The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting*

Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg
{malsabah,k4bauer,mtelahi,iang}@cs.uwaterloo.ca

Cheriton School of Computer Science
University of Waterloo

Abstract. Tor is the most popular low-latency anonymity network for enhancing ordinary users' online privacy and resisting censorship. While it has grown in popularity, Tor has a variety of performance problems that result in poor quality of service, a strong disincentive to use the system, and weaker anonymity properties for all users. We observe that one reason why Tor is slow is due to low-bandwidth volunteer-operated routers. When clients use a low-bandwidth router, their throughput is limited by the capacity of the slowest node.

With the introduction of *bridges*—unadvertised Tor routers that provide Tor access to users within censored regimes like China—low-bandwidth Tor routers are becoming more common and essential to Tor's ability to resist censorship. In this paper, we present Conflux, a dynamic traffic-splitting approach that assigns traffic to an overlay path based on its measured latency. Because it enhances the load-balancing properties of the network, Conflux considerably increases performance for clients using low-bandwidth bridges. Moreover, Conflux significantly improves the experience of users who watch streaming videos online.

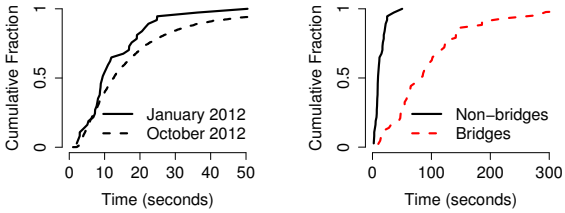
Through live measurements and a whole-network evaluation conducted on a scalable network emulator, we show that our approach offers an improvement of approximately 30% in expected download time for web browsers who use Tor bridges and for streaming application users. We also show that Conflux introduces only slight tradeoffs between users' anonymity and performance.

1 Introduction

Tor [14] is a widely used low-latency anonymity network, which offers strong privacy guarantees by tunnelling a user's Internet traffic through virtual circuits consisting of multiple intermediate overlay routers using a layered encryption scheme based on onion routing [44]. Beyond enabling anonymous communications online, Tor has become an essential tool in the fight against Internet censorship. Today, regimes around the world continue to aggressively filter [65], monitor [40], or explicitly block access [38] to certain types of online content. While Tor serves an estimated 400,000 users on a daily basis [58], its public infrastructure of over 3000 public relays can be easily blocked. In response, Tor uses special unlisted relays called *bridges* to aid users residing within regimes, such as China, that explicitly block the Tor network. Unfortunately, bridges generally provide a lower quality of service than Tor's public infrastructure.

Although Tor's primary goal is to support *real-time interactive* applications such as web browsing, the network suffers from a variety of performance problems [15] that

*This is an extended version of our PETS 2013 paper [3].



(a) 1 MiB download time (b) 1 MiB download time

Fig. 1: Figure 1(a) shows the time required to download 1 MiB files over Tor in Jan. and Oct. 2012. Observe the long tail of the download time distribution. Figure 1(b) shows a download time comparison between Tor users who use public entry guards and those who use bridges.

are manifested as high and variable delays which result in a poor user experience. This poor experience discourages Tor's adoption and ultimately results in a smaller user base and weaker anonymity for all users [13].

Dynamic Traffic Splitting for Tor. In this work, we recognize that the diversity of bandwidth provided by Tor's volunteer-operated routers, and in particular the low-bandwidth bridges, degrades performance. We also recognize the significance of improving the performance of some high-throughput applications, such as streaming web videos, for Tor users. We propose an unconventional approach to improving performance when using low-bandwidth routers and bridges: *Tor users should split their traffic across multiple semi-disjoint circuits.*

In the context of Tor, traffic splitting can improve load balancing. When routers become over-utilized and experience congestion, splitting traffic across semi-disjoint paths can ease the burden on the congested circuit; under our scheme, circuits need only share a common exit router. Second, by splitting data over multiple circuits, the user's throughput can achieve up to the aggregate throughput of all circuits rather than a single one. This is particularly useful when a circuit uses a low-bandwidth router. Tor's router selection algorithm favors routers that have higher bandwidths to ensure sufficient throughput to transport users' traffic and to balance the traffic load across Tor's routers. However, individual Tor routers can have vastly different bandwidth capacities, ranging from 20 KiB/s to over 20 MiB/s. Figure 1(a) shows a long-tailed distribution of download times for 1 MiB files over the course of two different months: January and October 2012.[1] These slower downloads often correspond to circuits that used at least one low-bandwidth router. By combining multiple circuits with low-bandwidth nodes, the attainable throughput is no longer bound by the bottleneck node, but is instead the aggregate of each individual circuit's throughput.

Our approach. We design, implement, and evaluate *Conflux*,[2] a novel congestion-aware traffic splitting and load balancing algorithm for anonymous communication networks. Conflux forwards a client's individual cells down multiple circuits that share a

[1] This data was obtained from The Tor Metrics Portal [56].

[2] Conflux: a flowing together of rivers or streams.

common exit router. Our algorithm dynamically measures the throughput of each constituent circuit and assigns traffic to each in proportion to its observed throughput. Our approach performs sub-stream traffic splitting, which provides a fine granularity of load balancing, as splitting can be performed at the individual cell level. This allows the traffic that is sent on a circuit to correspond to its desired load. The circuit's endpoints (the client and the exit router) are responsible for splitting the traffic at one endpoint and buffering, re-ordering, and delivering in-order cells to the application at the other end of the circuit. This approach can be deployed incrementally, as only clients and exit routers need to upgrade to support it.

To quantify the performance benefits of our proposed design, we perform a variety of live and whole-network experiments on an emulation-based Tor network testbed [6]. Our evaluation indicates that our approach can result in decreased queueing delays and increased throughput for users, particularly those who rely on low-bandwidth bridges to access the Tor network. We also find that, under light traffic loads, Conflux improves performance for clients who use Tor to access streaming videos (such as blocked YouTube videos[3]). Improving performance for such users is important, as streaming video websites are becoming a dominant source of Internet traffic [31, 45].

We also critically evaluate the security implications of utilizing additional circuits in light of the end-to-end traffic confirmation attack [47, 49]. Our analyses indicate that our scheme only slightly increases users' vulnerability to this attack. Anonymity is also slightly decreased when the adversary uses powerful selective denial of service tactics [5, 8, 11, 60] to maximize the number of circuits that can be compromised.

Contributions. This paper offers these contributions.

1. To improve performance for bridge and streaming application users, we design, implement, and evaluate a dynamic traffic splitting scheme that distributes the traffic load across circuits according to each circuit's bandwidth capacity.
2. Our live performance analysis indicates that Conflux results in an expected improvement of 30% in a typical Tor client's queueing delay and up to 75% in total download time. Whole-network experiments show that noticeable improvements are possible even when most or all clients adopt Conflux.
3. We analyze the security of Conflux and provide quantitative results showing that there is a small tradeoff between users' anonymity and performance gains.

Outline. The remainder of this paper is organized as follows: Section 2 describes Tor's design at a high level and Section 3 presents the Conflux design and an algorithm for splitting traffic in a manner that balances the traffic load over each circuit. We evaluate our proposal in Section 4 and offer a security analysis in Section 5. We discuss a variety of open issues and enumerate avenues of future work with our design in Section 6, contrast our contributions with related work in Section 7, and conclude in Section 8.

2 Background

How Tor works. The Tor overlay network involves three kinds of nodes: Onion Proxies (OP), Onion Routers (OR), and directory authorities. OPs run on end-users' machines

[3] Note that while Tor's browser bundle disables Flash by default, it is now possible to stream videos over Tor using HTML5. We expect this use case of streaming video over Tor to increase in popularity in the near term.

(clients). OPs start their operation by contacting the directory authorities to download the *network consensus*, which contains information about all ORs in the Tor network. ORs are volunteer-run relays (or *nodes*) that are responsible for relaying users' traffic to other nodes or to destinations outside of Tor, such as web servers.

The OP constructs a number of *circuits*—Tor-network paths through which the client's traffic is relayed in fixed-sized (512-byte) units called *cells*. To construct a circuit, the OP chooses three relays, each in a manner that weights a relay's selection in proportion to its bandwidth capacity. With Tor's decentralized architecture, only the exit node can observe the user's traffic and only the *entry guard* knows the identity of the user. Clients use the same set of three entry guards for a long period of time (currently 30–60 days), to mitigate the threat of the predecessor attack [64] and other attacks that seek to correlate entry and exit traffic to link senders with their respective receivers [39]. If both the entry guard and exit node cooperate, they can use traffic analysis to link the initiator to her destination [28, 49].

Circuits and streams. A single TCP connection is used between any two ORs in the network. However, this single TCP connection is used to multiplex several circuits that may or may not belong to the same user. To identify different circuits, each OR assigns different circuit IDs to circuits. In addition to circuit multiplexing, the OP can multiplex several TCP *streams* over one circuit, which generally has a lifetime of ten minutes.

To ensure flow control of data in flight, Tor employs an end-to-end window-based flow control mechanism in which every time the OP (or exit OR) receives 100 cells, it acknowledges windows of data cells using `SENDME` (or acknowledgement) cells.[4] We leverage these end-to-end control cells as a means to infer a circuit's bandwidth capacity, as we describe in Section 3.

Evading censorship with bridges. In addition to anonymous communications, Tor is an important tool in the fight against censorship. Tor helps users around the world visit blocked websites. In some cases, Tor's infrastructure of directory authorities and routers has been blocked, for example by the so-called "Great Firewall of China" [29]. To facilitate entry to the Tor network despite such blocking, Tor has introduced *bridges*, which are unlisted Tor routers that are distributed to censored users via out-of-band mechanisms such as HTTPS queries to `bridges.torproject.org`. To ensure that a censor cannot collect all bridges and simply block them, Tor currently limits the number of bridges that are distributed to each /24 IP address block [59]. Currently, clients use bridges in lieu of an entry guard, to keep the total circuit length at three routers.

While bridges provide an essential service to an estimated 30,000 censored users as of November 2012 [58], they are believed to be operated by Tor clients who often reside on low-bandwidth broadband networks. To confirm this hypothesis, we obtained 221, of approximately 700 [57], bridges in January 2012 using Tor's standard HTTPS request service from PlanetLab hosts on 55 different /24 IP networks.[5] Figure 1(b) compares the performance of a live Tor client that downloads 1 MiB files using entry guards versus bridges. Clearly, the low-bandwidth bridges are a significant source of poor performance. Furthermore, Tor bridges are becoming integrated into ubiquitous

[4] For more details about Tor's flow control mechanisms, see AlSabah *et al.* [4].

[5] This procedure for enumerating bridges is described in more detail by Ling *et al.* [30]. Automatically enumerating Tor bridges is more difficult at present because clients have to solve CAPTCHAs. For our experimental evaluation in Section 4.2, we use a smaller, more recent list of bridges.

devices such as wireless access points to simplify the process of configuring and running a bridge on a broadband Internet link at home [54]. Thus, because low-bandwidth bridges will likely become even more common in the near future, in this work we seek to improve performance for bandwidth-limited bridge clients.

Adapting Tor to the changing web. Unlike previous efforts which seek to enhance the experience of web browsers in Tor by throttling bulk downloads (specifically file-sharing applications) [23,35], we recognize that some emerging classes of bulk transfers should actually be improved rather than throttled. In fact, recent Internet traffic studies have revealed that file sharing applications are consuming less bandwidth,[6] while streaming video applications are starting to account for an increasingly large fraction of Internet traffic by volume [31, 45]. Therefore, in order to survive and continue to attract new users, it is crucial for Tor to meet the demands of its users and the changing web by improving the experience for streaming users. Although The Tor Project mainly welcomes web browsing, it is hard these days to separate streaming from web browsing. If a user visits a blocked news website via Tor, the user may also want to view videos associated with the stories accessed.

While streaming websites (such as Youtube and Netflix) sometimes use strategies that consist of buffering data followed by transmitting ON-OFF bursts to users, it has also been shown that one streaming strategy can be considered as simple file downloads [42]. We adopt the strategy of the simple file downloads to model streaming videos in our experiments in Section 4

3 Conflux's Design

We next shift attention to the design of our system. An OP that uses Conflux builds a number of circuits (two or more) that intersect at a common exit OR. We refer to the OP and common exit OR as the *end points* of a multipath. The OP receives and sends data to the client's application (such as a web browser), while the exit OR sends and receives data from an external server (such as a web server). Each end point receives data and splits it into cells, adding sequence numbers to the cell headers. Next, the end point divides the cells across the circuits of the multipath according to a traffic splitting scheme. When the other end point receives the cells, it collects and reorders them according to their sequence numbers before delivering their contents to their destinations. We note that this approach does not replace Tor's bandwidth-weighted router selection algorithm, but complements it.

Our approach to cell-level traffic splitting consists of three parts: 1) multipath construction, 2) throughput-informed sub-stream traffic splitting, and 3) sequencing, buffering, and reordering. We next describe each part in turn.

Constructing the multipath. As shown in Figure 2, the client constructs the first circuit, called the *primary circuit*, according to Tor's bandwidth-weighted router selection. Then, if the client wishes to use Conflux, the client forms another circuit that uses different entry and middle ORs, which are also selected according to the bandwidth-weighted algorithm. The only constraint our system requires on the second circuit is that its exit OR has to be the same as the primary circuit's exit OR. Next, the OP sends a new type of

[6] Note that P2P traffic on Tor is also likely to drop with the rise of UDP-based P2P applications [7].

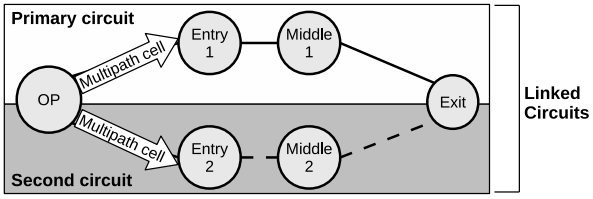


Fig. 2: Multipath construction and stream linking

command cell, which we call the "multipath" cell, through both circuits to the common exit OR. The multipath cell contains a 32-byte random nonce that is common to each of the OP's linked circuits. This nonce enables the exit router to associate the OP's TCP streams with its linked circuits.[7] The OP uses the primary circuit to command the exit OR to establish the TCP connections to Internet destinations. Closing a multipath is no different from closing circuits in Tor. If a circuit in a multipath exceeds its lifetime (ten minutes by default), and if it is idle, the circuit is torn down. Closing one circuit does not affect the operation of other circuits. Since a Tor client already builds many spare circuits by default, we do not expect any additional load being introduced by Conflux.

Dynamic sub-stream traffic splitting. To improve load balancing, we designed and implemented a dynamic load balancing algorithm where the splitting end point assigns different amounts of traffic to each circuit depending on its observed throughput relative to the other linked circuits. The advantage of this approach is that it is reactive to network dynamics, such as the congestion state of a circuit and its available capacity relative to the other circuits.

This scheme works as follows. First, the splitting end point (the OP for client-to-server traffic, or the exit OR for server-to-client traffic) measures the latency of each of the linked circuits. This can be done by storing the time every 100^{th} cell is sent down a particular circuit, and noting the time that the corresponding circuit-level SENDME arrives. This allows the splitting end point to compute the current round-trip-time (RTT) of cells on the circuit; this will be inversely proportional to the circuit throughput, as cells are of fixed size.

The splitting end point periodically updates the throughput measurements assigned to each linked circuit. Next, every time a data cell is ready to be transmitted on a multipath, the particular circuit used to send the cell is selected with a probability proportional to its throughput.

Sequencing, buffering, and reordering. Before any splitting can be performed on TCP streams across different circuits, we have to ensure that the receiver will be able to reorder cells before delivering them to their destination (client program or exit TCP connection). Therefore, we implement sequencing of data cells before sending them down our multipaths. Tor's standard data cell consists of a circuit identifier, a "relay" command type, a "data" sub-type, a "recognized" field to identify whether the cell is to be delivered locally, a stream identifier, a message digest to ensure integrity, a data length, and the data. We modify the cell format slightly, to reserve the first four bytes of the payload for the sequence numbers, as shown in Figure 3. This reduces the amount of data that can fit into each cell's payload by less than 1%.

[7] As usual, a malicious exit router can trivially observe all circuits it handles, including linked ones.

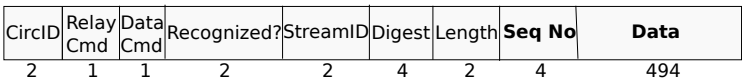


Fig. 3: The 512-byte data cell format with each field's length (in bytes) for Conflux.

Because we divide a single TCP stream across circuits, we expect that cells may arrive out of order. Therefore, the two end points of a multipath, the OP and exit OR, are responsible for buffering and reordering cells that arrive out of order. First, as long as the cells arrive in order, they are immediately delivered to the client application (or the TCP exit connection) when the receiver is the OP (or the exit OR). Also, we keep track of the sequence number of the last delivered cell. If a cell arrives out of order, it is stored in a sorted list of cells. When the next expected cell arrives, it is delivered to the OP (or TCP exit connection) along with any buffered cells with subsequent sequence numbers that have already arrived.

Implementation details. We implemented the multipath construction and cell sequencing, buffering, and reordering in the Tor source code (version 0.2.3.0-alpha-dev). We also implemented the weighted traffic splitting algorithm as described above. Conflux can be turned on or off as a configuration option. Note that only the circuit's end points (e.g., the client and the exit router) are required to upgrade to run Conflux. Thus, Conflux can be incrementally deployed as individual exit routers and clients upgrade. Our full implementation consists of roughly 2,000 lines of code.

4 Performance Evaluation

In order to empirically demonstrate the potential performance benefits of the proposed scheme, we present a series of live and large-scale experiments.

4.1 Experimental Setup

Live experiments. First, we seek to measure the potential benefits of deploying a modified Tor router on the currently deployed Tor network. Since only the Tor exit router needs to be modified in order to use Conflux, we deploy a single exit router and conduct a series of performance measurements using a Tor client that we control.

Each measurement is collected as follows: First, a Conflux multipath circuit is built using two entry routers $entry_1$ and $entry_2$, two middle routers $middle_1$ and $middle_2$, and our modified exit router *exit*. In order not to expose other clients' traffic, we set the exit policy of *exit* so that it can only connect to a specific web server. This means that *exit* will act as an exit router only for our traffic, but it can be a middle or an entry node for other clients' encrypted traffic. Using Conflux, our client fetches 320 KiB, 1 MiB and 5 MiB files. These file sizes were chosen to approximate web pages and larger files [41]. For comparison, the stock Tor client downloads the files over different circuits it builds using Tor's default bandwidth-weighted router selection algorithm.[8]

[8] To reduce any bias in the performance results due to the selection of particularly fast or slow entry guards, we disable the use of entry guards for this experiment.

Table 1: Network model for whole-network experiments

Attribute	*Data Source*
Pairwise link latency	King dataset [19]
Tor router bandwidth	Tor consensus (Nov. 2011)
Tor client bandwidth	Ookla Net Index dataset [37]
Traffic characteristics	Tor traffic study [32]

Measurements were collected from December 2012–January 2013, during which time our exit router was configured with a bandwidth rate of 200 KB/s.

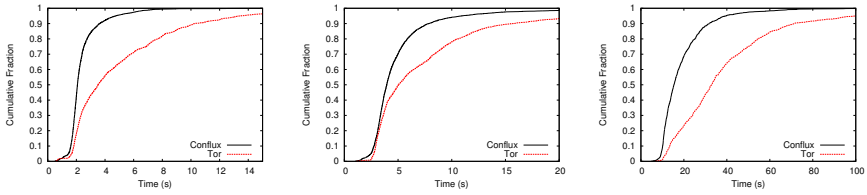
To evaluate the performance benefits for people using low-bandwidth Tor bridges, we also collect measurements where our client uses Tor bridges as its entry nodes. We collected 36 bridges by using Tor's standard HTTPS request service and we manually solved CAPTCHAs. The bridge clients work as follows. Every thirty minutes, our stock Tor and Conflux clients choose six bridges randomly from the list of 36 bridges we obtained, and use them as the first hop on each circuit they construct.

Whole-network experiments. One of the limitations of a live performance evaluation is that it is generally not possible to understand how performance might change when all participants of the network adopt the new design. To help understand these *whole-network effects*, we also perform experiments using the ExperimenTor testbed [6]. ExperimenTor is a Tor network testbed and toolkit that enables whole-network Tor experiments on a network topology with realistic delay, bandwidth, and other characteristics using the Modelnet [61] network emulation platform.

One challenge in conducting such an evaluation is that one must faithfully model the dynamics of the live network such as network latency, bandwidth, and traffic characteristics and replicate them in isolation. To enhance the realism of our experiments, we use a variety of empirical data sources, summarized in Table 1, to construct a network topology based on realistic link latencies, Tor router bandwidths sampled uniformly from a Tor consensus document from November 1, 2011, and asymmetric Tor client bandwidths assigned by sampling from the Ookla Net Index broadband data set [37] (the interquartile ranges are 4 –13 Mbit/s downstream, 0.5–1.9 Mbit/s upstream).

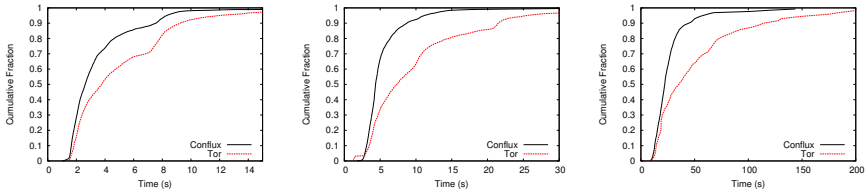
Client traffic models. In addition to building a realistic network topology, it is important to replicate the dynamics of the network's traffic. Since Tor's users are anonymous, it is inherently difficult to characterize real Tor traffic. One such study [32] exists, which reported that over 92% of TCP connections leaving a Tor exit router result from web browsing and make up nearly 60% of the network's aggregate traffic volume. However, BitTorrent accounts for only about 3% of connections, but comprises over 40% of the aggregate traffic volume. We employ these empirical observations in developing realistic traffic models for our whole-network experiments. Beyond modeling the dynamics of the past and present Tor network, we also consider emerging trends in Internet traffic.

We model two types of clients in our experiments: First, web browsing clients are modelled as fetching 320 KiB files (the median web page size on the Internet [41]) with random think time pauses between 1–30 seconds (chosen uniformly at random). A similar distribution of "think times" between web requests was measured by Hernández-Campos *et al.* [22]. Second, bulk clients (e.g., streaming video) download 5 MiB files with uniformly random delays of 1–5 minutes between fetches. This download size



(a) 320 KiB download time (b) 1 MiB download time (c) 5 MiB download time

Fig. 4: Live performance comparison between Tor and Conflux



(a) 320 KiB download time (b) 1 MiB download time (c) 5 MiB download time

Fig. 5: Download time performance comparison between Tor and Conflux using live Tor network bridges

and delay distribution approximates observations of YouTube video sizes and viewing durations [26]. As mentioned in Section 2, we model streaming clients by large file downloads [42].

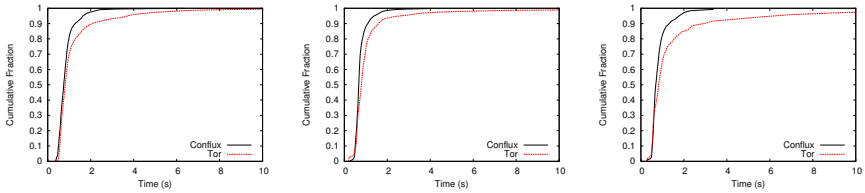
To highlight the performance benefits for bandwidth-deprived bridge clients, we also conduct whole-network experiments in which clients use a bridge in lieu of an entry guard. Since bridges are typically run by Tor clients themselves, we configure five bridges to run on asymmetric broadband-like Internet connections chosen from the Ookla Net Index data set.

4.2 Results

Performance metrics. To evaluate the performance of our technique, we measure *time-to-first-byte* and *download time*. The time-to-first-byte is the time it takes the client to receive the first cell of data after it issued a request; it is a good measure of both the responsiveness of the network and its congestion state. The other metric we consider is download time, which is simply the time it takes for the client to receive the last byte of data after issuing a request (download time includes the time-to-first-byte).

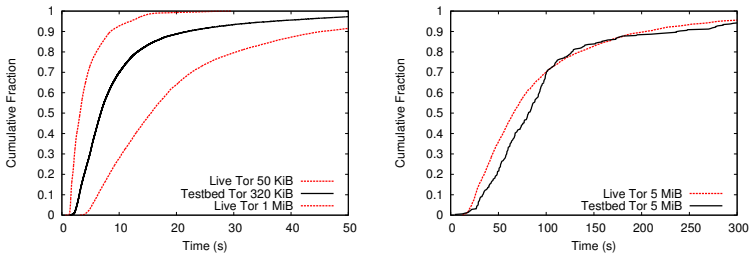
Live performance. Figure 4 compares the time for a client to download 320 KiB, 1 MiB, and 5 MiB files using Tor and Conflux. We observe a noticeable improvement in download times for all file sizes. However, improvements are more visible with larger file sizes. For example, the median improvement in download time for the 320 KiB and 1 MiB files is 42% and 25%, respectively. For the 5 MiB files, the improvement is around 54%. The reason why improvements are more visible for larger file sizes is the mechanics of TCP congestion control (see Appendix A for details).

Live performance for bridge users. When we apply Conflux with clients who use low-bandwidth bridges as their entry nodes, we also observe significant improvement in performance, regardless of the download size. Figures 5(a), 5(b) and 5(c) show that



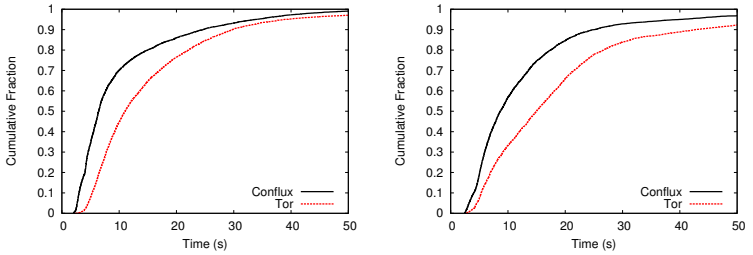
(a) 320 KiB time-to-first-byte (b) 1 MiB time-to-first-byte (c) 5 MiB time-to-first-byte

Fig. 6: Time-to-first-byte performance comparison between Tor and Conflux using live Tor network bridges



(a) Download times for small files (b) Download times for large files

Fig. 7: Performance comparison between ExperimenTor (testbed), when 370 web browsers and 30 bulk clients are used, and the live Tor network using torperf. These graphs show that our testbed setup realistically reproduces the performance of the live Tor network.



(a) 10 bulk and 390 web browsing clients (b) 30 bulk and 370 web browsing clients

Fig. 8: Download time comparison between Tor and Conflux for web clients that download 320 KiB using bridges in whole-network deployment experiments

the download times for 320 KiB, 1 MiB, and 5 MiB files are significantly improved; the performance improvement is most significant for the 5 MiB download, which experiences an improvement of over 50% relative to Tor.

Furthermore, Figures 6(a), 6(b) and 6(c) compare the times-to-first-byte for clients who use bridges to download 320 KiB, 1 MiB, and 5 MiB files using Tor and Conflux. Regardless of the download size, the Conflux clients experience faster response times compared to Tor. By using multiple circuits, if one circuit is congested, Conflux is able to send cells down a second, possibly uncongested circuit.

Whole-network deployment. We next seek to evaluate the performance of our technique if all our browsing bridge users and bulk clients in the network upgraded. In our large-scale experiments, we deploy a 20-router Tor network on our ExperimenTor testbed. Next, we fix the number of the total Tor clients to 400. Of the 400 clients, 30 clients act as bulk downloaders and 370 clients act as the interactive web browsers. Our initial experiments revealed that if all web clients who use entry guards use Conflux, those clients observe no performance benefits because the exit routers would be slower than the entry guards and would become the bottlenecks in circuits constructed by Conflux. Therefore, for our whole-network deployment experiments, we focus on browsing *bridge users*, since they have more performance incentives to use Conflux.

Before we present our whole-network deployment results, we first compare the stock Tor download time measurements obtained from our testbed, when 370 web and 30 bulk clients are used, with the live Tor network measurements maintained by the Tor metrics portal [56]. Figure 7(a) shows that the stock Tor download time distribution, of 320 KiB files obtained using ExperimenTor, fits between the download time distributions of the 50 KiB and 1 MiB files obtained from the live Tor network. Note that the Tor project only maintains the download times of 50 KiB, 1 MiB and 5 MiB file sizes over the live Tor network.

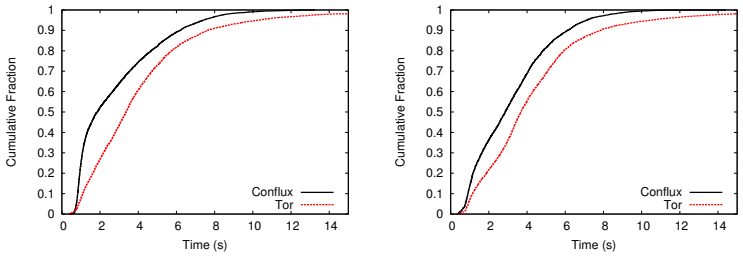
Second, Figure 7(b) demonstrates that our testbed's download time distribution of 5 MiB files closely approximates the respective distribution obtained from the live Tor network. In fact, the results look accurate for the fourth quartile of the download times, and for the first three quartiles, the testbed performance is only 15% slower than the live Tor network.

Although we believe that using 370 web and 30 bulk clients produces an accurate approximation, we also experiment with a lighter load of 390 web and 10 bulk clients, as there are continual efforts to reduce the load in the network by throttling the bulk downloaders [23, 35]. Furthermore, in each experiment, among the 390 or 370 web clients, we fix the number of bridge users to 50. We also run an additional five low-bandwidth routers that act as bridges.[9] Those bridges are neglected by non-bridge users.

Next, we present the results for the web browsing bridge users (for bulk clients' results, see Appendix B). Figure 8 shows the significant performance gains in download times experienced by bridge users using Conflux when the network is lightly loaded with 10 bulk clients and 390 web browsers. At the median, it takes a Tor client 11 seconds to finish downloading the 320 KiB file, whereas with Conflux, it takes only 6 seconds, which is approximately a 45% improvement. When we increase the load to 30 bulk clients to match the performance of the current Tor network, we still observe significant download time improvements. For Tor clients, the download times are degraded by 4 seconds, as the download time reaches 15 seconds at the median, whereas with Conflux, the degradation is only 2 seconds, as downloads complete in 8 seconds. Therefore, even with congestion, Conflux maintains the performance advantage of roughly 46%.

Figure 9 tells us a similar story for the time-to-first-byte results. Under a light load of 10 bulk clients, Conflux needs only 1.8 seconds before the browser starts changing for the bridge user at the median. The respective waiting time for the Tor client is 3.3 seconds, which is a 45% improvement in time-to-first-byte using Conflux. When we

[9] Since little is known about the total number and behavior of bridge users, we make reasonable assumptions in designing these experiments.



(a) 10 bulk and 390 web browsing clients (b) 30 bulk and 370 web browsing clients

Fig. 9: Time-to-first-byte comparison between Tor and Conflux for web clients that download 320 KiB using bridges in whole-network deployment experiments

increase the load to 30 bulk clients, the median time-to-first-byte is still improved by roughly 23% when Conflux is used. Therefore, our large-scale experiments confirm our live experiments, in which we observed large performance benefits when Conflux is used for bridge users. Table 2 summarizes our whole-network experimental results.

Table 2: Relative download time comparison at the 80th percentile for Conflux and Tor under increasing traffic loads

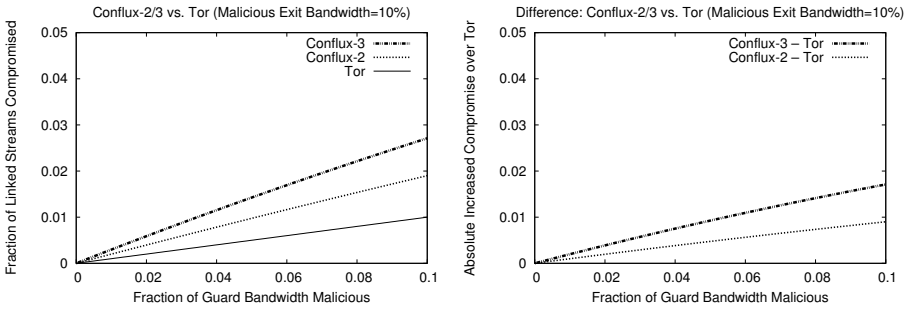
User type	*Light load*	*Regular load*
Web browsing bridge users	Improved by 32%	Improved by 34%
Streaming (bulk) users	Improved by 17%	Improved by 3%
Other clients	Not affected	Not affected

5 Security Analysis

We next investigate how the multipath routing scheme in Conflux affects the probability of the adversary linking together a circuit's source and destination. This linking, or *circuit compromise*, occurs when the adversary controls both endpoints of a given circuit [52] and applies timing analysis [28, 49] to reveal the communication patterns between the client and its corresponding destination. We first analyze the potential for path compromise due to passive attacks. Active attacks are considered in Section 5.3.

5.1 Identifying Bridge Users

Exit ORs can easily recognize which clients are using Conflux. Therefore, to prevent exit ORs from identifying bridge users, both bridge users and non-bridge users are encouraged to use Conflux. In fact, non-bridge browsing users can benefit from Conflux because browsing activity often involves streaming or downloading large files or images. In such situations, non-bridge users can observe between 3% to 17% performance improvements as we have seen in the results of our whole-network deployment experiments summarized in Table 2. This substantial improvement provides them with strong incentives to use Conflux and thereby aid in increasing the anonymity set of bridge users who adopt Conflux.



(a) Path compromise for Conflux and Tor (b) Difference between Conflux and Tor

Fig. 10: Security of Conflux with two and three circuits compared to stock Tor.

5.2 Path Compromise

We next examine the effect Conflux has on client exposure and compare it to Tor. We look at both client compromise rates as well as individual circuit compromise rates to provide a more thorough discussion of the security implications of Conflux.

For client compromise, we adopt the model used by Elahi *et al.* [17], which provides a realistic and empirical approach to determining client exposure due to real-world network phenomena such as guard churn. In their model, clients are considered compromised whenever there is a malicious relay in their guard set. Note that if a client has a guard set containing malicious guards, then the number of circuits created before choosing a malicious guard would be sightly smaller for Conflux as compared to Tor. However, such a client would be deanonymized with either Tor or Conflux, and so from a *client compromise* point of view, we consider either situation to be equally bad. (We will consider *circuit compromise*, in which one cares about how *often* a client's circuits are compromised, below.) This captures the upper bound on the absolute compromise levels of the client population. The reason why only clients with all guards honest are assured safety is that if even one of the guards is malicious then the client will eventually pick a malicious guard at the same time as picking a malicious exit, and will be exposed.

This model depends only on the distribution of the guards amongst the clients' guard sets and is not affected by the multipath scheme Conflux utilizes. Conflux does not alter how guards are selected by clients into their guard sets nor how guards are selected for circuits. This makes the guards clients pick with Conflux identical to those picked with Tor, and hence both systems have identical exposure rates.

Under certain circumstances, individual circuit compromise rates can be just as important as client compromise rates. The following formula captures the probability of a compromised multipath in Conflux.

$$P(\text{Compromised}) \stackrel{\Delta}{=} f_{xbw} \cdot (1 - (1 - f_{gbw})^k) \tag{1}$$

Table 3: Compromised circuit rates at different values of k (the number of entry guards used for a (multi)path) given m malicious relays in the user's guard set at 10% malicious guard bandwidth (for the computation of B_m) and 10% malicious exit bandwidth (for the computation of P_m)

		P_m(Compromised)		
Malicious Relays in Guard Set (m)	Probability of m (B_m)	$k=1$ (Tor)	$k=2$ (Conflux-2)	$k=3$ (Conflux-3)
0	72.9%	0%	0%	0%
1	24.3%	3.33%	6.67%	10%
2	2.7%	6.67%	10%	10%
3	0.1%	10%	10%	10%

Here, k is the number of guard nodes[10] used in the multipath, while the adversary controls a proportion f_{xbw} of the network's exit bandwidth and a proportion f_{gbw} of the network's guard bandwidth. The equivalent value for Tor is $f_{xbw} \cdot f_{gbw}$, the probability of selecting both a malicious exit and guard. Note that, as expected, this is the same as the expression for Conflux when $k = 1$.

In order to understand the implications of this formula, we compare Conflux — denoted Conflux-2 where two guards are used on a multipath and Conflux-3 where three guards are used — with Tor and plot the results in Figure 10(a). It is clear that while Conflux-2/3 increases the chance of a compromised circuit, it is very slight in absolute terms; at 10% compromised exit bandwidth it is 0.9% in Conflux-2 and 1.72% in Conflux-3, as shown in Figure 10(b). The tradeoff between this slight increase and that of increased performance can be evaluated depending on the needs of the client.

We further analyze the probability distribution (B_m) of a client having m malicious guards in its guard set as well as the probability (P_m) of constructing compromised circuits given m compromised guards. Assuming 10% malicious exit bandwidth, Table 3 provides the probabilities of picking 0 or more malicious guard relays as well as the probabilities of constructing malicious circuits under those conditions. *Note that because malicious guards are chosen infrequently, Conflux often adds no additional vulnerability to path compromise over and above Tor.*[11]

The same analysis follows for AS-level adversaries that can observe the connections between clients and their guards/bridges, and between exit relays and the destinations. Substituting the fraction of circuits observed by the adversary between the two end points, i.e. $f_{client-guard}$ and $f_{exit-destination}$, for f_{gbw} and f_{xbw} respectively in Equation 1, we can calculate compromise rates in similar fashion to those in Table 3. While it is difficult to estimate realistic adversarial AS coverage, Edman and Syverson [16] and Wacek *et al.* [62] provide values of between 18–25% for the parameters above.

[10] This analysis is also applicable to bridge relays since Conflux utilizes them in the same manner as guard relays.

[11] As of July 2012, the Tor network had roughly 1,200 MiB/s aggregate guard bandwidth and roughly 600 MiB/s aggregate exit bandwidth [55]. We believe that an adversary in possession of 10% of each of these bandwidths is a powerful, high-resource attacker. Thus, this should be considered a worst-case security analysis.

5.3 Selective Denial of Service

We now consider active attacks, in the form of selective denial of service (SDoS) [5, 8, 11, 60]. Under this attack the adversary actively breaks circuits he is part of if he finds that either end point is not controlled by him. This causes the client to create compromised circuits at a higher rate. The adversary can choose to either SDoS circuits immediately or be strategically patient.

Recall that Conflux establishes *primary circuits* first, to which further (secondary) circuits are linked to form multipaths. Hence, this mechanism provides two avenues for the SDoS attack: first at the primary circuit building stage and second at the circuit linking stage. We analyze each in turn.

We model the likelihood of primary circuit compromise with SDoS by inputing Formula 1 from Section 5.2 in the following formula, proposed by Das and Borisov [11], where f is the fraction of malicious relay bandwidth:

$$P(SDoS) = \frac{P(\text{Compromised})}{P(\text{Compromised}) + (1-f)^3} \tag{2}$$

With SDoS, the only primary circuits that are possible will either be compromised ($P(\text{Compromised})$) or entirely honest ($(1-f)^3$). At 10% malicious bandwidth this attack increases the likelihood to 2.54% from 1.9% for Conflux-2 and to 3.57% from 2.7% for Conflux-3. Compare this to Tor where the same attack increases the likelihood to 1.35% from 1%. The increases here are due to the primary circuit creation phase and not due to Conflux's multipath linking scheme.

The strategic adversary servicing a primary circuit with a compromised exit relay may gain an added advantage by performing SDoS exclusively on the secondary circuits being linked to the primary. At 10% malicious bandwidth the probability of this scenario is 9%. Under Conflux-2, employing SDoS on such secondary circuits exclusively gains the adversary 0.23% whereas Conflux-3 gains the adversary 0.41%. The adversary needs to balance the cost of this strategy in terms of bandwidth used to service the uncompromised primary and the added advantage it provides.

A possible countermeasure is to ensure that the OP will retry the linked circuit using the same guard and will not switch to another (potentially malicious) guard. After a few retries, if it is unable to build a *secondary circuit* for the linked stream, the OP will give up but without impacting the client whose traffic still flows over the uncompromised primary circuit.

6 Discussion

Congestion. Conflux does not introduce any additional traffic to the network as our multipaths are bounded by the window of the primary circuit and by the bandwidth of the exit routers. Indeed, in our large-scale experiments, we found that the windows are not exhausted when Conflux is used. As indicated by our results, congestion is reduced because Conflux dynamically senses latency on each circuit as a congestion indicator, which allows our traffic splitting approach to perform smart load balancing so that congestion can be avoided. Moreover, Conflux is complementary to incentive schemes that seek to increase the number of exit routers and bridges.

Computational Cost. One might wonder if Conflux adds more complexity to the operation of routers since it requires building more circuits resulting in more public key (PK) operations. Since Tor clients build spare idle circuits by default, Conflux can use those circuits as secondary circuits in order not to add more PK operations in building multipaths. For circuits that must be built on demand, Conflux doubles the cost of building circuits (assuming only two paths are used). However, those costs can be significantly reduced if Conflux uses Tor 0.2.4's ntor handshake [12], which eliminates PK decryptions at routers during circuit construction.

Experimental Limitations. The results we obtained show an improvement in a testbed environment. In our experiments, we emulate the network behaviour and traffic load using ExperimenTor, since it allows us to set up different network topologies with different bandwidth and link settings. For practical purposes, we scaled the network down to 20 routers, which is not reflective of the real Tor network. However, we strived in our experiments to perform and obtain realistic performance measurements under different traffic loads. Performing large-scale experiments using different network models such as the ones recently proposed by Jansen *et al.* [24] and Wacek *et al.* [62] and performing a large-scale performance evaluation on the live Tor network are areas for future work.

Future Work. Recall that our dynamic traffic splitting technique is based only on two circuits. One area we want to explore is using Conflux with more than two circuits. Since we found the performance improvements to be significant with two circuits, we suspect that we may find even more performance benefits with more than two circuits. We leave exploring the performance benefits of splitting traffic over more than two circuits for future investigation.

Another avenue for future work is to evaluate alternative methods for measuring circuit latency. Recall that we opportunistically measure the latency of a circuit in order to access its performance using Tor's existing circuit-level SENDME cells that a client sends to the exit router after receipt of every 100^{th} data cell. This allows us to measure circuit latency/congestion for free (e.g., with no additional overhead to the network). An alternative technique is to introduce a new cell type that clients send to exits and perform more frequent measurements. A potential advantage of this approach is that it might detect sudden changes in latency and thereby react to congestion faster. However, a downside is that this approach is more hostile to the network, because it places more probe traffic onto the network.

7 Related Work

Multipath TCP (MPTCP). MPTCP [21] is a transport layer protocol that allows applications to send data over several interfaces/addresses, whereas Conflux is implemented at the overlay layer. Because data is source-routed in Tor, we can choose our multipaths and ensure they are only joined at the exit routers. In MPTCP, routing decisions are done by IP routers, and a TCP source cannot choose the Internet path. MPTCP is more useful if divergent paths are available (multi-homing).

Multipath to improve anonymity. In the context of high-latency Chaumian mix networks, Serjantov and Murdoch [46] show that sending packets over multiple disjoint routes through the mix network may increase anonymity against a partial passive ad-

versary. Also, Feigenbaum *et al.* [18] proposed multipath routing to mitigate timing attacks [47, 49] in low-latency anonymity networks, using a layered mesh topology.

Multipath to improve performance. In the context of low-latency onion routing networks, MORE [27] routes every packet through a different path of onion routers, half of which are chosen by the sender and half chosen by the receiver. While this approach offers the ability to create highly dynamic paths—which can have desirable performance, load balancing, and anonymity properties—the requirement that communicating parties participate in the anonymity network may reduce MORE's practicality.

To increase throughput in Tor, Snader [50] presents preliminary experiments in which clients receive n different streams over n disjoint circuits. In particular, multiple circuits are shown to reduce the time to download 1 MB files. However, it is unclear whether performance is improved for smaller web-like streams; furthermore, since this scheme uses more entry and exit points into the Tor network, it may increase the threat of end-to-end traffic confirmation attacks (as in Bauer *et al.* [5]).

Combination with other Tor performance work. There are several proposals that aim to improve the performance of Tor in different ways such as congestion control and avoidance [4, 20, 43, 53, 63], improved router selection [48, 51], scalability [33, 34], and applying incentive schemes to increase the number and bandwidth of Tor relays [25, 35, 36]. Conflux is complementary to these previous approaches and we expect that even greater performance benefits are possible by combining these proposals together. We leave this for future investigation.

8 Conclusion

Motivated by the need to improve the performance of Tor, we presented the design, implementation, and analysis of Conflux, a dynamic congestion-aware traffic splitting scheme that is designed to improve the load balancing of the network which is particularly useful for clients using low-bandwidth routers such as bridges. We evaluated Conflux using a series of small-scale experiments on the live network, and using large-scale experiments on an isolated testbed. Our results indicate that significant performance benefits can be obtained at the expense of a slight decrease in anonymity.

Acknowledgements. We would like to thank Eugene Vasserman and the anonymous reviewers for their helpful input in improving the quality of this work. We also thank our sponsors: NSERC, ORF, The Tor Project Inc., and Qatar University.

References

1. Allman, M.: Internet Draft: Initial Congestion Window Specification. `http://tools.ietf.org/html/draft-allman-tcpm-bump-initcwnd-00` (November 2010)
2. Allman., M., Paxson, V., Blanton, E.: RFC 5681: TCP Congestion Control. `http://tools.ietf.org/html/rfc5681` (September 2009)
3. AlSabah, M., Bauer, K., Elahi, T., Goldberg, I.: The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. In: Proceedings of the 13th Privacy Enhancing Technologies Symposium (July 2013)
4. AlSabah, M., Bauer, K., Goldberg, I., Grunwald, D., McCoy, D., Savage, S., Voelker, G.M.: DefenestraTor: Throwing out Windows in Tor. In: Privacy Enhancing Technologies Symposium. pp. 134–154 (July 2011)

5. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-Resource Routing Attacks against Tor. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007). pp. 11–20 (October 2007)
6. Bauer, K., Sherr, M., McCoy, D., Grunwald, D.: ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In: Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET). pp. 51–59 (August 2011)
7. Blond, S.L., Manils, P., Chaabane, A., Kaafar, M.A., Castelluccia, C., Legout, A., Dabbous, W.: One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users. In: Proceedings of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats. LEET'11, USENIX Association (2011)
8. Borisov, N., Danezis, G., Mittal, P., Tabriz, P.: Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In: Proceedings of CCS 2007. pp. 92–102 (October 2007)
9. The Chromium Projects: SPDY: An Experimental Protocol for a Faster Web. http://dev.chromium.org/spdy/spdy-whitepaper, accessed February 2012
10. Chu, J., Dukkipati, N., Cheng, Y., Mathis, M.: Internet Draft: Increasing TCP's Initial Window. http://tools.ietf.org/html/draft-ietf-tcpm-initcwnd-00 (October 2010)
11. Das, A., Borisov, N.: Securing Tor Tunnels under the Selective-DoS Attack. In: Proceedings of Financial Cryptography and Data Security (February 2013)
12. Dingledine, R., Mathewson, N.: Tor protocol specification. https://gitweb.torproject.org/torspec.git/blob/HEAD:/tor-spec.txt
13. Dingledine, R., Mathewson, N.: Anonymity Loves Company: Usability and the Network Effect. In: Workshop on the Economics of Information Security. pp. 547–559 (June 2006)
14. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium. pp. 303–320 (2004)
15. Dingledine, R., Murdoch, S.: Performance Improvements on Tor or, Why Tor is Slow and What We're Going to Do about It. http://www.torproject.org/press/presskit/2009-03-11-performance.pdf (March 2009)
16. Edman, M., Syverson, P.F.: AS-awareness in Tor path selection. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS). pp. 380–389 (2009)
17. Elahi, T., Bauer, K., AlSabah, M., Dingledine, R., Goldberg, I.: Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2012). ACM (October 2012)
18. Feigenbaum, J., Johnson, A., Syverson, P.: Preventing Active Timing Attacks in Low-Latency Anonymous Communication. In: Proceedings of the 10th Privacy Enhancing Technologies Symposium. pp. 166–183 (2010), http://portal.acm.org/citation.cfm?id=1881151.1881161
19. Gil, T.M., Kaashoek, F., Li, J., Morris, R., Stribling, J.: King Data Set. http://pdos.csail.mit.edu/p2psim/kingdata, accessed August 2011
20. Gopal, D., Heninger, N.: Torchestra: Reducing Interactive Traffic Delays over Tor. In: Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society. pp. 31–42. WPES '12, ACM, New York, NY, USA (2012)
21. Han, H., Shakkottai, S., Hollot, C.V., Srikant, R., Towsley, D.: Multi-path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet. IEEE/ACM Trans. Netw. 14(6), 1260–1271 (Dec 2006)
22. Hernández-Campos, F., Jeffay, K., Smith, F.D.: Tracking the Evolution of Web Traffic: 1995-2003. In: Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS). pp. 16–25 (2003)
23. Jansen, R., Syverson, P., Hopper, N.: Throttling Tor Bandwidth Parasites. In: Proceedings of the 21st USENIX Security Symposium (August 2012)
24. Jansen, R., Bauer, K., Hopper, N., Dingledine, R.: Methodically Modeling the Tor Network. In: USENIX Workshop on Cyber Security Experimentation and Test (CSET) (August 2012)
25. Jansen, R., Hopper, N., Kim, Y.: Recruiting New Tor Relays with BRAIDS. In: Proceedings of ACM CCS. pp. 319–328 (October 2010)

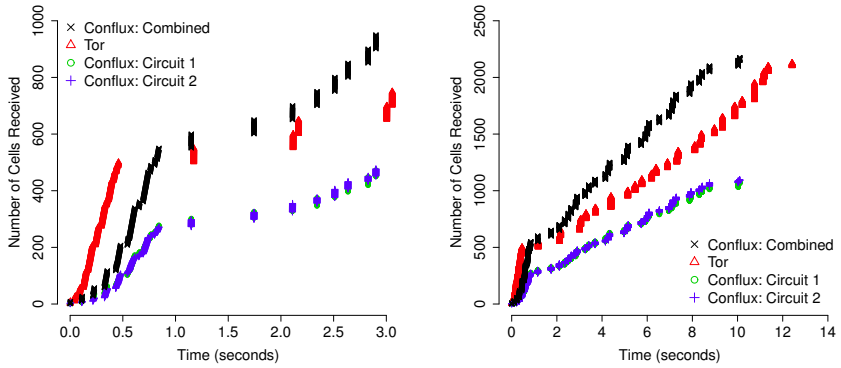
26. King., A.: Average Web Page Size Septuples Since 2003. Website Optimization, LLC. http://www.websiteoptimization.com/speed/tweak/average-web-page, accessed February 14, 2012
27. Landsiedel, O., Pimenidis, A., Wehrle, K., Niedermayer, H., Carle, G.: Dynamic Multipath Onion Routing in Anonymous Peer-to-Peer Overlay Networks. In: IEEE Global Telecommunications Conference, 2007. pp. 64–69 (Nov 2007)
28. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing Attacks in Low-Latency Mix-Based Systems. In: Proceedings of Financial Cryptography. pp. 251–265 (February 2004)
29. Lewman, A.: China Blocking Tor: Round Two. https://blog.torproject.org/blog/china-blocking-tor-round-two (March 2010), accessed August 2011
30. Ling, Z., Luo, J., Yu, W., Yang, M., Fu, X.: Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery. In: Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM) (March 2012)
31. Maier, G., Feldmann, A., Paxson, V., Allman, M.: On Dominant Characteristics of Residential Broadband Internet Traffic. In: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. pp. 90–102 (November 2009)
32. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining Light in Dark Places: Understanding the Tor Network. In: Proceedings of the 8th Privacy Enhancing Technologies Symposium. pp. 63–76 (July 2008)
33. McLachlan, J., Tran, A., Hopper, N., Kim, Y.: Scalable Onion Routing with Torsk. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 590–599. CCS '09, ACM, New York, NY, USA (2009)
34. Mittal, P., Olumofin, F., Troncoso, C., Borisov, N., Goldberg, I.: PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In: Proceedings of the 20th USENIX Security Symposium (August 2011)
35. Moore, W.B., Wacek, C., Sherr, M.: Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race with Tortoise. In: Proceedings of the 27th Annual Computer Security Applications Conference. pp. 207–216. ACSAC '11 (2011)
36. Ngan, T.W.J., Dingledine, R., Wallach, D.S.: Building Incentives into Tor. In: Proceedings of Financial Cryptography. pp. 238–256 (January 2010)
37. Ookla: Net Index by Ookla — Source Data. http://www.netindex.com/source-data, accessed on January 27, 2012
38. The OpenNet Initiative: YouTube Censored: A Recent History. http://opennet.net/youtube-censored-a-recent-history, accessed February 6, 2012
39. Øverlier, L., Syverson, P.: Locating hidden servers. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy. pp. 100–114 (May 2006)
40. Piatek, M., Kohno, T., Krishnamurthy, A.: Challenges and Directions for Monitoring P2P File Sharing Networks-or: Why My Printer Received a DMCA Takedown Notice. In: Proceedings of the 3rd conference on Hot topics in security. pp. 12:1–12:7 (July 2008)
41. Ramachandran, S.: Web Metrics: Size and Number of Resources. https://code.google.com/speed/articles/web-metrics.html, accessed August 2011
42. Rao, A., Legout, A., Lim, Y.s., Towsley, D., Barakat, C., Dabbous, W.: Network Characteristics of Video Streaming Traffic. In: Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. pp. 25:1–25:12. CoNEXT '11, ACM, New York, NY, USA (2011)
43. Reardon, J., Goldberg, I.: Improving Tor Using a TCP-over-DTLS Tunnel. In: Proceedings of the 18th USENIX Security Symposium (August 2009)
44. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous Connections and Onion Routing. Selected Areas in Communications, IEEE Journal on 16(4), 482–494 (May 1998)
45. Sandvine: Sandvine Global Internet Phenomena Report — Fall 2011. http://www.sandvine.com/downloads/documents/10-26-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Report%20-%20Fall%202011.pdf (October 2011)
46. Serjantov, A., Murdoch, S.J.: Message Splitting Against the Partial Adversary. In: Proceedings of Privacy Enhancing Technologies Workshop. pp. 26–39 (May 2005)

47. Serjantov, A., Sewell, P.: Passive Attack Analysis for Connection-Based Anonymity Systems. In: Proceedings of ESORICS. pp. 116–131 (October 2003)
48. Sherr, M., Blaze, M., Loo, B.T.: Scalable Link-Based Relay Selection for Anonymous Routing. In: PETS '09: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies. pp. 73–93. Springer-Verlag, Berlin, Heidelberg (2009)
49. Shmatikov, V., Wang, M.H.: Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In: Proceedings of ESORICS 2006. pp. 18–33 (September 2006)
50. Snader, R.: Path Selection for Performance- and Security-Improved Onion Routing. Ph.D. thesis, University of Illinois at Urbana-Champaign (2010)
51. Snader, R., Borisov, N.: A Tune-up for Tor: Improving Security and Performance in the Tor Network. In: Proceedings of the Network and Distributed Security Symposium (NDSS) (February 2008)
52. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an Analysis of Onion Routing Security. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability. pp. 96–114 (July 2000)
53. Tang, C., Goldberg, I.: An Improved Algorithm for Tor Circuit Scheduling. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 329–339. ACM, New York, NY, USA (2010)
54. The Tor Project: Codename: Torouter. https://trac.torproject.org/projects/tor/wiki/doc/TorouterAssignedTicketstothisproject, accessed August 2011
55. The Tor Project: Tor Metrics Portal: Bandwidth History by Relay Flags. https://metrics.torproject.org/network.html?graph=bwhist-flags&start=2012-07-01&end=2012-07-02&dpi=72#bwhist-flags, accessed July 2012
56. The Tor Project: Tor Metrics Portal: Data. https://metrics.torproject.org/data.html#performance, accessed November 2012
57. The Tor Project: Tor Metrics Portal: Network. http://metrics.torproject.org/network.html?graph=networksize&start=2012-01-01&end=2012-01-31&dpi=72#networksize, accessed November 2012
58. The Tor Project: Tor Metrics Portal: Users. http://metrics.torproject.org/users.html, accessed November 2012
59. The Tor Project: Tor Bridges Specification. https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/bridges-spec.txt (May 2009), accessed August 2011
60. Tran, A., Hopper, N., Kim, Y.: Hashing It out in Public: Common Failure Modes of DHT-based Anonymity Schemes. In: ACM Workshop on Privacy in the Electronic Society. pp. 71–80 (November 2009)
61. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostić, D., Chase, J., Becker, D.: Scalability and Accuracy in a Large-Scale Network Emulator. SIGOPS Oper. Syst. Rev. 36, 271–284 (December 2002)
62. Wacek, C., Tan, H., Bauer, K., Sherr, M.: An Empirical Evaluation of Relay Selection in Tor. In: Proceedings of the Network and Distributed Security Symposium (NDSS) (February 2013)
63. Wang, T., Bauer, K., Forero, C., Goldberg, I.: Congestion-aware Path Selection for Tor. In: Proceedings of Financial Cryptography and Data Security (FC'12) (February 2012)
64. Wright, M.K., Adler, M., Levine, B.N., Shields, C.: The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. ACM Trans. Inf. Syst. Secur. 7(4), 489–522 (2004)
65. Xu, X., Mao, Z.M., Halderman, J.A.: Internet Censorship in China: Where Does the Filtering Occur? In: PAM. pp. 133–142 (2011)

A Effects of TCP congestion control on Conflux

When a file download through a Tor circuit (or a Conflux circuit) starts, *slow start* initially sets the TCP congestion window (cwnd) of the connection between the exit router and destination server to a small value (typically one segment size). The congestion window exponentially increases by a factor of two every RTT until reaching the slow start threshold (ssthresh). At this point, the linear growth *congestion avoidance* phase starts [2].

The exponential growth during slow start and the linear growth during congestion avoidance can be seen in Figure 11(a). During slow start, the link between the exit router and the destination server is the slowest (bottleneck) link on the Conflux circuits and, consequently, sometimes we see no improvement in download time for clients whose transfers complete prior to the congestion avoidance phase. However, for larger downloads, or when exit is not the bottleneck in the circuit, Conflux offers a higher download rate relative to Tor during congestion avoidance, which translates to an improved quality of service for such downloads, as shown in Figure 11(b). Thus, we expect Conflux to be most effective at improving performance for users whose downloads do not complete during slow start; this includes the users of applications like streaming video or file sharing.[12,13]



(a) Download rate comparisons during the TCP slow start phase

(b) Download rate comparisons for a 1 MiB download

Fig. 11: Typical download rates for Conflux circuits individually, together and in comparison to a Tor circuit for a 1 MiB file download, as measured by an exit router. (a) indicates that the performance of smaller downloads that complete during TCP slow start experience little noticeable difference between Conflux and Tor because the TCP connection between the exit node and the destination server is the bandwidth bottleneck. (b) shows that Conflux is faster than Tor for downloads that are larger than 600 cells (300 KiB).

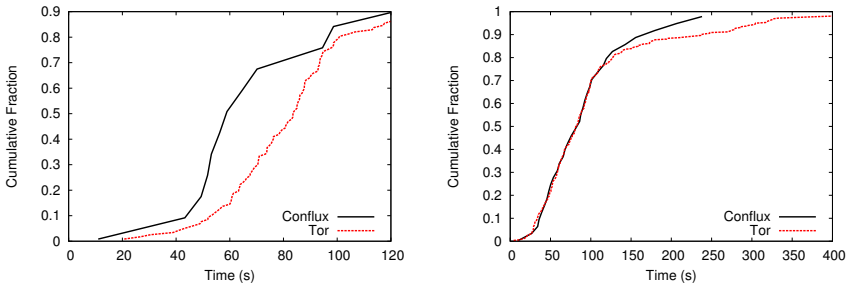
[12] We further confirm our conjecture regarding slow start by running a whole-network experiment where we used persistent TCP connections between exit nodes and servers. Indeed, the web browsing clients in that experiment observed improved performance because they were able to bypass the slow start effects.

[13] We note that this is a well-known performance problem due to the poor interaction between TCP with the short and bursty nature of web traffic [1, 10], not an issue inherent to Conflux or Tor. SPDY [9] has been proposed to ameliorate this problem.

B Whole-network experimental results for bulk clients

In this section, we present our results for the bulk clients (e.g. streaming users) for the whole-network experiments described in 4.2. Our whole-network experiments indicate that Conflux offers significant improvement, particularly for slower circuits. Under a light load of 10 bulk clients, shown in Figure 12(a), Conflux significantly improves the performance for bulk downloads compared to stock Tor. For example, at the median, it takes approximately 85 seconds to finish downloading a 5 MiB file for the Tor clients, whereas with Conflux, it takes approximately 60 seconds.

Under the regular load of 30 bulk clients, as seen in Figure 12(b), Conflux and Tor offer a similar performance for 80% of the downloads as the amount of spare bandwidth in the network becomes less available. However, for 20% of the downloads, Conflux still outperforms Tor.



(a) 10 bulk and 390 web browsing clients (b) 30 bulk and 370 web browsing clients

Fig. 12: Whole-network deployment experiments for the bulk downloaders that download 5 MiB files