Privacy-Preserving Dynamic Learning of Tor Network Traffic

Akhil Thomas
XXXXXXX , University of Cincinnati

ABSTRACT

The goal of this paper is to better understand Tor traffic and its characteristics so that we can more accurately generate traffic in private Tor networks and simulators. Moreover, it provides a more powerful method for the future researches on Tor security and performance. In this work, we design a novel technique for dynamically learning Tor network traffic models using hidden Markov modeling and privacy-preserving measurement techniques.

ACM Reference Format:
Akhil Thomas. 2018. Privacy-Preserving Dynamic Learning of Tor Network Traffic. In *Proceedings of ACM XXXXX, XX, XX, XX, XX 2018 (ACM XXXXX'18)*, 7 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

TOR (The Onion Router) is an anonymous communication system to stay safe while browsing the internet and communicating online. In a TOR network multiple set of encryptions can be done on each path to increase security, nested like an onion. Each relay has only data of previous one, no relays know the complete path of transmission. The TOR network distributes your transactions over several places on the Internet, so no single point can link to your destination.

2 MEASURING TOR

Measuring of a Tor network also includes analysis of (a) The number of active and inactive clients. (b) The number of active and inactive circuits and their distribution per client. (c) The number and types of streams and their distribution per circuit. (d) The number of inbound and outbound bytes and their distribution per stream. For Measuring TOR, we used the measurement tool called PrivCount (Privacy Preserving Counting). In this type of measuring we achieve forward privacy during measurement. That is the adversary cannot learn the state of the measurement before time of compromise. It also provides differential privacy of the results by preventing confirmation of the actions of a specific user given the output.

2.1 ARCHITECTURE

The set up for PrivCount deployment consist of 1 tally server, 3 share keepers, and 17 data collectors each connecting to a distinct Tor relay (run on 6 data collectors for exit and 11 data collectors for non-exit relays or entry relays). An overview of the deployment is shown in Figure 1.

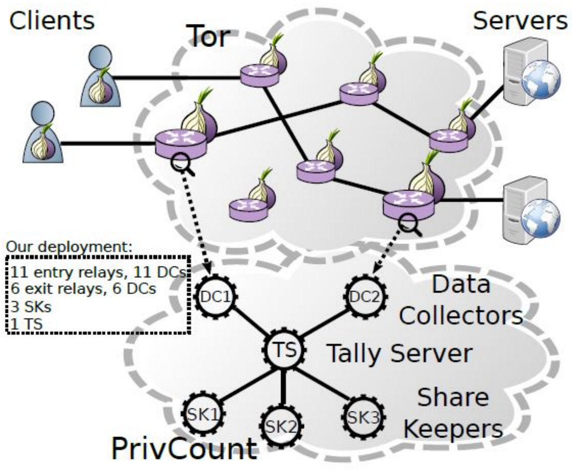


Figure 1: A simplified view of the 17 Tor relays (11 entry, 6 exit) and 21 PrivCount nodes (17 DCs, 3 SKs, and 1 TS) used in our deployment. See Table 2 for the total weights of our entry and exit relays during the periods in which they were used to collect measurements.

Figure 1: apparatus

Table 1

The PrivCount nodes and the Tor relays were run by 3 different operators in 3 different countries: Canada, France, and the United States

2.2 Measuring Phase

The measuring is done at two phases, one is General Measurement phase (overall information of Tor) which include analysis of total number of clients, circuits, streams, and bytes and their breakdown into traffic classes. The distribution of these data is also studied here. Second phase is traffic model measurement phase in which we measure Hidden Markov Models for stream and packet. The purpose of each measurement and the mean combined entry or exit consensus weight of our deployment during each measurement is summarized in Table 2.

3 MEASURING TRAFFIC STATISTICS

PrivCount counts the number of unique clients at the end of every 10 minutes period in order to limit the amount of time that client IP addresses are stored in RAM. For consistency, we report all measurements as 10 minute means by dividing the daily count total by 144 (the number of 10 minute periods during each 24-hour measurement period).

Table 2: Mean Combined Consensus Bandwidth Weights of all Relays in PrivCount Deployment during Measurements

	#	Purpose of Measurement	Weight*
Entry	1	Total clients and circuits	1.26%
	2	Circuits per client	1.13%
Exit	3	Total circuits and streams	2.13%
	4	Total bytes on streams	2.14%
	5	Streams per circuit, bytes per stream (All)	2.27%
	6	Streams per circuit, bytes per stream (Web)	2.29%
	7	Streams per circuit, bytes per stream (Other)	2.54%
	8	Hidden Markov packet model	1.49%
	9	Hidden Markov stream model	1.33%

* Weights correspond to the relay measurement position.

Table 2: weights

3.1 Entry Statistics

Entry relays have a limited view of Tor traffic types since they do not directly observe stream information. However, they can observe clients and circuits, and hence, the distribution of circuits per client. This information is helpful in producing accurate Tor client models. The results from our measurement of the total number of unique clients and circuits in a 10-minute interval are shown in Table 3, as well as the breakdown into active and inactive classes.

Table 3: The 10 minute mean entry statistics collected during period 1 (see Table 2). The error associated with the addition of privacy-preserving noise is shown with 95% confidence.

Statistic	Count ($\times 10^3$)	% of Total
Unique Clients	14.01 ± 0.513 (3.66%)	–
Active	9.978 ± 0.347 (3.48%)	71.2% ± 3.60%
Inactive	3.395 ± 0.185 (5.45%)	24.2% ± 1.59%
Circuits	679.3 ± 3.64 (0.535%)	–
Active	287.8 ± 2.94 (1.02 %)	42.4% ± 0.488%
Inactive	391.5 ± 2.15 (0.548%)	57.6% ± 0.441%

Table 3: entry statistics

We see from Table 3 that our entry relays observed 14,010 unique clients per 10 minute period on average, 71.2 percent of which are active while 24.2 percent are inactive. Our entry relays observed 679,300 circuits during an average 10 minutes, 42.4 percent of which are active and 57.6 percent of which are inactive. Although initially surprising, we speculate that the large number of inactive clients may be due to users who have Tor browser open but are not using it. We believe that the large number of inactive circuits are caused by Tor clients preemptively building circuits (part of TorâĂŹs design) that are never used.

The number of active and inactive clients are observed to understand how clients build circuits. The results are shown in Figure 2.

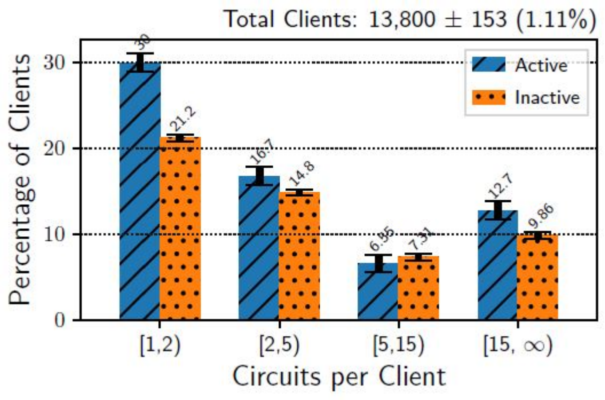


Figure 2: The 10 minute mean number of circuits per client collected during period 2 (see Table 2). The 10 minute mean number of unique clients was 13,800 ± 153 (1.11%) with 95% confidence.

Figure 2: Clients

For both types of circuits, we observed that a single circuit per client is the most common, followed by two, three, or four circuits per client. Although we observed that 12.7 and 9.86 percent of clients build 15 or more active and inactive circuits, respectively. These results indicate that most Tor users only use a handful of circuits in an average 10 minutes, suggesting that many Tor Browser users are only lightly browse the web. Note that about 34 and about 47 percent of clients build zero active and inactive circuits, respectively, in an average 10 minute period, which again may be caused by idle users.

3.2 Exit Statistics

Unlike entry relays, exit relays are unable to observe clients but can observe streams. Therefore, exits have access to traffic meta-data such as the port to which a stream connects. PrivCount classifies streams to port 80 or 443 as web, streams to common file sharing ports as file sharing, and streams to unclassified ports as other.

Table 4: The 10 minute mean exit circuit statistics collected during period 3 (see Table 2). The error associated with the addition of noise is shown with 95% confidence.

	Circuit Stat.	Count ($\times 10^3$)	% of Total
	Total	61.34 ± 2.20 (3.59%)	–
Total	*Active*	31.78 ± 1.06 (3.35%)	51.8% ± 2.54%
	Inactive	28.28 ± 1.14 (4.02%)	46.1% ± 2.48%
Active	*Web*	28.15 ± 0.93 (3.29%)	88.6% ± 4.16%
	FileSharing	0.850 ± 0.02 (2.66%)	2.68% ± 0.114%
	Other	4.76 ± 0.12 (2.44%)	15.0% ± 0.621%

Table 4: exit circuit

Results for our exit circuit statistics are shown in Table 4. These results show that 52.8 percent of circuits are active while 46.1 percent are inactive, which is comparable to our entry circuit statistics. We do observe about 10 times fewer circuits overall on our exits than on our entries (61,340 exit circuits compared to 679,300 entry circuits), which could be attributed to internal Tor network circuits (e.g., directory, bandwidth test, and onion service circuits) that do not utilize exit relays, circuits that fail before reaching the exit, or a bug in PrivCount. We observed that 88.6 percent of the active circuits carry web traffic, whereas only 2.68 percent carry traffic to well-known file sharing ports and 15 percent carry traffic on other ports.

Table 5: The 10 minute mean exit stream statistics collected during period 3 (see Table 2). The error associated with the addition of noise is shown with 95% confidence.

	Stream Stat.	Count ($\times 10^3$)	% of Total
	Total	263.5 ± 11.6 (4.39%)	–
Total	*Web*	238.4 ± 10.4 (4.38%)	90.4% ± 5.61 %
	FileSharing	1.374 ± 0.066 (4.80%)	0.52% ± 0.034%
	Other	19.91 ± 1.06 (5.34%)	7.56% ± 0.523%

Table 5: exit stream statistics

Results for our exit stream statistics are shown in Table 5. Again, we observed that a large majority of streams (90.4 percent) carry traffic to web ports, while an insignificant number of streams (0.52 percent) carry traffic to known file sharing ports and a small number of streams (7.56 percent) carry other traffic.

Table 6: The 10 minute mean exit byte statistics collected during period 4 (see Table 2). The error associated with the addition of noise is shown with 95% confidence.

	Byte Stat.	Count ($\times 2^{20}$)	% of Total
	Total	25,587 ± 2 (0.01%)	–
Total	*Inbound*	23,914 ± 2 (0.01%)	93.5% ± 0.009%
	Outbound	1,672 ± 0.1 (0.01%)	6.53% ± 0.001%
Total	*Web*	18,547 ± 1 (0.01%)	72.5% ± 0.007%
	FileSharing	263 ± 0.01 (0.002%)	1.03% ± 0.001%
	Other	6,744 ± 0.5 (0.01%)	26.4% ± 0.003%
Inbnd.	*Web*	17,569 ± 1 (0.01%)	73.5% ± 0.007%
	FileSharing	212 ± 0.01 (0.002%)	0.89% ± 0.001%
	Other	6,106 ± 0.4 (0.01%)	25.5% ± 0.003%
Outbnd.	*Web*	997 ± 0.03 (0.003%)	58.5% ± 0.003%
	FileSharing	51 ± 0.001 (0.003%)	3.08% ± 0.001%
	Other	637 ± 0.06 (0.01%)	38.1% ± 0.004%

Table 6: exit byte statistics

Results for our exit byte statistics are shown in Table 6. Over the 24 hour measurement period, the relays in our PrivCount deployment transferred 25 GB of exit stream data in total. Of the traffic, 93.5 percent was inbound, i.e., forwarded toward the circuit initiator; the remaining 6.53 percent was outbound, i.e., forwarded toward the Tor-external service. Of the inbound traffic, 73.5 percent was for web ports 80 and 443, while 0.89 percent was for default file sharing ports and 25.5 percent was for other ports. The share of traffic that is web-related fell to just 58.5 percent of outbound traffic, while traffic on file sharing ports rose to 3.08 percent and traffic on other ports rose to 38.1 percent.

The conclusion from the exit measurements We conclude from our exit measurements that an unsurprisingly large majority of circuits (88.6 percent), streams (90.4 percent), and bytes (72.5 percent) are associated with web ports 80 and 443, while most of the remaining traffic is associated with other ports. Based on our observations, we conclude that traffic to default file sharing ports is relatively insignificant. For this reason, we focus on web and other traffic in the remainder of our exit measurements.

3.3 Traffic Distributions

To get a better understanding of how clients build streams and transfer bytes, we now explore distributions of streams-per-circuit and bytes-per-stream using PrivCount histogram counters.

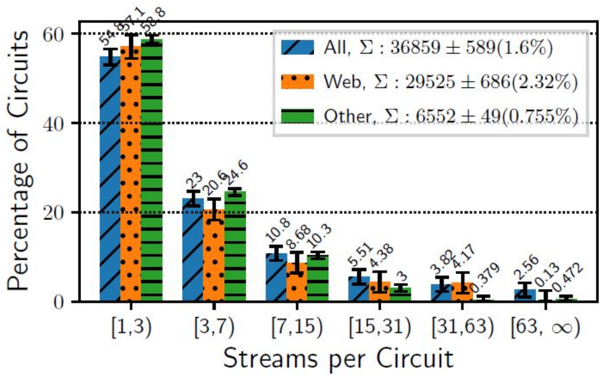


Figure 3: The distributions of streams per active circuit collected during periods 5, 6, and 7 (see Table 2). The total number of circuits of each type is shown in the legend with 95% confidence.

Figure 3: streams

The distribution of the number of streams per active circuit is shown in Figure 3. It was observed that about 55 to 59 percent of active circuits carry only one or two streams and about 21 to 25 percent of circuits carry 3 to 6 streams. Another 9 to 11 percent of circuits carry 7-14 streams, and about 4-11 percent carry 15 or more streams. This suggests that Tor Browser users may experience the web differently than non-Tor users.

The distribution of the number of bytes per stream is shown in Figure 4, with inbound bytes shown in Figure 4a and outbound bytes shown in Figure 4b. We observed that about 70 to 80 percent

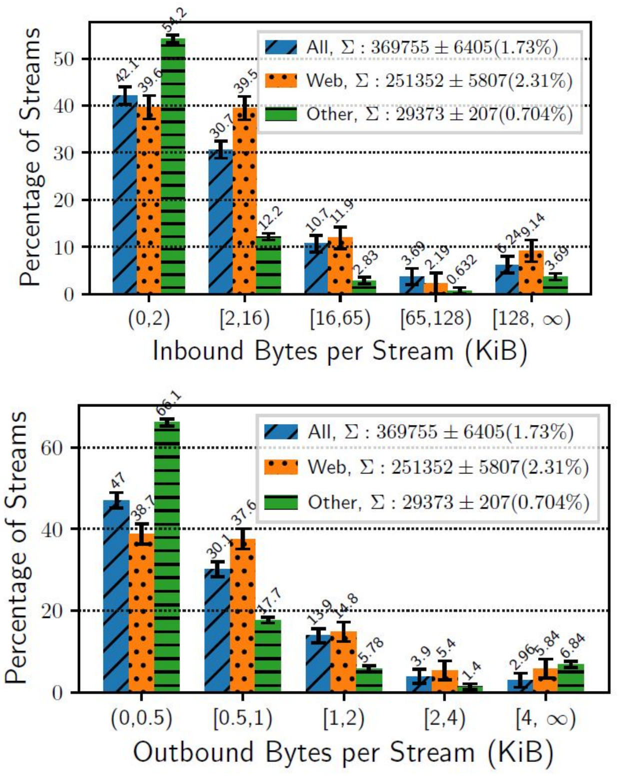


Figure 4: The distributions of bytes per stream collected during periods 5, 6, and 7 (see Table 2). The total number of streams of each type is shown in the legend and with 95% confidence.

of streams receive less than 16 KiB inbound, while about 75 to 85 percent of streams send less than 1 KiB outbound. We also observed a significant difference between web and other traffic: it is about 15 and 30 percent more likely that other streams will carry less than 2 KiB inbound and less than 512 bytes outbound, respectively, compared to web streams.

4 LEARNING TOR TRAFFIC

The method we use to learn Tor traffic is Hidden Markov Model (HMM). In HMM, states of the model are hidden. Each state can emit an output which is observed. In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters, while in the hidden Markov model, the state is not directly visible, but the output (in the form of data or "token" in the following), dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states; this is also known as pattern theory. The method we use here is hidden Markov models (HMM) and an iterative PrivCount measurement process to model both the creation of streams and the traffic on a stream (i.e., packets) In figure 5 we show how the HMM measurement is done at different steps and the iterations happening. The different steps of HMM measurement are, Step 1: Initiate relays with the current HMM parameters. Step 2: Update sums for the number of observations of each state, the number of transitions between each pair of states, and summary statistics for the observations for each state. Step 3: The aggregated counters were used to compute new parameters for the HMM.

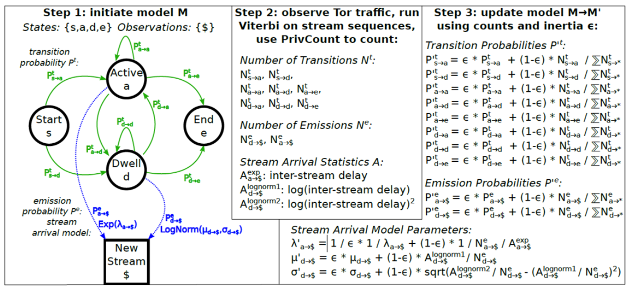

Step 1: initiate model M
States: {s,a,d,e} Observations: {$}
transition probability P^t:
Active a
Start s
End e
Dwell d
$P^t_{a\to a}$ $P^t_{s\to a}$ $P^t_{a\to e}$ $P^t_{d\to a}$ $P^t_{d\to d}$ $P^t_{a\to d}$ $P^t_{s\to d}$ $P^t_{d\to e}$
emission probability P^e: stream arrival model:
$P^e_{a\to\$}$ $Exp(\lambda_{a\to\$})$
$P^e_{d\to\$}$ $LogNorm(\mu_{d\to\$},\sigma_{d\to\$})$
New Stream $
Step 2: observe Tor traffic, run Viterbi on stream sequences, use PrivCount to count:
Number of Transitions N^t:
$N^t_{s\to a}, N^t_{s\to d}, N^t_{a\to a}, N^t_{a\to d}, N^t_{a\to e}, N^t_{d\to a}, N^t_{d\to d}, N^t_{d\to e}$
Number of Emissions N^e:
$N^e_{d\to\$}, N^e_{a\to\$}$
Stream Arrival Statistics A:
$A^{exp}_{a\to\$}$: inter-stream delay
$A^{lognorm1}_{d\to\$}$: log(inter-stream delay)
$A^{lognorm2}_{d\to\$}$: log(inter-stream delay)^2
Step 3: update model M→M' using counts and inertia ϵ:
Transition Probabilities P'^t:
$P'^t_{s\to a} = \epsilon * P^t_{s\to a} + (1-\epsilon) * N^t_{s\to a} / \sum N^t_{s\to *}$
$P'^t_{s\to d} = \epsilon * P^t_{s\to d} + (1-\epsilon) * N^t_{s\to d} / \sum N^t_{s\to *}$
$P'^t_{a\to a} = \epsilon * P^t_{a\to a} + (1-\epsilon) * N^t_{a\to a} / \sum N^t_{a\to *}$
$P'^t_{a\to d} = \epsilon * P^t_{a\to d} + (1-\epsilon) * N^t_{a\to d} / \sum N^t_{a\to *}$
$P'^t_{a\to e} = \epsilon * P^t_{a\to e} + (1-\epsilon) * N^t_{a\to e} / \sum N^t_{a\to *}$
$P'^t_{d\to a} = \epsilon * P^t_{d\to a} + (1-\epsilon) * N^t_{d\to a} / \sum N^t_{d\to *}$
$P'^t_{d\to d} = \epsilon * P^t_{d\to d} + (1-\epsilon) * N^t_{d\to d} / \sum N^t_{d\to *}$
$P'^t_{d\to e} = \epsilon * P^t_{d\to e} + (1-\epsilon) * N^t_{d\to e} / \sum N^t_{d\to *}$
Emission Probabilities P'^e:
$P'^e_{a\to\$} = \epsilon * P^e_{a\to\$} + (1-\epsilon) * N^e_{a\to\$} / \sum N^e_{a\to *}$
$P'^e_{d\to\$} = \epsilon * P^e_{d\to\$} + (1-\epsilon) * N^e_{d\to\$} / \sum N^e_{d\to *}$
Stream Arrival Model Parameters:
$\lambda'_{a\to\$} = 1 / \epsilon * 1 / \lambda_{a\to\$} + (1-\epsilon) * 1 / N^e_{a\to\$} / A^{exp}_{a\to\$}$
$\mu'_{d\to\$} = \epsilon * \mu_{d\to\$} + (1-\epsilon) * A^{lognorm1}_{d\to\$} / N^e_{d\to\$}$
$\sigma'_{d\to\$} = \epsilon * \sigma_{d\to\$} + (1-\epsilon) * sqrt(A^{lognorm2}_{d\to\$} / N^e_{d\to\$} - (A^{lognorm1}_{d\to\$} / N^e_{d\to\$})^2)$

Figure 5: Overview of HMM measurement and update process using PrivCount and exemplified with our stream model.

5 RESULTS

Two separate measurement experiments, first performing a series of 14 iterations with the packet model, and then performing a series of 14 iterations with the stream model. Each iteration involved a 24-hour measurement period (The âĂIJinertiaâĂİ value equals to 0.5, meaning that it gives both parameter sets equal weight).

5.1 Parameter Convergence

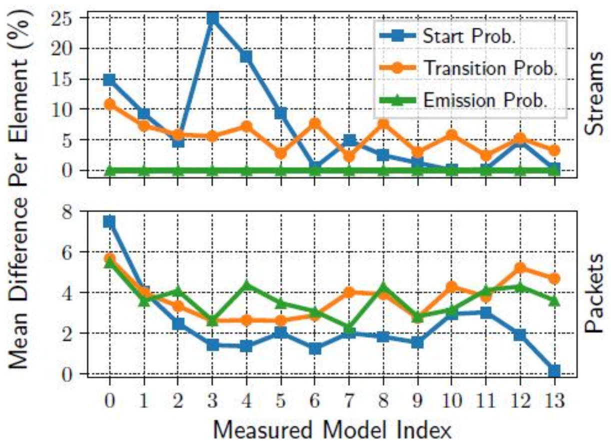
Fig 6a shows the variations in parameter per iterations for different models. The change in parameters in the final model are significantly lower than the changes in the first model, and that the differences generally decreased over measurement iterations.

6 RESULTS

6.1 Model Improvement

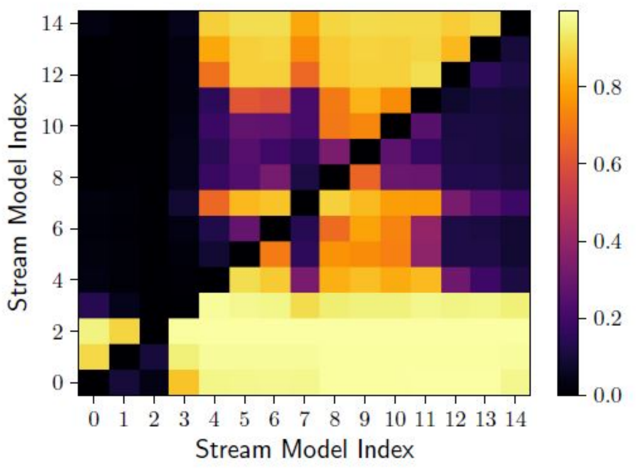
How this approach improved this model apart from the previous model to reduce the decrease in parameter variation is shown through some steps. Step1: The Relay computes the most likely sequence of states under each model M(i) and the log likelihood of the observed sequence, âĎŞ(i). Step2: The relay compares each pair âĎŞ(i), âĎŞ(j) to determine which model gave the sequence a higher likelihood and added this result to a counter. Step3: After 24 hours, these pairwise counters were taken as the output of the experiment. Intuitively, A better model for a process should assign higher likelihood to the output of the process more often.

The results are shown in Figure 7 and Figure 8. For both the stream arrival model (Figure 6b) and the packet model (Figure 6c), the later models are generally superior fits to new data than the earliest models, but we can also see that potentially anomalous measurement periods can cause some iterations to produce inferior models that then continue to improve in following iterations. Using the results of these measurements, we chose the stream and packet



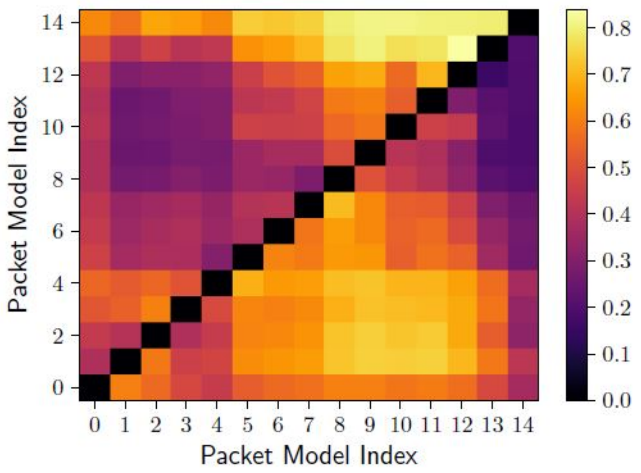
(a) Summed differences from previous model parameters per iteration.

Figure 6: Overview of HMM measurement and update process using PrivCount and exemplified with our stream model.



(b) Fraction of observed stream sequences more likely under stream model x than stream model y.

Figure 7: Fraction of observed stream sequences more likely under stream model x than stream model y.



(c) Fraction of observed packet sequences more likely under packet model x than packet model y.

Figure 8: Fraction of observed packet sequences more likely under packet model x than packet model y.

models that had the best performance as the basis for our Shadow experiments. The stream and packet models at index 9 have the best performance.

7 MODELLING FOR TRAFFIC

For traffic generation, the authors design and develop a set of modeling semantics and a traffic generation tool called TGen that can be used to create arbitrarily complex patterns of behavior; TGen enables configurable control over the creation of TCP connections and the size, schedule, and duration of packet streams. TGen parses standard graph files to extract the actions, parameters, and edge ordering. The architecture of TGen is shown in Fig. 9.

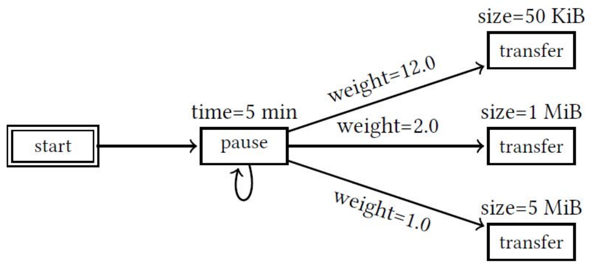
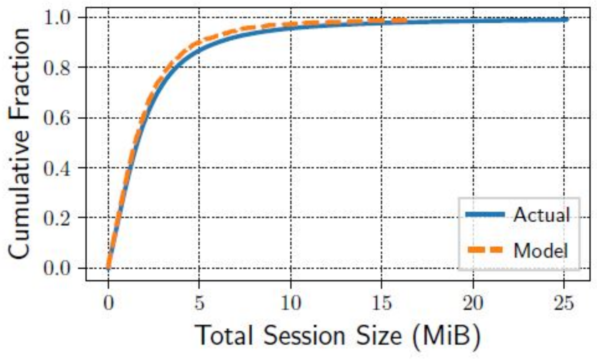


Figure 9: A simplified tgen model graph capturing the behaviour of Tor's performance benchmarking process

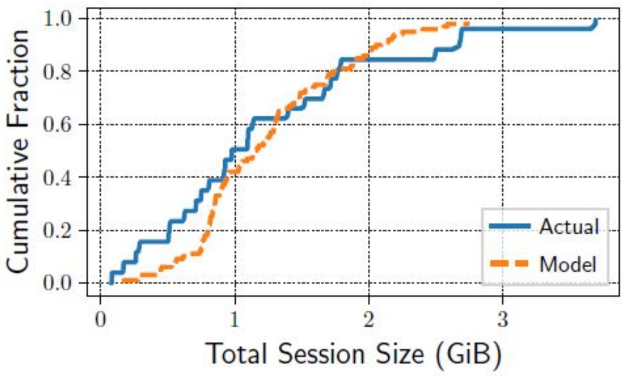
The simple model shows that a client downloads one of three differently-sized files (50 KB, 1 MB, and 5 MB) every five minutes from a server, where each transfer has a weighted chance of being chosen after each five minutes pause completes. Because of the modeling flexibility, The Tor Project now uses TGen to collect performance benchmarks. Also, they use three different client models to build the traffic and they are as follows. 1. Single File Models. In Single File Models, generally, there are two kinds of clients: (1) âĂIJwebâĂİ client type that downloads 320 KB files and pauses in between repeated downloads (to mimic user âĂIJthink timeâĂİ). (2) âĂIJbulkâĂİ client type that repeatedly downloads 5 MB files without pausing. 2. Protocols Models. In this model, they design a web user model that generates traffic that is like real Internet website traffic and a BitTorrent user model that generates traffic similar to real-world BitTorrent traffic; 3. PrivCount Model: Their PrivCount Model Uses the Tor measurement results of this paper and HMM stream and packet models as the basis for traffic generation. Most importantly, each client generates streams and packets in the PrivCount model according to the HMMs.

8 RESULTS

After generating the traffic with TGen, they show some experiment results.



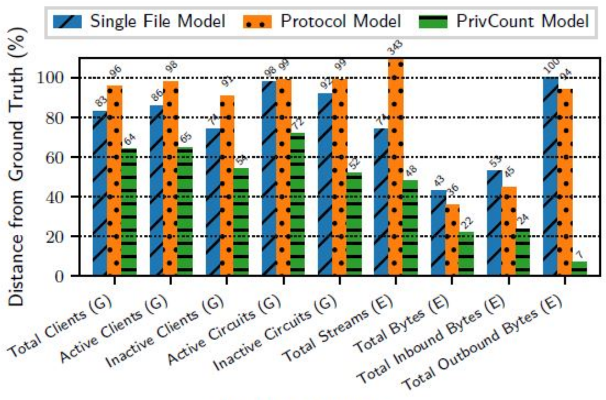
(a) HTTP Archive Model



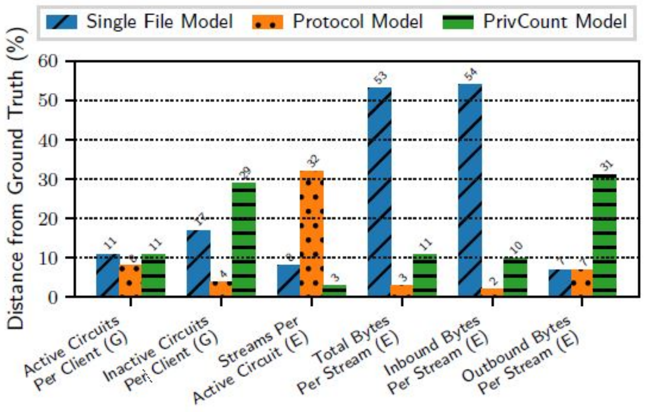
(b) BitTorrent Model

Figure 10: Total session size distribution in our general user models compared to actual session sizes in the full datasets.

They show the distribution of the modeled total session sizes of HTTP compared to actual total page sizes in Figure 10a; a KolmogorovâĂŞSmirnov test of the distributions yields a result of 0.045 ($p < 0.05$) which indicates that our model approximates the dataset reasonably well. Also, the resulting distribution of modeled session sizes, where session size is the sum of the bytes received across all connections, is plotted against actual session sizes in Figure 10b; a KolmogorovâĂŞSmirnov test of the distributions yields a result of 0.22 ($p < 0.005$) which indicates that our model approximates the dataset reasonably well. They use Shadow to evaluate and compare the traffic models. Shadow is a network simulator that runs real applications, including Tor and TGen. For Single File Model:2000 Tor relays, there are 5000 TGen servers and 60,000 TGen clients; For Protocol Model, there are 11,943 clients and half of them are âĂIJbulkâĂİ clients; For PrivCount Model, 128,519 TGen clients are included.



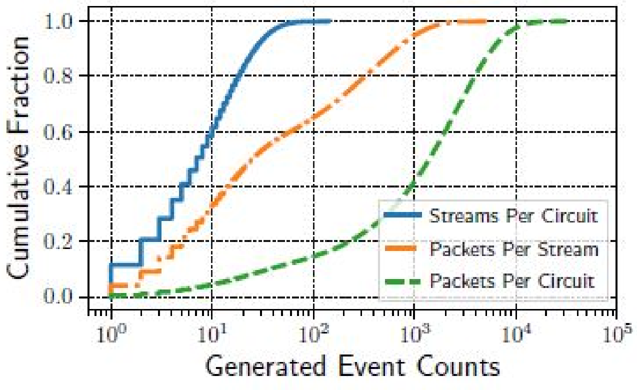
(a) Single Counter Type



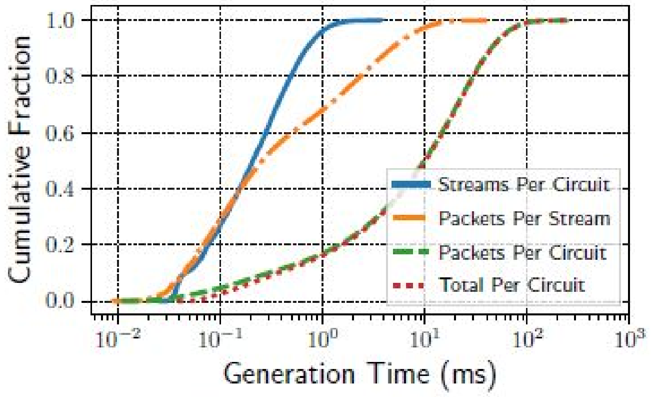
(b) Histogram Counter Type

Figure 11: Wasserstein distance (a.k.a., earth moverâĂŹs distance) between PrivCount measurements as a percentage of ground truth, where a (G) indicates a measurement taken from an entry relay position and an (E) indicates a measurement taken from an exit relay position. (10a) The cumulative percentage distance across all nine single counters was 703% for the single file model, 1001% for the protocol model, and 408% for the PrivCount model. (10b) The cumulative percentage distance across all six histogram counters was 150% for the single file model, 56% for the protocol model, and 95% for the PrivCount model.

We present the Wasserstein distances for each Shadow experiment as percentages of the total ground truth values in Figure 11. Compare the measurement results from each Shadow experiment to the ground truth Tor measurements using earth moverâĂŹs distance as a metric. The cumulative percentage distance across all nine single counters was 703% for the single file model, 1001% for the protocol model, and 408% for the PrivCount model. The cumulative percentage distance across all six histogram counters was 150% for the single file model, 56% for the protocol model, and 95% for the PrivCount model. They believe that the generator times are negligible when compared to the time to send and receive network traffic (which happens as a result of the generated events).



(a) Number of Generated HMM Events



(b) Time to Generate HMM Events

Figure 12: Performance overhead of our HMM generator processes.

The performance results of this model are shown in fig 12. From Figure 12a we can see that, fewer than 165 streams per circuit, 7,211 packets (âĹij10.5 MB) for a single stream and did not generate more than 35,568 packets (51.9 MB) for all streams in a circuit. And the maximum total generator time for all streams and packets in a circuit was 278 milliseconds in figure 12b.

9 CONCLUSION

In this paper, they conducted a significant general measurement study of Tor, measuring clients, circuits, streams, bytes, and their distributions for modeling purposes. Additionally, they collected measurements that they used to dynamically learn hidden Markov stream arrival and packet models. Using our measurements, they designed a traffic generation model for private Tor networks and demonstrated that their model yields a more realistic network than previous and alternative models.

REFERENCES

[1]R. Jansen, M. Traudt, N. Hopper, "Privacy-preserving Dynamic Learning of Tor Network Traffic" ACM Computer and Communications Security (ACM CCS), 2018.