Multi Path Considerations for Anonymized Routing: Challenges and Opportunities

Hasan T. Karaoglu, Mehmet Burak Akgun, Mehmet Hadi Gunes and Murat Yuksel
Department of Computer Science and Engineering
University of Nevada, Reno
Email: { karaoglu, makgun, mgunes, yuksem}@cse.unr.edu

***Abstract*—Recently, there has been a surging interest on anonymizer technologies as a result of increasing privacy concerns and censorship barriers on the Internet. Tor network has emerged as the most popular option in avail. However, popularity of Tor network is under threat of growing user frustration over low throughput and long latencies. Coincidentally, multi-path routing proposals have become popular in tackling similar performance issues in various contexts from the data center networks to the inter-domain routing. In this work, we apply multi-path techniques to overcome limitations in Tor's path construction and rigid congestion avoidance mechanisms which are major factors behind the unsatisfactory user experience. Multi-path mechanisms nicely complement underlying onion-routing mechanisms of Tor via exploiting diversity in the Tor network. Our preliminary evaluation on live Tor network revealed significant potential of performance improvements especially increasing throughput i.e., up to 4 times speed up in data transmission durations. Also, multi-path mechanisms may increase reliability of the overall system against congestion or service-interruption based attacks in return of an acceptable anonymity tradeoff.**

I. Introduction

As the Internet revolutionizes the world through social media, privacy, anonymity, and free access to the knowledge have become crucial components of society. Although these rights are not granted by governments or legislation in every country, there has been significant effort to provide them by means of technology [1]. Currently, Tor Project [2] is the largest network of volunteers who help running a low-latency traffic mixing network which makes anonymous access to the Internet possible for many around the world.

Tor network has been built on onion-routing scheme which basically prevents the matching of flows entering and leaving an overlay network to ensure anonymity to a certain level via multi-layered encryption and traffic-mixing [3]. Within that scheme, Onion-Proxy (OP), which is the Tor software running on end-user computer, connects to a node which can be a well-known, reliable, high bandwidth entry guard or a well-hidden bridge so as to get information about other nodes in the network. Then, OP builds paths called "circuits" which connects OP to the Internet and redirect its traffic over other relay nodes in an overlay manner as depicted in Figure 1.

A typical circuit consists of three relay routers: an entry node, a middle Onion Router (OR) in addition to an exit node. Established circuits are periodically built and torn down after a time period to ensure anonymity of the users. In a typical web browsing scenario as depicted in Figure 1, browser running at

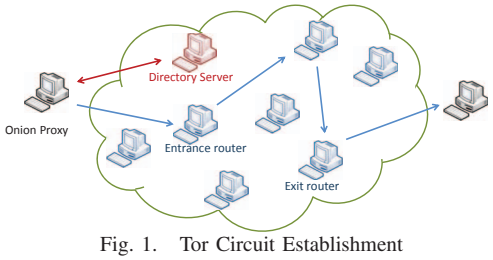

Fig. 1. Tor Circuit Establishment

user-host creates a TCP connection to a website through Tor proxy running on her computer. Tor proxy creates a stream whose cells, i.e. unit of transmission with a fixed size of 512 bytes, would be forwarded along the circuit to the exit node. Exit node in return reconstructs TCP segments from these fixed-size cells and connects to the website as if the exit node itself were the origin of the web request.

While building a circuit, OP agrees on a pairwise symmetric key with each relay node on the circuit [3]. So, payload in a cell is encrypted within three layers of encryption upon leaving the OP. Each layer of encryption is striped off by relay nodes along the path and, finally, the exit node forwards original data to its destination without encryption as shown in Figure 2.

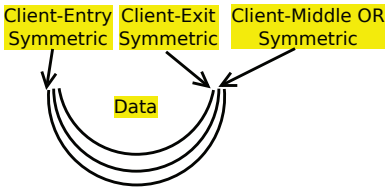

Fig. 2. Tor Circuit Establishment

In its current design, Tor network favors nodes who advertise high bandwidth capacity in the Tor directories so that these high-capacity nodes are most probably selected by OPs querying these directories [4]–[6]. OPs build circuits in a source-based routing manner without much coordination from Tor network, mostly based on self-advertised bandwidth capacity information and circuit establishment latencies that are nearly oblivious to congestion on relay nodes. Hence, overall load-balancing performance in Tor network may result in several highly congested periods of time on certain relays whereas other regions of the network may have low traffic utilization [5], [6].

In addition to its path construction method, Tor network is increasingly criticized over its static congestion window management [6], [7]. According to its token-bucket based algorithm, there can be at most 1000 cells (500 KB) concur-

978-1-4673-0229-6/12/$31.00 ©2012 IEEE

rently in transmission through a single circuit. For stream-level control, this limit is 500 cells. When there is no token left at the bucket, OP stops sending cells until it receives a *SENDME* message from the other end of the circuit which adds a token worth of 50 KB at circuit-level (and 25 KB at stream-level) to the bucket.

Moreover, introduction of low-bandwidth bridge relays, which provides a means of access to the Tor network for users living in countries with active Internet censorship policies, intensifies the problem of low-throughput and long-latencies experienced by users. As pointed out by the authors of a recent work [7], slowing growth rate of Tor network user base could be the result of user frustration over Tor performance. This poses a major threat for Tor network whose mixing-anonymization performance directly relies on the size and diversity of its user base.

In this paper, we investigate the possibility of multi-path routing on the Tor network to tackle performance issues related to the congestion and low throughput. Interest in multi-path routing has been recently revived by the urgent need of high-performance transmission solutions for the data-center networks [8] and proliferation of computers equipped with multiple network interfaces, e.g., wireless and wired, which can enjoy significant throughput improvements by concurrent data transmission over multiple media [9].

In its simplest form, multi-path routing in the Tor network can increase overall throughput by exploiting concurrent data transmission over multiple low-bandwidth circuits. Since multi-path routing dynamically distributes traffic among multiple circuits according to performance of underlying overlay paths, temporary congestions on relay nodes or artificial delays due to Tor congestion management (at circuit and stream levels) can be circumvented. Moreover, overall load-balancing performance of the Tor network can be significantly improved. Better traffic mixing can be achieved via exploiting path diversity of the Tor network by means of intelligent path construction mechanisms [4], [6], [10] coupled with multi-path routing.

Our preliminary analysis on live Tor network revealed significant improvements in throughput via multi-circuit transmissions. In this work, we analyze the differences in throughput speed-ups achieved by varying the number of circuits, and the differences in performance improvements in short and bulk data transfers. We also conducted experiments to understand how node diversity and locality affect improvements achieved by multi-path routing using Amazon EC2 nodes on different regions of the world. Finally, we investigate how multi-path mechanisms may affect the size of buffers on Tor relays, especially onion proxies and exit nodes. This analysis is particularly important as it is widely known that multi-path schemes may cause overall high memory usage, e.g., receiver buffer in MultiPath TCP [9].

The rest of the paper is organized as follows: In Section II, we describe principal ideas and mechanisms related to the design of multi-path onion routing. In Section III, first we explain our experimental setup in detail, then we present the evaluation results for performance improvements in various setup cases along with cost analysis regarding required growth in buffer size. In Section IV, we summarize some of the recent developments in anonymous routing research on Tor performance. Finally, in Section V, we conclude our paper.

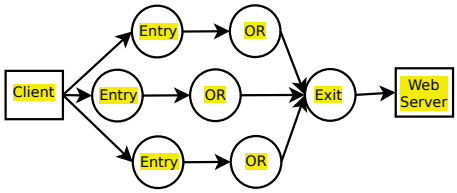

Fig. 3. Tor with Multi-path Approach

II. Multi-path Onion Routing

A. In-band Signaling

Multi-path routing scheme requires introduction of many changes to the original Tor software. First of all, in-band signaling of multi-circuit transmission should be implemented so that OP and exit node could combine multiple-streams over separate circuits into one logical stream. In our design, we envision existence of several circuits which share a single exit node as shown in Figure 3. Common exit node combines all the data received from related streams over multiple-circuits and forwards it to the Internet as a single stream of traffic. Entry nodes also can be common however it is less likely to perform as well as the entry-node disjoint multi-circuit transmission in case of a congested or low-capacity entry node.

Similar to MultiPath-TCP [9], if both OP and exit nodes are multi-path capable, OP informs the exit node of its intention to initiate multi-path transmission. Then, both ends of communication should exchange security tokens (i.e., nonce) and related circuit information over master stream [9], [10]. Both ends should verify these exchanged tokens again for each sub-stream flowing over each circuit. Finally, the exit node and OP logically join these streams together as they belong to a single logical stream.

B. Traffic Distribution over Sub-streams

For dynamic distribution of traffic among multiple-circuits, estimation of round-trip-times (RTT) with relatively fine granularity is necessary. This will help preferring high-performing (i.e., high-throughput) circuits over low-performing circuits while traffic distribution decisions are made at per-cell transmission level. Also, in the short-run, temporarily congested high-capacity circuits can be avoided by forwarding most of the cells over non-congested circuits.

C. Sequencing and Receiver Windows

Since dynamic traffic distribution is made at per-cell basis and characteristics of transmitting circuits can be quite different in terms of latency and bandwidth, reordering of data should be handled by both ends of the communication via sequence numbers and packet buffering. In an analogy with TCP, a receiver buffer should keep packets in according to their sequence numbers. If an incoming cell has the next expected sequence number, it can be directly delivered to higher layers (i.e., a user program on OP or a TCP socket at the exit node).

Otherwise, it should be kept in the receiver window. Once the cell with the expected sequence number is received, the buffered data can be delivered to the higher layers as a block of continuous cells. Size of the receiver window is directly affected by the performances of the concurrently transmitting streams. In fact, if the receiver window is full, high-performing streams should wait for the delivery of the next expected cell by low-performing stream. Although original Tor congestion window management may help limiting an unbounded growth of receiver window, still the receiver window should be sized significantly large in order to avoid performance degradations due to wide performance differences between concurrently transmitting circuits.

D. Black-box Experiment Design for Performance Evaluation

Our multi-path transmission scheme necessitates both ends of the communication to implement buffering and reordering mechanisms in order to combine data transferred over multiple circuits into its original form. Consequently, both client-side Onion Proxy and exit-node should be running our modified software. However, such a requirement would restrict us to choose exit nodes among a couple of modified nodes installed by us while building circuits during our experiments. Since we believe that such a scenario would not be capable of reflecting neither diversity nor realistic conditions of Tor network, we rather preferred making a black-box analysis of multi-path routing on Tor network.

Our experimental setup involves a multi-path routing scenario where traffic between a Tor proxy and web server is transferred over multiple circuits. Our web server running on the Utah Emulab Testbed [11] is listening on multiple-ports and combines the data received from these ports into a single stream. Our client side software implements a Tor controller [12] which can manage the circuit establishment and attachment of streams into preferred circuits. Our Tor proxy software also implements packet buffering, reordering and dynamic traffic distribution over multiple-circuits according to RTT estimations for each circuit.

For achieving fine-granularity RTT estimation, we introduce acknowledgment packets. This could be an overkill in a realistic Tor implementation, as there are several techniques which can achieve the similar task in a relatively coarse-grained but efficient manner. For instance, using estimates based on periodic circuit-level and stream-level *SENDME* messages [10] as well as more costly active probing techniques can be considered [6].

In analogy with TCP and its congestion window management, we implemented a dynamic-traffic distribution algorithm similar to the coupled congested control mechanism of the MultiPath-TCP as described in [9]. In the algorithm, multiple sub-flows share a common congestion window whose size is represented by $cwnd_{tot}$. Each flow rate-limits its transmission to its share, i.e. $cwnd_i$, from this combined congestion window. To make sure congestion window of each sub-flow to grow in coordination ensuring TCP-friendliness of overall transmission, multi-path TCP scheme given in [9] introduces an α parameter. Using RTT estimates made for each sub-flow, α parameter is calculated by the Equation 1.

$$\alpha = cwnd_{tot} \times \frac{max_i \left(\frac{cwnd_i * mss_i^2}{RTT_i^2} \right)}{\left(\sum_{i=1}^{n} \frac{cwnd_i * mss_i}{RTT_i} \right)^2}. \quad (1)$$

Upon receipt of an acknowledgment packet, congestion window of an individual sub-flow grows according to the value calculated by the formula given in Equation 2.

$$min(\frac{\alpha}{cwnd_{tot}}, \frac{1}{cwnd_i}) \quad (2)$$

We implement the original Jacobson RTT estimation and dynamic retransmission timer interval calculation as described in [13]. If an observed RTT value exceeds RTO estimate, it will signal sender about the congestion experienced on the path of this particular sub-stream. In such a case, we decrement the size of sub-stream congestion window, $cwnd_i$ to its half as described in [9].

III. Evaluation

In this section we further describe our experiment setup and present evaluation results for performance improvements in various setup cases along with cost analysis regarding required growth in buffer sizes.

A. Experimental Setup

In our experiments, we use a unidirectional scenario where a client-side software uploads a file to a web server over the Tor network. Client-side software, which is a Tor controller, establishes multiple socket connections to the web server and attaches each resultant stream to a separate circuit. Moreover, client-side software receives acknowledgment messages over multiple circuits, calculates RTT values on each circuit and dynamically distributes the traffic over streams as described in the previous section. We experimented with file sizes of 1.5 MB to model bulk file transfers and 300 KB to model the case representing a typical HTTP web transaction. To investigate varying multi-path performance gains with regional differences in node diversity, we installed our client software on our campus at Reno, USA and Amazon EC2 sites at Singapore, Ireland, and North Virginia. Our web server fixed at the Emulab Utah facility, is listening on multiple ports and combines packets received from these ports into a single logical connection.

B. Test Procedure

We have conducted 33 tests from December 18 to January 1, 2012. For each test, a client Tor proxy establishes 10 circuits, selects 4 of them and then uploads the file to our web server at Utah over these circuits. Once upload is complete, total download time, throughput, and RTT statistics are recorded. Then, client proxy tears down all 4 circuits, waits for 2 minutes and reestablishes 3 of these circuits again using the same relay nodes whose information was recorded. This procedure is repeated with 2 circuits and a single circuit as well. Finally,

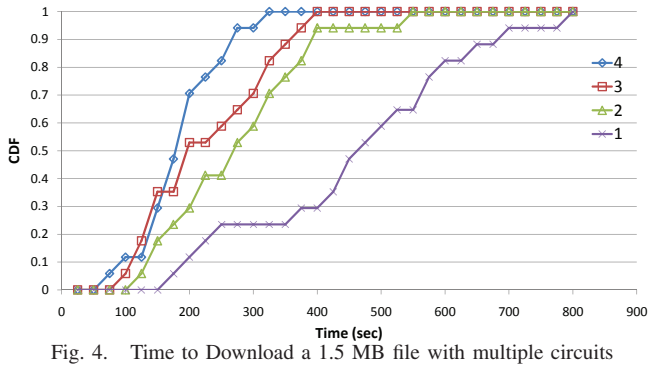

Fig. 4. Time to Download a 1.5 MB file with multiple circuits

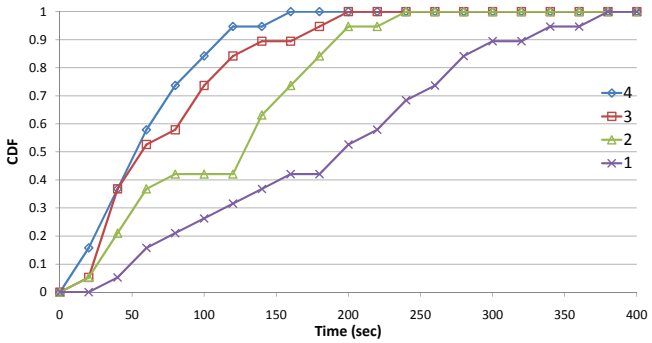

Fig. 5. Time to Download a 300 KB file with multiple circuits

one of the four tests is repeated again. If the performance of the repeated test, e.g., previous test with 3 circuits, matches the recent performance within a reasonable difference margin, we consider this particular set of tests as reliable. Otherwise, we discard the results for this set of tests due to the possibility of inconsistency within the test set where transmissions may have experienced significantly different congestion conditions.

C. Overall Data Transfer Performance

In Figure 4, we plot the empirical cumulative distribution function (CDF) of bulk data transfer durations for files of 1.5 MB between our campus and the Emulab Utah facility over the Tor network. As shown in Figure 4, dynamic traffic distribution over multiple-circuits significantly reduces the overall download time in compared to the single circuit case. Download times are significantly improved for all multi-circuit cases where performance improvements with 3 and 4 circuits are significantly better than with 2 circuits. However, performance improvements with 3 and 4 circuits are relatively close which indicates that using 4 circuits may be an overkill considering the cost of establishing additional circuits.

With smaller files, performance improvements get more obvious in comparison to case of bulk file transfer. Similar to 1.5 MB files, 3 and 4 circuits provide considerable performance improvements. Download times with 3 and 4 circuits significantly differ from the transfer performance of 2 circuits. Note that, overall download times in our test cases are significantly larger than the median values shared in the Tor metric portal [14] but are below timeout and failure limits considerably. Since we compare our results against the single-circuit case, which does not include any RTT estimation or traffic distribution algorithm, we believe that our results reflect the latencies usually experienced by Tor users [15].

File Size	Number of Circuits		
	4	3	2
300 KB	3.92/1.45	3.01/0.88	1.85/0.34
1.5 MB	2.49/0.41	2.28/0.54	1.73/0.25

TABLE I
SPEEDUP FOR DOWNLOAD TIMES (MEAN/MOE, 90% CI)

In Table I, we share average throughput increases with varying number of circuits along with the margin of error for 90th percentile confidence interval. Throughput improvements for small files are significantly better than large file uploads although they are more prone to the performance fluctuations.

D. Receiver Window Size

Next, we investigate the tradeoff between increasing the throughput via multi-path routing and consequent increase in the receiver window size. In our tests, we do not limit the receiver window size at the web server to achieve maximum throughput improvement without being limited by low-performing circuits. However, we calculate required receiver window size on average to achieve such performance improvements by applying Equation 3 on RTT statistics gathered from our experiment.

$$rbuf = 2 \times (\sum_{i=1}^{n} BW_i \times RTT_{max}) \quad (3)$$

As indicated by Barre et. al. [9], these values are considered for the cases where fast-retransmit timers are in use so that fast retransmission of lagging packets are made without waiting for a retransmission timeout.

File Size	Number of Circuits		
	4	3	2
300 KB	161.11/35.25	127.51/36.63	92.01/21.38
1.5 MB	283.85/88.97	254.20/69.04	251.07/67.04

TABLE II
RECEIVER WINDOW SIZE (KB) (MEAN/MOE, 90% CI)

As seen in Table II, increase in receiver window size can be over 250 KB for bulk data transfers and 160 KB for short interactive transfers. These values correspond to significant increases in overall memory usage in exit nodes. Since the Web traffic is mostly asymmetric for HTTP, we can expect that the increases in receiver buffer sizes will most likely impact overall memory usage of the onion proxies running at the client side instead of the Tor exit nodes. However, significant peer-to-peer traffic observed in Tor network may pose a significant security and performance threat to the exit nodes due to increasing buffer sizes.

Increase in receiver window size is parallel to the increase in the number of circuits used in the concurrent transmission. These findings underline the importance of intelligent circuit establishment mechanisms and dynamic circuit management schemes as previously proposed by several researchers [4], [6], [7]. Via dynamic circuit management mechanisms, low-performing circuits can be torn down to prevent unacceptable growth of receiver window size (or adverse effects of slow-circuits on other circuits in the case of limited receiver buffer).

E. Regional Differences

Finally, we analyze the differences in performance improvements varying with the location of onion proxies. We repeat the test cases 7 times for each location where our Tor controller client code is deployed on our campus in Reno, USA, and Amazon EC2 sites in Ireland and Singapore. Our web server is fixed at the Emulab facility at Utah, USA. Since most of the Tor relay nodes are deployed in USA and Europe, in parallel to our expectations, tests with 300 KB files result in better speed-up performance in the overall throughput for these regions with a high node diversity as seen in Table III.

EC2 Region	Number of Circuits		
	4	3	2
USA	4.32/2.52	3.66/2.59	2.31/0.99
Ireland	4.54/1.45	3.02/1.82	1.78/0.51
Singapore	2.46/0.75	2.06/0.51	1.32/0.32

TABLE III
REGIONAL DIFFERENCES IN SPEEDUP (300 KB) (MEAN/MOE, 90% CI)

IV. RELATED WORK

Recently, there have been several proposals targeting overhauls in node-selection algorithms of Tor [4], [6]. Snader et al. proposed several bandwidth estimation techniques including passive observance of achieved bandwidth values and active/periodic measurement of temporal bandwidth capacity of nodes, so as to increase the chance of establishing high-performing circuits instead of relying on the self-advertised capacity values by relay nodes [4]. They also proposed a tunable mechanism which allows users to manage the tradeoffs between performance and anonymity by adjusting the probability of selecting nodes out of a small-set of high-capacity nodes or large-set of nodes with homogeneously distributed bandwidth capacities [4]. Their findings showed that significant improvements can be achieved by little compromise on anonymity [4]. On a similar study, Wang et al. proposed consideration of node congestion while establishing circuits [6]. Authors explained how to apply latency measurement techniques for achieving effective node-based congestion estimation by using active probing and opportunistic use of control messages [6]. They also introduced mechanisms to avoid temporal and long-term congestion by fast-circuit switching and by tracking long-term congestion performance of nodes in Tor network [6].

AlSabah et al. addressed the Tor performance issues by improving the static congestion window mechanisms of Tor [7]. They experimented with adaptive window sizing mechanisms by comparing explicit congestion signaling performance of backpressure mechanisms with the performance of dynamic window sizing. Their preliminary findings showed that backpressure congestion signaling and dynamic queue management approaches perform better than dynamic window sizing [7].

In a recent technical report, Mashael et al. described a scenario where multi-path routing is applied on Tor network [10]. They have reported considerable improvements on two-circuits scenarios for several performance metrics, e.g., time to receive first byte, overall download time. They experimented with modified Tor proxy and exit nodes. In their design, they exploited *SENDME* control messages to estimate round trip times in an aggregate manner for every 100 cells. In their study, they also removed stream-level Tor congestion windows. Moreover, according to the security analysis given in the same work [10], risk of compromising user anonymity is increased only by 7 % due to three-circuit multi-path scheme for the cases where some of the entry and exit nodes in Tor network is compromised.

V. CONCLUSION

In this work, inspired by recent MultiPath TCP studies [8], [9], we applied dynamic traffic distribution mechanisms on the Tor anonymizer software. Our experiments with bulk data files and short interactive data transactions showed that multi-path routing has great potential for improving the overall throughput and decreasing the latencies experienced by users of the Tor network. Our findings also revealed that multi-path routing is able to effectively exploit the underlying regional node diversity. As pointed out by a recent study [10], anonymity implications of multi-path routing is relatively small which does not expose the Tor network to additional risks. Our work showed that despite of its great potential in improving the Tor network performance, multi-path routing may introduce large buffer costs on Tor proxies unless intelligent circuit management algorithms are deployed. Since we work with unmodified Tor software, we prefer working with circuits not sharing a common exit node, so as to avoid unfair queue delays on exit node. As a future work, we would like to investigate the possible side-effects of this experimental setup.

REFERENCES

[1] I. Goldberg, D. Wagner, and E. Brewer, "Privacy-enhancing technologies for the internet," in *Compcon '97. Proc., IEEE*, feb 1997, pp. 103 –109.
[2] "Tor project: Anonymity online," "https://www.torproject.org/".
[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proc. of USENIX SEC*, 2004, pp. 21–21.
[4] R. Snader and N. Borisov, "A tune-up for tor: Improving security and performance in the tor network," in *Proc. of NDSS*, February 2008.
[5] R. Dingledine and S. J. Murdoch, "Performance improvements on tor or, why tor is slow and what were going to do about it," March 2009, "www.torproject.org/press/presskit/2009-03-11-performance.pdf".
[6] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for tor," in *Proc. of FC*, February 2012.
[7] M. AlSabah, K. S. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. M. Voelker, "Defenestrator: Throwing out windows in tor," in *PETS*, ser. LNCS, vol. 6794, 2011, pp. 134–154.
[8] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 266–277, Aug. 2011.
[9] S. Barre, C. Paasch, and O. Bonaventure, "Multipath tcp: from theory to practice," in *Proc. of IFIP TC 6 conference on Networking*, 2011, pp. 444–457.
[10] M. AlSabah, K. Bauer, T. Elahi, and I. Goldberg, "Tempura: Improved tor performance with multipath routing," Centre for Cryptographic Research, University of Waterloo, Tech. Rep., August 2011.
[11] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *USENIX ATC*, 2008, pp. 113–128.
[12] N. Mathewson, "Tc: A tor control protocol (version 1)," "https://gitweb.torproject.org/torspec.git/blob/HEAD:/control-spec.txt".
[13] V. Paxson and M. Allman, "Computing tcp's retransmission timer," RFC 2988, November 2000.
[14] "Tor metrics portal," "http://metrics.torproject.org/".
[15] K. Bauer, M. Sherr, D. McCoy, and D. Grunwald, "Experimentor: a testbed for safe and realistic tor experimentation," in *Proc. of CSET*, ser. CSET'11. n, 2011, pp. 7–7.