Poster: PredicTor: Predicting Fast Circuits For A Faster User Experience in Tor

Armon Barton, Mohsen Imani, Matthew Wright, and Jiang Ming

***Abstract*—The Tor anonymity system provides online privacy for millions of users, but it is slower than typical web browsing. To improve performance in Tor, we propose *PredicTor*, a path selection technique that uses a kNN classifier trained on a set of 125,000 simulated Tor paths in order to predict the performance of a proposed path. If the path is predicted to be fast, then a circuit is built using those relays. We implemented PredicTor in the Tor source code and show through simulations in Shadow that PredicTor improves Tor network performance by 29% compared to Vanilla Tor and by 21% compared to the previous state-of-the-art scheme.**

***Index Terms*—Tor, Anonymous Communications**

I. Introduction

Tor is a low-latency anonymity system designed for TCP-based applications [1]. The Tor network comprises approximately 7000 relays [2] that are deployed throughout the world by volunteer operators. It was recently shown by Jansen et al. [3] that Tor has approximately 550,000 active users at any given time. Each user's Tor client selects a path of three relays and builds a *circuit* over them to pass anonymized traffic back and forth. In the circuit, the first, middle, and last hops are called *guard*, *middle*, and *exit* relays respectively. Circuits are built based on the onion routing protocol, where clients negotiate session keys incrementally with each successive hop in the path until the final hop is reached.

Relays are selected for paths such that traffic is evenly distributed over the available relay bandwidth. To enable this, each client receives hourly a *consensus document* from the *directory servers* containing bandwidth weights for all relays. Based on this information, the client uses the consensus bandwidth weights to calculate weights for the relays. Finally, the client uses the calculated weights to select the relays, which ensures load balancing but does not ensure fast circuits.

Related Work. Tor is slower than typical web browsing, and a number of research groups have attempted to address this. Wacek et al. [4] examined multiple approaches and determined that *Congestion-Aware Routing* (CAR) [5] offered the best performance-anonymity trade-off. In this paper, we thus use CAR as as a benchmark for comparison.

II. Path Classification

Our goal is to classify potential Tor circuits into two classes—fast and slow. To do this, we first ran a Tor network simulation with 1000 clients using Shadow [6], a discrete event simulator that runs the Tor source code. We generated a training set of 125,000 streams, where each stream consisted of a client downloading a file from a server through a circuit.

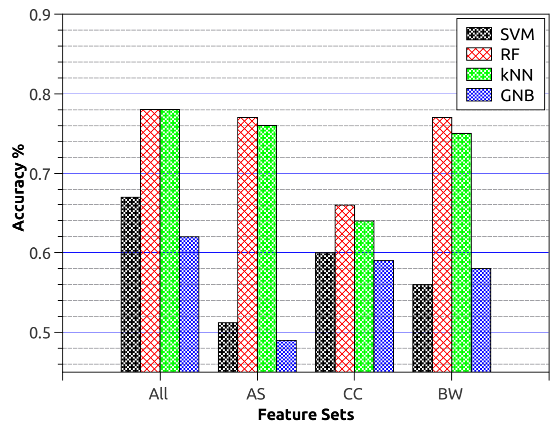

Fig. 1. Accuracy of machine learning algorithms in predicting circuit performance with All features compared to AS, CC, and BW features only.

We label each stream based on the time-to-last-byte (TTLB) download time that was measured from the client during the simulation. For all streams in the training set, the median TTLB value was $1.8s$. Therefore, we set a threshold parameter $\tau = 1.8s$; the label for a given stream was then set to 0 or 1, if $\tau < 1.8s$ and $\tau \geq 1.8s$ respectively.

In Tor circuits, there is a relationship between download times and consensus bandwidth of each relay, as well as between download times and network location of each relay. Due to these relationship, we believe that a recognizable pattern exists such that download times can be predicted (to some degree) by inspecting bandwidth and network location of each relay in a circuit. As such, we resolve each relay into three features: 1) Autonomous System (AS), 2) Country Code (CC), and 3) Consensus Bandwidth (BW). Our feature set F then consists of nine features, three for each of the three relays.

To perform the classification of circuits into either the fast class 0 or the slow class 1, we used distance-weighted k-NN with $k = 9$, Gaussian Naive Bayes, Support Vector Machine (SVM) with a $3rd$ degree polynomial kernel, and Random Forests.

Figure 1 shows the accuracy of these machine learning algorithms in predicting circuit performance using *All* features compared to *AS* features, *CC* features, and *BW* features alone. The results indicate that k-NN and Random Forests perform better than SVM and Naive Bayes. The best classification accuracy was 78% using either k-NN or Random Forests when *All* features were used. Using just *BW* features led to

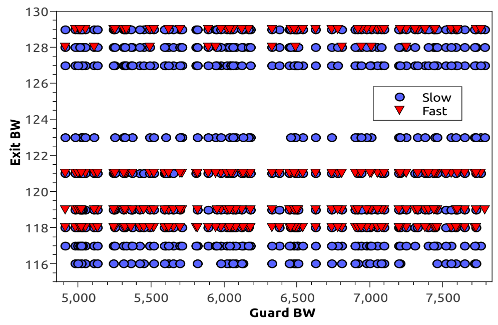

Fig. 2. Fast and slow circuits with respect to their guard vs. exit consensus bandwidth weights.

an accuracy that was almost as high when compared to *All features*; the importance of bandwidth in path selection was previously demonstrated by Wacek et al. [4]. When country code features were used, the accuracy for k-NN and Random Forests was 64% and 66%, respectively. One surprising result was that the accuracy of k-NN and Random Forests was quite high for *AS* features, as good as that of *BW* features. This may be because some ASes have either high-bandwidth relays (e.g. a server provider like OVH) or low-bandwidth relays (e.g. consumer ISPs). It may also show that some ASes are too remote or poorly connected to the rest of the Tor network to be of much use for fast circuits.

Visualizing The Data. In Figure 2, we plot fast and slow circuits with respect to their guard vs. exit consensus bandwidth weights. This data helps us visualize the fact that it is difficult to draw one decision boundary in 2D space that would separate the data well. This helps to explain why k-NN and Random Forests outperformed SVM. Additionally, another advantage of k-NN over SVM is the fact that SVM must be trained on a much larger training set to have high accuracy. In a real deployment, however, large training sets would result in a longer wait time for clients to receive their updated classification model. Additionally, large training sets would result in a larger consensus file.

III. PredicTor

We now describe PredicTor, a system that applies path performance classification to speed up circuit selection in Tor. In PredicTor, the guard selection policy is identical to Vanilla Tor; clients use a single guard selected randomly using the consensus bandwidth weights [7]. Middle and exit relays are first selected according to consensus bandwidth weights as per the standard Tor protocol, and the result path of three relays is then classified by one of the models described in Section II. If the performance of the proposed circuit is predicted to be fast (class 0), the circuit is built. Otherwise, new middle and exit relays are selected, and this is repeated until a fast circuit is found.

IV. Results

We implemented PredicTor in the Tor source code and tested performance in a scaled-down Tor network using Shadow.

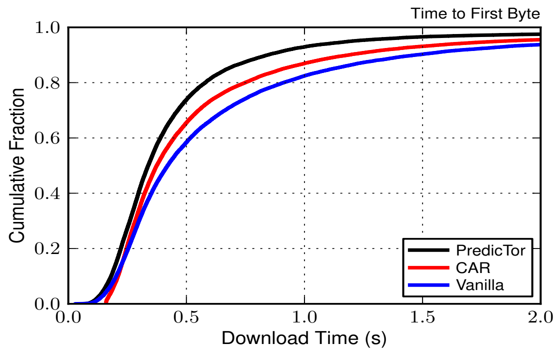

Fig. 3. Time to first byte

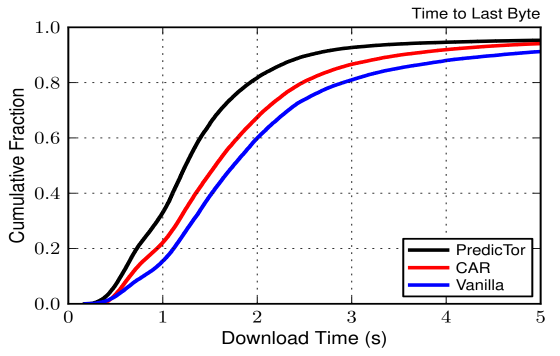

Fig. 4. Time to last byte

We note that even though each client is making its own classification decisions about circuits, the Shadow simulations are network-wide, incorporating activity from 1000 clients all using PredicTor. For the classification model, we used a distance weighted k-NN with $k = 9$ and *All* features.

Figures 3 and 4 show time-to-first-byte (TTFB) and time-to-last-byte (TTLB) values, respectively, for PredicTor compared to congestion-aware routing (CAR) and Vanilla Tor. In the median case, PredicTor had a 22% improvement in TTFB compared to Vanilla Tor and an 11% improvement compared to CAR. For TTLB, Predictor had a median 29% improvement compared to Vanilla Tor, and a 21% improvement compared to CAR. This resulted in PredicTor downloading files in the median case $0.5s$ and $0.4s$ faster compared to Vanilla and CAR, respectively, and for the 90th percentile, $2.5s$ and $1.1s$ faster compared to Vanilla and CAR, respectively.

V. Conclusion

Our results indicate that PredicTor significantly improves network performance in Tor. In future work, we plan on measuring the anonymity of PredicTor, evaluating it with different threshold values τ, and combining PredicTor with a recently proposed AS-Aware path selection scheme called DeNASA [8] and show how the combination improves both AS-awareness and performance in Tor. This material is based upon work supported by the National Science Foundation under Grant No. 1423139.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The Second-Generation Onion Router,” in *USENIX Security*, 2004.

[2] Tor Metrics, https://metrics.torproject.org.

[3] R. Jansen and A. Johnson, “Safely Measuring Tor,” in *CCS*, 2016.

[4] C. Wacek, H. Tan, K. S. Bauer, and M. Sherr, “An Empirical Evaluation of Relay Selection in Tor.” in *NDSS*, 2013.

[5] T. Wang, K. Bauer, C. Forero, and I. Goldberg, “Congestion-Aware Path Selection for Tor,” in *FC*, 2012.

[6] R. Jansen and N. Hopper, “Shadow: Running Tor in a Box For Accurate and Efficient Experimentation,” in *NDSS*, 2012.

[7] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson, “One Fast Guard for Life (or 9 Months),” in *HotPETs*, 2014.

[8] A. Barton and M. Wright, “DeNASA: Destination-Naive AS-Awareness in Anonymous Communications,” in *PETS*, 2016.