrow \mathcal{B}[i]$;
        $\mathcal{U}[i] \leftarrow \left(\mathcal{B}[i] \cdot \frac{1}{ratio}\right)$;
    else $\mathcal{U}[i] \leftarrow \mathcal{D}[i] \leftarrow \mathcal{B}[i]$;
  else if $\mathcal{R}[i] > 0$ **and** $\mathcal{W}[i] > 0$ **then**
    $\mathcal{U}[i] \leftarrow \mathcal{W}[i]$;
    $\mathcal{D}[i] \leftarrow \mathcal{R}[i]$;
  else $\mathcal{U}[i] \leftarrow \mathcal{D}[i] \leftarrow \mathcal{C}[i]$;

and access link capacities, the most important properties affecting the resources each relay provides and the expected client performance in our modeled network. Rate limits are taken from the public server descriptors [12] of our sampled relays. Capacities must be estimated.

Since a relay's ISP access link capacities are not directly measured or published, we estimate these values using historical bandwidth measurements published in server descriptors and extra info documents, and the weights published in the consensus. The published documents include: bandwidth weights—values used during circuit construction to help distribute client load to faster relays; observed bandwidth—the smaller of the maximum sustained input and output over any ten second interval; and read/write bandwidth histories—the maximum sustained input and output over any fifteen-minute interval. We prefer the observed bandwidth as the best estimate of capacity. Since only the smaller of the input and output observed bandwidth is published, we use the read/write histories to infer to which the published value corresponds, and the ratio of read/write histories to estimate the unpublished observed value. In the absence of observed bandwidth information, we use read/write histories directly, and otherwise fall back on the bandwidth weights. A detailed specification is provided in Algorithm 2. The distribution on relay bandwidths computed using Algorithm 2 is shown in Figure 2a.

Note that a relay's observed bandwidth is only a good estimator of capacity when the relay was not limiting its rate during at least one ten second interval, and the relay had enough clients to consume its available bandwidth. Otherwise, the observed bandwidth is an underestimate of a relay's true capacity. This is corroborated in Figure 2a: upstream and downstream estimates are mostly symmetric due to the reliance on observed Tor bandwidth and Tor's circuit design, and the relay capacities appear far less than the expected upstream and downstream capacities from Net Index. We plan to explore passive measurement techniques, such as packet trains [22], to directly measure relay capacities in future work. Such measurements would provide a significantly better data source for modeling capacities than currently available.

The last part of modeling relays is adjusting their Tor configuration. As mentioned above, we sample relays that will exit Tor traffic separate from those that won't. Both exit and non-exit relays require the `ORPort` option to configure it as a relay while exit relays additionally require a configured `ExitPolicy`. (Exit policies may be found in relays' server descriptors.) Other notable configurations include `TestingTorNetwork` to help with bootstrapping in our test environment, and `DirServer` to specify our custom directory authorities.

Tor Authorities. Tor *directory authorities* are responsible for creating, signing, and distributing the consensus document—a list of all available relays and their associated bandwidth weights. Tor *bandwidth authorities* measure the expected performance of each relay and use the relative measured performance to compute the consensus weights used by clients for relay selection. In the live Tor network, the bandwidth measurement functionality is provided by a set of scripts known as TorFlow [13].

Our model selects the fastest sampled non-exit relay as the directory authority (all Tor directory authorities are currently non-exit relays). Since our test network lacks TorFlow, we must ensure that the bandwidth weights that appeared in the live network consensus also appear in our test network consensus. This is done by writing a `.v3bw` file with the live network bandwidth weights in the directory authority's data directory, as is done in live Tor. Lacking a valid `.v3bw` bandwidth file, the authorities will fall back on relays' reported observed bandwidth. In this case, we must remove a software-defined limit[8] on the observed bandwidth to allow relays to report the correct consensus weight. Note that although clients will be selecting relays in our test network using the same weights as the live network, the probability that each re-

sumes by configuring a token bucket rate-limiter: the token bucket size and refill rate can be configured by setting `BandwidthBurst` and `BandwidthRate` in the configuration file.

[8] Directory authorities will not trust any self-reported relay bandwidth over `DEFAULT_MAX_BELIEVABLE_BANDWIDTH`, which is set to a default value of 10 MiB/s.

Table 1: The ten countries with the highest reported Tor connecting user counts [12] during January, 2012.

Country	%	Country	%
United States	16.46	Spain	5.08
Iran	12.63	Russia	3.46
Germany	9.99	Republic of Korea	2.66
Italy	6.96	United Kingdom	2.39
France	6.30	Saudi Arabia	2.38

Table 2: The ten countries with the highest number of servers in the Alexa top 1 million data set [1] during January, 2012.

Country	%	Country	%
United States	47.94	France	3.64
Germany	8.65	Russia	3.40
China	4.50	Netherlands	2.86
United Kingdom	4.20	Canada	2.10
Japan	3.73	Italy	1.48

lay is selected necessarily increases (we downsampled the relays and the sum of the probabilities must equal 1).

Tor Clients. In our model, Tor clients are the main source of network load, producing all of the exit-bound traffic routed through Tor while simultaneously serving to measure network performance. Clients perform synchronous `HTTP GET` requests to download files through our modeled Tor network. Clients choose `HTTP` servers from which to request each download uniformly at random. Since the requests are synchronous, each client will be responsible for at most one stream through Tor at any time. Each client measures the time from when it initiates a connection to the SOCKS application proxy to the first byte and last byte of the file payload, indicating network responsiveness and performance.

Our model classifies clients into two broad categories: web clients and bulk clients. Each web client requests 320 KiB files, the average webpage size according to recent web metrics [30]. After completing a download, a web client will pause for a time drawn uniformly at random from a range of $[1,20]$ seconds before initiating the next download to simulate the time a user takes to consume the web page content. Each bulk clients requests 5 MiB files without pausing between the completion of one download and the initiation of the next. Our client model is based on work characterizing Tor exit traffic by McCoy *et al.* [27]. This work found that roughly 60% of the bytes and 95% of the connections exiting Tor were attributable to `HTTP` traffic while roughly 40% of the bytes and 5% of the connections were attributable to BitTorrent traffic. Therefore, we use a 19:1 *ratio* of web to bulk clients. The total *number* of clients is dependent on the number of relays and their capacities (see Section 4).

Each client is assigned a geographical location and the corresponding network vertex in our topology according to Tor's directly connecting user statistics [12,21]. These statistics specify the country from which clients connect when directly downloading Tor directory information. The top ten countries from a recent version of this data are shown in Table 1. When assigning a client to a vertex, the assignment is weighted by the given percentages. Each client's ISP connection upstream and downstream capacities are taken from the default vertex properties as measured by Net Index [9] (see Section 3.1).

Internet Servers. In our model, `HTTP` servers are the destinations of our client requests and the sources of the files downloaded through Tor. In order to attribute changes in performance to Tor itself while minimizing effects external to the network, we assign Internet servers 100 MiB/s bandwidth capacities. This high capacity will prevent our Internet servers from becoming bottlenecks during our client downloads. The geographic locations of Internet servers are assigned using the Alexa Top Sites data set [1]. Since the Alexa *ranking* may not capture the usage patterns of Tor users well, we instead produce a distribution on location of the reported top one million sites.[9] The top ten countries with the most sites in the Alexa data set are given in Table 2. Our assignment of server to topology vertex is weighted by this distribution, similar to our client vertex assignment.

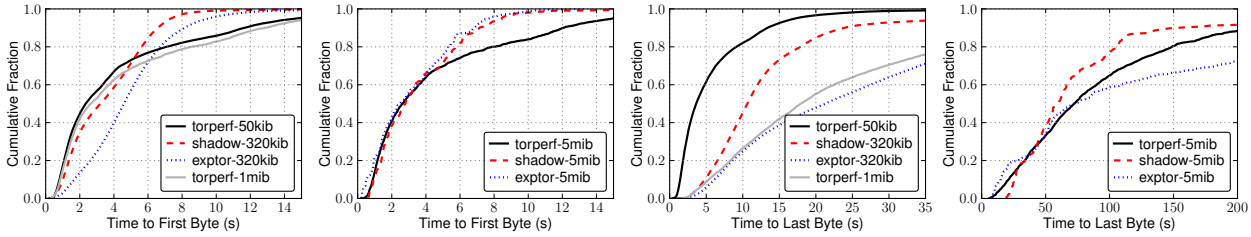
4 Methodology and Experiments

To determine the accuracy of and increase the confidence in our Tor network model, we instantiate it using two state-of-the-art Tor experimentation tools: Shadow [23] and ExperimenTor [16] (see Section 2 for background). This section compares the performance and load characteristics of the environments produced with each tool to that of the live Tor network, illustrating the effectiveness of our modeling strategies from Section 3. We choose network performance and load because Tor already measures these characteristics on the live network, allowing for a direct comparison of results. Further, these metrics represent the gauges in which clients and relays are generally interested, and are most useful when developing new algorithms that improve the state of the network.

We test our model with two different network sizes, both of which are scaled down versions of the live Tor network. In our *small* network, we configure 50 relays and 500 clients that communicate with 50 `HTTP` file servers. In our *large* network, we configure 100 relays and 1000 clients that communicate with 100 `HTTP` file servers. The small and large networks are approximately fifty and twenty-five times smaller than the size of Tor, respectively. Both Shadow and ExperimenTor use instantiated versions of our Tor network model[10] and are configured to run a vanilla instance of version 0.2.3.13-alpha of the Tor software for ninety virtual minutes. Download results are ignored during the first thirty minutes of each experiment to allow for Tor's bootstrap-

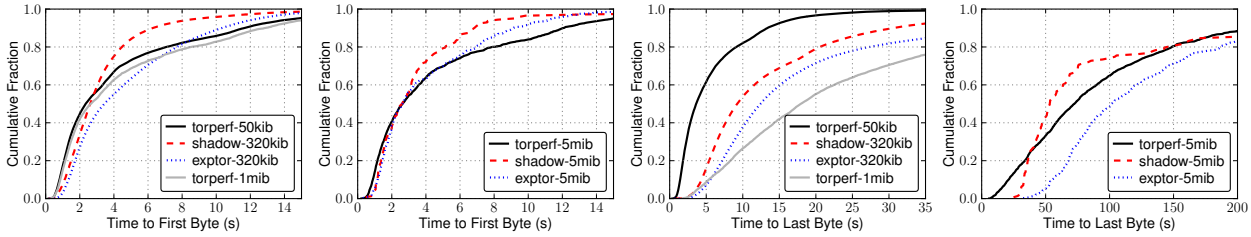
[9] We find locations with standard DNS queries and GeoIP lookups.

[10] The topology files are available on the Shadow website [11].



(a) 320 KiB clients (b) 5 MiB clients (c) 320 KiB clients (d) 5 MiB clients

Figure 3: Performance for live Tor and our small modeled network of 50 relays and 500 clients in Shadow and ExperimenTor. Time to the first byte of the data payload is shown in (a) and (b), and time to the last byte in (c) and (d), for various download sizes.



(a) 320 KiB clients (b) 5 MiB clients (c) 320 KiB clients (d) 5 MiB clients

Figure 4: Performance for live Tor and our large modeled network of 100 relays and 1000 clients in Shadow and ExperimenTor. Time to the first byte of the data payload is shown in (a) and (b), and time to the last byte in (c) and (d), for various download sizes.

ping process. File download timings during the remaining period are utilized as discussed below.

Note that we explored various numbers of clients and found that a 10:1 client-to-relay ratio in our experiments resulted in load and network performance that reasonably approximated that of the live Tor network [12]. We stress that this client-to-relay ratio is due to our client modeling strategies; alternative client behaviors may require an adjusted ratio to produce the network characteristics that best approximate Tor. Accurately modeling Tor client behaviors is an open research problem which future work should consider.

4.1 Network Performance

We compare client performance measured in our test environments to client performance in Tor during the same period we are modeling.[11] We measure the time to the first and last byte of the data payload of our 320 KiB and 5 MiB file downloads as indications of network responsiveness and throughput. We compare our results to live Tor network performance measured with torperf [14], a tool that monitors live Tor network performance by downloading files of sizes 50 KiB, 1 MiB, and 5 MiB. Performance for our small and large networks are respectively shown in Figures 3 and 4.

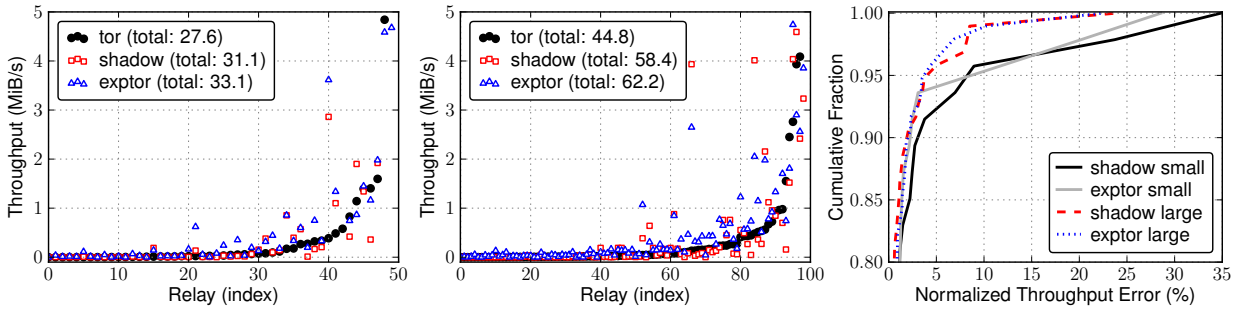
We expect client performance in our test environments to be similar to that in Tor. In particular, the time-to-first-byte should be consistent regardless of the size of the file being downloaded. As can be seen in Figures 3a, 3b, 4a, and 4b, our model produces accurate time-to-first-byte performance in both tools, although the tools tend to lose some accuracy above the eightieth percentile. Under the time-to-last-byte metric, we expect our 320 KiB web downloads to complete somewhere between the torperf 50 KiB and 1 MiB downloads, and our 5 MiB download times to be consistent with torperf. Web download times are more accurate in ExperimenTor in the large network (Figure 4c) than the small (Figure 3c), and all downloads tend to take slightly longer in ExperimenTor than in live Tor. Shadow approximates web download times reasonably well (Figures 3c and 4c), and bulk downloads complete slightly faster in Shadow than in Tor (Figures 3d and 4d). Overall, we are impressed that our model enables both tools to characterize Tor performance closely, even with scaled-down Tor networks.

4.2 Network Load

Each relay in Tor tracks byte histories: the number of bytes read and written over time. We use these statistics to calculate the throughput of each relay included in our small and large networks, and directly compare throughputs from Tor with throughputs from our experimentation environments. The results are shown in Figure 5.

The aggregate throughput for all the relays we chose in our small network (Figure 5a) totaled 27.6 MiB/s for live Tor, 31.1 MiB/s in Shadow, and 33.1 MiB/s in ExperimenTor. In our large network (Figure 5b), the aggregate throughput was 44.8 MiB/s in live Tor, 58.4 MiB/s in Shadow, and 62.2 MiB/s in ExperimenTor. These results indicate that our experimental networks were too heav-

[11] This work models Tor as it existed during January, 2012.



(a) Small network (b) Large network (c) Error comparison

Figure 5: Load in live Tor and (a) our small modeled network of 50 relays and 500 clients, and (b) our large modeled network of 100 relays and 1000 clients. Throughput is indexed by each relay chosen in our model and the sum is shown in the legend. (c) The distribution on the normalized experimental throughput error from reported live Tor throughput.

ily loaded, and the absolute error increased with the network size. The distribution on the normalized individual relay throughput error is shown in Figure 5c. The distributions have long tails: the maximum normalized error was 34.9% for Shadow and 28.6% for ExperimenTor in the small network, and 23.9% for Shadow and 22.5% for ExperimenTor in the large network. Although the absolute error increased with the network size, the individual errors decreased in the larger network. Our analysis found that most throughput error was attributable to bootstrapping issues: recently added fast relays were under-utilized in Tor but fully utilized in our experiments. Despite these issues, over 95% of relays in the small network and 98% of relays in the large network had less than 10% throughput error.

5 Lessons Learned

Modeling a distributed system is a complex process. During this process, we found that it is important to use real Internet and system measurements to eliminate arbitrary modeling decisions, as this tends to have a significant impact on how accurately the experimental environment replicates the real distributed system. However, measurements should not be used until they are fully understood (what they mean and how they are useful), or they may harm accuracy.

We also found it important to determine useful metrics that allow for a comparison between the experimental platform and the real distributed system being modeled. Useful metrics and proper comparisons of measurements increase confidence in the obtained results. Useful metrics assist in understanding the strengths and weaknesses of a model, and help determine if the environment produced from the model is suitable for the given research.

We discovered that it's very useful to replicate experiments on multiple experimental platforms. This can help identify errors or peculiarities caused by a specific tool. For example, this process allowed us to discover that packet header overhead on TCP packets without a data payload were not consuming bandwidth on Shadow's virtual network interfaces. Shadow's accuracy improved greatly after accounting for TCP packet header overhead on both data and control packets.

Finally, it is important to understand that ExperimenTor and Shadow have fundamentally different approaches to experimentation: Shadow simulates all network properties including jitter and packet loss on links due to the presence of background Internet traffic. In contrast, ExperimenTor emulates link properties simply as a function of the Tor traffic load, ignoring any effects due to background Internet traffic. As in our experiments, differences between other tools may also contribute to the differences in experimental performance and load, and should be considered when analyzing comparative experiments.

6 Conclusion

This paper explored modeling the distributed Tor network. We provided precise and detailed specifications of our modeling choices and their effect on the resulting experimental environment. We validated our model by instantiating it in two state-of-the-art Tor experimentation tools: Shadow [23] and ExperimenTor [16]. We compared network performance and network load from our experiments to real Tor data and found that our model leads to environments that characterize the live Tor network well. Finally, we provided insights into the lessons we learned while replicating our experiments with both experimentation tools.

Future Work. There are several ways in which our model could be improved. First, we could increase the size of the our network by improving the software support and acquiring the hardware resources necessary for handling larger networks in the available experimentation tools. Running at or near scale means we may reduce experimentation artifacts, such as those created because relay selection probabilities necessarily change when using only a subset of the existing relays. Larger networks

will also provide a more realistic experimentation environment and more realistic results.

Second, our model may benefit from capacity and link characteristics gathered directly from Tor relays. This would give us precise statistics about the specific nodes we are modeling and reduce our reliance on external sources of more generic information for links between relays. One possible approach to capacity measurement involves using packet trains [22], but more work is needed to determine the efficacy of such techniques in the context of the Tor network. At the same time, many research questions may require a more detailed topology structure than that modeled in this paper. Higher fidelity of the underlying network topology may be possible by combining data from iPlane [7] and CAIDA [3].

Third, determining a better client model would further increase confidence in experimental results. Producing a more robust client model will likely require the development of algorithms for collecting client statistics in a way that mitigates privacy risks. While this is challenging since client behaviors are dynamic and hard to capture in a representative fashion, it would allow us to increase faithfulness to the live Tor network and its users. Finally, modeling malicious adversaries and their behaviors may be of specific interest to future research that analyzes the security of Tor or its algorithms.

Acknowledgments. The authors thank Ian Goldberg, Aaron Johnson, Harsha Madhyastha, Micah Sherr, Paul Syverson, and Chris Wacek for discussions regarding this work, and the anonymous reviewers for their comments. This work was supported by ONR and DARPA.

References

[1] Alexa The Web Information Company. Top 1 million sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. Retrieved January 31, 2012.

[2] Bandwidth Speed Test. http://speedtest.net/.

[3] CAIDA Data. http://www.caida.org/data.

[4] DETER Testbed. http://www.isi.edu/deter.

[5] Emulab Homepage. http://www.emulab.net.

[6] ExperimenTor Homepage. http://crysp.uwaterloo.ca/software/exptor/.

[7] iPlane: Data. http://iplane.cs.washington.edu/data.

[8] MaxMind GeoIP. http://www.maxmind.com/.

[9] Net Index Dataset. http://www.netindex.com/source-data/.

[10] Ping Test. http://pingtest.net/.

[11] Shadow Homepage. https://shadow.cs.umn.edu/.

[12] Tor Metrics Portal. http://metrics.torproject.org/.

[13] TorFlow. https://gitweb.torproject.org/torflow.git/.

[14] TorPerf. https://gitweb.torproject.org/torperf.git/.

[15] BAUER, K., MCCOY, D., GRUNWALD, D., KOHNO, T., AND SICKER, D. Low-Resource Routing Attacks against Tor. In *Proc. of Workshop on Privacy in the Electronic Society* (2007).

[16] BAUER, K., SHERR, M., MCCOY, D., AND GRUNWALD, D. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *Proc. of the 4th Workshop on Cyber Security Experimentation and Test* (2011).

[17] BLOND, S. L., MANILS, P., CHAABANE, A., KAAFAR, M. A., CASTELLUCCIA, C., LEGOUT, A., AND DABBOUS, W. One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users. In *Proc. of the 4th Workshop on Large-Scale Exploits and Emergent Threats* (2011).

[18] BORISOV, N., DANEZIS, G., MITTAL, P., AND TABRIZ, P. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *Proc. of 14th Conference on Computer and Communication Security* (2007).

[19] DHAMDHERE, A., AND DOVROLIS, C. Twelve Years in the Evolution of the Internet Ecosystem. *IEEE/ACM Transactions on Networking 19*, 5 (Sep 2011), 1420–1433.

[20] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The Second-Generation Onion Router. In *Proc. of the 13th USENIX Security Symposium* (2004).

[21] HAHN, S., AND LOESING, K. Privacy-preserving Ways to Estimate the Number of Tor Users. Tech. rep., The Tor Project, November 2010. https://metrics. torproject. org/papers/countingusers-2010-11-30. pdf, 2010.

[22] JAIN, R., AND ROUTHIER, S. Packet Trains–Measurements and a New Model for Computer Network Traffic. *IEEE Journal on Selected Areas in Communications 4*, 6 (1986), 986 – 995.

[23] JANSEN, R., AND HOPPER, N. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proc. of the 19th Network and Distributed System Security Symposium* (2012).

[24] JANSEN, R., HOPPER, N., AND KIM, Y. Recruiting New Tor Relays with BRAIDS. In *Proc. of the 17th Conference on Computer and Communication Security* (2010).

[25] LOESING, K., MURDOCH, S. J., AND DINGLEDINE, R. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Proc. of the Workshop on Ethics in Computer Security Research* (2010).

[26] MADHYASTHA, H., ISDAL, T., PIATEK, M., DIXON, C., ANDERSON, T., KRISHNAMURTHY, A., AND VENKATARAMANI, A. iPlane: An Information Plane for Distributed Services. In *Proc. of the 4th Operating Systems Design and Implementation* (2006), pp. 367–380.

[27] MCCOY, D., BAUER, K., GRUNWALD, D., KOHNO, T., AND SICKER, D. Shining Light in Dark Places: Understanding the Tor Network. In *Proc. of the 8th Privacy Enhancing Technologies Symposium* (2008), pp. 63–76.

[28] MURDOCH, S. J., AND WATSON, R. N. M. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Proc. of the 8th Privacy Enhancing Technologies Symposium* (2008).

[29] PETERSON, L., MUIR, S., ROSCOE, T., AND KLINGAMAN, A. PlanetLab Architecture: An Overview. Tech. rep., PlanetLab Consortium, 2006.

[30] RAMACHANDRAN, S. Web metrics: Size and number of resources. http://code.google.com/speed/articles/web-metrics.html, 2010.

[31] SNADER, R., AND BORISOV, N. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proc. of the 16th Network and Distributed Security Symposium* (2008).

[32] SPRING, N., MAHAJAN, R., AND WETHERALL, D. Measuring ISP Topologies with Rocketfuel. In *Proc. of the SIGCOMM Conference* (2002), pp. 133–145.

[33] TANG, C., AND GOLDBERG, I. An Improved Algorithm for Tor Circuit Scheduling. In *Proc. of the 17th Conference on Computer and Communication Security* (2010).

[34] VAHDAT, A., YOCUM, K., WALSH, K., MAHADEVAN, P., KOSTIĆ, D., CHASE, J., AND BECKER, D. Scalability and Accuracy in a Large-scale Network Emulator. *SIGOPS Operating Systems Review 36* (2002), 271–284.

[35] YOOK, S., JEONG, H., AND BARABÁSI, A. Modeling the Internet's Large-Scale Topology. *Proc. of the National Academy of Sciences 99*, 21 (2002), 13382.