The Path Less Travelled: Overcoming Tor's Bottlenecks with Multipaths

Mashael AlSabah, Kevin Bauer, Tariq Elahi, Ian Goldberg
Cheriton School of Computer Science, University of Waterloo
{malsabah,k4bauer,mtelahi,iang}@cs.uwaterloo.ca

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 multipath circuit construction and stream-splitting approach that 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 our scheme has no significant impact on users' security or anonymity.

1. INTRODUCTION

Privacy enhancing technologies are gaining attention and widespread use among ordinary Internet users, in part due to an increasing awareness of the many privacy threats present on the Internet today. Tor [13], a system for low-latency anonymous communications, 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 [42].

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 [63], monitor [39], or explicitly block access [37] to certain types of online content. While Tor offers online privacy and censorship resistance to an estimated 400,000 users on a daily basis [57], its public infrastructure of over 2,500 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. We discover that bridges generally provide a lower quality of service than Tor's public infrastructure, and we propose and evaluate a multipath routing scheme designed to improve performance for censored users.

Why is Tor slow? Tor's primary goal is to provide a *low-latency* service that is sufficient to support real-time interactive applications such as web browsing, instant messaging, and secure shell. However, it is well known that Tor suffers from a variety of performance problems [14] that are manifested as high and variable delays which result in a poor user experience. This poor experience discourages Tor adoption and ultimately results in a smaller user base and weaker anonymity for all users [12]. Therefore, there is a pressing need for Tor to improve its performance in order to attract more users, which will lead to a larger anonymity set and improved anonymity to all users.

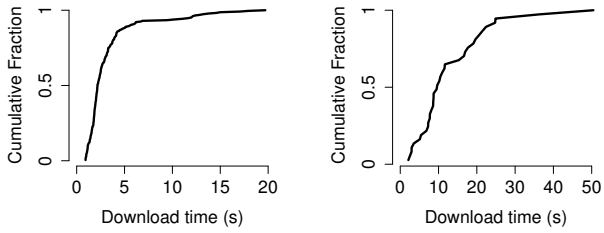
Prior work has diagnosed many of the causes of Tor's poor performance, which include the following:

- Tor handles congestion poorly [3, 41];
- Tor's path selection algorithm does not optimally load balance its traffic [50];
- Peer-to-peer downloaders are consuming too much of the network's scarce bandwidth [28]; and
- Tor may not have sufficient bandwidth to handle its traffic load and there are few incentives [19, 35] to encourage users to donate bandwidth.

In this work, we recognize that the diversity of bandwidth provided by Tor's volunteer-operated routers, and in particular the low-bandwidth bridges, degrade 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.*

Multipath routing for Tor. At present, Tor users construct circuits periodically and use each for approximately ten minutes. Each user's traffic is divided into fixed-size (512-byte) cells, and these cells are transported along the circuits. In the context of Tor, multipath routing offers the following benefits:

- *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.
- *Increase throughput.* 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.
- *Improve performance with low-bandwidth relays.* 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 10 MiB/s. Figure 1 shows a long-tailed distribution of download times for



(a) 50 KiB (b) 1 MiB

Figure 1: Time required to download files over Tor during January 2012.

50 KiB and 1 MiB files over the course of one month.[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 traffic splitting algorithm that forwards a client's individual cells down multiple circuits that share a common exit Tor router. Our algorithm dynamically measures the throughput of each constituent circuit and assigns traffic to each in proportion to its observed throughput. The circuit's endpoints (the client and the exit router) are responsible for splitting the traffic at one endpoint and buffering, reordering, 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 [5]. Our evaluation indicates that utilizing multiple circuits with Conflux 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 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 [27, 43].

We also critically evaluate the security implications of utilizing additional circuits in light of the well-studied end-to-end traffic confirmation attack [45, 47]. Our analyses indicate that our scheme does not significantly increase users' vulnerability to this attack, even when the adversary uses powerful selective denial of service tactics [4, 7, 11, 59] to maximize the number of circuits that can be compromised.

Contributions. This paper offers these contributions.

1. We motivate for the crucial need to improve the experience for bridge and streaming application users. To our knowledge, this is the first work that sheds light on the performance problems for those two emerging classes of users.
2. 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.
3. 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.
4. We analyze the security of our scheme and argue that the additional risks to users' anonymity are low.

[1] This data was obtained from The Tor Metrics Portal [55].

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

[3] Note that while Tor's browser bundle disables Flash by default, it is still possible to stream videos over Tor using HTML5.

2. BACKGROUND

How Tor works. Tor is the most widely used low-latency anonymity network; its design is based on onion routing [42], but the currently deployed network incorporates improvements over the original design in terms of security and efficiency. The Tor overlay network involves three kinds of nodes: Onion Proxies (OP), Onion Routers (OR), and directory authorities.

OPs run on end-users' machines, and allow them to connect anonymously to Internet hosts by interfacing between a user's application, such as a web browser, and the rest of the Tor network. Volunteer-run ORs (or *nodes*) are responsible for relaying users' traffic to other nodes or to destinations outside of Tor, such as web servers.

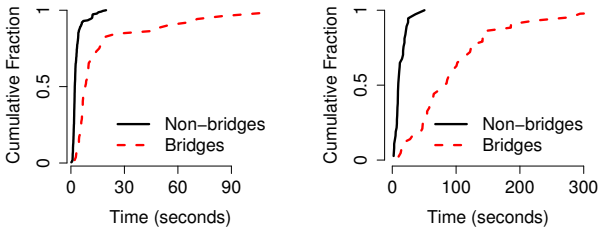
Tor's directory authorities are responsible for collecting and distributing information about ORs in the network. Before the client can connect anonymously through the Tor network, it must first bootstrap by downloading the *network consensus*, a document that contains the network information, including the contact information (IP address and port) of every OR in the network, their capacities, public keys, etc. This consensus is signed by a majority of the directory authorities. Next, the OP constructs a number of *circuits*—Tor-network paths through which the client's traffic is relayed in 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 [62] and other attacks that seek to correlate entry and exit traffic to link senders with their respective receivers [38]. If both the entry guard and exit node cooperate, they can use traffic analysis to link the initiator to her destination [24, 47].

Circuits and streams. There is generally one TCP connection 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 3.1.

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" [25]. 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 en-

[4] For more details about Tor's flow control mechanisms, see AlSabah *et al.* [3].



(a) 50 KiB (b) 1 MiB

Figure 2: Download time comparison between Tor users who use public entry guards and those who use bridges

sure 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 [58]. 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 80,000 censored users in January 2012 [57], 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 [56], bridges in January 2012 using Tor's standard HTTPS request service from PlanetLab hosts on 55 different /24 IP networks. (This procedure for enumerating bridges is described in more detail by Ling *et al.* [26].) Figure 2 compares the performance of a live Tor client that downloads 50 KiB and 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 devices such as wireless access points to simplify the process of configuring and running a bridge on a broadband Internet link at home [53]. 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) [20, 32], 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,[5] while streaming video applications are starting to account for an increasingly large fraction of Internet traffic by volume [27, 43]. 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. For example, if a user visits a blocked news website via Tor, the user may also want to view videos associated with the stories accessed.

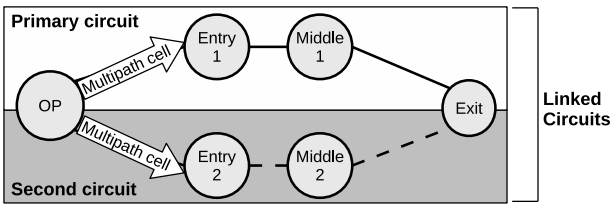


Figure 3: Multipath construction and stream linking

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

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.

3.1 Traffic Splitting

An important aspect of load balancing is the *traffic splitting problem* [21]. Traffic splitting can be performed on a whole-stream or sub-stream basis. In a whole-stream splitting approach, the OP, when creating a stream, will assign it *in its entirety* to one particular circuit of the multipath according to the desired load on each circuit. (Note that different streams may be assigned to different circuits.) This approach is easy to implement; however, because TCP streams differ in their load and bandwidth requirements, performing traffic splitting at the whole-stream level may result in ineffective load balancing. Worse, if streams use different exit routers, then the risk of traffic confirmation attacks may increase. Sub-stream splitting, on the other hand, has the advantage that it allows a finer 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. It also does not increase the likelihood of attack.

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

Constructing the multipath. As shown in Figure 3, 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 multipath routing, the client forms another circuit that uses different entry and middle ORs, which are also selected according to the weighted-bandwidth 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 control 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.[6] 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 per-cell traffic splitting. One approach to performing sub-stream traffic splitting is to perform traffic splitting in a round-robin fashion: cells in a stream are sent one (or a small fixed-sized batch) at a time down each circuit in turn. However, because differ-

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

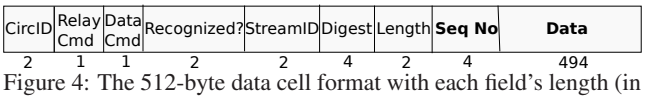


Figure 4: The 512-byte data cell format with each field's length (in bytes) for Conflux.

ent circuits may have different throughput capacities, sending equal amounts of traffic down each circuit may not result in optimal load balancing. To solve that problem, 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 4. This reduces the amount of data that can fit into each cell's payload by less than 1%.

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 latest development branch of the Tor source code (version 0.2.3.0-alpha-dev). We also implemented the weighted traffic splitting algorithm as described in Section 3.1. 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.

Table 1: Network model for whole-network experiments

Attribute	*Data Source*
Pairwise link latency	King [16] dataset
Tor router bandwidth	Tor consensus (Nov. 2011)
Tor client bandwidth	Ookla dataset [36]
Traffic characteristics	Tor traffic study [28]

4. PERFORMANCE EVALUATION

In order to empirically demonstrate the potential performance benefits of the proposed scheme, we present a series of experiments. In particular, we wish to understand the potential benefits of Conflux on the currently deployed live Tor network. We also conduct experiments in an isolated, network emulation environment to explore how Conflux might perform at scale, when adopted by many or all Tor clients and routers.

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.

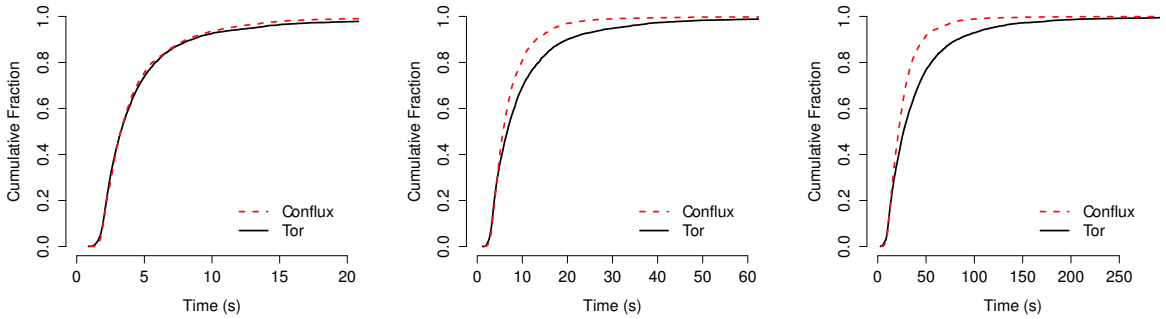
Each measurement is collected as follows: First, a Conflux multipath circuit is built using two entry routers *entry1* and *entry2*, two middle routers *middle1* and *middle2*, and our modified exit router *exit*. In order not to expose other clients' traffic, we set the exit policy of *exit* so that 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 300 KiB, 1 MiB and 5 MiB files. These file sizes were chosen to approximate web pages and larger files. For comparison, the client also downloads the files over each circuit *entry1*, *middle1*, *exit* and *entry2*, *middle2*, *exit*, independently. Note that for all positions of the circuit, Tor's default bandwidth-weighted router selection algorithm is used.[7] Measurements were collected from January 7 – February 11, 2012, during which time our exit router sustained approximately 4.5 MiB/s of background Tor traffic.

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 guards. The performance evaluation is conducted in the same manner as described above, only we use a Tor bridge as the first hop on each circuit, using the method for finding bridges described in Section 2. In total, the clients used one of 221 bridges and the measurements were collected from January 24 – February 11, 2012.

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 [5]. 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 [60] network emulation platform.

One challenge in conducting such an evaluation is that one must faithfully model the important dynamics of the live network such as network latency, bandwidth, and traffic characteristics and replicate them in isolation. In an effort to enhance the realism of our

[7] 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.



(a) 300 KiB download time (b) 1 MiB download time (c) 5 MiB download time

Figure 5: Live performance comparison between Tor and Conflux

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 Speedtest broadband data set [36] (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 [28] 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 modelling 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 [40]) 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.* [18]. Second, bulk clients (e.g., streaming video and BitTorrent users) download 5 MiB files with uniformly random delays of 1–5 minutes between fetches. This download size and delay distribution approximates observations of YouTube video sizes and viewing durations [22].

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 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. Improving the time-to-first-byte is especially important for interactive applications such as web browsing, as it directly informs the time between a user click and when the screen starts to change. In fact, prior work found that interactive users (such as web browsing users) have a low tolerance to delays beyond a few seconds [34]; thus, ensuring fast responses for interactive traffic is essential to Tor's usability. 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).

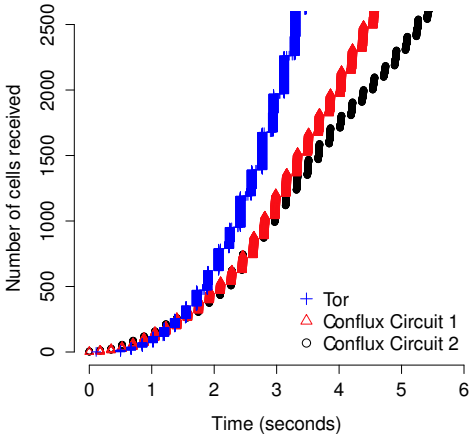


Figure 6: Download rates for Conflux circuits compared to Tor, as measured by a Tor exit router. The bursty arrivals of cells are shown by the groups of vertical points; the increasing number of bytes received in each group increases over the course of the download. The effect of TCP slow start can be seen during the first few hundred cells, or roughly the first two seconds.

Live performance. Figure 5 compares the time for a client to download 300 KiB, 1 MiB, and 5 MiB files using Tor and Conflux. We observe a noticeable improvement in download times for the 1 MiB and 5 MiB files; however, the smaller file downloads are unaffected by the use of multiple circuits. The reason for this is due to *TCP slow start*: when the file download starts, the TCP congestion window starts at a small value (typically one segment size) and TCP conservatively increases the window by a factor of two for each received acknowledgement [2]. This effect can be seen in Figure 6. During slow start, the link between the exit router and the destination server is the slowest (bottleneck) link on the Conflux circuits and, consequently, we see no improvement in download time for clients whose downloads complete during the slow start phase of congestion control.[8]

Live performance for bridge users. When we apply Conflux with clients who use low-bandwidth bridges as their entry nodes, we observe a more significant improvement in performance, regardless

[8] 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 [1,9], not an issue inherent to Conflux or Tor. SPDY [8] has been proposed to ameliorate this problem.

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

User type	*Light load*	*Medium load*	*High load*
Web browsing bridge users	Improved by 17%	Improved by 23%	Improved by 38%
Streaming users	Improved by 17%	Improved by 5%	Improved by 3%
Other clients	Not affected	Not affected	Not affected

Table 3: Relative download time comparison at the 80th percentile for Conflux and Tor under increasing traffic loads for a partial deployment where 50% of clients adopt Conflux

User type	*Light load*	*Medium load*	*High load*
Upgraded streaming users	Improved by 32%	Improved by 13%	Improved by 5%
Non-upgraded streaming users	Degraded by 31%	Degraded by 30%	Degraded by 7%
Other clients	Not affected	Not affected	Not affected

of the download size. Figure 7 compares the time-to-first-byte for a client who uses a bridge to download 300 KiB, 1 MiB, and 5 MiB files with Tor and Conflux. Regardless of the download size, the Conflux clients experience a faster response time compared to Tor. In fact, the figure shows that the 80th percentile of requests take more than 3 seconds before users start to see items displayed in their browser. With Conflux, on the other hand, it takes less than 2 seconds to start seeing changes in their browser window. Note that this is a huge improvement from the user perspective, as the threshold of user tolerance has been found to be approximately 2 seconds [34]. By using multiple circuits, if one circuit is congested, Conflux is able to send cells down a second, possibly uncongested circuit.

Furthermore, Figure 8 shows that the overall download times for 300 KiB, 1 MiB, and 5 MiB files are significantly improved for the majority of download trials; the performance improvement is most significant for the 5 MiB download, who experiences an improvement of over 75% relative to Tor.

Whole-network deployment. In our previous experiments, we have shown the performance benefits for a single client made possible by using Conflux. In this section, we seek to evaluate the performance of our technique if all 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.[9] In order to capture different network loads, we vary the number of bulk clients from 10 to 30. Increasing the number of bulk clients increases the load on the network, which helps us understand the effects of increasing network congestion on the performance of Conflux. Furthermore, in each experiment, among the 390 to 370 web clients, we fix the number of bridge users to 30. We also run an additional five low-bandwidth routers that act as bridges.[10] Those bridges are neglected by non-bridge users.

The whole-network experiments indicate that Conflux offers significant improvement particularly for slower circuits. Tables 2 and 3 provide a concise summary of the results. Under a light load of 10 bulk clients, as shown in Figure 9(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. Increasing the number of bulk clients to 20 reduces the available bandwidth in the network; however, in Figure 9(b), we still see a significant performance benefit for Conflux, as 80% of the requests experience an improvement between 13% to 15%. Finally, under an exaggerated load of 30 bulk clients,[11] 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.

Next, we present the results for the web browsing bridge users. Figure 10(a) shows the performance gains experienced by bridge users when using Conflux when the network is under a light load of 10 bulk clients. Although the performance of stock Tor bridge users is relatively fast, Conflux users still experience faster download times for 50% of the requests. For instance, with Conflux, the 80th percentile of downloads complete in 15 seconds, whereas with Tor, it takes 18 seconds.

As we introduce more congestion by increasing the number of bulk clients to 20, we observe that the performance of Tor degrades quickly, whereas Conflux remains more robust to congestion. As shown in Figure 10(b), with Conflux, the 80th percentile of requests complete downloading in 16 seconds, whereas with Tor, they take 21 seconds. Also, 65% of the requests experience faster downloads with Conflux compared to Tor.

Finally, the advantages of using multipath routing become more substantial for bridge users with extreme cases of congestion. It can be seen in Figure 10(c) that increasing congestion can significantly affect the variability of the download times of stock Tor bridge users. Also, although 20% of Tor bridge users experience slightly faster download times, the 80th percentile of download times are significantly improved with Conflux. For example, 20% of requests complete in 18 seconds with Conflux, whereas with Tor, it takes approximately 36 seconds. As for the web-browsing clients who do not use bridges, our experiments revealed that their performance is unaffected by the introduction of Conflux clients.

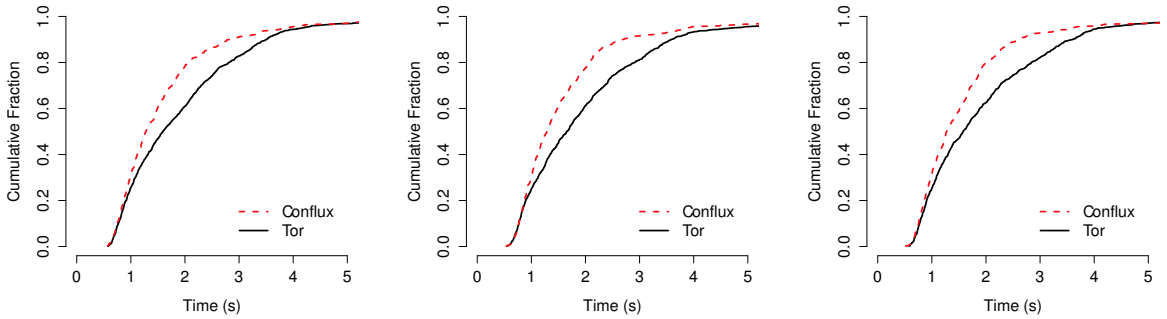
Partial deployment. We next present results for partial deployment experiments. It is important to understand how gradually upgrading the network with Conflux-enabled bulk clients might impact the performance of other clients. We modify our large-scale experiments so that half of our bulk clients run Tor while the other half run Conflux. The results are shown in Figure 11. As expected, the 50% Conflux-enabled clients outperform stock Tor users under different traffic loads. We also observe that web clients are unaffected by the partial Conflux adoption, across all traffic loads.

One interesting result that was revealed in this experiment is that when a fraction of the network uses Conflux, the performance of the rest of the network might get slightly worse compared to when no Conflux users are present in the network. The reason is that

[9] Since we do not have the computational resources to emulate the entire live Tor network at scale, we must scale down these experiments. However, the distribution of relay bandwidths and client traffic models are faithful to the live Tor network, as described in Section 4.1.

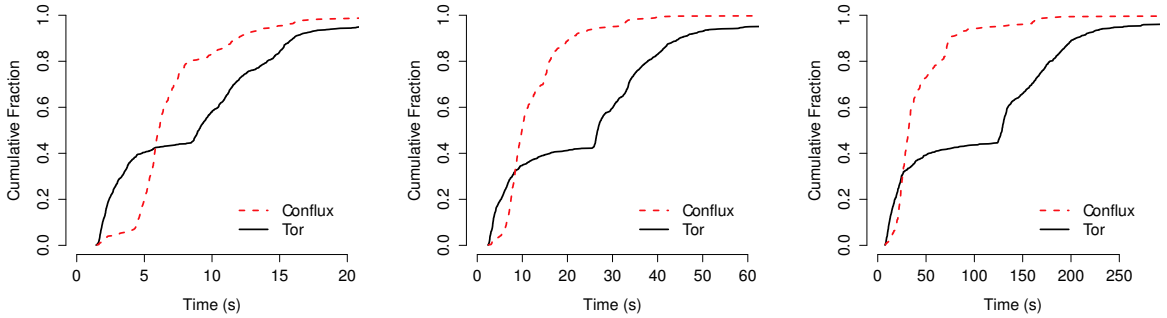
[10] Since little is known about the total number and behavior of bridge users, we make reasonable assumptions in designing these experiments.

[11] Note that it is usually assumed in the Tor literature that the bulk downloaders constitute approximately 3% of all clients [28, 32].



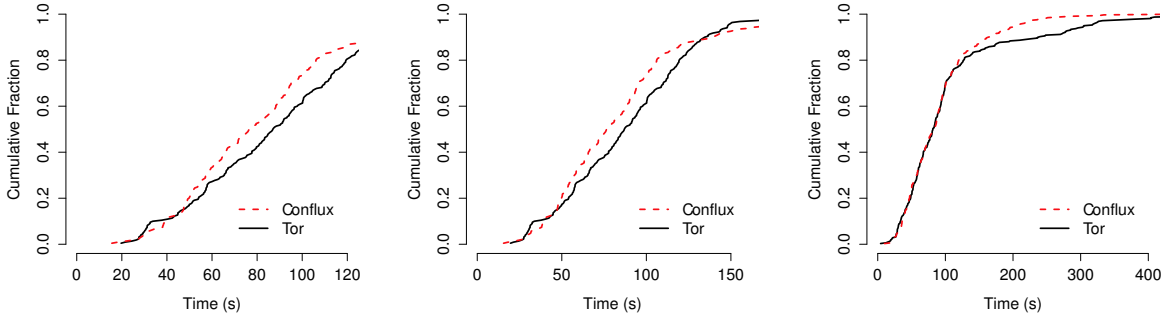
(a) 300 KiB time-to-first-byte (b) 1 MiB time-to-first-byte (c) 5 MiB time-to-first-byte

Figure 7: Time-to-first-byte comparison between Tor and Conflux using live Tor network bridges



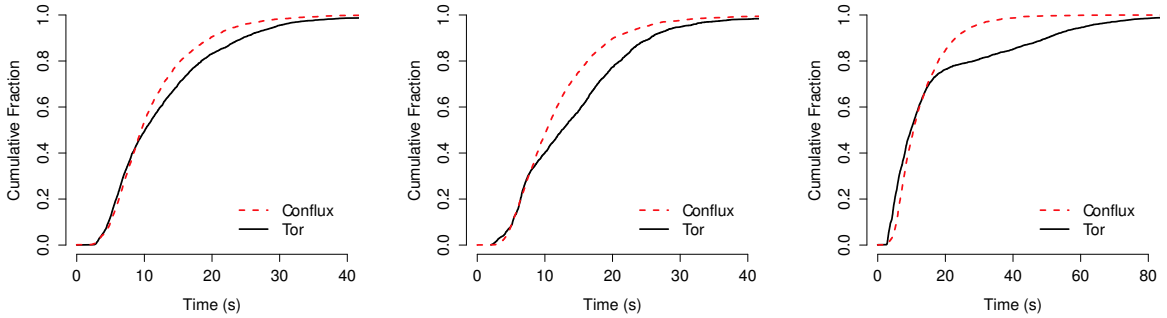
(a) 300 KiB download time (b) 1 MiB download time (c) 5 MiB download time

Figure 8: Download times comparison between Tor and Conflux using live Tor network bridges



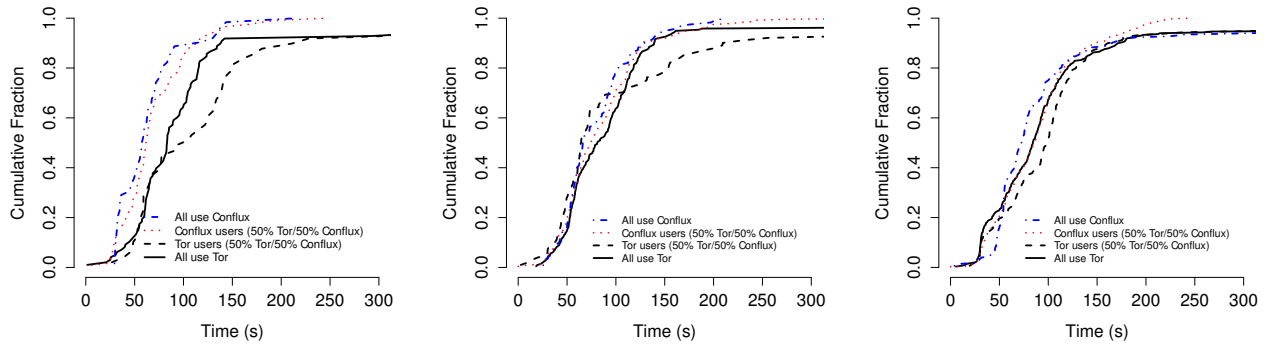
(a) 10 bulk and 390 web browsing clients (b) 20 bulk and 380 web browsing clients (c) 30 bulk and 370 web browsing clients

Figure 9: Whole-network deployment experiments for the bulk downloaders that download 5 MiB files



(a) 10 bulk and 390 web browsing clients (b) 20 bulk and 380 web browsing clients (c) 30 bulk and 370 web browsing clients

Figure 10: Whole-network deployment experiments for web clients that download 320 KiB using bridges



(a) 10 bulk and 390 web browsing clients (b) 20 bulk and 380 web browsing clients (c) 30 bulk and 370 web browsing clients

Figure 11: Expected download times as different fractions of bulk downloaders adopt Conflux

since Conflux clients use more network resources, ORs will spend more time and bandwidth servicing the traffic of Conflux-enabled clients, which affects the performance of non-upgraded users. This is an incentive for clients to upgrade and obtain better performance.

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 [51] and applies timing analysis [24, 47] to reveal the communication patterns between the client and its corresponding destination. Since Conflux uses more circuits than Tor and shares common exit relays amongst them, this might increase the potential for compromised circuits. We first analyze the potential for path compromise due to passive attacks. Active attacks are considered in Section 5.5.

5.1 Threat Model

We assume that the adversary controls a small set of malicious Tor routers. This means that the adversary can view traffic on some portion of the network and generate, modify, delete, or delay network traffic. This is the same threat model Tor's designers envisioned [13]. We also assume that the amount of compromised relays — both in terms of number and bandwidth — that exist in the network is no more than 20%; rates of compromise beyond this level are considered to be beyond Tor's threat model of a partial (non-global) adversary. The adversary's goal is to identify pairs of communicating parties, thereby defeating Tor's anonymity.

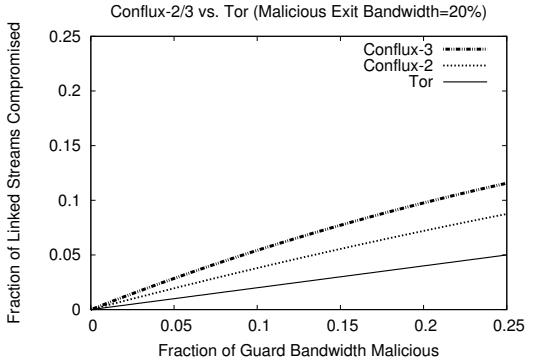
5.2 Identifying Bridge Users

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.

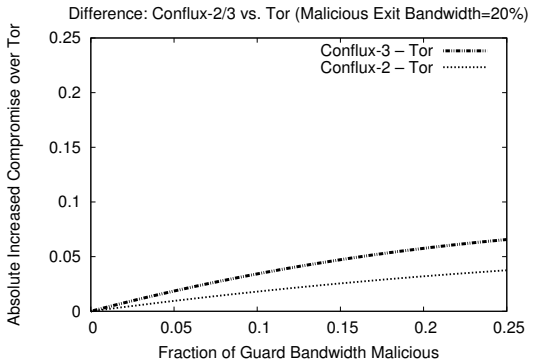
5.3 Path Compromise

Due to Tor's decentralized design, only guard relays can identify a client and only exit relays can identify the destination. The middle relays play no part in passively compromising a path. Conflux augments this design by allowing multipaths to share a common exit relay while having multiple guard relays. We examine the extent this increases a user's exposure to malicious exit and/or guard relays. The following formula captures the probability of a compromised multipath in Conflux.

$$P(\text{Compromised}) \triangleq f_{xbw} \cdot (1 - (1 - f_{gbw})^k) \qquad (1)$$



(a) Path compromise for Conflux and Tor



(b) Difference between Conflux and Tor

Figure 12: Security of Conflux with two and three circuits compared to stock Tor.

Here, k is the number of guard nodes 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 $k = 1$. See Section 5.6 for supporting details of this analysis.

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 12(a). It is clear that Conflux-2/3 does increase the chance of a compromised circuit but only slightly; at 20% compromised exit bandwidth it is 3.2% in Conflux-2 and 5.76% in Conflux-3, as shown in Figure 12(b). The tradeoff be-

tween this slight increase and that of increased performance can be evaluated depending on the needs of the client.

5.4 Malicious Bridges

Current bridge protection measures do not provide benefits to bridge clients. The focus is mainly to prevent enumeration and blocking of bridges [26, 48] and the identification of bridge operators [29] — clients are unprotected from malicious bridges. Contrast this with regularly listed guard routers where clients have some path compromise resistance at the first hop.

Bridges are not vetted by any authority for maliciousness.[12] Unlike guard nodes, however, there is no bandwidth or uptime requirement to become a bridge. It is therefore much easier for an adversary to deploy many malicious bridges than to deploy many malicious guard nodes; an adversary simply needs access to compute power and IP addresses. Products like Amazon Web Services, amongst others, provide easy access to both; The Tor Project even offers bridge images to run in the Amazon cloud [54]. Also, the costs of operating bridges are low due to the modest computational and bandwidth requirements. Putting the above together, the adversary can easily provision nodes and launch a dangerous attack.

This applies equally well to both regular Tor users and Conflux users; in either case, if a malicious adversary is running almost all of the bridges, the probability of path compromise is just f_{xbw} — the probability of picking a malicious exit. Conflux presents no security degradation over Tor in this case.

5.5 Selective Denial of Service

We now consider active attacks, in the form of selective denial of service (SDoS) [4, 7, 11, 59]. 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 more strategic and patient.

Let `M` denote a malicious relay and `H` an honest one. Each circuit, denoted by `G-X`, is comprised of an guard relay at the `G` position and an exit relay at the `X` position. Hence, `M-M` stands for a compromised circuit and `H-H`, `H-M` and `M-H` denote uncompromised circuits. The adversary wants to achieve at least one `M-M` within a given multipath. Note that malicious relays in any position in a circuit — guard, middle or exit — can launch a SDoS attack.

Recall that Conflux establishes *primary circuits* first to which further 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.

First, under a strictly SDoSing adversary the only primary circuits that can be created are those that have malicious end points or those that are completely honest — denoted by `M-M` and `H-H` respectively. In the former event the adversary need not further employ the SDoS in the linking stage since he is already part of the multipath circuit. In the latter, an SDoS has no influence and the adversary can only wait for circuit linking to occur.

Second, at the circuit linking stage if an `H-H` primary circuit is attached to an `M-H` circuit an SDoSing guard or middle relay does not provide the adversary any advantage. This is because Conflux will not change the exit relay due to an SDoS and hence a compromised circuit will not be possible. Meanwhile, traffic continues to flow over the primary circuit leaving the client unaffected.

[12] Since Tor exit nodes can view and/or modify unencrypted exit traffic, Tor's operators currently scan all exit routers for man-in-the-middle attacks or other injection/modification attacks. Bridges, however, cannot be scanned for such attacks.

An SDoSing adversary may also allow primary circuits of the form `H-M` with the expectation that one of the subsequent linked circuits attached to the multipath may be of the form `M-M`. This is governed by the client's guard set and the adversary does not know if it has any relays within it. If no guards are malicious then a `M-M` circuit cannot occur and the adversary will never succeed.

In the event that malicious relays are present in the guard set the adversary waits until either an `M-M` or another `H-M` linked circuit occurs. In the former case the adversary has succeeded, but without employing the SDoS, thus devolving to a passive path compromise attack discussed above. In the latter case, adversarial guard or middle relays can SDoS the linked circuit and hope that the remaining guard is malicious and a circuit beginning with it is created. If this occurs then the adversary will have succeeded with an SDoS.

A possible countermeasure is to ensure that the OP will retry the linked circuit using the same guard and will not switch to the another relay in the guard set. 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 `H-M` primary circuit. By not switching to another guard, which could be malicious, this countermeasure removes the advantage of the SDoS.

We model the likelihood of compromise with SDoS by using Formula 1 from Section 5 in the following formula, proposed by Das and Borisov [11], where f is the fraction of malicious relay bandwidth:

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

With SDoS, the only circuits that are possible will either be compromised ($P(\text{Compromised})$) or entirely honest ($(1-f)^3$). At 20% malicious bandwidth this attack increases the likelihood to 12.33% from 7.2% for Conflux-2 and to 16% from 9.76% for Conflux-3. Compare this to Tor where the same attack increases the likelihood to 7.25% from 4%. The increases here are due to the primary circuit creation phase and not due to Conflux's multipath linking scheme.

The analysis and discussion above confirm that, although it is as susceptible to SDoS attacks as Tor, Conflux does not introduce significant further anonymity or security vulnerabilities.

5.6 Effect of Entry Guards on Security

For each linked circuit the chance of being compromised is equal to the chance that 1) a malicious exit is picked and 2) there is at least one malicious guard relay in the client's guard set and 3) at least one of those malicious guards is an endpoint in the linked stream. Tor's guard design refreshes users' guard sets periodically, so given a long enough time frame all users will eventually have malicious relays in their guard set. We have incorporated Tor's current entry guard design in which clients select precisely three entry guards to use for all circuits into our analysis.

The probability distribution for the number m of malicious guards in a client's guard set of size t is given by:

$$B_m \triangleq \binom{t}{m} \cdot f_{gbw}^m \cdot (1 - f_{gbw})^{t-m} \tag{2}$$

while the probability of compromise given m malicious guards is:

$$P_m(\text{Compromised}) \triangleq f_{xbw} \cdot \left[1 - \frac{\binom{t-m}{k}}{\binom{t}{k}} \right] \tag{3}$$

Table 4: 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 20% malicious guard bandwidth (for the computation of B_m) and 20% 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	51.2%	0%	0%	0%
1	38.4%	6.67%	13.3%	20%
2	9.6%	13.3%	20%	20%
3	0.8%	20%	20%	20%

where k is the number of distinct guards used in the multipath. (As usual, we set $\binom{t-m}{k} = 0$ when $t-m < k$.) For Tor's current entry guard design, $t = 3$. It is easy to check analytically that Formulas 2 and 3 above agree with Formula 1 because

$$\sum_{m=0}^{t} B_m \cdot P_m(\text{Compromised}) = f_{xbw} \cdot (1 - (1 - f_{gbw})^k)$$

as expected.

From this analysis we learn that, at 20% malicious guard bandwidth, 51.2% of users do not have malicious relays in their guard sets (of size 3) and hence will be safe from circuit compromise for all of their circuits; this applies equally to Tor and Conflux. They are safe until they refresh their list of guards, currently done randomly between 30 to 60 days, at which point they have a 51.2% probability again of not picking any malicious guards.

Given 20% malicious exit bandwidth, Table 4 provides the probabilities of picking 0, 1, 2, or 3 malicious guard relays as well as the probabilities of constructing malicious circuits under those conditions. These details concur with and provide support for the security analysis provided in Section 5.3.

6. RELATED WORK

Multipath for IP and peer-to-peer networks. Multipath routing is a well-studied problem in the networking literature. Common core network routing protocols, such as Open Shortest Path First (OSPF) [33] and Interior Gateway Routing Protocol (IGRP) [17], implement load balancing, where routers distribute traffic among several possibly disjoint paths that have equal (or unequal) costs. These routing protocols measure the cost of each path and distribute traffic accordingly. The purpose is to relieve congestion and cope with failures. Another example is BitTorrent [10], which is a peer-to-peer file-sharing protocol that utilizes multipath routing to quickly and efficiently distribute files. BitTorrent peers maintain many simultaneously active connections with other peers, using an rarest first data request strategy with an "optimistic unchoking" algorithm to mitigate selfishness.

Multipath for anonymity networks. In the context of high-latency Chaumian mix networks, Serjantov and Murdoch [44] show that sending packets over multiple disjoint routes through the mix network may increase anonymity against a partial passive adversary.

Multipath routing has also been studied in the context of onion routing networks. MORE [23] 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 [49] 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 [4]).

Multipath routing has also been proposed to mitigate timing attacks [45, 47] in low-latency anonymity networks. Feigenbaum *et al.* [15] introduces a new anonymous communication structure, called a layered mesh topology, that defends against such attacks without delaying the user's traffic (as in mix networks). In the layered mesh, circuits carrying data from the client to a destination server are composed of w paths, where every path consists of ℓ relays, and all paths intersect at a common final (exit) router. The key difference between this work and Conflux is that Conflux aims to use multiple circuits to increase performance; in contrast, Feigenbaum *et al.* focus only on defeating timing attacks.

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 [3, 41, 52, 61], improved router selection [46, 50], scalability [30, 31], and applying incentive schemes to increase the number and bandwidth of Tor relays [19, 32, 35]. 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.

7. CONCLUSION

This work is motivated by the need to improve performance for Tor clients who utilize low-bandwidth bridge nodes to access the Tor network from locations that block access to Tor. We presented the design, implementation, and analysis of Conflux, a dynamic multipath traffic splitting scheme that offers higher throughput service even when clients utilize low-bandwidth nodes. Conflux is easy to deploy on the current Tor network, as it only requires upgrades for Tor clients and exit routers.

Our whole-network performance evaluation demonstrates a potential for decreased latency and increased throughput with our scheme, particularly for low-bandwidth bridge users. Interestingly, we also observed that users of emerging web applications such as streaming video also benefit from Conflux. Furthermore, experiments conducted on the live Tor network show that real Tor users today could experience better performance with Conflux. Our hope is that this work improves Tor's performance and usability, leading to a larger user base and greater anonymity properties.

8. 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., AND BLANTON, E. RFC 5681: TCP Congestion Control. `http://tools.ietf.org/html/rfc5681`, September 2009.

[3] ALSABAH, M., BAUER, K., GOLDBERG, I., GRUNWALD, D., MCCOY, D., SAVAGE, S., AND VOELKER, G. M. DefenestraTor: Throwing out Windows in Tor. In *Privacy Enhancing Technologies Symposium* (July 2011), pp. 134–154.

[4] BAUER, K., MCCOY, D., GRUNWALD, D., KOHNO, T., AND SICKER, D. Low-Resource Routing Attacks against

Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)* (October 2007), pp. 11–20.

[5] BAUER, K., SHERR, M., MCCOY, D., AND GRUNWALD, D. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET)* (August 2011), pp. 51–59.

[6] BLOND, S. L., MANILS, P., CHAABANE, A., KAAFAR, M. A., CASTELLUCCIA, C., LEGOUT, A., AND 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* (2011), LEET'11, USENIX Association.

[7] BORISOV, N., DANEZIS, G., MITTAL, P., AND TABRIZ, P. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *Proceedings of CCS 2007* (October 2007), pp. 92–102.

[8] THE CHROMIUM PROJECTS. SPDY: An Experimental Protocol for a Faster Web. `http://dev.chromium.org/spdy/spdy-whitepaper`. Accessed February 2012.

[9] CHU, J., DUKKIPATI, N., CHENG, Y., AND MATHIS, M. Internet Draft: Increasing TCP's Initial Window. `http://tools.ietf.org/html/draft-ietf-tcpm-initcwnd-00`, October 2010.

[10] COHEN, B. BitTorrent Protocol Specification. `http://wiki.theory.org/BitTorrentSpecification`, June 2011. Accessed August 2011.

[11] DAS, A., AND BORISOV, N. Securing Tor Tunnels under the Selective-DoS Attack. *Technical Report available at `http://arxiv.org/abs/1107.3863v1`* (July 2011).

[12] DINGLEDINE, R., AND MATHEWSON, N. Anonymity Loves Company: Usability and the Network Effect. In *Workshop on the Economics of Information Security* (June 2006), pp. 547–559.

[13] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium* (2004), USENIX Association, pp. 303–320.

[14] DINGLEDINE, R., AND 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.

[15] FEIGENBAUM, J., JOHNSON, A., AND SYVERSON, P. Preventing Active Timing Attacks in Low-Latency Anonymous Communication. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium* (2010), pp. 166–183.

[16] GIL, T. M., KAASHOEK, F., LI, J., MORRIS, R., AND STRIBLING, J. King Data Set. `http://pdos.csail.mit.edu/p2psim/kingdata`. Accessed August 2011.

[17] HEDRICK, C. L. An Introduction to IGRP. `http://www.cisco.com/image/gif/paws/26825/5.pdf`, August 1991.

[18] HERNÁNDEZ-CAMPOS, F., JEFFAY, K., AND 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)* (2003), pp. 16–25.

[19] JANSEN, R., HOPPER, N., AND KIM, Y. Recruiting New Tor Relays with BRAIDS. In *Proceedings of ACM CCS* (October 2010), pp. 319–328.

[20] JANSEN, R., SYVERSON, P., AND HOPPER, N. Throttling Tor Bandwidth Parasites. *University of Minnesota - Computer Science and Engineering Technical Report 11-019* (September 2012).

[21] KANDULA, S., KATABI, D., SINHA, S., AND BERGER, A. Dynamic Load Balancing without Packet Reordering. *SIGCOMM Comput. Commun. Rev. 37* (March 2007), 51–62.

[22] 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.

[23] LANDSIEDEL, O., PIMENIDIS, A., WEHRLE, K., NIEDERMAYER, H., AND CARLE, G. Dynamic Multipath Onion Routing in Anonymous Peer-to-Peer Overlay Networks. In *IEEE Global Telecommunications Conference, 2007* (Nov. 2007), pp. 64–69.

[24] LEVINE, B. N., REITER, M. K., WANG, C., AND WRIGHT, M. K. Timing Attacks in Low-Latency Mix-Based Systems. In *Proceedings of Financial Cryptography* (February 2004), pp. 251–265.

[25] LEWMAN, A. China Blocking Tor: Round Two. `https://blog.torproject.org/blog/china-blocking-tor-round-two`, March 2010. Accessed August 2011.

[26] LING, Z., LUO, J., YU, W., YANG, M., AND 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).

[27] MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference* (November 2009), pp. 90–102.

[28] MCCOY, D., BAUER, K., GRUNWALD, D., KOHNO, T., AND SICKER, D. Shining Light in Dark Places: Understanding the Tor Network. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium* (July 2008), pp. 63–76.

[29] MCLACHLAN, J., AND HOPPER, N. On the Risks of Serving Whenever You Surf: Vulnerabilities in Tor's Blocking Resistance Design. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009)* (November 2009), ACM.

[30] MCLACHLAN, J., TRAN, A., HOPPER, N., AND KIM, Y. Scalable Onion Routing with Torsk. In *Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 590–599.

[31] MITTAL, P., OLUMOFIN, F., TRONCOSO, C., BORISOV, N., AND GOLDBERG, I. PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In *Proceedings of the 20th USENIX Security Symposium* (August 2011).

[32] MOORE, W. B., WACEK, C., AND 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 (2011), ACSAC '11, pp. 207–216.

[33] MOY, J. RFC 2328: OSPF Version 2. http://tools.ietf.org/html/rfc2328, April 1998.

[34] NAH, F. F.-H. A Study on Tolerable Waiting Time: How long are Web Users Willing to Wait? *Behaviour Information Technology 23*, 3 (2004), 153–163.

[35] NGAN, T.-W. J., DINGLEDINE, R., AND WALLACH, D. S. Building Incentives into Tor. In *Proceedings of Financial Cryptography* (January 2010), pp. 238–256.

[36] OOKLA. Net Index by Ookla — Source Data. http://www.netindex.com/source-data. Accessed on January 27, 2012.

[37] THE OPENNET INITIATIVE. YouTube Censored: A Recent History. http://opennet.net/youtube-censored-a-recent-history. Accessed February 6, 2012.

[38] ØVERLIER, L., AND SYVERSON, P. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy* (May 2006), pp. 100–114.

[39] PIATEK, M., KOHNO, T., AND 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* (July 2008), pp. 12:1–12:7.

[40] RAMACHANDRAN, S. Web Metrics: Size and Number of Resources. https://code.google.com/speed/articles/web-metrics.html. Accessed August 2011.

[41] REARDON, J., AND GOLDBERG, I. Improving Tor Using a TCP-over-DTLS Tunnel. In *Proceedings of the 18th USENIX Security Symposium* (August 2009).

[42] REED, M. G., SYVERSON, P. F., AND GOLDSCHLAG, D. M. Anonymous Connections and Onion Routing. *Selected Areas in Communications, IEEE Journal on 16*, 4 (May 1998), 482 –494.

[43] 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.

[44] SERJANTOV, A., AND MURDOCH, S. J. Message Splitting Against the Partial Adversary. In *Proceedings of Privacy Enhancing Technologies Workshop* (May 2005), pp. 26–39.

[45] SERJANTOV, A., AND SEWELL, P. Passive Attack Analysis for Connection-Based Anonymity Systems. In *Proceedings of ESORICS* (October 2003), pp. 116–131.

[46] SHERR, M., BLAZE, M., AND LOO, B. T. Scalable Link-Based Relay Selection for Anonymous Routing. In *PETS '09: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 73–93.

[47] SHMATIKOV, V., AND WANG, M.-H. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *Proceedings of ESORICS 2006* (September 2006), pp. 18–33.

[48] SMITS, R., JAIN, D., PIDCOCK, S., GOLDBERG, I., AND HENGARTNER, U. BridgeSPA: Improving Tor Bridges with Single Packet Authorization. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)* (October 2011), ACM.

[49] SNADER, R. *Path Selection for Performance- and Security-Improved Onion Routing*. PhD thesis, University of Illinois at Urbana-Champaign, 2010.

[50] SNADER, R., AND 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).

[51] SYVERSON, P., TSUDIK, G., REED, M., AND LANDWEHR, C. Towards an Analysis of Onion Routing Security. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (July 2000), pp. 96–114.

[52] TANG, C., AND GOLDBERG, I. An Improved Algorithm for Tor Circuit Scheduling. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2010), ACM, pp. 329–339.

[53] THE TOR PROJECT. Codename: Torouter. https://trac.torproject.org/projects/tor/wiki/doc/TorouterAssignedTicketstothisproject. Accessed August 2011.

[54] THE TOR PROJECT. Tor Cloud. https://cloud.torproject.org/. Accessed February 2012.

[55] THE TOR PROJECT. Tor Metrics Portal: Data. https://metrics.torproject.org/data.html#performance. Accessed January 2012.

[56] 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 January 2012.

[57] THE TOR PROJECT. Tor Metrics Portal: Users. http://metrics.torproject.org/users.html. Accessed January 2012.

[58] THE TOR PROJECT. Tor Bridges Specification. https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/bridges-spec.txt, May 2009. Accessed August 2011.

[59] TRAN, A., HOPPER, N., AND KIM, Y. Hashing It out in Public: Common Failure Modes of DHT-based Anonymitys Schemes. In *ACM Workshop on Privacy in the Electronic Society* (November 2009), pp. 71–80.

[60] VAHDAT, A., YOCUM, K., WALSH, K., MAHADEVAN, P., KOSTIĆ, D., CHASE, J., AND BECKER, D. Scalability and Accuracy in a Large-Scale Network Emulator. *SIGOPS Oper. Syst. Rev. 36* (December 2002), 271–284.

[61] WANG, T., BAUER, K., FORERO, C., AND GOLDBERG, I. Congestion-aware Path Selection for Tor. In *Proceedings of Financial Cryptography and Data Security (FC'12)* (February 2012).

[62] WRIGHT, M. K., ADLER, M., LEVINE, B. N., AND SHIELDS, C. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Trans. Inf. Syst. Secur. 7*, 4 (2004), 489–522.

[63] XU, X., MAO, Z. M., AND HALDERMAN, J. A. Internet Censorship in China: Where Does the Filtering Occur? In *PAM* (2011), pp. 133–142.