Once is Never Enough: Foundations for Sound Statistical Inference in Tor Network Experimentation

Rob Jansen, U.S. Naval Research Laboratory
Justin Tracey, University of Waterloo
Ian Goldberg, University of Waterloo
Anonymous Communication with Tor

- Separates *identification* from *routing*
- Provides unlinkable communication
- Protects user privacy and safety online
Anonymous Communication with Tor

- Separates identification from routing
- Provides unlinkable communication
- Protects user privacy and safety online

Research to Improve Tor Performance

- A faster Tor means privacy is accessible to more humans
- Many ways to improve performance
- Need methods and tools to help us safely conduct Tor experiments
- We want experimentation results to be accurate and dependable so they can help inform real world decisions
Steps to Run Tor Test Network Experiments

(1) Model a Tor test network configuration

(2) Use Shadow\(^1\) to run Tor experiments

(3) Analyze and compare experimental results
Steps to Run Tor Test Network Experiments

(1) Model a Tor test network configuration
   • How many clients? How many relays? How to sample relays?
   • What are their node characteristics (location, bandwidth, rate limits, relay position)?

(2) Use Shadow\(^1\) to run Tor experiments

(3) Analyze and compare experimental results
Steps to Run Tor Test Network Experiments

(1) Model a Tor test network configuration
   - How many clients? How many relays? How to sample relays?
   - What are their node characteristics (location, bandwidth, rate limits, relay position)?
     😞 Previously done by considering the state from a single consensus (1 hour)
     😊 We consider the state of the network over time when sampling relays for test networks

(2) Use Shadow\(^1\) to run Tor experiments

(3) Analyze and compare experimental results

\(^1\)shadow.github.io
Steps to Run Tor Test Network Experiments

(1) Model a Tor test network configuration
   • How many clients? How many relays? How to sample relays?
   • What are their node characteristics (location, bandwidth, rate limits, relay position)?
     😞 Previously done by considering the state from a single consensus (1 hour)
     😊 We consider the state of the network over time when sampling relays for test networks

(2) Use Shadow\(^1\) to run Tor experiments
     😞 Large RAM and computational requirements
     😊 We reduced RAM usage by 64% and run time by 94%, enabling larger-scale experiments

(3) Analyze and compare experimental results

\(^1\)shadow.github.io
Steps to Run Tor Test Network Experiments

(1) Model a Tor test network configuration
   • How many clients? How many relays? How to sample relays?
   • What are their node characteristics (location, bandwidth, rate limits, relay position)?
     😞 Previously done by considering the state from a single consensus (1 hour)
     😊 We consider the state of the network over time when sampling relays for test networks

(2) Use Shadow\(^1\) to run Tor experiments
    😞 Large RAM and computational requirements
    😊 We reduced RAM usage by 64% and run time by 94%, enabling larger-scale experiments

(3) Analyze and compare experimental results
    😞 Previously, one experiment done with vanilla Tor and each research variant
    😊 We present methods for quantifying the statistical significance across a set of experiments

\(^1\)shadow.github.io
Outline
(1) Model a Tor test network
(2) Use Shadow to run Tor experiments
(3) Analyze and compare experimental results
Step (1) Modeling a Tor Test Network

(1) We use data available on metrics.torproject.org to make informed modeling decisions

- The network has high churn
Step (1) Modeling a Tor Test Network

(1) We use data available on metrics.torproject.org to make informed modeling decisions

- The network has high churn
- We consider the state of the network over time in order to capture network diversity
- We sample the true relay distribution using the following weights:
  - Relay uptime over the modeling period
  - Relay consensus bandwidth weight
Step (1) Modeling a Tor Test Network

(1) We use data available on metrics.torproject.org to make informed modeling decisions

- The network has high churn
- We consider the state of the network over time in order to capture network diversity
- We sample the true relay distribution using the following weights:
  - Relay uptime over the modeling period
  - Relay consensus bandwidth weight

(2) Simulate multiple users in each Tor client process to save RAM
Outline

(1) Model a Tor test network
(2) Use Shadow to run Tor experiments
(3) Analyze and compare experimental results
Step (2) Use Shadow to Run Tor Experiments

- Conducted performance audit of Shadow using the Linux perf tool
  - Fixed several performance bottlenecks
  - Added feature to shorten Tor bootstrapping
  - Enabled run-time optimizations

- Improved Shadow networking
  - Fixed non-determinism bugs
  - Improved network stack
Step (2) Use Shadow to Run Tor Experiments

- Conducted performance audit of Shadow using the Linux perf tool
  - Fixed several performance bottlenecks
  - Added feature to shorten Tor bootstrapping
  - Enabled run-time optimizations

- Improved Shadow networking
  - Fixed non-determinism bugs
  - Improved network stack

### Table 2: Scalability improvements over the state of the art

<table>
<thead>
<tr>
<th>Model</th>
<th>Scale ( \times )</th>
<th>RAM</th>
<th>Bootstrap Time</th>
<th>Total Time</th>
<th>( \Omega )°</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCS'18 [38]†</td>
<td>31%</td>
<td>2.6 TiB</td>
<td>3 days, 11 hrs.</td>
<td>35 days, 14 hrs.</td>
<td>1850</td>
</tr>
<tr>
<td>This work†</td>
<td>31%</td>
<td>932 GiB</td>
<td>17 hrs.</td>
<td>2 days, 2 hrs.</td>
<td>79</td>
</tr>
<tr>
<td>This work‡</td>
<td>100%</td>
<td>3.9 TiB</td>
<td>2 days, 21 hrs.</td>
<td>8 days, 6 hrs.</td>
<td>310</td>
</tr>
</tbody>
</table>

* \(31\%: \approx 2k\) relays and \(\approx 250k\) users; \(100\%: 6,489\) relays and \(792k\) users

\(\Omega\): ratio of real time / simulated time in steady state (after bootstrapping)

† Using 8×10-core Intel Xeon E7-8891v2 CPUs each running @3.2 GHz.
‡ Using 8×18-core Intel Xeon E7-8860v4 CPUs each running @2.2 GHz.
Step (2) Use Shadow to Run Tor Experiments

- Conducted performance audit of Shadow using the Linux perf tool
  - Fixed several performance bottlenecks
  - Added feature to shorten Tor bootstrapping
  - Enabled run-time optimizations

- Improved Shadow networking
  - Fixed non-determinism bugs
  - Improved network stack

<table>
<thead>
<tr>
<th>Table 2: Scalability improvements over the state of the art</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>CCS'18 [38]†</td>
</tr>
<tr>
<td>This work†</td>
</tr>
<tr>
<td>This work‡</td>
</tr>
</tbody>
</table>

* 31%: \( \approx \)2k relays and \( \approx \)250k users; 100%: 6,489 relays and 792k users

\( \Omega \): ratio of real time / simulated time in steady state (after bootstrapping)

† Using 8×10-core Intel Xeon E7-8891v2 CPUs each running @3.2 GHz.
‡ Using 8×18-core Intel Xeon E7-8860v4 CPUs each running @2.2 GHz.
Step (2) Use Shadow to Run Tor Experiments

- Conducted performance audit of Shadow using the Linux perf tool
  - Fixed several performance bottlenecks
  - Added feature to shorten Tor bootstrapping
  - Enabled run-time optimizations

- Improved Shadow networking
  - Fixed non-determinism bugs
  - Improved network stack

<table>
<thead>
<tr>
<th>Model</th>
<th>Scale $s^*$</th>
<th>RAM</th>
<th>Bootstrap Time</th>
<th>Total Time</th>
<th>$\Omega^\circ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCS'18 [38]†</td>
<td>31%</td>
<td>2.6 TiB</td>
<td>3 days, 11 hrs</td>
<td>35 days, 14 hrs</td>
<td>1850</td>
</tr>
<tr>
<td>This work†</td>
<td>31%</td>
<td>932 GiB</td>
<td>17 hrs.</td>
<td>2 days, 2 hrs.</td>
<td>79</td>
</tr>
<tr>
<td>This work‡</td>
<td>100%</td>
<td>3.9 TiB</td>
<td>2 days, 21 hrs</td>
<td>8 days, 6 hrs.</td>
<td>310</td>
</tr>
</tbody>
</table>

* 31%: $\approx$2k relays and $\approx$250k users; 100%: 6,489 relays and 792k users
$\Omega$: ratio of real time / simulated time in steady state (after bootstrapping)
† Using 8×10-core Intel Xeon E7-8891v2 CPUs each running @3.2 GHz.
‡ Using 8×18-core Intel Xeon E7-8860v4 CPUs each running @2.2 GHz.

64% reduction in RAM usage
94% reduction in run time
Step (2) Use Shadow to Run Tor Experiments

- Conducted performance audit of Shadow using the Linux perf tool
  - Fixed several performance bottlenecks
  - Added feature to shorten Tor bootstrapping
  - Enabled run-time optimizations

- Improved Shadow networking
  - Fixed non-determinism bugs
  - Improved network stack

| Table 2: Scalability improvements over the state of the art |
|---------------------------------|----------|---------------|---------------|-------------|-------------|
| **Model** | **Scale $^*$** | **RAM** | **Bootstrap Time** | **Total Time** | **$\Omega^\circ$** |
| CCS’18 [38]† | 31% | 2.6 TiB | 3 days, 11 hrs | 35 days, 14 hrs. | 1850 |
| This work† | 31% | 932 GiB | 17 hrs. | 2 days, 2 hrs. | 79 |
| This work‡ | 100% | 3.9 TiB | 2 days, 21 hrs. | 8 days, 6 hrs. | 310 |

*$31\%$: $\approx2k$ relays and $\approx250k$ users; $100\%$: $6,489$ relays and $792k$ users

$\Omega$: ratio of real time / simulated time in steady state (after bootstrapping)

† Using $8 \times 10$-core Intel Xeon E7-8891v2 CPUs each running @3.2 GHz.
‡ Using $8 \times 18$-core Intel Xeon E7-8860v4 CPUs each running @2.2 GHz.

64% reduction in RAM usage
94% reduction in run time
Supports larger test networks
Outline

(1) Model a Tor test network
(2) Use Shadow to run Tor experiments
(3) Analyze and compare experimental results
Running experiments involves two levels of sampling:

1. Sampling a test network model at some scale ≤ 100%
2. Simulating the sampled test network with a seed

Previous work uses one simulation for each research variant

• Ignores sampling error
Running experiments involves two levels of sampling:

1. Sampling a test network model at some scale ≤ 100%
2. Simulating the sampled test network with a seed

Previous work uses one simulation for each research variant

- Ignores sampling error

One simulation is never enough

- We need repeated sampling of test networks (not just sim seeds)
- We quantify the sampling error by computing CIs over empirical CDFs
- Allows us to make statistical arguments for the observed results
Each simulation in each test network produces an empirical CDF for our metric of interest.

(E.g., time to last byte)
Estimating the True CDF with CIs

Each simulation in each test network produces an empirical CDF for our metric of interest. We use the mean to estimate the true CDF, and sampling and measurement error to compute CIs.
Case Study: Tor Usage and Performance

Demonstrate how to apply our methods with an example

Hypothesis:
- Increasing user traffic load by 20% will decrease Tor performance for existing clients

Experiment setup
- 100% and 120% traffic loads
- 1%, 10%, and 30% scale factors
- 420 total simulations

How does network scale affect the conclusions we can draw from the results?

<table>
<thead>
<tr>
<th>Scale</th>
<th>Load</th>
<th>Number of Simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>100%</td>
<td>100</td>
</tr>
<tr>
<td>1%</td>
<td>120%</td>
<td>100</td>
</tr>
<tr>
<td>10%</td>
<td>100%</td>
<td>100</td>
</tr>
<tr>
<td>10%</td>
<td>120%</td>
<td>100</td>
</tr>
<tr>
<td>30%</td>
<td>100%</td>
<td>10</td>
</tr>
<tr>
<td>30%</td>
<td>120%</td>
<td>10</td>
</tr>
</tbody>
</table>
Scale = 1%

CIs overlap even with 100 sims
CIs Inform the Analysis of Results

Scale = 1%
CIs overlap even with 100 sims

Scale = 10%
CIs are close with 5 sims, but separate with 10 sims
CIs Inform the Analysis of Results

Scale = 1%
CIs overlap even with 100 sims

Scale = 10%
CIs are close with 5 sims, but separate with 10 sims

Scale = 30%
CIs clearly separate with either 5 or 10 sims
CIs Inform the Analysis of Results

Scale = 1%
CIs overlap even with 100 sims

Scale = 10%
CIs are close with 5 sims, but separate with 10 sims

Scale = 30%
CIs clearly separate with either 5 or 10 sims

More simulations needed at smaller scales, fewer at larger scales to reach a certain CI precision
## Summary

<table>
<thead>
<tr>
<th>Primary Contributions</th>
<th>Main Results</th>
</tr>
</thead>
</table>
| (1) New methods for constructing Tor test networks considering the state of the network over time rather than at a static point | We create many test Tor networks in Shadow  
   ➔ a Tor network with up to 6,489 relays  
   ➔ traffic of up to 792k simultaneously active users |
| (2) New/improved experimentation tools, optimized to run Tor faster and at larger scales than previously possible | We enhanced Shadow to reduce:  
   ➔ RAM usage by 64%  
   ➔ run time by 94% |
| (3) New methodology for conducting statistical inference using results collected from experiments in smaller-scale Tor networks | Running multiple simulations in independently sampled Tor networks  
   ➔ necessary for statistical significance |
| (4) Demonstrated how to apply our methodologies to conduct sound Tor performance research | To reach a desired precision requires:  
   ➔ more simulations in smaller-scale networks  
   ➔ fewer simulations in larger-scale networks |

**Artifacts:** [https://neverenough-sec2021.github.io](https://neverenough-sec2021.github.io)  
**Contact:** rob.g.jansen@nrl.navy.mil, robgjansen.com, @robgjansen