ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation

Kevin Bauer
University of Waterloo

Micah Sherr
Georgetown University

Damon McCoy
University of California, San Diego

Dirk Grunwald
University of Colorado

Abstract

Tor is one of the most widely-used privacy enhancing technologies for achieving online anonymity and resisting censorship. Simultaneously, Tor is also an evolving research network on which investigators perform experiments to improve the network's resilience to attacks and enhance its performance. Existing methods for studying Tor have included analytical modeling, simulations, small-scale network emulations, small-scale PlanetLab deployments, and measurement and analysis of the live Tor network. Despite the growing body of work concerning Tor, there is no widely accepted methodology for conducting Tor research in a manner that preserves realism while protecting live users' privacy. In an effort to propose a standard, rigorous experimental framework for conducting Tor research in a way that ensures safety and realism, we present the design of ExperimenTor, a large-scale Tor network emulation toolkit and testbed. We also report our early experiences with prototype testbeds currently deployed at four research institutions.

1 Introduction

Tor [13] is one of the most widely-used privacy enhancing technologies for achieving online anonymity and resisting censorship, with an estimated 250,000 daily users [20] and over 2,500 volunteer-operated Tor routers. Ordinary Internet users from around the world use Tor to enhance the privacy of their web browsing, e-mail, instant messaging, and file sharing. Users who reside in jurisdictions that enforce censorship policies also use Tor to enable free and open web browsing and online speech.

Despite its popularity, Tor is still an evolving research network on which researchers work to harden Tor's design against attacks on its anonymity [9, 10, 24] and improve the network's quality of service [29, 31, 35]. In spite of this large and continually growing body of work, there is no standard or widely accepted methodology for conducting Tor research in a manner that is *safe* (*e.g.*, researchers do not risk jeopardizing live users' privacy or quality of service) and *realistic*, faithfully reproducing the salient features of the Tor network in a way that enables the results of non-live experiments to improve our understanding of the live Tor network.

Prior Tor research has employed a variety of methods including analytical modeling [10, 24], simulations [17, 24–26], small-scale network emulations [12], small-scale PlanetLab deployments [9, 35], and live experimentation, measurement, and analysis of the real Tor network [11, 18, 22, 23]. While each respective experimental method offers its own advantages and drawbacks, no approach offers a high degree of realism without experimenting with live users on the real Tor network. However, experiments that involve real users might inadvertently cause harm to the users' anonymity or degrade their quality of service. Thus, in an effort to enable realistic experiments without any potential for a negative impact on live users, we present the design and our early experiences with an isolated whole-network Tor emulation toolkit and testbed.

Tor, like other large and complex distributed systems, presents significant obstacles to accurate replication and deployment in a testbed environment. First, it is necessary to accurately model all relevant dynamics of the live Tor network, including (but not limited to) the distribution of Tor router bandwidth, Tor router exit policies, and Tor client behaviors and application traffic models. Also, it is necessary to emulate the underlying network topology and end-host and link properties in a manner that is faithful to the live Tor network. Finally, it is essential to have the ability to run both unmodified Tor code as well as unmodified applications (e.g., web browsers and peer-to-peer file sharing software) within the testbed.

To meet these challenges, we present ExperimenTor, a large-scale Tor network emulation toolkit and testbed that enables realistic experimentation by modeling the distribution of Tor router bandwidth, client traffic loads and applications, and other important aspects of the real Tor network. In contrast to shared and relatively small

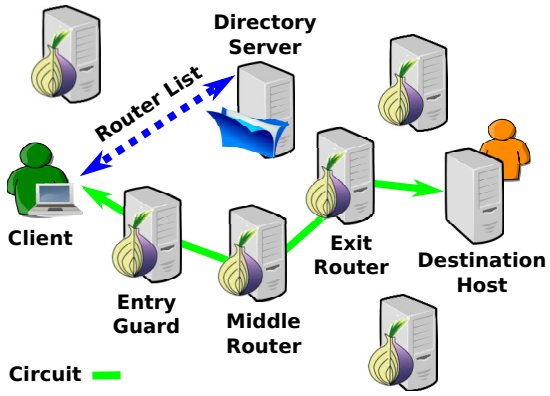

Figure 1: The Tor network's system architecture

network emulation platforms such as Emulab [2] or DETER [1], ExperimenTor is built on top of the ModelNet [36] network emulation platform and runs entirely on inexpensive commodity hardware, can be scaled up using a cluster of machines, and can be deployed at the researcher's local institution. Prototypes of the publicly-available[1] testbed are currently operational at four research institutions and we detail our experiences using the testbed for Tor research.

Contributions. In summary, this paper offers the following contributions to the science of cyber-security experimentation:

1. We offer a taxonomy of past experimental methods used in Tor research. The sheer variety of past approaches highlights the lack of a standard best practice for conducting experimental Tor research.
2. We argue that whole-system Tor emulation offers the greatest degree of realism without placing live Tor users at risk to lost anonymity or degraded quality of service.
3. We present the design and implementation of ExperimenTor, a network emulation-based toolkit and testbed that enables realistic, scalable, and safe Tor experiments. We also discuss our early experiences and "lessons learned" through building ExperimenTor and using it for Tor research.

2 Tor Background

Tor is a network of volunteer-operated routers that enables users to communicate privately in the presence of eavesdroppers who have local (non-global) views of the Internet [13].

To use the Tor network, clients tunnel TCP connections through bidirectional *circuits* that consist of a small randomly selected subset of Tor routers. Typical circuits consist of three routers: an *entry guard*, a *middle router*, and an *exit router* (see Figure 1). Router discovery is accomplished by querying Tor *directory servers* that publish routers' network addresses, public keys, and bandwidth capacities. The termination point of a circuit – the exit router – forwards TCP flows to/from their intended destination using direct IP communication. Reply traffic traverses circuits in the opposite direction.

Tor uses a layered encryption technique similar to onion routing [34] to thwart traffic analysis. The initiator of communication establishes session keys with each router in its chosen circuit through a *telescoping* technique: the guard router's session key is derived via authenticated Diffie-Hellman; session keys for the middle and exit routers are established by tunneling the authenticated Diffie-Hellman through the preceding hops, effectively increasing the length of the circuit one router at a time. Once session keys are established, data are multiply encrypted with the session keys, allowing each router to "peel back" its encryption and forward the result to the subsequent hop. Tor routers handle reply traffic in the opposite manner – data are encrypted at each step until reaching the client.

The application of layered encryption exposes only the identities of the previous and next routers at each hop along the circuit. Since the entry guard communicates directly with the client, it trivially learns the client's network address; similarly, the exit router learns the identity of the receiver. In principle, the guard and exit routers cannot determine that they belong to the same circuit.[2] Importantly, an adversary who intercepts communication along the circuit discovers only the endpoints of the monitored link(s); the identities of the communicants are protected through the layered encryption.

Tor is primarily intended to provide privacy by hiding senders' network addresses. However, the system is also a fully featured overlay routing system, supporting congestion control, rate limiting, and multiplexing of circuits across common routers. It is highly configurable, enabling users to specify average bandwidth rates, the frequency of directory fetches, policies for including or excluding specific routers in circuits, the duration of anonymous circuits, among many other options. As Tor's usage has increased, so has its features and complexities.

3 Prior Methods for Tor Experimentation

The Tor network has grown in popularity from the publication of its design in 2004 [13], in part, due to a close collaboration with the research community. Since 2004, researchers have studied Tor's security, anonymity, and performance in an effort to understand the network's resilience to various attacks, propose mitigation strategies for attacks, and improve Tor's performance to make the

[1] Available at `http://crysp.org/software/exptor`

[2] However, timing-based attacks demonstrate that colluding entry and exit nodes can perform such linkage with high probability [19].

system attractive to more users. Tor's status today as the defacto standard technique to achieve anonymous communications and resist censorship on the Internet has been largely influenced by the role that researchers have played in proposing and evaluating improvements to Tor's design and implementation.

Despite Tor's successful collaboration with the research community, there is no standard and accepted methodology for conducting Tor research in a manner that ensures the safety of real Tor users and also achieves a high degree of realism. Prior Tor research has used a wide variety of experimental techniques, ranging from abstract analytical modeling of specific details of Tor's design to launching potentially dangerous attacks on the live Tor network with a possibility of harming the anonymity or quality of service of live users.

In the remainder of this section, we present an overview of the various experimental methods used in past Tor research, highlighting the advantages and drawbacks of each respective approach and, ultimately, concluding that a fresh approach is needed to standardize Tor experimentation. Note that our objective in this section is not to be critical of prior work, but instead to evaluate the current state-of-the-art of Tor experimentation.

Analytical modeling. To understand how attacks on reliability can be used to facilitate the de-anonymization of Tor users, Borisov *et al.* [10] abstractly model Tor's router selection process and estimate the expected likelihood of an adversary controlling Tor circuits' entry and exit positions through denial-of-service attacks against non-compromised circuits. While their analysis offers valuable insights about this attack in the abstract, their analytical model assumes that clients choose Tor routers by uniform selection. However, Tor clients weigh router selection by bandwidth, and thus, it is unclear whether the attack's precise behavior as observed by Borisov *et al.* would hold on the live Tor network. An empirical analysis of denial-of-service attacks on the live network may be ill-advised, as it might harm real users' anonymity or quality of service; we note that such a study could be safely conducted in a testbed environment.

Murdoch and Watson [24] study the impact of different router selection algorithms on users' security and performance. To determine users' expected waiting time to complete a request, the authors apply queuing theory to model a Tor router as an M/D/1 queue with a Poisson input process. These assumptions may have some degree of fidelity to the live network; however, it has been previously shown that Poisson processes do not accurately model real Internet traffic [27]. In Murdoch and Watson's work, the use of an analytic model was likely motivated by its ability to evaluate a *global change* to Tor's design (e.g., a network-wide adoption of a new router selection algorithm). In contrast, simply modifying a small fraction of Tor clients on the live network might provide an incomplete and potentially misleading analysis. In general, to properly study the effects of protocol changes at scale, it is necessary to deploy an entire network of routers with the modifications. Thus, a large-scale Tor emulation platform fuelled by empirically-derived traffic models would be an ideal tool for this type of experiment.

Simulations. Murdoch and Watson [24] also *simulate* router selection algorithms to study the effect of the router selection strategy on an adversary's ability to compromise circuits and de-anonymize Tor clients. Tor's router selection algorithm is highly complex and difficult to abstractly model, thus, it is necessary to implement the selection mechanism in simulation and observe circuit compromise rates. We note that the simulator used data from Tor's directory servers to model Tor routers. Alternatively, this type of analysis could be done with emulated Tor clients running the real Tor code in a testbed environment.

In addition, parts of Tor's design and operation have been implemented as discrete-event simulators. Using the Scalable Simulation Framework [6], O'Gorman and Blott [26] develop a discrete-event Tor simulator capable of simulating a Tor network with up to 4,500 web clients, 100 web servers, and over 950 Tor routers. Their work represents an ambitious effort to simulate the Tor network at scale. Their implementation simulates Tor's circuit establishment process, Tor's fixed-size 512-byte cells, and Tor's multiplexing of several streams over individual circuits. Jansen *et al.* [17] build a custom discrete-event Tor simulator that includes circuit establishment and realistic empirical traffic models to simulate web browsing and file sharing clients. Their simulator is well-suited for evaluating multiple Tor queuing models as well as examining the effects of network growth on bandwidth performance. However, experience with the Java-based simulator indicates it does not scale beyond 50 Tor routers, 1,500 web clients, and 50 file sharers. Ngan *et al.* [25] implement a packet-level discrete-event Tor simulator that is capable of simulating on a commodity machine a Tor network with traffic generated by 20 BitTorrent clients and 2,000 web clients. However, the simulator's source code is not publicly available.

PlanetLab deployments. In general, a significant shortcoming of PlanetLab as an experimental platform is that results are often not reproducible [28]. Nonetheless, many researchers have utilized PlanetLab to study Tor.

In order to study the influence of Tor's bandwidth-weighted router selection algorithm on an adversary's ability to compromise Tor circuits, Bauer *et al.* deployed small Tor networks on PlanetLab [9]. Two sets of experiments were conducted with 40 Tor routers and 60 clients, and with 60 Tor routers and 90 clients. However, in order to accurately model the live Tor network's router band-

width distribution, it is necessary to exhaustively search for individual PlanetLab nodes with specific bandwidth properties – a challenging task since PlanetLab nodes are shared among many researchers and each node's available bandwidth may vary over time. Also, due to PlanetLab's limited resources, the experimental Tor networks were much smaller than the live Tor network, in terms of the number of Tor client and router nodes and the network's aggregate bandwidth.

Tang and Goldberg [35] seek to improve performance by replacing Tor's round-robin circuit scheduling algorithm with a prioritization scheme that services web clients' traffic ahead of bulk clients' downloads. As a component of a larger evaluation, the authors present results from a PlanetLab deployment with three Tor routers, three Tor clients, and a single web server. A small-scale evaluation on PlanetLab enabled the authors to study the whole-network effects of their proposed improvements. However, with a large-scale testbed such as ExperimenTor, it would also be possible to scale this experiment beyond only a few nodes and perform a whole-network evaluation at scale.

Small-scale emulations. Chakravarty *et al.* [12] propose a traffic analysis attack on Tor that uses available bandwidth estimation techniques to identify a Tor client and their chosen circuit. In this attack, it is essential to accurately model real network dynamics such as bandwidth and delays. To demonstrate and evaluate the proposed attack, the authors conduct small-scale experiments on DETER with roughly one dozen end-hosts acting as either Tor clients or Tor routers. While such a small experiment is sufficient for a proof-of-concept to demonstrate the attack, in order to provide a deeper understanding of how the attack works in practice, it may be necessary to scale the experiment beyond what is currently feasible with DETER.

Live experiments. Many prior Tor studies have performed experiments, measurements, or analysis that in some way involved the live Tor network [11, 16, 18, 22, 23, 31, 35]. Studies that directly observe or analyze the live Tor network have the ability to offer great insight into real Tor behavior; however, we observe that many of these prior studies involving the live Tor network suffer from certain limitations and even potential hazards.

To illustrate one common limitation of live Tor experimentation, consider the following experiments. To give Tor users the ability to better manage the security and performance trade-offs that result from Tor clients' router selection strategy, Snader and Borisov [31] proposed a tunable router selection algorithm that allows users to tune the degree to which they weigh router selection toward high bandwidth routers. To evaluate this approach, the authors deploy a single Tor client that uses the tunable router selection strategy and report on their client's security and performance over the course of several months of participation in the Tor network. While these experiments offer valuable insight into the proposed selection technique, they do not analyze the behavior of the proposed algorithm when many or all Tor clients use various configurations of tunable selection.

Similarly, Tang and Goldberg [35] evaluate their proposed circuit scheduling algorithm by deploying a single modified Tor router on the live Tor network, where all other Tor routers are unmodified. The authors observe that their proposed circuit scheduling algorithm offers noticeable improvements in download time relative to Tor's default algorithm when Tor clients use circuits with their modified Tor router as a middle node and unmodified routers as their entry guards and exit routers. However, from the experiments presented, it is not clear how much improvement might be possible if the whole Tor network used their proposed scheduling algorithm.

These examples highlight one inherent limitation of live Tor experiments: *Evaluating small-scale modifications to a few Tor clients or Tor routers may not provide much information about how the modifications scale to the Tor network as a whole*.

Another potential limitation of live experimentation involves the possible risks to real users' anonymity or quality of service. Many recent studies have captured live Tor traffic in an effort to understand how Tor is used in practice [11, 22, 23], while other studies have launched large-scale de-anonymization attacks on real Tor users [18]. In response to this body of work, a debate within the research community has begun that is focused on establishing acceptable community standards and best practices for conducting such Tor studies in an ethical and legal manner [32], culminating in the development of the Tor Metrics Portal [7], a community-driven, public repository for aggregated, statistical Tor data [21].

These issues illustrate a potential hazard in live Tor experimentation: *In the absence of clearly articulated community standards and accepted norms for conducting such research, live Tor experimentation may raise ethical or legal concerns*.

A case for a whole-Tor network testbed. In summary, there has been a wide diversity of experimental approaches used in past Tor research, each having its own set of advantages and potential drawbacks. In an effort to (i) offer realism and fidelity to the dynamics of the live Tor network and (ii) provide the ability to make global changes to Tor clients or Tor routers without incurring any risk to live Tor users, we propose ExperimenTor, a whole-Tor network emulation-based testbed environment. In the next section, we outline the specifics of the proposed testbed and its accompanying toolkit, and

describe how they may be used to efficiently configure, run, and analyze Tor experiments.

4 Experimentation with ExperimenTor

In order to perform large-scale experiments with the Tor network in a manner that ensures safety to real Tor users while not sacrificing much realism, we present the design and implementation of ExperimenTor, a network emulation-based Tor testbed and accompanying experiment toolkit. In this section, we describe the challenges that are involved in replicating the salient features of the Tor network in an isolated testbed environment, present the high-level design of ExperimenTor, and detail ExperimenTor's implementation.

4.1 Challenges of Building a Tor Testbed

There have been many different approaches toward the goal of faithful Tor experimentation (see Section 3). Of the techniques that we surveyed, no prior work has utilized a large-scale Tor testbed for experimentation. This is because there are many significant challenges involved in the design and implementation of a large-scale Tor network testbed. We next enumerate and describe the most significant challenges.

Modeling the live Tor network is difficult. To construct a testbed that is faithful to the live Tor network, it is essential to accurately model the distribution of Tor router bandwidth. Since the Tor network is composed of volunteer-operated Tor routers, there is great diversity in the amount of bandwidth that each respective Tor router has available to devote to relaying traffic. If the distribution of bandwidth is not accurately modelled in the testbed, the results of experiments may not be useful or generalizable to the real Tor network. Similarly, it is necessary to accurately model the distribution of bandwidth that is available for entry guard, middle node, and exit node routers. For exit routers, it is also necessary to model the exit routers' various exit policies, which indicate the destination IP addresses and ports with which the exit router is willing to communicate.

In addition to accurate Tor router models, it is also necessary to accurately model Tor clients. Such models should accurately describe the number of Tor clients and their exit traffic in terms of the distribution of applications, number of connections, and traffic volume.

Need large-scale network emulation. In addition to faithfully modeling the dynamics of real Tor clients and Tor routers, it is necessary to accurately model the underlying network. Network emulation platforms such as Emulab [2] or DETER [1] have been built for this precise purpose; however, these emulation testbeds have limitations. First, both Emulab and DETER have limited computing and networking resources, which restricts the scale of experiments that can be run on these testbeds. In addition, both testbeds are shared resources, which introduces contention among researchers for experiment time.

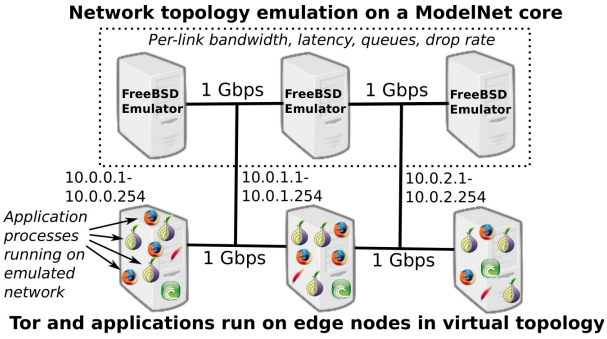

Figure 2: ExperimenTor system architecture

Should run native Tor and application code. Prior approaches to Tor experimentation have relied on simulation of various aspects of Tor's design. However, rather than re-implementing specific components of Tor, we wish to run the unmodified and complete Tor code in order to provide a higher degree of realism. In addition, running the real Tor code also reduces the likelihood of introducing (re-)implementation errors that might influence an experiment's results. It is also sometimes necessary to conduct experiments using real applications (e.g., web browsers and e-mail clients). Emulators should support the same SOCKS interface that enables Tor users to connect their applications to the Tor network.

Some network simulators such as `ns2` [5] provide *emulation modes* that support access to a live network; however, the inability to execute native applications makes them unsuitable for experimentation with Tor.

4.2 Meeting the Design Challenges

We next describe the details of ExperimenTor and, in particular, how our design overcomes the challenges of large-scale network emulation and accurate modeling of Tor routers, clients, and typical applications.

Large-scale network emulation with ModelNet. To enable large-scale network emulation on commodity hardware and operating systems, we build ExperimenTor on top of the ModelNet [36] network emulation platform. ModelNet allows the experimenter to build network topologies using standard network topology generation tools such as `inet` [3] and assign realistic bandwidth, latency, queue length, and other properties to links. Network emulation is performed by forwarding real application traffic through one or more *emulator* machines, which are responsible for emulating the specified network using a specialized kernel module. Applications run without modification[3] on *edge nodes* scattered

[3]Users set the `LD_PRELOAD` environment variable to enable applications to use ModelNet. The applications are unaware of ModelNet.

throughout the virtual network topology, but these applications are physically executed on commodity machines connected over a local area network to the emulator machine(s). More specifically, each virtual end-host is assigned its own virtual network interface in the 10.0.0.0/8 address space on one of the commodity machines connected to the emulator. An example ExperimenTor configuration is shown in Figure 2.

The main advantage of leveraging ModelNet in the design of ExperimenTor is that it allows the testbed to run native, unmodified Tor code with typical applications. In addition, ModelNet can be scaled to support more end-hosts than is currently possible with shared emulation platforms such as Emulab or DETER. Finally, because ModelNet runs on commodity hardware and operating systems, researchers can deploy their own private ExperimenTor testbed at low cost within their labs.

Modeling Tor routers using directory data. As described in Section 2, Tor uses a set of trusted directory authorities to distribute information about Tor routers. These directory authorities sign and publish information needed to reach Tor routers such as their IP addresses, listening ports, and public keys, and additionally advertise a significant amount of metadata about each Tor router. For instance, Tor clients can learn each router's sustained bandwidth capacity over the course of its operation, its status as an entry guard, and its exit policy.

Since this directory information is made publicly available through a standard HTTP interface, it is possible to fetch all Tor router metadata and effectively replicate the state of the Tor network by configuring Tor routers with the same distribution of entry guards, middle-only routers, and exit routers both by number of nodes and by bandwidth. In addition to replicating the state of the live Tor network, an experimenter might also use the directory information to scale the network size up or down in terms of number of Tor routers and aggregate Tor network bandwidth, while preserving the correct proportions of guard, middle-only, and exit nodes and bandwidths.

Modeling Tor clients and their traffic. Accurately modeling Tor clients is important for replicating the dynamics of real Tor network in a testbed. Such modeling presents a challenge, as little empirical data is available about the precise number of Tor clients and their behaviors. However, we can leverage publicly-available data sets on the estimated number of live Tor clients over time from the Tor Metrics Portal [7] and observations about the characteristics of live Tor clients' traffic obtained by McCoy *et al.*, who analyzed a live Tor exit router's traffic [22]. They observe that 92% of exit TCP connections are HTTP connections making up 57% of Tor's exit traffic volume, and 3% of exit connections come from BitTorrent file sharing and comprise 40% of Tor's exit traffic volume. Of the HTTP traffic, the vast majority appears to result from interactive web browsing, with only 3.5% of connections transporting over 1 MiB. These empirical observations can be used to model the distribution of client traffic in the testbed by connection and volume.

In addition, to model interactive web traffic, data sets [15] consisting of web request and response sizes, the number of web objects per page, and the time between a client's successive web requests can be leveraged to enhance the realism of the emulated clients. For maximum realism, live network traces can be used to generate emulated traffic that is faithful to the user, application, and network behaviors captured in the traces [33, 37].

4.3 ExperimenTor Details

We have built and deployed ExperimenTor prototypes at four research institutions in the United States and Canada. Our initial prototypes are relatively simple in their construction, consisting of a single FreeBSD 6.3 machine with the ModelNet emulator kernel module and a single "edge node" machine running Linux 2.6.32 for the Tor routers, clients, and application processes within the emulated topology.

In addition to the generalized ExperimenTor system architecture, we developed an extensive, publicly-available toolkit for configuring, running, and analyzing ExperimenTor experiments. We next describe the toolkit.

Generating realistic topologies. The first step before running experiments in the testbed is to generate a network topology and deploy it on the ModelNet emulator. To accurately model the real Tor network's distribution of router bandwidth, the configuration tool first obtains the router information from a Tor directory server and extracts each router's estimated sustained bandwidth capacity. Next, it maps each router to an end-host located within the virtual topology and assigns it a real bandwidth value. Finally, the toolkit assigns realistic network latencies to each end-host. Currently, the toolkit extends ModelNet's topology generation tools to map measured latencies from the King data set [14] onto the topology.

Configuring Tor routers and directories. After the topology has been created, it is necessary to deploy the Tor routers. The first step is to create directory servers for the testbed Tor deployment, which is accomplished by simply generating public/private key pairs for each directory server to be used in the testbed (five directories are sufficient to distribute the traffic load of Tor client requests). To ensure that all emulated Tor clients and routers use the testbed's directories instead of the real directories, it is necessary to record each testbed directory's public key fingerprint and distribute these to the Tor routers and clients in the testbed. Also, similar to how bandwidths from Tor routers are assigned to Tor routers in the testbed, entry guard status and exit poli-

cies may also be assigned in the same way. Each router, after being configured, joins the Tor network testbed.

Configuring Tor clients and applications. Our tools also automatically configure a desired number of Tor clients and their applications. Our initial prototype supports HTTP clients that fetch web objects of various sizes, to approximate either interactive web users who engage in web surfing or bulk downloaders. It is also possible to configure the testbed to support more complex traffic generation, for example, using one of the models discussed in Section 4.2. To help generate HTTP traffic, the testbed runs a lightweight multi-threaded web server (`lighttpd` [4]) to serve web objects to clients. In addition, BitTorrent file shares may be configured and a desired fraction of clients can run BitTorrent-over-Tor.

Running and analyzing experiments. The ExperimenTor toolkit contains routines for configuring the network topology, directory servers, Tor routers, and Tor clients. Additionally, the toolkit also manages the execution of experiments: a single master script creates an instance of the testbed, runs the experiments, stops them after a specified amount of time, and collects data that were generated during the course of the experiment. Additionally, the toolkit provides scripts for synthesizing and analyzing the collected results (for example, to plot throughput and latency).

5 Lessons Learned

We next describe our early experiences with ExperimenTor and discuss the limitations of Tor emulation.

5.1 Initial Experience with ExperimenTor

Evaluating design changes. We have applied ExperimenTor to evaluate the effects of performing *link-based router* selection [30] in Tor. Such an evaluation would be difficult to perform either through analysis or simulation. The choice of routers affects network congestion, packet/cell arrival rates, and network-, OS- and application-layer queuing delays. Developing an analytical model that considers anticipated as well as *unforeseen* effects at the network and application layers is likely intractable; we know of no simulators that support this complexity.

A more practical approach is to experiment with the actual Tor code base. Testing alternative router selection policies using the deployed Tor network produces (by definition) accurate results, but is limited to the study of small-scale adoptions. That is, it measures the effects of only the experimental (altered) Tor nodes on the network. ExperimenTor offers more flexibility by enabling us to investigate the effects of various adoption rates (up to 100%) on Tor. ExperimenTor has supported our ongoing research by allowing us to efficiently "deploy" our router selection techniques and measure their performance in many client and network configurations.

Diagnosing performance problems in Tor. In addition to evaluating changes to aspects of Tor's design, we have also used the testbed to help diagnose and fix performance problems in Tor [8]. From experiments run on the testbed, we observed that Tor handles congestion poorly and uses an end-to-end window-based flow control scheme that offers suboptimal flow control; both issues result in unnecessarily high delays for end-users.

To improve Tor's performance, we developed and evaluated new congestion control and flow control strategies, using ExperimenTor to analyze the proposals with whole-Tor network experiments that realistically model the distribution of Tor router bandwidth and client traffic. These experiments led us to conclude that the proposed improvements likely would offer similar improvements when deployed at-scale on the live Tor network.

5.2 Limitations

As with any emulator, ExperimenTor cannot perfectly represent all characteristics of the deployed Tor network. Although ExperimenTor does not impose any fixed scalability limitations, the emulated network is effectively restricted by its available computational and network resources. Since ExperimenTor supports multiple "edge node" machines, constructing large topologies is not difficult. However, the size of the emulation is effectively bound by the network capacity of the ModelNet instance. In our experimentation, we have successfully scaled ExperimenTor to 1,000 Tor nodes using one "edge node" machine and one ModelNet emulator machine connected by a 1 Gbps switch.

Since ExperimenTor cannot emulate Tor's (estimated) 250,000 daily users [20], experiments must down-sample their behavior from the deployed Tor network. In general, it is unclear how to best "scale down" traffic characteristics to an emulated environment. Although intuitive approaches such as dividing the usage observed on the real Tor network by the emulation's scaling factor provide some level of realism, the decrease in absolute traffic volume has sometimes subtle consequences to performance (for example, by incurring shorter application-layer queueing delays at Tor routers). Techniques for best capturing "realism" in emulation environments remains an open challenge.

Additionally, how well ExperimenTor reflects the actual Tor network is dependent on the investigators' ability to accurately represent the configurations of real Tor users. Although ExperimenTor runs unmodified instances of Tor, correctly emulating the diverse versions and configurations of Tor users – proportionately to those employed in the real network – is another challenge.

Finally, due to its reliance on synthetic behavior, ExperimenTor is ill-suited for measurement studies of traffic characteristics or real-world Tor usage.

6 Conclusion and Future Work

ExperimenTor is a valuable testbed and toolkit for understanding Tor's security and performance, as well as evaluating changes to Tor's design. It allows researchers to perform large-scale experiments that faithfully reproduce many of the important features of the Tor network, including Tor client traffic generation and Tor router topologies, on a dedicated testbed that requires minimal hardware. With ExperimenTor, researchers can adjust client and router behaviors along with network configurations to directly observe the effects of adoption rates up to 100% on Tor. ExperimenTor is currently being used to support ongoing research at four institutions and has been released to the community as open source software (available at `http://crysp.org/software/exptor`).

To date, ExperimenTor prototypes have been built and tested with a single ModelNet emulator. We are extending the prototypes to multiple emulators to realize testbeds that scale to the size of the current Tor network and beyond. We are also working on improving traffic generation tools that will enable investigators to configure proportions of traffic in an intuitive and standardized format. Finally, while we recognize that there are limitations, we argue that ExperimenTor is an efficient, cost-effective, safe, and realistic testbed for studying Tor.

Acknowledgements

We thank our shepherd, Sean Peisert, and the anonymous reviewers for their helpful comments and suggestions. We also thank Ken Yocum for his invaluable guidance with ModelNet, Chris Wacek for assisting with development, and Stefan Savage and Geoffrey M. Voelker for informative discussions about network emulation and for introducing us to ModelNet.

This work was supported by NSF award DGE-0841423, DARPA contract N66001-11-C-4020, and a CCC-CRA-NSF Computing Innovation Fellowship. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] DETERlab testbed. `http://www.isi.edu/deter`.

[2] Emulab. `http://www.emulab.net`.

[3] Inet. `http://topology.eecs.umich.edu/inet`.

[4] lighttpd. `http://www.lighttpd.net`.

[5] The network simulator. `http://www.isi.edu/nsnam/ns`.

[6] SSFNet. `http://www.ssfnet.org/homePage.html`.

[7] Tor metrics portal. `https://metrics.torproject.org`.

[8] AlSabah, M., Bauer, K., Goldberg, I., Grunwald, D., McCoy, D., Savage, S., and Voelker, G. M. Defenestra-Tor: Throwing out windows in Tor. In *PETS* (2011).

[9] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-resource routing attacks against Tor. In *WPES* (2007).

[10] Borisov, N., Danezis, G., Mittal, P., and Tabriz, P. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In *CCS* (2007).

[11] Chaabane, A., Manils, P., and Kaafar, M. A. Digging into anonymous traffic: A deep analysis of the Tor anonymizing network. In *Conference on Network and System Security* (2010).

[12] Chakravarty, S., Stavrou, A., and Keromytis, A. D. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *ESORICS* (2010).

[13] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *USENIX Security* (2004).

[14] Gummadi, K. P., Saroiu, S., and Gribble, S. D. King: Estimating latency between arbitrary Internet end hosts. *SIGCOMM Comput. Commun. Rev. 32*, 3 (2002).

[15] Hernández-Campos, F., Jeffay, K., and Smith, F. D. Tracking the evolution of web traffic: 1995-2003. In *MASCOTS* (2003).

[16] Hopper, N., Vasserman, E. Y., and Chan-Tin, E. How much anonymity does network latency leak? In *CCS* (2007).

[17] Jansen, R., Hopper, N., and Kim, Y. Recruiting new Tor relays with BRAIDS. In *CCS* (2010).

[18] Le Blond, S., Manils, P., Abdelberi, C., 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 *LEET* (2011).

[19] Levine, B. N., Reiter, M. K., Wang, C., and Wright, M. K. Timing attacks in low-latency mix-based systems. In *FC* (2004).

[20] Loesing, K. Measuring the Tor network: Evaluation of client requests to the directories. *Tor Project Technical Report* (2009).

[21] Loesing, K., Murdoch, S., and Dingledine, R. A case study on measuring statistical data in the Tor anonymity network. In *WECSR* (2010).

[22] McCoy, D., Bauer, K., Grunwald, D., Kohno, T., and Sicker, D. Shining light in dark places: Understanding the Tor network. In *PETS* (2008).

[23] Mulazzani, M., Huber, M., and Weippl, E. Tor HTTP usage and information leakage. In *IFIP CMS* (2010).

[24] Murdoch, S. J., and Watson, R. N. M. Metrics for security and performance in low-latency anonymity systems. In *PETS* (2008).

[25] Ngan, T.-W. J., Dingledine, R., and Wallach, D. S. Building Incentives into Tor. In *FC* (2010).

[26] O'Gorman, G., and Blott, S. Large scale simulation of Tor: Modelling a global passive adversary. In *ASIAN* (2007).

[27] Paxson, V., and Floyd, S. Wide-area traffic: The failure of Poisson modeling. *SIGCOMM Comput. Commun. Rev. 24* (October 1994), 257–268.

[28] Peterson, L., Pai, V., Spring, N., and Bavier, A. Using PlanetLab for network research: Myths, realities, and best practices. PDN-05-028, June 2005.

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

[30] Sherr, M., Blaze, M., and Loo, B. T. Scalable link-based relay selection for anonymous routing. In *PETS* (2009).

[31] Snader, R., and Borisov, N. A tune-up for Tor: Improving security and performance in the Tor network. In *NDSS* (2008).

[32] Soghoian, C. Enforced community standards for research on users of the Tor anonymity network. In *WECSR* (2011).

[33] Sommers, J., and Barford, P. Self-configuring network traffic generation. In *IMC* (2004).

[34] Syverson, P. F., Goldschlag, D. M., and Reed, M. G. Anonymous Connections and Onion Routing. In *IEEE Symposium on Security and Privacy* (1997).

[35] Tang, C., and Goldberg, I. An improved algorithm for Tor circuit scheduling. In *CCS* (2010).

[36] Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostić, D., Chase, J., and Becker, D. Scalability and accuracy in a large-scale network emulator. In *OSDI* (2002).

[37] Vishwanath, K. V., and Vahdat, A. Realistic and responsive network traffic generation. In *SIGCOMM* (2006).