ShorTor: Improving Tor Network Latency via Multi-hop Overlay Routing

Kyle Hogan
MIT CSAIL

Sacha Servan-Schreiber
MIT CSAIL

Zachary Newman
MIT CSAIL

Ben Weintraub
Northeastern University

Cristina Nita-Rotaru
Northeastern University

Srinivas Devadas
MIT CSAIL

***Abstract*—We present ShorTor, a protocol for reducing latency on the Tor network. ShorTor uses *multi-hop overlay routing*, a technique typically employed by content delivery networks, to influence the route Tor traffic takes across the internet. In this way, ShorTor avoids slow paths and improves the experience for end users by reducing the latency of their connections while imposing minimal bandwidth overhead.**

ShorTor functions as an overlay on top of onion routing—Tor's existing routing protocol—and is run by Tor relays, making it independent of the path selection performed by Tor clients. As such, ShorTor reduces latency while preserving Tor's existing security properties. Specifically, the routes taken in ShorTor are in no way correlated to either the Tor user or their destination, including the geographic location of either party. We analyze the security of ShorTor using the AnoA framework, showing that ShorTor maintains all of Tor's anonymity guarantees. We augment our theoretical claims with an empirical analysis.

To evaluate ShorTor's performance, we collect a real-world dataset of over 400,000 latency measurements between the 1,000 most popular Tor relays, which collectively see the vast majority of Tor traffic. With this data, we identify pairs of relays that could benefit from ShorTor: that is, two relays where introducing an additional intermediate network hop results in *lower* latency than the direct route between them. We use our measurement dataset to simulate the impact on end users by applying ShorTor to two million Tor circuits chosen according to Tor's specification.

ShorTor reduces the latency for the 99th percentile of relay pairs in Tor by 148 ms. Similarly, ShorTor reduces the latency of Tor circuits by 122 ms at the 99th percentile. In practice, this translates to ShorTor truncating tail latencies for Tor which has a direct impact on page load times and, consequently, user experience on the Tor browser.

I. Introduction

Tor is the foremost deployed system for anonymous communication. Millions of people around the world use Tor every day to escape censorship and avoid surveillance of their browsing habits [27,58]. This broad user base is a critical component of Tor's privacy guarantees. Tor users are anonymous only amongst each other—not within the general internet population. That is, an internet censor may be able to know that *some* Tor user visited a blocked site, but not *which* Tor user. Because of this, the degree of anonymity Tor provides in practice grows with the total number of concurrent users on the network [25].

This relationship between the privacy of individual users and the overall popularity of Tor makes user experience a major concern for Tor. A poor experience relative to non-private browsing results in lower adoption of Tor and, ultimately, limits the degree of anonymity Tor is capable of providing. A major factor contributing to positive user experience is latency. Internet users are very sensitive to latency, and increased page load times discourage user interaction [10,11,24]. Unfortunately, anonymous communication incurs higher latency than typical internet connections [8,31,53,54,56,65,83,84].

In Tor, much of this overhead is due to the underlying structure of its connections [26,27]. Tor is a network composed of ∼7,000 volunteer-run servers, or *relays*, used to route client traffic. Rather than connecting directly to their destination, Tor clients tunnel their traffic through a series of Tor relays in a process known as *onion routing*. This drastically increases the path length for Tor traffic, and, in turn, latency.

A substantial body of prior work aims to reduce latency in Tor by changing the relay selection process [5,7,9,16,42,74,79,87,90]. By default, Tor clients select relays for their circuit at random, weighted by relay bandwidth, and do not consider path length or circuit latency in the process. In contrast, proposals that aim to reduce latency often prioritize selecting circuits that have low latency between relays [5,9,16,42,74,79]. Unfortunately, preferentially choosing circuits in this way *also* selects relays that are correlated with the identity of the user or their destination [12,14,60]. Many attacks show how this can be exploited to deanonymize Tor users, allowing a passive observer to identify information about user locations [12,14,36,60,61,74,85,86].

In this paper, we propose ShorTor, an entirely different approach to reducing the latency of Tor traffic. Rather than alter the circuit selection process, ShorTor exploits a technique used by content delivery networks (CDNs) known as *multi-hop overlay routing* [22,81]. Multi-hop overlay routing, like Tor, functions by introducing intermediate hops into its connections, but does so for the explicit purpose of *reducing latency*. In the wild, CDNs use multi-hop overlay routing to influence the path internet traffic takes. They do this by inserting their own servers as intermediate points in client connections, avoiding slow default routes by forcing traffic to travel through their server, rather than directly to its destination. The success of this technique is due to the existence of sub-optimal default routes across the internet [28] and the distributed nature of CDN-controlled nodes. The broad presence of CDN controlled servers gives them *many* possible routes to choose from, and consequently, increases their odds of finding a faster route to send traffic through. In practice, this allows CDNs to avoid outages, congestion, or other delays along the default path.

With ShorTor, we ask:

> *Can multi-hop overlay routing reduce latency in Tor without compromising anonymity?*

While multi-hop overlay routing is widely successful for CDNs (which share some similarities to the Tor network), Tor is much smaller with ∼7,000 nodes [67] compared to the ∼300,000 operated by a CDN [3]. In addition to the difference in scale, Tor relays are volunteer run and their placement is not optimized for fast routing. ShorTor is the first proposal to apply and analyze the impact of multi-hop overlay routing on the Tor network, and is likely of independent interest to other distributed communication systems.

A. ShorTor

To reduce the latency experienced by end users of Tor, ShorTor uses multi-hop routing as an additional overlay layer on top of Tor's onion routing protocol. Crucially, ShorTor is independent of Tor's circuit selection algorithm and the client, operating only between relays. ShorTor introduces additional hops, which we call *via* relays, that tunnel traffic between relays on a Tor circuit. Acting as a via is simply a *role* that a normal Tor relay may take in addition to its usual function on circuits. Via relays, unlike circuit relays, can be introduced after circuit establishment in response to changing network conditions without client involvement or any modification to the circuit itself. While the basic idea of ShorTor is simple in retrospect, multi-hop overlay routing has security implications for anonymous communication that are not present in CDNs.

Security: We demonstrate that ShorTor can find faster paths across Tor *without* the loss in anonymity experienced by other approaches. ShorTor selects via relays based *solely* on the adjacent circuit relays. This process ensures that malicious vias cannot lie about their performance to artificially increase their selection probability. Specifically, ShorTor operates as an overlay routing layer, requiring no modification to Tor's onion routing or encryption, preserving Tor's security guarantees. We provide a formal security analysis of the impact ShorTor has on Tor's anonymity using the AnoA framework, which was introduced by Backes et al. [12] to analyze the anonymity guarantees of Tor [13,14]. Using AnoA, we show that ShorTor has minimal impact on security when compared to baseline Tor. However, we find that when used in conjunction with alternative,[1] location-aware path selection algorithms such as LASTor [5], ShorTor can exacerbate the existing leakage. We validate these claims through an empirical analysis on data collected from the Tor network.

Latency Measurements: To quantify the benefits of ShorTor, we conduct latency measurements between approximately 400,000 pairs of the 1,000 most popular Tor relays. We collect measurements ourselves, rather than use a general-purpose source for internet measurements such as RIPE Atlas [2], for two main reasons. (1) internet routing operates at a scale and complexity that cannot easily be simulated [75] and (2) ISPs often treat Tor packets differently from other internet traffic [23]. Using our own pairwise latency dataset we determine that, despite being much smaller than a typical CDN, Tor can still benefit from multi-hop overlay routing.

Ethics: Our measurements were conducted on the live Tor network, but did *not* involve any observations on Tor users or their traffic. We underwent Tor's security review process and followed best practices to limit our impact on the Tor network. Details can be found in Section IV-A5.

Practicality: While ShorTor does require modifications to Tor relays, it does *not* rely on participation of all, or even a majority of, relays and makes no assumptions about or modifications to client behavior. Tor circuits can benefit from multi-hop overlay routing as long as any two adjacent relays on the path both support it. The majority of our evaluation assumes that only the 1,000 most popular Tor relays participate, but we find ShorTor is beneficial with even fewer relays participating. ShorTor achieves a latency reduction of 178 ms at the 99.9th percentile with only the 500 most popular relays supporting the protocol. As such, ShorTor can be deployed incrementally and still provide meaningful reductions of tail latency on Tor.

Limitations: Our dataset of pairwise latencies was collected from the 1,000 most popular Tor relays. While these relays do see the majority of traffic on Tor [39], they are not representative of the full network. The less popular relays, while not as likely to be included in circuits, may benefit similarly from ShorTor, and could broaden the pool of available via relays. A deployed version of ShorTor, however, would naturally include all available relays regardless of popularity. As such, the scale of our dataset is strictly a limitation of our evaluation, not of ShorTor's effectiveness in practice.

Using this data, we find that ShorTor primarily impacts *tail* latencies on the Tor network. On average, ShorTor reduced the RTT between a pair of relays from 42.6 ms to 23.5 ms, while at the 99.9th percentile the RTT dropped much more substantially from 487 ms to 125 ms. As a result, the speedups ShorTor offers disproportionally benefit a relatively small fraction of Tor users—approximately 20,000 out of two million daily users select circuits that ShorTor can speed up by 120 ms or more.

Contributions: We propose ShorTor, the first protocol to apply multi-hop overlay routing to an anonymous communication network. ShorTor is designed to improve performance while preserving the security guarantees of baseline Tor, preventing adversarial relays from gaining an advantage by participating in ShorTor. We evaluate ShorTor using measured latencies from the live Tor network and show that ShorTor can significantly improve tail latencies on the Tor network with minimal bandwidth overhead.

In summary, this paper contributes:

1) ShorTor: a protocol for multi-hop overlay routing on Tor which reduces the latency experienced by Tor circuit traffic by 122 ms in the 99th percentile.

[1] Tor's only deployed path selection algorithm is independent of user location.

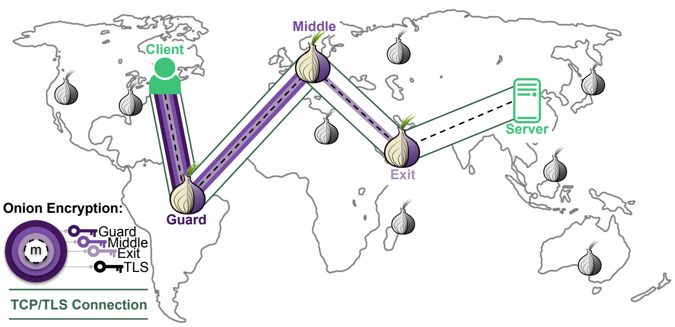

Fig. 1: A Tor circuit between a client and server: Tor relays are represented by onions. The circuit is a series of connections between three relays carrying onion-encrypted Tor cells.

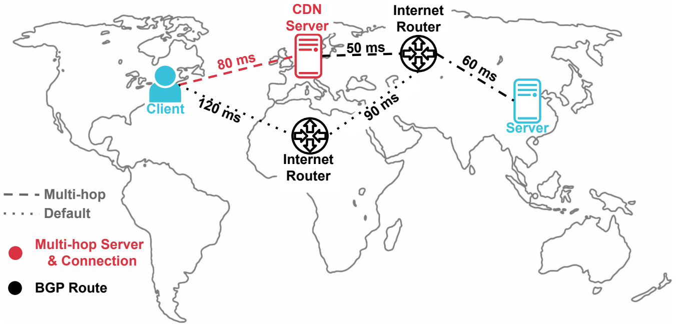

Fig. 2: Multi-hop overlay routing as in a CDN: the client avoids a slow BGP route to the blue server by addressing data to the red CDN server, which then forwards the traffic.

2) An evaluation of ShorTor's performance at various levels of deployment based on measured latency between the thousand most popular Tor relays.
3) A security analysis of ShorTor in the AnoA framework, demonstrating minimal impact to user anonymity.

II. Background

Here, we provide background on Tor and multi-hop overlay routing, which we combine in Section III to design ShorTor.

A. Tor

The Onion Router (Tor) is a network for anonymous communication comprising approximately 7,000 [67] volunteer-run relays that carry user traffic. We provide a brief overview of Tor's architecture and security guarantees. For more details on Tor, see the Tor specification [26] or paper [27].

1) Onion Routing: Tor users send their traffic through the Tor network using *onion routing*. Rather than communicating directly with their destination, clients send their traffic through "layers:" encrypted connections to three (or more) Tor relays in sequence. These relays form a *circuit* and have fixed roles:

Guard relays connect directly to the client and serve as an entry point into the Tor network,

Exit relays connect directly to the server and proxy communication on behalf of the client, and

Middle relays pass traffic between the Guard and Exit.

Figure 1 shows a single Tor circuit between a client and server including the connections and layers of encryption involved in Tor's onion routing protocol. The traffic flowing over a circuit is carried in fixed-size packets called *cells* which are onion encrypted. That is, cells have a layer of encryption for *each* relay on the circuit. Tor relays remove their layer of encryption when forwarding cells in the client-to-server direction and add their layer when returning the responses. This ensures that only the Exit can remove the innermost onion layer, protecting the client's privacy without requiring destination servers to handle onion encrypted data.

2) Path/Circuit Selection: Path—or circuit—selection is the process by which Tor clients select the set of relays that will form their circuit. This is a randomized process to ensure that the selection of relays is neither predictable nor correlated with the identities of either the client or server. It is, however, not *uniformly* random as relays have highly variable capacities and not every relay can support the same volume of traffic. As such, path selection is weighted based on a relay's available bandwidth, along with security considerations.

3) Tor's Adversarial Model: Tor is intended to provide *anonymity* to its users. Specifically, no adversary should be able to link the source and destination of any traffic stream across Tor. Tor's threat model considers adversaries in the form of malicious relays as well as external observers such as users' internet service providers. Anonymity in Tor is provided among all concurrent Tor users. While onion routing prevents any individual relay or localized network observer from directly linking a client to their destination, it does *not* hide the fact that a client is connected to the Tor network in the first place. Similarly, onion routing alone does not hide which servers are the destination of Tor connections. As such, both the *volume* and *diversity* of Tor users influence the degree of anonymity Tor is able to provide. In a well-known example of this principle, the sender of a 2013 Harvard bomb threat was identified despite their use of Tor because they were the *only* client connecting to Tor from Harvard's campus at the time [21].

4) Traffic Analysis: Traffic analysis attacks are a type of anonymity-compromising attack against Tor that identify features of encrypted traffic stream, such as packet interarrival times [57], to either: 1) recognize a previously observed stream [38,47,50,62], linking it across Tor or, 2) observe a pattern corresponding to a website *fingerprint* and infer the destination of traffic [20,45,72,80,88]. Both styles give the adversary an advantage in linking a client to their destination, compromising Tor's anonymity by making clients, servers, or client-server pairs more identifiable. We give additional details on the capabilities of such adversaries and their impact on Tor in Section V.

B. Multi-hop Overlay Routing

Multi-hop overlay routing is a technique that introduces intermediate waypoints into the connection between a client and server for the purpose of altering the route their traffic takes across the internet. There are many motivations for this technique—Tor's onion routing is itself an example of multi-hop overlay routing that provides anonymity by masking the direct relationship between client and destination server. More commonly, CDNs route their traffic over a multi-hop overlay in order to reduce the latency of their connections as illustrated in Figure 2 [22,81]. Two such examples are Cloudflare's Argo [55] and Akamai's SureRoute [4]. Rather than relying solely on the Border Gateway Protocol (BGP) to decide routes for their traffic, both Argo and SureRoute instead establish intermediate connections via their own servers. By routing their traffic via these intermediate waypoints they are able to identify and use a route which may be faster than the route selected by BGP. This is possible because BGP is subject to routing policies based on business relationships, not solely on shortest paths [77].

Importantly, this technique is an *overlay*—it runs on top of standard BGP without modifying any of the underlying protocols. This is achieved by establishing pairwise TCP connections between each of the intermediate points on the multi-hop overlay route rather than a single direct connection between the client and server. As such, standard BGP handles the route used *between* hops while the overlay protocol adjusts the ultimate path between client and server via the *placement* of its waypoints.

III. ShorTor

We propose ShorTor, a protocol for reducing the latency of connections over Tor. Like other such proposals, ShorTor preferentially selects faster routes across the Tor network for client circuits. In prior work, fast *routes* across Tor are equivalent to fast Tor *circuits*—Tor clients simply optimize for latency when selecting relays for their circuits.

Instead, ShorTor creates a *multi-hop overlay* on top of the Tor protocol to improve latency as shown in Figure 3. Rather than altering the circuit selection process to favor faster paths, ShorTor changes the routing *between* relays on existing circuits. It does this by offering circuit relays the option to route their traffic through an additional Tor relay rather than directly to the next hop. These intermediate hops, called *via relays*, are chosen on-demand by the relays themselves instead of in advance by clients. Via relays are **not** part of client circuits and do not participate in onion routing or encryption.

By routing as an overlay rather than altering circuit composition, ShorTor avoids security pitfalls of prior works while still providing a substantial reduction in latency on the Tor network. Directly optimizing for faster circuits, as in past proposals, has the unfortunate side effect of creating a correlation between the relays a client chooses and the client's location. Via relays in ShorTor are chosen *only* based on the circuit relays and inherit their relationship to the client—if circuits are chosen independently, as in the default Tor circuit selection, then via relay choices leak no information about the client. We discuss the implications of running ShorTor with alternative circuit selection techniques in Section V and Section VII.

ShorTor's design gives it several advantages over proposals that modify circuit selection:

1) **Security:** Routes in ShorTor are independent of the client and destination.

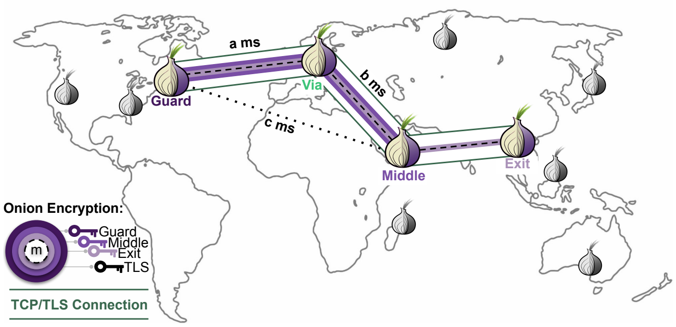

Fig. 3: A Tor circuit routing using a via relay between the guard and middle. A via relay will be used when the latency over the via (a+b ms) is less than that of the direct connection (c ms). The via does *not* participate in onion routing. For clarity, and because Shortor only operates between Tor relays, the client and server are not shown.

2) **Agility:** ShorTor can modify its routes as needed, not just during circuit construction.
3) **Compatibility:** ShorTor operates with *any* circuit selection algorithm, making it modular and compatible with future changes to the Tor protocol.

While we describe ShorTor in Tor-specific terms, we note that it applies to other distributed communication systems as well.

A. Security Model

ShorTor inherits Tor's adversarial model, as described in Section II-A3. It is designed to preserve the same anonymity guarantees against an adversary identifying the sender or recipient of a traffic stream which we define more formally in Section V. In particular, ShorTor requires no modification to Tor's baseline circuit selection or encryption and preserves independence between circuit choice and the identities of the client and server. However, ShorTor *does* necessarily change the number and distribution of relays that may see a given traffic stream, which could potentially be exploited by an adversarial Tor relay to deanonymize a larger share of Tor traffic. We formally consider the anonymity impact of ShorTor in our security analysis (Section V).

B. ShorTor Protocol

ShorTor introduces only one additional step into Tor's routing procedure. Rather than forwarding cells solely along previously established circuits, relays establish transient alternate routes between themselves and the next hop on their circuits. These alternate routes forward traffic via an additional Tor relay rather than sending it directly to the next relay on the circuit. As such, we refer to the intermediate hops between circuit relays as *via relays*, the connection between a circuit relay and a via relay as a *via connection*, and the communications over this connection as *via traffic*.

Note that the 'circuit' and 'via' modifiers denote different roles a relay may play in ShorTor, but do *not* correspond to different physical entities. A via relay is simply a regular Tor relay that has been chosen as an intermediate hop for some circuit rather than as part of the circuit itself. Any relay in Tor

Protocol 1: ShorTor

```
CIRCUIT RELAY
// Circuit relays conduct data races to determine if a suitable via exists
// between themselves and the next relay on the circuit.
// Parameter: self, Tor ID for this relay.
// State: Routes, the routing table for each circuit.
• SHORTOR.CHOOSEVIA(circ, dst, candidates)
    via ← RACE.RUN(candidates, circ, dst) // (Protocol 3).
    if via = ⊥, then return.      // no via faster than default route.
    Routes[circ].via ← via.

• SHORTOR.HANDLETRAFFIC(cell)
    if cell.cmd = VIA then return SHORTOR.HANDLEVIA(cell).
    candidates ← LATENCIES.VIASFOR(cell.next). // (Protocol 2)
    if Routes[cell.circ] = ⊥ then         // no routing table entry.
        SHORTOR.CHOOSEVIA(cell.circ, cell.next, candidates).
    via ← Routes[cell.circ].via.
    if via = ⊥ then proceed with default cell routing and return.
    Set cell.cmd = VIA and cell.prev = self.
    Send cell to relay via.
    if no response from via then
        candidates ← candidates \ {via}
        SHORTOR.CHOOSEVIA(cell.circ, cell.next, candidates).
        SHORTOR.HANDLETRAFFIC(cell)

VIA RELAY
// Via relays forward cells between circuit relays.
// Via relays do not perform onion decryption and only forward traffic
// if they have the available resources (i.e., bandwidth).
• SHORTOR.HANDLEVIA(cell)
    route ← Routes[cell.circ]
    if under heavy load or route = ⊥ then drop cell and return.
    Forward cell to route.next.
    Forward response from route.next to route.prev.
```

can act as both a circuit and a via relay simultaneously for different traffic streams.

1) ShorTor Protocol Stages: The ShorTor protocol proceeds in several stages. On an ongoing basis, relays take measurements of their round-trip latencies with other relays (LATENCIES.UPDATE(), Protocol 2). Circuit relays use these measurements to choose candidate via relays for outgoing traffic (SHORTOR.CHOOSEVIA(), Protocol 1). They perform a "data race" to choose the fastest path (RACE, Protocol 3). If a route with a via relay is faster than the default path, the circuit relay updates its routing table. In the steady state, the circuit relay handles traffic for its circuits as usual, but directs it to the via relay rather than to the next circuit hop.

Establishment: When establishing a connection for a given circuit, relays on that circuit will run LATENCIES.VIAFOR() (Protocol 2) to obtain a shortlist of potential vias. These vias are those that have recently been observed to provide the largest latency improvements over the default path between this relay and the next hop on its circuit. The circuit relay then performs a data race over each of the candidate vias (RACE.RUN(), Protocol 3).

The finish line of this race is the next relay on the relevant circuit which can report to the starting relay which of the data race cells arrived first. We provide details on the selection of candidate via relays in Section III-B2.

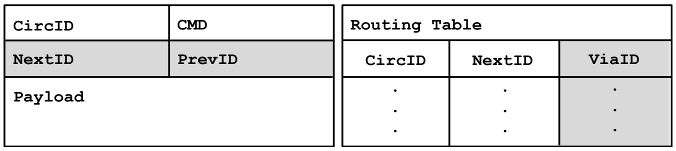

Fig. 4: Via Cell and Routing Table (fields added to baseline Tor highlighted in grey): Via cells contain additional header fields with routing info. This information is used by circuit relays to populate the routing table with which via (if any) each circuit should be routed through, while via relays use it to record where to forward cells from a given circuit.

Routing: While establishing a via connection, both the circuit and via relays must update their routing tables: circuit relays note which via to send cells to, while vias record which circuit relay should receive their forwarded traffic. To do this, we simply introduce new fields in Tor cell headers and routing tables, described in Figure 4. These allow relays to recognize traffic streams and route them to the correct next hop.

Steady-State: Traffic over via connections that have already been established is handled similarly to regular Tor traffic. Via relays simply forward the received traffic stream according to their routing table for the circuit. As via relays are not part of client circuits, they do *not* perform onion decryption/encryption prior to forwarding cells. Circuit relays also function as in baseline Tor except in cases where their routing table for a circuit contains a via relay. Then, the relay will alter the header on cells for these circuits as shown in Figure 4 and send the cells to the indicated via rather than to the next relay on the circuit. Periodically, relays on a circuit can repeat the data race to determine whether a given via is still the fastest option based current network status.

2) Latency Measurements: ShorTor relies on two forms of latency measurements (1) an up-to-date table of probable via candidates for each relay pair (Protocol 2) and (2) the data race that determines the fastest of the candidates (Protocol 3).

3) Pairwise Latency: ShorTor requires latency measurements between Tor relays to narrow down the set of potential via relay options for a circuit. In LATENCIES.UPDATE() (Protocol 2), each Tor relay collects this latency information as needed, distributing the involved storage, computation, and network load across the Tor network. This is in contrast to the centralized measurement methodology we use to evaluate ShorTor in Section IV which, while useful for this work, would not meet the performance needs of the live ShorTor protocol.

To participate in the distributed latency measurements of Protocol 2, each relay maintains their estimated round-trip latency to every other relay along with a list of "candidate" via relays. The candidates are computed by each relay using the round-trip latency tables for itself and for the destination relay based on latencies provided by the *destination*. ShorTor uses latencies reported from the destination for security reasons: an honest destination will not recommend a dishonest via relay

```
Protocol 2: LATENCIES

ALL RELAYS
// Parameter: $\ell$, how many routes to keep.
// Parameter: IDs of all $n$ active Tor relays: $\mathsf{Tor} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$.
// State: table RTTs: ping times to each other relay.
// State: table NextHop: for each relay, the top $\ell$ candidate vias $(\mathsf{id}, \mathsf{rtt})$.

• LATENCIES.UPDATE()
  // Keep RTTs and NextHop tables up-to-date.
  // Run periodically (once per day).
  for $\mathsf{id} \in \mathsf{Tor}$:
      Ping relay id to estimate round-trip time (RTT).
      Set RTTs[ID] to estimated value.
      Remove $(\_, \mathsf{rtt}) \in \mathsf{NextHop}[\mathsf{id}]$ with $\mathsf{rtt} \geq \mathsf{RTTs}[\mathsf{id}]$.
      $\mathsf{RTTs}_{\mathsf{id}} \leftarrow$ LATENCIES.RTTS() (remote call to relay id).
      for $\mathsf{via} \in \mathsf{Tor}$
          $\mathsf{rtt} \leftarrow \mathsf{RTTs}[\mathsf{via}] + \mathsf{RTTs}_{\mathsf{id}}[\mathsf{via}]$.
          if $\mathsf{rtt} \geq \mathsf{RTTs}[\mathsf{id}]$ then continue        // no speedup.
          Add $(\mathsf{via}, \mathsf{rtt})$ to NextHop[id], keeping fastest $\ell$ entries.

• LATENCIES.RTTS()
  Output RTTs.

• LATENCIES.VIASFOR(id)
  Output Routes[id].        // up to $\ell$ candidate via relays.
```

```
Protocol 3: RACE

CIRCUIT RELAY:
// Find the fastest via relay for reaching the destination relay.
// Parameter: myId, Tor ID of this relay.
// State: Seen, a set of circuit IDs for which this relay has seen data race packets.

• RACE.RUN(vias, circ, dst)
  Input: Candidate vias $\mathsf{vias} = \{v_1, \ldots, v_\ell\}$, destination relay dst
  Output: Fastest via $v$ if one exists; $\perp$ otherwise.
  Create data race cell cell with fields cmd = RACE, prev = self,
      next = dst, and circ = circ.
  for $\mathsf{via} \in \mathsf{vias}$: send cell to via.
  Send data race cell directly to dst.
  $\mathsf{resp} \leftarrow$ response from dst.
  Output resp.via.        // May be $\perp$ if no via provides speedup.

• RACE.RESPOND(cell)
  Input: Data race cell from source relay (sender).
  if $\mathsf{cell.circ} \in \mathsf{Seen}$ then drop cell and return.
  Add cell.circ to Seen.
  if $\mathsf{cell.prev} = \mathsf{sender}$ then $\mathsf{via} \leftarrow \perp$
  else $\mathsf{via} \leftarrow \mathsf{sender}$.
  Send response resp to cell.prev with $\mathsf{resp.via} = \mathsf{via}$.

VIA RELAY:
// Via relays update their routing tables to forward traffic on a stream,
// provided sufficient resources are available to do so.
// State: Routes, the routing table for each circuit.

• RACE.VIAFORWARD(cell)
  Input: Data race cell cell with cell.cmd = RACE.
  if under heavy load then drop cell and return.
  Add cell.prev and cell.next to Routes[cell.circ].
  Forward the cell to relay cell.next.
```

disproportionately often, while a dishonest destination was *already* on the circuit and gains nothing by lying. The list of candidate vias is used to inform the data race which will select the fastest via from the list at the time of the race.

4) Data Race: Directly choosing via connections based on measured latencies has several potential drawbacks. First, the measured latencies are round-trip, while network paths are directional: the fastest path from relay A to relay B might be different from the fastest path from relay B to relay A. Timestamping at the destination halfway through the round trip *does not* address this issue, as it becomes impossible to distinguish between imperfect clock synchronization and path asymmetry. Second, latencies change in real-time in response to network conditions, like congestion at relays or on internet links. Third, latencies might be inaccurate due to measurement errors or even misreporting by malicious relays; we must take care to prevent such relays from seeing disproportionate amounts of traffic. As such, measured latencies alone are insufficient.

Instead circuit relays choose the fastest via using a "data race:" sending packets along different routes to see which arrives at the destination first (RACE.RUN(), Protocol 3). The starting relay *simultaneously* sends a copy of a data race cell to each prospective via relay, and one copy directly to the destination. The destination relay, which is the next hop on the circuit, responds *only* to the first of these cells to arrive.

Data races are *directional*—relays can identify the fastest path in each direction separately. Additionally malicious via relays cannot report lower latencies to artificially increase their odds of being selected. Data race cells are not forgeable by the via, so the via must wait to receive the cell from the source circuit relay before delivering it to the destination relay. Thus, the via cannot artificially reduce its perceived latency below the true time it takes to forward the cell.

5) Avoiding Traffic Loops: We define a *loop* to occur when the same traffic stream passes through a relay more than once. This is an issue as such relays could utilize traffic correlation to identify the previously seen traffic stream, thus learning a larger portion of its path through Tor than they should have been privy to. Tor only builds circuits using distinct, unrelated relays to ensure that circuits contain no loops. However, because ShorTor selects via relays separately from the circuit selection process, care must be taken to avoid loops.

In order to provide the same guarantee as Tor, we require that ShorTor is applied only to circuits of length exactly three (the default in Tor) and that only a single via is used between any pair of relays. This ensures that the middle relay of a circuit is capable of observing *all* vias on that circuit and enforcing the same guarantees as for circuit relays. That is, in either direction of a circuit, the middle relay will not choose to use a via that is already in use for the prior hop or is related to a relay on the circuit. We elaborate on security in Section V, but note here that a malicious middle relay gains no advantage by failing to enforce this guarantee, as it already knows the identities of both the guard and exit relays and does not need to correlate traffic across the via to get this information.

6) Stability: ShorTor's distributed via selection protocol must avoid *oscillations* where circuit relays swap back and forth between vias. As an example, without appropriate precautions, a cycle could form where traffic streams dropped from an overwhelmed via all divert to the same alternate via, subsequently overwhelming that via and causing the streams to revert to the original choice, and so on. We note that this situation is not prohibitive: CDNs use a similar overlay routing technique in practice, carrying substantial portions of internet traffic, without such stability problems.

We mitigate the risk of this situation in ShorTor through backoff and capacity parameters in the data race. We note that races are an integral component of the ShorTor protocol and are conducted to identify faster routes for traffic, not for stability purposes. These backoff and capacity parameters simply ensure that races avoid oscillations between vias.

Circuits never attempt to send traffic through a via without first conducting a race (though they may fall back to their direct path at any point). First, vias without available capacity will drop data race cells, preventing them from being selected at the small cost of processing a single packet. Second, upon being dropped from a via path, circuit relays will apply randomized exponential backoff and will not include that via in data races again until a set period of time has passed. The exact parameters for via capacity and backoff timing are network-dependent and may evolve based on the current state of the Tor network. However, because ShorTor is an optional performance-enhancement, these values can initially be set conservatively and then decreased adaptively.

C. Integration with Tor

While the ShorTor approach has potential applications to other networks, we designed and evaluated ShorTor's protocol to integrate with Tor specifically. Maintaining Tor's existing security guarantees is one main focus of this design and informed the structure of our data races and avoidance of loops. However, successful integration with Tor also requires that ShorTor be *deployable*. In this section, we discuss the components of ShorTor that are most relevant Tor deployment, including support for load balancing and fairness, required modification to Tor relays, and incremental deployment.

1) Load Balancing & Fairness: Fairness to circuit traffic and load balancing are both necessary to ensure that ShorTor does not inadvertently increase latency for some circuits as a consequence of reducing it on others. This could happen if via traffic was allowed to consume more resources than a relay had available to spare, resulting in increased processing times or congestion at the relay. ShorTor provides both fairness and load balancing through the same mechanism: prioritizing circuit traffic over via traffic. Tor already recognizes different traffic priorities—web browsing is prioritized over large file downloads [26]. We extend this to ensure that relays will preferentially schedule traffic from circuit queues over via queues (Figure 5).

Circuits will select vias that have lower latency than the default path, *including* the transit time through the via itself. This is very important as relay congestion and the associated queuing delays are a primary source of latency in Tor [43,44,46].

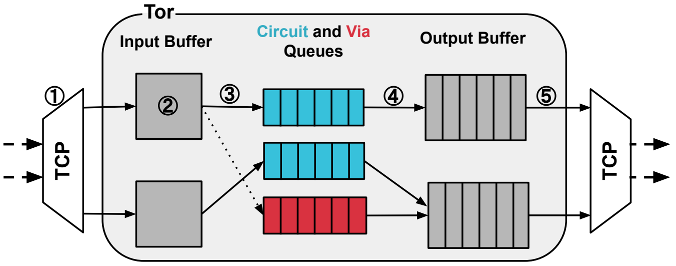

Fig. 5: Tor relay with circuit and via traffic (queuing architecture unmodified from baseline Tor). ① Via and circuit traffic are multiplexed on a TCP connection entering the Tor relay. ② The TLS layer is decrypted and circuit cells are onion encrypted/decrypted. ③ Circuit and via cells are sent to their individual queues. ④ Cells are scheduled for release to the output buffer based on priority order. ⑤ The contents of the output buffer are encrypted using TLS, then sent to the kernel for transit over a TCP connection to the next hop.

Congestion at a via will appear naturally during the data race in the form of increased latency or could be indicated explicitly by dropping race packets. In addition, vias are *transitory* and can be dropped or swapped at will with minimal cost compared to that of circuit construction/teardown.

As such, ShorTor ensures that: (1) circuit traffic on a relay is never delayed by via traffic and (2) load from via traffic is distributed only across relays with available capacity.

2) Tor Modifications: ShorTor's primary modification to Tor is the introduction of data races, all other components are simple extensions of Tor's existing mechanisms for routing and prioritizing circuits. To support ShorTor, Tor relays (though not clients) require additional protocol messages, a new data path for via traffic, and state for managing via traffic. The protocol requires new cell headers for data races, ping, and via traffic.

Specifically, via traffic needs a new priority level lower than circuit traffic (optionally, this level can be higher than that of bulk download traffic, such as torrenting, which is currently of lower priority than circuit traffic) [26]. Incoming via traffic needs a new data path that bypasses onion encryption and decryption. Relays must also handle ping and data race traffic as specified in Protocol 2 and Protocol 3. Finally, relays must hold two additional pieces of state: first, a new field in the routing table to indicate the via (if any) for each circuit; second, the list of candidate via nodes for each possible next hop (see Section III-B2 for details).

These modifications are relatively minor, do not touch Tor's onion encryption layer, and represent an optional overlay on Tor's routing. We discuss more details of required modifications in Appendix B, but note here that the high up-front cost of integrating and deploying modifications to the Tor protocol was a large factor in the ultimate design of ShorTor. This consideration motivated ShorTor's construction as an extension to Tor's existing architecture that operates largely separately from the baseline protocol.

Furthermore, ShorTor's modifications are configurable, trivially backwards compatible,[2] and support incremental deploy-

[2] Relays lacking support for ShorTor simply route as usual without any vias.

ment. This allows relay operators to choose whether to support ShorTor and how much capacity to dedicate to the protocol. As shown in Section IV-C3, ShorTor can substantially reduce tail latencies even with relatively low support. This is important, as it minimizes the risk of up front development efforts being wasted due to slow deployment.

3) Incremental Deployment: Tor relays are volunteer run and notoriously slow to update [58]. As such, any proposal that requires support of all—or even a majority of—Tor relays is unlikely to be effective. ShorTor is incrementally deployable and improves the latency of any Tor circuits that meet the following two requirements: (1) two adjacent circuit relays support ShorTor and (2) some other relay supporting ShorTor provides a faster path between the two circuit relays. Because Tor does not select its relays with uniform probability, a small set of popular relays could meet these conditions for many circuits without support from the rest of the network. We demonstrate this concretely in Section IV-C3.

Security is another important consideration—incremental deployment inherently creates differences between Tor clients or relays that have adopted a modification and those who have not. This has been an issue for client side proposals as anonymity in Tor relies on all clients behaving *uniformly* [12, 14,36,60,61,74,85].

ShorTor avoids this issue entirely as it is a fully server-side protocol that does not require participation from, or modify the behavior of, Tor clients in any way. So, while ShorTor is an observable modification to the Tor protocol,[3] it is in no way correlated to client identity. As such, support for ShorTor *cannot* be used to distinguish between clients. In fact, Tor clients should *not* try to preferentially select relays with support for ShorTor. While this would improve their latency, it would also differentiate them from Tor clients following Tor's baseline circuit selection algorithm, reducing their anonymity.

IV. Evaluation

We evaluate ShorTor using a dataset of approximately 400,000 latency measurements we collected from the live Tor network over the course of 42 days during summer 2021. Our measurements allow us to compare the direct latency between relays to the latency when routing through an intermediate hop, as in ShorTor.

Using measured latencies allows us to avoid relying on simulated or approximate data. While simulations can be a useful tool, prior work [75] has shown that routing protocols are best evaluated using live internet paths rather than through a simulation with, necessarily, reduced scale and complexity.

We evaluate the performance of ShorTor in terms of its direct impact on the latency between pairs of Tor relays as well as its ability to reduce the latency of Tor circuits. Evaluating on circuits as well as pairs allows us to account for the relative popularity of relays and more closely model the expected reduction in latency ShorTor can provide to Tor's end users.

A. Measurement Methodology

1) Ting: For our measurements, we adapt the Ting method of Cangialosi et al. [23] for estimating latencies between Tor relays. Ting creates a set of three circuits involving *observers*, which are Tor relays run solely for the purpose of obtaining latency measurements. Specifically, to obtain the latency between two Tor relays, Relay_A and Relay_B, we run two observer relays Obs_1 and Obs_2 along with a measurement client on the same physical machine. Once each circuit is established, the measurement client "pings" itself through the circuit to estimate round-trip latencies for the following circuits:

1) $\mathsf{rtt}_{AB} = \mathsf{RTT}\left(\mathsf{Obs}_1 \rightarrow \mathsf{Relay}_A \rightarrow \mathsf{Relay}_B \rightarrow \mathsf{Obs}_2\right)$
2) $\mathsf{rtt}_{A} = \mathsf{RTT}\left(\mathsf{Obs}_1 \rightarrow \mathsf{Relay}_A \rightarrow \mathsf{Obs}_2\right)$
3) $\mathsf{rtt}_{B} = \mathsf{RTT}\left(\mathsf{Obs}_1 \rightarrow \mathsf{Relay}_B \rightarrow \mathsf{Obs}_2\right)$

With these, we estimate the round-trip time between Relay_A and Relay_B as $\mathsf{rtt}_{AB} - \frac{1}{2}(\mathsf{rtt}_A + \mathsf{rtt}_B)$. This approximates the RTT including the forwarding delay at both Relay_A and Relay_B as forwarding is inherently a component of the latency experienced by via traffic. We repeat this process in order to find the minimum observed latency between Relay_A and Relay_B: in our observations, after 10 iterations, 95.5 % of circuits are within 5 % or 5 ms of the minimum observed in 100 samples.

2) Directional Latencies: The Ting protocol does not account for *directional* latencies where the outgoing latency between two nodes may not be equivalent to that of the return trip. Specifically, the method for computing rtt_{AB} described above assumes that latency $(\mathsf{Obs}_1 \rightarrow \mathsf{Relay}_X) \approx$ latency $(\mathsf{Relay}_X \rightarrow \mathsf{Obs}_2)$. To detect asymmetry in our RTT measurements we include a timestamp in our measurements halfway through the round-trip "ping" (all timestamps are with respect to the same clock). In our dataset (Section IV-A6), the median asymmetry was 2.4 % and only 0.2 % of measurements had an asymmetry of 2× or greater. Importantly, asymmetric RTTs impact **only** our evaluation as, when deployed, ShorTor naturally accounts for directional latencies using data races (Section III-B4).

3) Infrastructure: To collect latency measurements at scale, we adapted the Ting protocol to support parallel measurements across multiple machines. Our larger scale also required changes to respect a safe maximum load on the Tor network (see Section IV-A5): we impose a global maximum limit on concurrent measurements and spread measurements of individual relays across time. Our infrastructure also compensates for the high churn in the Tor network (13 % of relays we observed were online less than half the time) by enqueueing measurement jobs based on the currently online relays, with automated retries. We handled hardware and power failures using a fault-tolerant system design: we separated data persistence, measurement planning, and the measurements themselves.

We deployed to a private OpenStack [78] cloud, but provide a Terraform [41] template supporting any provider. Our open-source [1] measurement infrastructure is approximately 3,300 lines of code, consisting primarily of Python and shell scripts.

4) Geographic Location of Relays: We obtain country codes for the relays in our dataset using the GeoIP database [59],

[3] Both adversarial relays and network adversaries can likely detect when traffic is routed using ShorTor as opposed to baseline Tor.

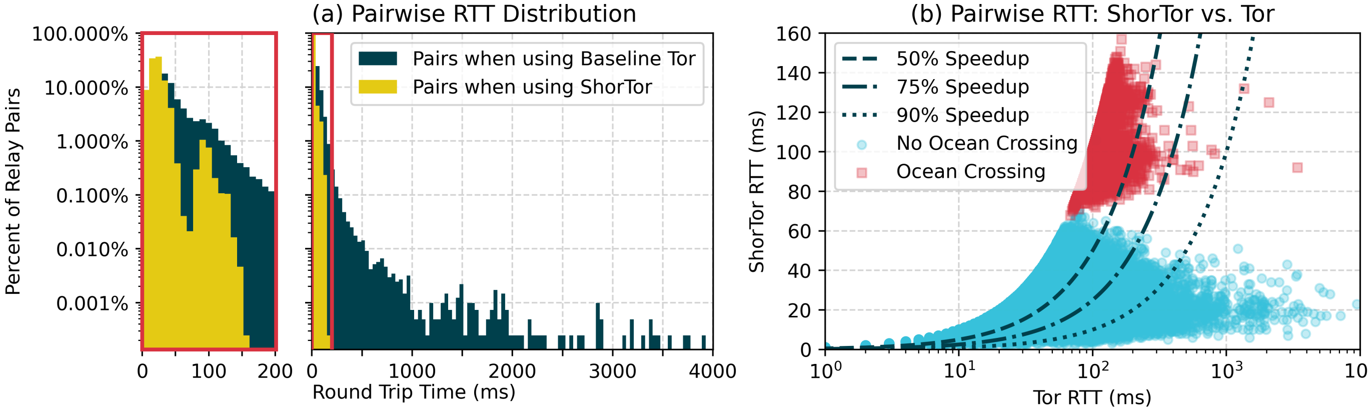

Fig. 6: *(a) Left:* Round-trip times (RTTs) measured between Tor relays vs. RTTs of those *same* pairs when using ShorTor as a percentage of all 406,074 pairs. We show an expanded view of the first 200 ms on the left. For readability, this figure omits 10 pairs with RTTs between 4,019 ms and 9,415 ms. *(b) Right:* Relative latency of each relay pair when routing via ShorTor vs. the default route. 25.4 % of pairs have a latency reduction of at least 50 %, 9.4 % at least 75 %, and 1.2 % at least 90 %.

which is also used by Tor in practice. However, GeoIP locations are not guaranteed to be 100 % accurate [37,51,66]. Indeed, upon careful inspection, we observed a handful of relay pairs with *physically impossible* RTTs for their purported locations. All of these pairs involved the same twelve relays allegedly located in the US. We determined that these twelve relays have *higher* average RTTs inside the Americas than they do to relays located in other regions. Because of this, all location-related figures (Figure 6(b), Figure 9, Figure 10) exclude these relays as, while we are confident that their reported location is incorrect, we cannot accurately determine their true location.

5) Ethics and Safety: We designed our measurement process to minimize impact on Tor users and relay operators and to comply with security best practices for Tor. To this end, we submitted a proposal to the Tor Research Safety Board [69] for review prior to measurement and adhered to their recommendations. We also received an IRB exemption from each author's institution for this work.

Collecting our data required us to run several live Tor relays. These relays recorded **only** our measurement traffic—at no point did we observe or record any information about any traffic from Tor users. We also minimized the likelihood of a user choosing our relays for their circuits by advertising the minimum allowed bandwidth of 80 KiB/s [26].

Our measurement collection was spread over 42 days to reduce concurrent load, including a limit on simultaneous measurements (detailed in Section IV-A3). We also notified a Tor relay operator mailing list and allowed operators to opt out of our measurements; we excluded four such relays.

In light of recent work by Schnitzler et al. [76] on the security implications of fine-grained latency measurements for Tor circuits, we have not published our full latency dataset. However, we will share this data with researchers upon request and are in communication with our reviewers from the Tor Research Safety Board about safely releasing it in the future.

6) Latency Dataset: In this work, we directly measure pairwise latencies within Tor rather than relying on outside estimates. We focus our measurements on the 1,000 most popular Tor relays (by consensus weight) for two main reasons:

1) Measuring all 36,325,026 possible pairs of the 8,524 Tor relays we observed was intractable for this work.
2) These popular relays are present on over 75 % of circuits [39] and thus can provide disproportionate utility.

Our dataset contains 406,074 measured latencies or 81.3 % of all pairs of the 1,000 most popular relays.[4]

B. Applying ShorTor to Relay Pairs

We begin evaluating ShorTor by comparing the potential latency between our set of relay pairs when routing via ShorTor to our measured latencies observed using Tor's default routing. Figure 6(a) shows the relative frequency of RTTs experienced by pairs of Tor relays using ShorTor and when routing normally while Figure 6(b) focuses on the relationship between default RTT and ShorTor RTT for each relay pair. Using ShorTor, all of Tor's high tail latencies were resolved: ShorTor sees a maximum absolute RTT of 157 ms, while 0.09 % of pairs in Tor had RTTs of over half a second. In other words, the 99.9th percentile of relay pairs see a reduction in RTT from 487 ms in Tor to 125 ms in ShorTor. Additionally, 25.4 % of relay pairs cut their RTT in half (or more) using ShorTor.

Figure 6(b) also shows that ShorTor's RTT values largely correspond the physical distance between the endpoints: relay pairs that are across an ocean necessarily experience a higher latency than those in the same region.

C. ShorTor Circuits

We model the expected reduction in latency for end users by applying ShorTor to Tor circuits. Due to Tor's non-uniform relay selection probabilities, our pairwise latency dataset does not directly account for how *probable* any of the observed RTTs are. As such, we include an evaluation of ShorTor on two million Tor circuits built according to Tor's default parameters. Because

[4] Missing measurements are largely due to churn in the Tor network causing relay pairs to not be live simultaneously.

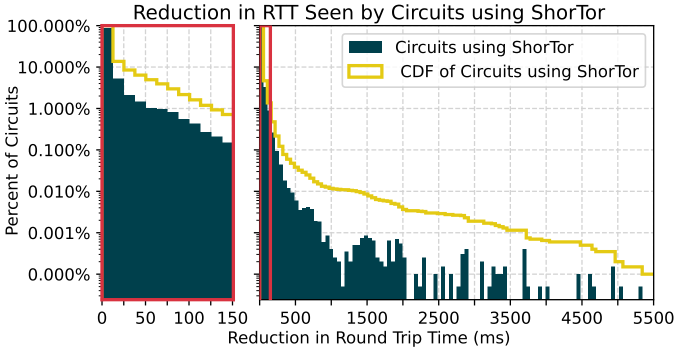

Fig. 7: Reduction in Round Trip Time experienced by 2M Tor circuits when routing with ShorTor. We show an expanded view of the first 150 ms on the left. The CDF line is reversed: for instance, 0.01 % of circuits see a speedup of at least 1,000 ms.

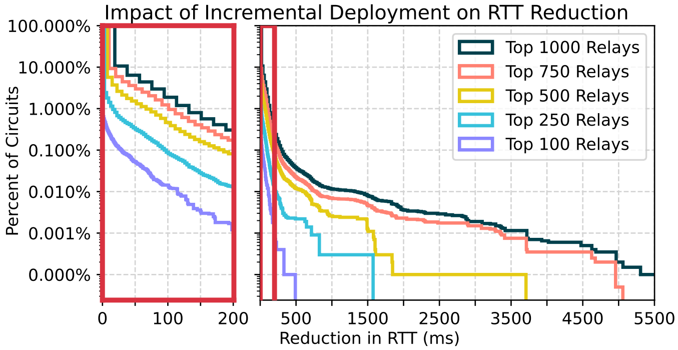

Fig. 8: Reversed CDF for incremental deployment speedups.

Tor averages 2M daily users, this roughly approximates the expected distribution of circuits over one day of use.

1) Circuit Selection: Using the Tor path selection simulator (TorPS) [47], we choose two million circuits over *all* 8,524 relays we observed. Because we collect latency measurements from only the 1,000 largest Tor relays by consensus weight, many of these circuits have incomplete latency data. We select circuits from the full set of relays, despite incomplete latency data, to ensure that our distribution of circuits closely resembles that of real Tor users. We handle the gaps in our data by reporting on the *reduction* in latency provided by ShorTor rather than absolute RTTs. All circuit legs with missing measurements are reported as a speedup of 0 ms (equivalent to baseline Tor). The speedups we observe can thus be thought of as the *minimum* that our set of simulated circuits would experience using ShorTor.

2) Latency of ShorTor Circuits: In this section, we evaluate the performance of ShorTor on our set of 2M circuits. We only indicate a speedup for a circuit if: (1) it contains two adjacent relays that are present in both our measurement dataset and in the set of relays supporting ShorTor and (2) some other relay that supports ShorTor can provide a faster route between the circuit relays. 68.0 % of circuits have no available measurement data for either leg and are shown with the default speedup of 0 ms. Of the 32.0 % of circuits with a latency measurement for at least one leg, 83.7 % see a speedup with ShorTor.

As shown in Figure 7, 1 % of the 2M circuits see a latency improvement of 122 ms or greater and 0.012 % of circuits saw a speedup of over a second. For details on the relationship between RTTs and the page load times experienced by users, see Section IV-E

3) Incremental Deployment: As previously described in Section III-C3, ShorTor is designed to function at relatively low levels of deployment. Our previous evaluation (Figure 7) assumed that all 1,000 of the relays we measured supported ShorTor. In Figure 8, we show that ShorTor is also capable of reducing latency for Tor circuits even at substantially lower levels of deployment. As before, we only apply ShorTor when all relays involved support the protocol and assume that all unmeasured pairs of relays have no speedup. We find that circuits at the 99.9th percentile see latency reductions of 178 ms even when only the top 500 relays support ShorTor.

D. Cost of ShorTor

ShorTor's primary cost, aside from the one-time startup cost of modifying the Tor protocol, is in terms of bandwidth overhead from its longer paths. In the steady state, Tor circuits will use extra bandwidth for each hop using a via relay: an overhead of 1/3 above the original traffic. If we do this for *every* hop with *any* speedup, no matter how small, this uses 10.9% extra bandwidth over Tor right now. However, if we only use ShorTor when it offers a speedup above a certain cutoff, this overhead quickly declines:

Cutoff	**0 ms**	**10 ms**	**25 ms**	**50 ms**	**100 ms**
Overhead	10.9%	6.6%	3.8%	2.3%	0.8%

Further, relays will only carry via traffic when they have excess capacity. Tor reports consistently under 50% bandwidth utilization across the network [67].

A more minor source of bandwidth overhead is control information around routing tables and data races. First, relays must keep their latency tables up to date following LATENCIES.UPDATE() (Protocol 2 in Section III-B3). This requires each relay send its latency table to every other relay once per day. Using 16 bits per RTT estimate, each relay must send about 100 MB per day[5] in total—about 0.05% of the minimum recommended relay bandwidth [68]. Second, they must establish via connections using RACE.RUN() (Protocol 3 in Section III-B4), which sends a small, configurable number of extra packets (e.g., 5), equivalent to about 2.5 kB[6] of extra data. Data races have tunable frequency and will only occur if latency estimates indicate a speedup above the cutoff. Assuming two million circuits per day with each circuit participating in a data race at both hops, data races will consume 10 GB over the course of the day across the entire network. Currently, Tor advertises bandwidth of 600 GB/s and consumes less than 300 GB/s [58].

[5] Assuming 7000 concurrently active relays in the Tor network: 16 bits per RTT × 7000 relays per table × 7000 relays sending their table each day.

[6] Tor cells are ≈ 0.5 kB each.

Thus, ShorTor's bandwidth overhead is dominated by the longer path lengths and is parameterizable based on cutoffs for latency reduction as shown above. This overhead may *not* be distributed evenly among relays, but we note that participation in ShorTor is fully optional (see discussion of incremental deployment in Section III-C3), so resource constrained relays may simply choose not to participate at any stage of ShorTor or decide not to support the protocol entirely if overhead is a concern.

E. Impact of ShorTor on User Experience

Perceived latency in the form of page load times (PLT) has a demonstrable impact on users in anonymity systems. In qualitative user experience research, Tor users specifically cite latency as an issue keeping them from adopting Tor [33–35]. Köpsell [52] finds a linear relationship between latency and number of users for an anonymous communication system.

In many cases, latency (not bandwidth) is the limiting factor for page loads: increases in RTTs cause linear increases to PLT, often with a 10× multiplier. Netravali et al. [63] find that increasing RTT from 25 ms to 50 ms increases 95th-percentile PLT across 350 popular sites from 1.5 s to 3.4 s, and increasing to 100 ms raises the PLT to 6.1 s. Many factors contribute to this multiplier, including TCP congestion control, TLS handshakes, and complex web sites where an initial page fetch may spawn additional requests [63,70,71].

To bridge the gap between our RTT-based evaluation of ShorTor and the more intuitive usability metric of PLTs, we simulate the impact of network delays on page load times over Tor, finding that small increases in delays lead to large increases in page load times. First, we measure the time it takes to load the New York Times and Google homepages over ten Tor circuits, chosen by the Tor path selection algorithm, without modification or delay. We then model changes in RTT, such as those from a link delay, by using the Linux `tc` utility to introduce an artificial delay for each packet sent over the same ten circuits.

To evaluate ShorTor, we selected delays that correspond to speedups seen by ShorTor circuits in Section IV-C to obtain an estimate for the potential difference in page load times experienced by end users. Of the circuits in our measurement dataset, 5.04 % experience a speedup of at least 50 ms, 1.66 % of at least 100 ms, and 0.04 % of at least 500 ms.

We report the median change in PLT for fetching `google.com` and `nytimes.com` over these ten circuits when traffic is delayed by 50 ms, 100 ms, and 500 ms:

		Network Delay		
Website		**50 ms**	**100 ms**	**500 ms**
google.com	ΔPLT:	0.98 s	1.96 s	10.40 s
nytimes.com	ΔPLT:	1.66 s	2.34 s	15.80 s

We find that Tor follows the trend seen in prior work with even 50 ms changes in RTT increasing PLTs by approximately a second. In the context of ShorTor, 1 %, or 20k, of the 2M circuits we evaluated saw a reduction in their RTT of at least 120 ms which corresponds to an expected two second drop in PLT. As Tor sees approximately two million daily users, each building at least one circuit, ShorTor's impact on tail latencies is likely to improve the experience of tens of thousands of Tor users daily.

V. Security Analysis

In this section, we analyze the security of ShorTor. We consider how ShorTor's use of via relays might impact an adversary's ability to deanonymize Tor traffic in practice. To do so, we examine the change in the adversary's view of the Tor network when using ShorTor as compared to the baseline Tor protocol. While via relays never observe the sender or recipient of Tor traffic *directly*, they are able to observe traffic streams and other relays on the circuit, which could *indirectly* deanonymize the sender or recipient. For this purpose, we use the AnoA [12,60] framework for analyzing the anonymity guarantees of anonymous communication protocols to help us determine the potential anonymity impact of vias in ShorTor.

A. AnoA Analysis

Backes et al. [13, 14] apply the AnoA framework to analyze the anonymity of the Tor network and the impact proposed protocol modifications might have on Tor's anonymity. AnoA uses ideas from differential privacy [29] to determine an adversary's advantage in a challenge-response game, which models the ability to distinguish between traffic streams. In this game, the adversary statically corrupts a set of "traffic observation points" (i.e., Tor relays) and attempts to distinguish between two possible scenarios involving different senders and recipients for each traffic stream. The adversary's ability to distinguish between these scenarios models the overall anonymity of the Tor network.

Definition V.1 (Anonymity Notions [12]; simplified)**.** Let $\mathcal{A}$ be a passive network adversary consisting of a set of corrupted relays and capable of observing a subset of network traffic through these relays. The anonymity notions are:

Sender anonymity: the probability that $\mathcal{A}$ can distinguish between two potential senders of a given traffic stream.

Recipient anonymity: the probability that $\mathcal{A}$ can distinguish between two potential recipients of a given traffic stream.

Relationship anonymity: the probability that $\mathcal{A}$ is capable of determining which sender is communicating with which recipient. The anonymity game is defined for all pairs of senders (Alice and Bob) and recipients (Charlie and Diana); $\mathcal{A}$ wins by successfully linking a traffic stream to the correct communicating pair.

Tor relays are not all created equal: stable, high bandwidth relays see a larger fraction of Tor's traffic, but are also more costly. To accurately analyze an adversary's impact on anonymity, it is necessary to decide *which* subset of relays are most beneficial to corrupt. Backes et al. [13, 14] develop MATor [19] to model different adversarial corruption strategies. Following Backes et al. [13, 14], we consider four adversarial corruption strategies: k-collusion, bandwidth, monetary, and geographic.

With the k-collusion strategy, the adversary corrupts k relays that provide it with the most advantageous view of Tor's network. With the other three strategies, the adversary has a similarly fixed "budget" (e.g., cumulative bandwidth) which constrains the optimal set of relays to corrupt. The MATOR tool [19] optimizes each adversarial strategy based on the allocated budget and anonymity notion to empirically compute the worst-case anonymity bound under AnoA.

B. Differential Advantage

We define how we measure the theoretical impact of ShorTor on anonymity in Tor in terms of the difference in an adversary's advantage in ShorTor vs. baseline Tor.

Notation: We write $\mathcal{N}$ for the set of all Tor relays and $\mathcal{C}$ for the set of all Tor circuits (consisting of three independent relays). We denote the path selection and via selection algorithms by PS, and VS, respectively. We note that VS is unique to ShorTor and is separate from the path selection algorithm PS used in Tor. We use $\perp$ for a "null" element.

Definition V.2 (Via Relay)**.** Let $\mathsf{C} \in \mathcal{N} \times \mathcal{N} \times \mathcal{N}$ be a Tor circuit consisting of three circuit relays. A *via relay* $v \in \mathcal{N}$ is a Tor relay routing packets between a pair of consecutive circuit relays in C.

Remark 1. A via relay is semantically equivalent to a *wire* connecting two consecutive Tor relays in the circuit. Via relays only forward traffic and are *not* involved in circuit establishment or any of Tor's onion-encryption operations.

We first define *adversary observations* on the network. We then use this to define the *differential advantage*—the impact that ShorTor introduces relative to baseline Tor.

Definition V.3 (Adversary Observations)**.** Let $\mathcal{V} \subseteq \mathcal{N}$ be the set of candidate via relays and let $\mathcal{C}$ be the set of all three-relay circuits. Fix a set $\mathcal{N}^* \subseteq \mathcal{N}$ of adversary-corrupted relays. We define the function: $\mathsf{Obs}_{\mathcal{N}^*} : \mathcal{C} \times \mathcal{V} \cup \{\perp\} \times \mathcal{V} \cup \{\perp\} \to \mathsf{O}$, which takes as input a circuit and a pair of via relays (possibly $\perp$), and outputs the set of observation points (adversary-corrupted circuit and via relays).

Definition V.3 captures the "view" of the adversary for a given circuit. For example, an adversary corrupting the middle relay on a single circuit sees the guard and exit relays on the circuit, but not the sender or recipient.

Definition V.4 (Differential Advantage)**.** Let $\mathcal{V} \subseteq \mathcal{N}$ be the set of candidate via relays. Let PS be a randomized path selection algorithm and VS be a via relay selection algorithm for a circuit. Fix $\mathcal{N}^* \subseteq \mathcal{N}$ to be the set of adversary-corrupted relays and let $\mathcal{C}$ be a set of all three-relay circuits output by PS. For a circuit $\mathsf{C} \in \mathcal{C}$ and $(v_1, v_2) \in \mathsf{supp}_{\mathsf{C} \in \mathcal{C}} |\mathsf{VS}(\mathsf{C})|$, the adversary is said to have a *differential advantage* when for

$$\mathsf{O}_{\text{tor}} \leftarrow \mathsf{Obs}_{\mathcal{N}^*}(\mathsf{C}, \perp, \perp) \text{ and } \mathsf{O}_{\text{via}} \leftarrow \mathsf{Obs}_{\mathcal{N}^*}(\mathsf{C}, v_1, v_2),$$

the set $\mathsf{O}_{\text{tor}} \subset \mathsf{O}_{\text{via}}$, where Obs is as defined in Definition V.3.

In words, an adversary has a differential advantage in ShorTor when *new* observations are gained as a result of introducing via relays. We now examine scenarios in which the adversary *does* gain differential advantage by corrupting via relays. We formalize these scenarios in Lemma 1.

Lemma 1. *Let $\mathcal{V} \subseteq \mathcal{N}$ be the set of candidate via relays. Fix $\mathcal{N}^* \subseteq \mathcal{N}$, the set of adversary-corrupted relays. For all sets of observations O_{tor} and O_{via} for a circuit $\mathsf{C} \in \mathcal{C}$, as in Definition V.4, $\mathsf{O}_{\text{tor}} \subset \mathsf{O}_{\text{via}}$ if and only if there exists at least one via relay in $\mathcal{N}^*$ between two consecutive non-corrupted relays in C.*

Proof. Let $r_a, r_b \in \mathsf{C}$ be any two consecutive circuit relays in C (either $\{\text{guard}, \text{middle}\}$ or $\{\text{middle}, \text{exit}\}$) with via relay v connecting r_a and r_b. Corrupting either r_a or r_b provides the adversary with a view of the wire, which is equivalent to the view obtained from corrupting the via (see Remark 1). For any circuit $\mathsf{C} \in \mathcal{C}$, the set of observation points gained from corrupting v is a strict subset of the set of observation points gained from corrupting either r_a or r_b individually. Therefore, we have that the adversary only obtains an additional observation ($\mathsf{O}_{\text{tor}} \subset \mathsf{O}_{\text{via}}$) if r_a and r_b are *not* corrupted while the via relay v *is* corrupted. ■

Claim 1. An adversary-corrupted via relay observes strictly less than an adversary-corrupted circuit middle relay in Tor.

Proof. By Lemma 1, we have that the adversarial advantage from corrupting a via relay is strictly less than corrupting any middle relay. Via relays are positioned either between the guard and middle relays or middle and exit relays. As such, corrupting a middle relay in a circuit tightly upper bounds the observation points gained from corrupting both vias. ■

In Claim 2, we argue that ShorTor does not advantage the adversary in any of the anonymity notions of Definition V.1 (we empirically confirm this result in Section V-C).

Claim 2. ShorTor applied to the baseline Tor network with path selection algorithm $\mathsf{PS} : \mathcal{N} \to \mathcal{C}$ which outputs circuits independently of the sender and recipient (as is currently the case in Tor [26,27]), does not impact the anonymity notions of Definition V.1 of the AnoA framework.

Proof. Under the AnoA framework, corrupting the middle relay does not change the adversary's ability to deanonymize either sender, recipient, or relationship anonymity when the circuit is constructed *independently* of the sender and recipient (see analysis of Backes et al. [14]). This is because each middle relay is equally probable in all communication scenarios, giving the adversary no advantage [14]. As a consequence, by leveraging Claim 1, corrupting one or both via relays when ShorTor is applied to Tor does not advantage the adversary in the AnoA anonymity game of Definition V.1. ■

Claim 2 shows that ShorTor does not impact anonymity of Tor. However, when the middle relay is *not* chosen independently of the sender or recipient (for example, when using location-aware path selection proposals; see Section VI), then ShorTor can exacerbate the negative impact on anonymity.

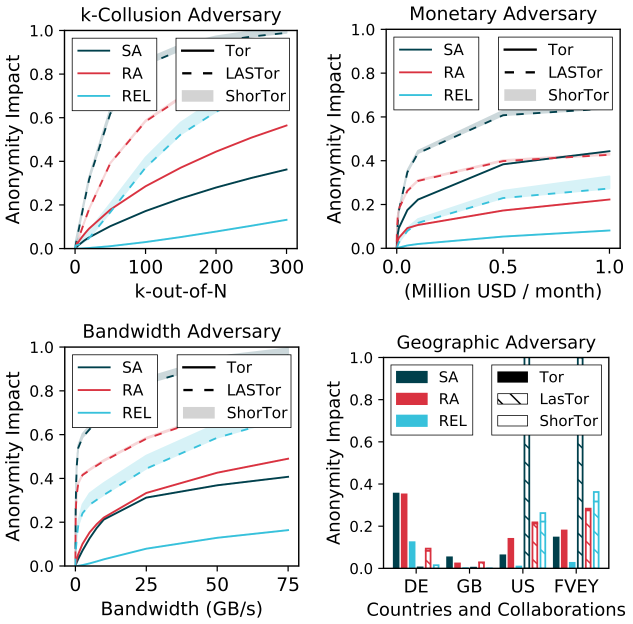

Fig. 9: Anonymity impact of ShorTor, compared with baseline Tor (client-independent) and LASTor (dependent on client location) path selection. Each plot shows sender (SA), recipient (RA), and relationship (REL) anonymity (Definition V.1) for a different adversarial relay corruption strategy. Shaded region represents the difference in the MATOR-computed anonymity bounds with and without ShorTor. "FVEY" refers to "Five Eyes" intelligence alliance member countries. Under ANoA, ShorTor affects all anonymity notions for LASTor, though not baseline Tor. Extended plots provided in Appendix A.

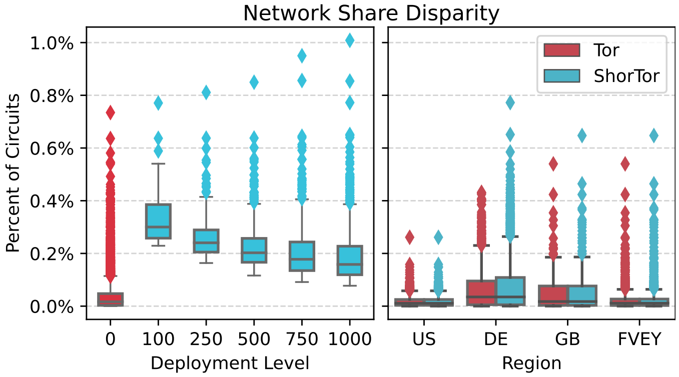

Fig. 10: Fraction of circuits seen by relays in Tor vs. ShorTor. Left side considers incremental deployment (Section IV-C3). As more relays begin to support ShorTor (x-axis), the average network share goes down, though some outliers see more traffic. Right side looks at regional network share, or the fraction of circuits seen by relays in different regions.

We quantify this advantage using MATOR by applying ShorTor to LASTor [5], a location-biased path-selection proposal. We emphasize that LASTor is *not* integrated in Tor and has known security flaws [86]—we include it as an illustrative example of a location-aware path selection scheme.

C. Quantifying Anonymity of ShorTor

We now turn to empirically computing the *worst-case* anonymity impact under the AnoA framework when ShorTor is applied to Tor and proposed modifications thereof. We modify MATOR (our code is open-source [1]; written in C++ and Python) to incorporate the use of via relays as described in Section III-B. We report our quantitative results in Fig. 9.

ShorTor applied to Tor: We confirm the results of Claim 2 on the adversarial impact of ShorTor used with baseline Tor: the worst-case anonymity bounds computed by MATOR for baseline Tor and ShorTor are equal, as relays are selected independently of both the sender and recipient.

ShorTor Applied to LASTor: We examine the impact of ShorTor when combined with *biased* path selection algorithms (e.g., path selection that takes client location into account). We use the LASTor [5] proposal for this purpose. We find that ShorTor applied to LASTor decreases anonymity under all three anonymity notions of Definition V.1, as via relays offer additional observations points for the skewed distribution of guards and exits used by LASTor.

D. Network Share and Traffic Analysis Attacks

A limitation of AnoA is that it does not take into account traffic analysis attacks that can be conducted by a single point of observation [60], such as a via relay. Most notably, this includes website fingerprinting attacks [20,45,72,80,88].

To analyze the adversarial advantage in orchestrating such attacks in ShorTor, we consider the relative *network share* disparity between baseline Tor and ShorTor. By this we mean the relative fraction of circuits seen by a relay when acting as part of a circuit vs. as a via. Regional network share is a concern for Tor users primarily due to varying policies on surveillance in different jurisdictions [48]. Separately, determining network share of individual relays at different levels of deployment allows us to assess the potential security impact of incrementally deploying ShorTor.

In Fig. 10, we plot the network share using the same circuits as in Section IV-C2. We vary the deployment level of ShorTor to measure the expected change in network share as a function of relays supporting ShorTor (Section IV-C3). We find that ShorTor increases the median network share (as traffic traverses more nodes when using ShorTor). However, median network share decreases with larger deployments. The worst-case network share increases from about 0.4 % to about 0.8 % for relays located in Germany (which is by far the country with the most Tor relays). It is important to note that the overall network shares remain low, indicating a small disparity in expected network traffic observed.

Our analysis does not take into account *adversarial* placement of relays with fast network connections to boost their via selection probability. However, this is not unique to ShorTor: relays in Tor already have a high disparity in their network share based on *bandwidth* influencing their circuit selection probability. As such, we believe that the impact of ShorTor on traffic analysis attacks is modest and in-line with Tor's existing assumptions about adversarial placement of relays.

VI. Related Work

In this section, we outline past works that focus on reducing latency in Tor though optimized routing decisions. We note that all works here operate at the *circuit* layer and are proposed modifications to Tor's circuit selection protocol. There is additionally a large body of work that alters path selection in Tor for purposes of *security* [13,15,30,40,48,64,73,82,89]. While important, these works are orthogonal to ShorTor and often result in substantially degraded performance [61,74] without clear security advantages over Tor's current protocol [36,85,86].

Traffic Splitting: Rather than selecting a *single* faster circuit, Karaoglu et al. [49] and AlSabah et al. [7] split traffic across multiple circuits. This distributes the load of the circuit across a larger number of relays, improving latency by reducing congestion on relays in the circuit. Conflux [7] can achieve an average reduction in time-to-first-byte of 23 % over baseline Tor. Splitting traffic across multiple circuits solves an orthogonal problem to that addressed in ShorTor and combining both protocols could be an interesting future direction.

Location-Aware Path Selection: Alternative path selection proposals that reduce latency on Tor share a common theme: they all, either directly or indirectly, account for the location of the client or destination server when choosing a circuit [5,9,16,42,74,79,87]. This makes intuitive sense, as fast paths are likely to also be geographically short and, in particular, are unlikely to contain multiple ocean crossings.

Imani et al. [42] propose to improve performance of Tor circuits by having clients build multiple circuits, then preferentially select from those according to a series of strategies focusing on circuit length, RTT, and congestion. In addition to latency, Sherr et al. [79] include measurements of jitter and packet loss when selecting relays.

Wang et al. [87] opportunistically selects relays with low latency to construct circuits that avoid congested relays. NavigaTor [9] applies a similar strategy to Wang et al. [87] and demonstrates improved performance by using latency (specifically round-trip time) to discard slow Tor circuits. PredicTor [16] avoids the overhead of constructing then discarding multiple circuits by using a random forest classifier trained on Tor performance data to *predict* the performance of a circuit prior to building it. CLAPS (CLient Aware Path Selection) [74] solves a weight-optimization problem to ensure a strict bound on anonymity degradation when selecting location-biased circuits.

There additionally exists a large body of work on the security implications of location awareness in path selection as this can make Tor clients more identifiable by creating a correlation between their geographic location and the relays chosen for their circuit [12,14,36,60,61,74,85,86]. As shown in Section V, ShorTor does not share this problem and is able to reduce latency for Tor clients without the security pitfalls of location-aware circuit selection techniques.

VII. Discussion

Other sources of delay in Tor: The type of overhead that ShorTor addresses is not the largest source of delay in Tor. Limited congestion control [6,32], under-optimized multiplexing of circuits on TCP connections [17,18], and high queuing delays [43,44,46] are likely larger contributors to latency in Tor than suboptimal BGP routes. Despite this, we believe ShorTor to be of interest. The source of delays addressed by ShorTor and the techniques applied are both completely independent of other delays in Tor. Because of this, the decrease in latency provided by ShorTor will trivially *stack* with any future improvements to congestion control, circuit multiplexing, or queueing. As such, we believe ShorTor to be a valuable contribution to improving the latency of Tor connections.

Compatibility with security-focused path selection: ShorTor is also fully compatible with any modifications to Tor's path selection algorithm. Prior work has shown that existing proposals, overviewed briefly in Section VI, suffer from poor load balancing and non-uniform client behavior, hurting performance and client anonymity [12,14,36,60,61,74,85,86]. However, this does not preclude some *future* path selection proposal from improving upon Tor's current algorithm. In this case, ShorTor is again agnostic to the choice of the path selection algorithm and would require no modification to continue improving latency on top of the new algorithm.

Generality: While we apply multi-hop overlay routing to Tor specifically, we note that it is a general purpose technique. Evaluating its effectiveness for other relatively small scale, distributed communication networks is an interesting direction for future work. However, as shown by prior work [75] and confirmed here for Tor, accurate evaluation of multi-hop overlay routing cannot be done with general purpose latency data and requires measurements from the specific network involved.

VIII. Conclusion

We presented ShorTor, an incrementally-deployable protocol for improving the latency of Tor's connections. We evaluated the performance and security of ShorTor, demonstrating that it provides substantial improvements to tail latencies on Tor circuits, with minimal impact to security. As part of our evaluation we collected a dataset of pairwise latencies between the thousand most popular Tor relays. This dataset allowed us to determine the reduction in latency ShorTor provides to Tor circuits *directly* without relying on simulation or approximated data. Finally, while we proposed and evaluated ShorTor specifically for Tor, the protocol is general and has foreseeable applications to other distributed communication networks.

IX. Acknowledgements

We would like to thank Arthur Berger for invaluable suggestions and discussion surrounding internet routing, Frank Cangialosi for conversations about Ting and tips on measuring the Tor network, and Anish Athalye for his ideas around evaluating page load times and feedback on our security analysis. We are also grateful to the Tor community for their support of this project and, in particular, Roger Dingledine, Georg Koppen, and Tariq Elahi for extensive advice related to running our experiments.

References

[1] Source code for ShorTor measurement infrastructure and MaTor security analysis. https://github.com/sachaservan/ShorTor.

[2] RIPE Atlas. https://atlas.ripe.net/, 2021. Accessed December 2021.

[3] Akamai. Content delivery networks — what is a CDN? https://www.akamai.com/our-thinking/cdn/what-is-a-cdn, 2021. Accessed December 2021.

[4] Akamai. SureRoute. https://developer.akamai.com/article/sureroute, 2021. Accessed December 2021.

[5] Masoud Akhoondi, Curtis Yu, and Harsha V Madhyastha. LASTor: A low-latency AS-aware Tor client. In *2012 IEEE Symposium on Security and Privacy*, pages 476–490. IEEE, 2012.

[6] Mashael AlSabah, Kevin Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. DefenestraTor: Throwing out windows in Tor. In Simone Fischer-Hübner and Nicholas Hopper, editors, *Privacy Enhancing Technologies*, pages 134–154, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22263-4.

[7] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. The path less travelled: Overcoming Tor's bottlenecks with traffic splitting. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, 2013.

[8] Sebastian Angel and Srinath Setty. Unobservable communication over fully untrusted infrastructure. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 551–569, 2016.

[9] R. Annessi and M. Schmiedecker. NavigaTor: Finding faster paths to anonymity. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 214–226, 2016.

[10] Ioannis Arapakis, Xiao Bai, and B. Barla Cambazoglu. Impact of response latency on user behavior in web search. In *Proceedings of the 37th International ACM SIGIR conference on Research & Development in Information Retrieval*, pages 103–112, 2014.

[11] Christiane Attig, Nadine Rauh, Thomas Franke, and Josef F. Krems. System latency guidelines then and now - is zero latency really considered necessary? In Don Harris, editor, *Engineering Psychology and Cognitive Ergonomics: Cognition and Design - 14th International Conference, EPCE 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part II*, volume 10276 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2017. doi: 10.1007/978-3-319-58475-1_1. URL https://doi.org/10.1007/978-3-319-58475-1_1.

[12] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. In *2013 IEEE 26th Computer Security Foundations Symposium*, pages 163–178. IEEE, 2013.

[13] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (Nothing else) MATor(s) monitoring the anonymity of Tor's path selection. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 513–524, 2014.

[14] Michael Backes, Sebastian Meiser, and Marcin Slowik. Your choice MATor(s): Large-scale quantitative anonymity assessment of Tor path selection algorithms against structural attacks. *Proceedings on Privacy Enhancing Technologies*, 2016(2):40–60, 2016.

[15] Armon Barton and Matthew Wright. DeNASA: Destination-naive AS-awareness in anonymous communications. *Proceedings on Privacy Enhancing Technologies*, 2016(4):356–372, 2016.

[16] Armon Barton, Matthew Wright, Jiang Ming, and Mohsen Imani. Towards predicting efficient and anonymous Tor circuits. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 429–444, 2018.

[17] Lamiaa Basyoni, Aiman Erbad, Mashael Alsabah, Noora Fetais, and Mohsen Guizani. Empirical performance evaluation of QUIC protocol for Tor anonymity network. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 635–642, 2019. doi: 10.1109/IWCMC.2019.8766609.

[18] Lamiaa Basyoni, Aiman Erbad, Mashael Alsabah, Noora Fetais, Amr Mohamed, and Mohsen Guizani. QuicTor: Enhancing Tor for real-time communication using QUIC transport protocol. *IEEE Access*, 9:28769–28784, 2021. doi: 10.1109/ACCESS.2021.3059672.

[19] Markus Bauer. *The MATor Addon: Accessible Quantification of Tor Anonymity for the Tor Browser*. PhD thesis, Saarland University, 2015.

[20] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-CNN: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(4):292–310, Oct 2019. ISSN 2299-0984. doi: 10.2478/popets-2019-0070. URL http://dx.doi.org/10.2478/popets-2019-0070.

[21] The Privacy Blog. Why Tor failed to hide the bomb hoaxer at Harvard. https://theprivacyblog.com/blog/anonymity/why-tor-failed-to-hide-the-bomb-hoaxer-at-harvard, 2013. Accessed December 2021.

[22] Claudson F Bornstein, Timothy K Canfield, Gary L Miller, Satish B Rao, and Ravi Sundaram. Optimal route selection in a content delivery network, July 2 2013. US Patent 8,477,630.

[23] Frank Cangialosi, Dave Levin, and Neil Spring. Ting: Measuring and exploiting latencies between all Tor nodes. In *Proceedings of the 2015 Internet Measurement Conference*, pages 289–302, 2015.

[24] Anita Crescenzi, Diane Kelly, and Leif Azzopardi. Impacts of time constraints and system delays on user experience. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 141–150, 2016.

[25] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *WEIS*, 2006.

[26] Roger Dingledine and Nick Mathewson. Tor protocol specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt, 2008.

[27] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association. URL https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[28] Nick Duffield, Kartik Gopalan, Michael R Hines, Aman Shaikh, and Jacobus E Van Der Merwe. Measurement informed route selection. In *International Conference on Passive and Active Network Measurement*, pages 250–254. Springer, 2007.

[29] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[30] Matthew Edman and Paul Syverson. AS-awareness in Tor path selection. In *Proceedings of the 16th ACM conference on Computer and Communications security*, pages 380–389, 2009.

[31] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1775–1792. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/eskandarian.

[32] Felix Fiedler, Christoph Döpmann, Florian Tschorsch, and Sergio Lucia. Predictor: Predictive congestion control for the tor network. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 863–870, 2020. doi: 10.1109/CCTA41146.2020.9206384.

[33] Kevin Gallagher. *Measurement and Improvement of the Tor User Experience*. PhD thesis, New York University Tandon School of Engineering, 2020. URL https://www.proquest.com/docview/2370441192.

[34] Kevin Gallagher, Sameer Patil, and Nasir D. Memon. New me: Understanding expert and non-expert perceptions and usage of the tor anonymity network. In *Thirteenth Symposium on Usable Privacy and Security, SOUPS 2017, Santa Clara, CA, USA, July 12-14, 2017*, pages 385–398. USENIX Association, 2017. URL https://www.usenix.org/conference/soups2017/technical-sessions/presentation/gallagher.

[35] Kevin Gallagher, Sameer Patil, Brendan Dolan-Gavitt, Damon McCoy, and Nasir D. Memon. Peeling the onion's user experience layer: Examining naturalistic use of the tor browser. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1290–1305. ACM, 2018. doi: 10.1145/3243734.3243803. URL https://doi.org/10.1145/3243734.3243803.

[36] John Geddes, Rob Jansen, and Nicholas Hopper. How low can you go: Balancing performance with anonymity in Tor. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, pages 164–184, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[37] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. A look at router geolocation in public and commercial databases. In *Proceedings of the 2017 Internet Measurement Conference*, pages 463–469, 2017.

[38] Yossi Gilad and Amir Herzberg. Spying in the dark: TCP and Tor traffic analysis. In *International symposium on privacy enhancing technologies symposium*, pages 100–119. Springer, 2012.

[39] Andre Greubel, Steffen Pohl, and Samuel Kounev. Quantifying measurement quality and load distribution in tor. In *Annual Computer Security Applications Conference*, ACSAC '20, page 129–140, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450388580. doi: 10.1145/3427228.3427238. URL https://doi.org/10.1145/3427228.3427238.

[40] Hans Hanley, Yixin Sun, Sameer Wagh, and Prateek Mittal. DPSelect: A differential privacy based guard relay selection algorithm for Tor. *Proceedings on Privacy Enhancing Technologies*, 2019(2):166–186, 2019. doi: doi:10.2478/popets-2019-0025. URL https://doi.org/10.2478/popets-2019-0025.

[41] Mitchell Hashimoto. Terraform. https://www.terraform.io, 2014. Accessed December 2021.

[42] M. Imani, M. Amirabadi, and M. Wright. Modified relay selection and circuit selection for faster Tor. *IET Communications*, 13(17):2723–2734, 2019.

[43] Rob Jansen and Matthew Traudt. Tor's been KIST: A case study of transitioning Tor research to practice. *arXiv preprint arXiv:1709.01044*, 2017.

[44] Rob Jansen, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. Never been KIST: Tor's congestion management blossoms with kernel-informed socket transport. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 127–142, 2014.

[45] Rob Jansen, Marc Juarez, Rafa Galvez, Tariq Elahi, and Claudia Diaz. Inside job: Applying traffic analysis to measure Tor from within. In *NDSS*, 2018.

[46] Rob Jansen, Matthew Traudt, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. KIST: Kernel-informed socket transport for ToR. *ACM Transactions on Privacy and Security (TOPS)*, 22(1):1–37, 2018.

[47] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 337–348, 2013.

[48] Aaron Johnson, Rob Jansen, Aaron D. Jaggard, Joan Feigenbaum, and Paul Syverson. Avoiding the man

on the wire: Improving Tor's security with trust-aware path selection. *Network and Distributed System Security Symposium (NDSS)*, 2017.

[49] H. T. Karaoglu, M. B. Akgun, M. H. Gunes, and M. Yuksel. Multi path considerations for anonymized routing: Challenges and opportunities. In *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2012.

[50] Katharina Kohls and Christina Pöpper. DigesTor: Comparing passive traffic analysis attacks on Tor. In Javier Lopez, Jianying Zhou, and Miguel Soriano, editors, *Computer Security*, pages 512–530, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99073-6.

[51] Katharina Kohls, Kai Jansen, David Rupprecht, Thorsten Holz, and Christina Pöpper. On the challenges of geographical avoidance for tor. In *NDSS*, 2019.

[52] Stefan Köpsell. Low latency anonymous communication - how long are users willing to wait? In Günter Müller, editor, *Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006, Freiburg, Germany, June 6-9, 2006, Proceedings*, volume 3995 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2006. doi: 10.1007/11766155_16. URL https://doi.org/10.1007/11766155_16.

[53] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 406–422, 2017.

[54] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable messaging system with cryptographic privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, 2020.

[55] Rustam Lalkaka. Introducing Argo — A faster, more reliable, more secure internet for everyone. https://blog.cloudflare.com/argo/, 2017. Accessed December 2021.

[56] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 711–725, 2018.

[57] Shuai Li, Huajun Guo, and Nicholas Hopper. Measuring information leakage in website fingerprinting attacks and defenses. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1977–1992, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356930. doi: 10.1145/3243734.3243832. URL https://doi.org/10.1145/3243734.3243832.

[58] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS. Springer, January 2010.

[59] MaxMind. GeoLite2 Free Geolocation Data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data, 2021. Accessed December 2021.

[60] Sebastian Wilhelm Ludwig Meiser. *Quantitative anonymity guarantees for Tor*. PhD thesis, Saarland University, 2016.

[61] Asya Mitseva, Marharyta Aleksandrova, Thomas Engel, and Andriy Panchenko. Security and performance implications of BGP rerouting-resistant guard selection algorithms for Tor. In Marko Hölbl, Kai Rannenberg, and Tatjana Welzer, editors, *ICT Systems Security and Privacy Protection*, pages 219–233, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58201-2.

[62] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. DeepCorr: Strong flow correlation attacks on Tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1962–1976. ACM, 2018.

[63] Ravi Netravali, Vikram Nathan, James Mickens, and Hari Balakrishnan. Vesper: Measuring time-to-interactivity for web pages. In Sujata Banerjee and Srinivasan Seshan, editors, *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, pages 217–231. USENIX Association, 2018. URL https://www.usenix.org/conference/nsdi18/presentation/netravali-vesper.

[64] Rishab Nithyanand, Oleksii Starov, Phillipa Gill, Adva Zair, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/measuring-mitigating-as-level-adversaries-against-tor.pdf.

[65] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix anonymity system. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, 2017.

[66] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. Ip geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[67] The Tor Project. Tor metrics: relays and bridges. https://metrics.torproject.org/networksize.html, 2021. Accessed December 2021.

[68] The Tor project. Relay requirements. https://community.torproject.org/relay/relays-requirements/, 2021. Accessed December 2021.

[69] The Tor project. Research safety board. https://research.torproject.org/safetyboard/, 2021. Accessed December 2021.

[70] Mohammad Rajiullah. *Towards a low latency internet: understanding and solutions*. PhD thesis, Karlstad University Press, 2015. URL https://www.diva-portal.org/smash/get/diva2:846109/FULLTEXT01.pdf.

[71] Mohammad Rajiullah, Andra Lutu, Ali Safari Khatouni, Mah-Rukh Fida, Marco Mellia, Anna Brunström, Özgü Alay, Stefan Alfredsson, and Vincenzo Mancuso. Web

experience in mobile networks: Lessons from two million page visits. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1532–1543. ACM, 2019. doi: 10.1145/3308558.3313606. URL https://doi.org/10.1145/3308558.3313606.

[72] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. *Proceedings 2018 Network and Distributed System Security Symposium*, 2018. doi: 10.14722/ndss.2018.23105. URL http://dx.doi.org/10.14722/ndss.2018.23105.

[73] Florentin Rochet and Olivier Pereira. Waterfiling: Balancing the Tor network with maximum diversity. *CoRR*, abs/1609.04203, 2016. URL http://arxiv.org/abs/1609.04203.

[74] Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. CLAPS: Client-location-aware path selection in Tor. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 17–34, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. doi: 10.1145/3372297.3417279. URL https://doi.org/10.1145/3372297.3417279.

[75] Brandon Schlinker, Todd Arnold, Ítalo Cunha, and Ethan Katz-Bassett. PEERING: Virtualizing BGP at the edge for research. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 51–67, 2019.

[76] Theodor Schnitzler, C. Pöpper, Markus Dürmuth, and K. Kohls. We built this circuit: Exploring threat vectors in circuit establishment in Tor. In *2021 IEEE European Symposium on Security and Privacy (EuroS P)*, 2021.

[77] Srinivasan Seetharaman and Mostafa Ammar. Inter-domain policy violations in multi-hop overlay routes: Analysis and mitigation. *Computer Networks*, 53(1):60–80, 2009. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2008.09.014. URL https://www.sciencedirect.com/science/article/pii/S1389128608003009.

[78] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. OpenStack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42, 2012.

[79] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable link-based relay selection for anonymous routing. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 73–93. Springer, 2009.

[80] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943. ACM, 2018.

[81] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabián E. Bustamante. Drafting behind Akamai: Inferring network conditions based on CDN redirections. *IEEE/ACM Transactions on Networking*, 17(6):1752–1765, 2009. doi: 10.1109/TNET.2009.2022157.

[82] Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. Counter-RAPTOR: Safeguarding Tor against active routing attacks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 977–992. IEEE, 2017.

[83] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.

[84] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[85] Chris Wacek, Henry Tan, Kevin S Bauer, and Micah Sherr. An empirical evaluation of relay selection in Tor. In *NDSS*, 2013.

[86] Gerry Wan, Aaron Johnson, Ryan Wails, Sameer Wagh, and Prateek Mittal. Guard placement attacks on path selection algorithms for Tor. *Proceedings on Privacy Enhancing Technologies*, 2019(4):272–291, 2019. doi: doi:10.2478/popets-2019-0069. URL https://doi.org/10.2478/popets-2019-0069.

[87] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. Congestion-aware path selection for Tor. In *International Conference on Financial Cryptography and Data Security*, pages 98–113. Springer, 2012.

[88] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA, 2014. USENIX Association. ISBN 978-1-931971-15-7. URL https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao.

[89] Lei Yang and Fengjun Li. Enhancing traffic analysis resistance for Tor hidden services with multipath routing. In Bhavani Thuraisingham, XiaoFeng Wang, and Vinod Yegneswaran, editors, *Security and Privacy in Communication Networks*, pages 367–384, Cham, 2015. Springer International Publishing. ISBN 978-3-319-28865-9.

[90] Lei Yang and Fengjun Li. mTor: A multipath Tor routing beyond bandwidth throttling. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 479–487, 2015. doi: 10.1109/CNS.2015.7346860.

Appendix A
Extended MaTor Plots

In this section we provide additional data from our MaTor analysis described in Section V-C. We refer to Section V for details of the analysis and results.

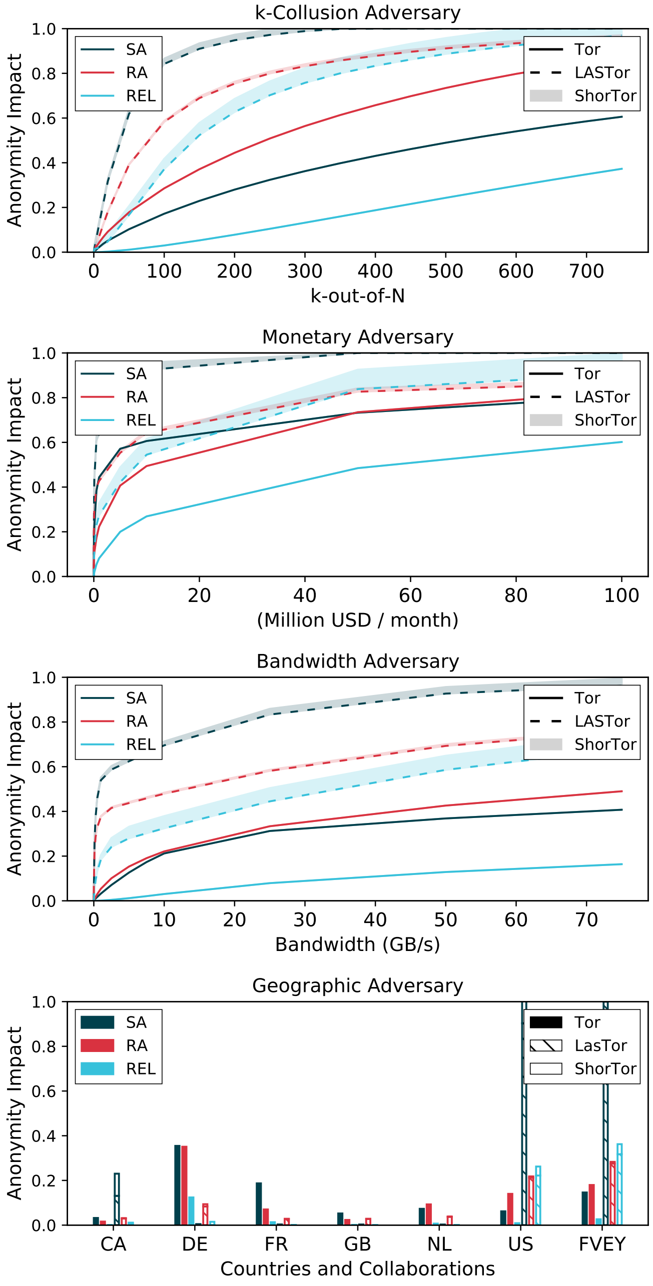

Fig. 11: Extended experiments for the anonymity impact of ShorTor. Anonymity impact of ShorTor, compared with baseline Tor (client-independent) and LASTor (dependent on client location) path selection. Each plot shows sender (SA), recipient (RA), and relationship (REL) anonymity (Definition V.1) for a different adversarial relay corruption strategy. Shaded region represents the difference in the MaTor-computed anonymity bounds with and without ShorTor. "FVEY" refers to "Five Eyes" intelligence alliance member countries. Under ANoA, ShorTor affects all anonymity notions for LASTor, though not baseline Tor.

Appendix B
Extended Integration with Tor

Section III-C covers the high-level design decisions we made for ShorTor to support integration with Tor. Here, we provide specific details on the necessary modifications and extensions which are required to integrate ShorTor with Tor. ShorTor has three primary components not present in baseline Tor: (1) on-demand data races, (2) periodic sharing of latency information, and (3) a separate data path for via traffic. We design these components to be minimal extensions to Tor's existing protocol.

A. *Data Races*

In ShorTor, data races represent the majority of required modifications. To successfully conduct a data race for traffic on a circuit, relays must: (1) recognize data race traffic as separate from steady state traffic, (2) decide when a race should be conducted while observing appropriate backoff parameters, (3) interpret the new latency tables to discover potential vias, and (4) update routing information to include vias.

We now describe how these four requirements can be implemented into Tor relay logic.

(1) can be achieved using the existing `CMD` field in Tor's cell header (see Figure 4). By introducing a new `CMD` value to indicate that traffic is part of a data race, both vias and circuit relays can immediately recognize (and potentially drop) race traffic with minimal processing. Importantly, using the `CMD` field allows ShorTor to conduct data races *without* altering the content of the cell. Because of this, data race cells are simply normal cells from a client's traffic stream and clients will see no interruption or delay while the race is run.

(2) is described in Section III-B and Section III-C. The only additional detail is that this process will need to tie into the queuing architecture shown in Figure 5 such that, when a data race is to be conducted for a circuit, the next cell out of its queue will be duplicated, have its header modified according to Figure 4, and have the copy rerouted to each of the vias in the race prior to reaching the output buffer. This is due to the fact that, as the cells are now going to different destinations, they will also be on different TCP/TLS connections and must be sent to the appropriate output buffer.

(3) is a simple matter of reading the latest latency table and selecting the vias with the lowest recorded latencies to participate in the race.

(4) is handled by a new field for the `ViaID` in Tor's routing table and the new header information in Tor cells containing the `IDs` of the two adjacent circuit relays (see Figure 4). This information allows the via relay and both adjacent circuit relays to recognize which circuit a traffic stream is from (and, consequently, where it should be routed) even when the traffic stream arrives on a different TCP/TLS connection than is usual for the circuit (i.e., arrives through a via, not from the previous circuit relay).

B. Latency Tables

Maintaining and disseminating up to date latency information is a simple process detailed in Protocol 2 presented in Section III-B. Unlike data races, it can be run entirely independently from the main Tor protocol and does not involve any circuit traffic. As such, it requires no actual integration with the Tor protocol, but is simply an entirely separate functionality run by Tor relays. The table of latency information produced as part of this process is accessed to inform the selection of candidate vias for a data race, but is not otherwise involved and, in particular, is not used during the steady-state of routing traffic through vias.

C. Data Path for Via Traffic

The data path for via traffic is almost identical to that of regular, non-via, Tor traffic and follows the queuing architecture of baseline Tor as shown in Figure 5. The two differences are that traffic is not onion encrypted/decrypted while being forwarded by a via and that via traffic is considered lower priority than circuit traffic. Via traffic, like data races, can utilize the `CMD` field in the cell header to identify itself as soon as the TLS layer has been decrpyted. This lets the via relay know that it is not expected to operate on the onion encryption layer (and does not have the keys necessary to do so) and that it should send the cell directly to the appropriate queue instead. Finally, while Tor already prioritizes browsing traffic over bulk download traffic, ShorTor requires a new priority level for via traffic that is below that of circuit traffic. This priority level is applied when cells are dequeued to be scheduled to the output buffers. Deprioritizing via traffic is necessary to ensure that it never outcompetes circuit traffic sharing the same relay (see paragraph in Section III-C on load balancing and fairness).