2012 IEEE Symposium on Security and Privacy

LASTor: A Low-Latency AS-Aware Tor Client

Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha
Department of Computer Science and Engineering
University of California, Riverside
{makho001,cyu,harsha}@cs.ucr.edu

***Abstract*—The widely used Tor anonymity network is designed to enable low-latency anonymous communication. However, in practice, interactive communication on Tor—which accounts for over 90% of connections in the Tor network [1]—incurs latencies over 5x greater than on the direct Internet path. In addition, since path selection to establish a circuit in Tor is oblivious to Internet routing, anonymity guarantees can breakdown in cases where an autonomous system (AS) can correlate traffic across the entry and exit segments of a circuit.**

In this paper, we show that both of these shortcomings in Tor can be addressed with only client-side modifications, i.e., without requiring a revamp of the entire Tor architecture. To this end, we design and implement a new Tor client, *LASTor*. First, we show that *LASTor* can deliver significant latency gains over the default Tor client by simply accounting for the inferred locations of Tor relays while choosing paths. Second, since the preference for low latency paths reduces the entropy of path selection, we design *LASTor*'s path selection algorithm to be tunable. A user can choose an appropriate tradeoff between latency and anonymity by specifying a value between 0 (lowest latency) and 1 (highest anonymity) for a single parameter. Lastly, we develop an efficient and accurate algorithm to identify paths on which an AS can correlate traffic between the entry and exit segments. This algorithm enables *LASTor* to avoid such paths and improve a user's anonymity, while the low runtime of the algorithm ensures that the impact on end-to-end latency of communication is low. By applying our techniques to measurements of real Internet paths and by using *LASTor* to visit the top 200 websites from several geographically-distributed end-hosts, we show that, in comparison to the default Tor client, *LASTor* reduces median latencies by 25% while also reducing the false negative rate of not detecting a potential snooping AS from 57% to 11%.

I. Introduction

Tor [2] is a widely used and deployed network for anonymous communication on the Internet. Unlike other systems that facilitate anonymous communication [3], [4], Tor distinguishes itself by enabling low-latency communication. Indeed, a vast majority of users—accounting for over 90% of TCP connections [1] on Tor—use Tor for interactive traffic.

However, several measures for increasing client anonymity in Tor fundamentally inflate communication latencies. For example, the default Tor client sets up a tunnel between itself and a destination via three relays selected at random, with some preference for relay stability and access link bandwidth. This random selection of relays can lead to circuitous routing of tunnels around the globe, resulting in high latencies. Previous solutions for improving performance on Tor have either focused on increasing throughput [5], or those that focused on improving latencies mandate a revamp of the Tor network, e.g., by having all Tor relays participate in a network coordinate system [6], [7] or by modifying traffic management at relays [8]. Due to the undoubtedly significant development effort required to implement these changes, these solutions are yet to be deployed.

In addition, Tor's anonymity guarantees breakdown in some cases due to its path selection being oblivious to Internet routing. For example, on some paths, an Autonomous System (AS) may be present on the Internet routes both between the client and the entry relay and between the exit relay and the destination. Such an AS can statistically correlate traffic on the entry and exit segments of the path and potentially infer the destination with which the client communicated. Though this problem has been recognized previously [9], [10] and the default Tor client attempts to preempt such cases by ensuring that no two relays in a path are in the same /16 IP prefix, we find that this heuristic is insufficient for detecting most instances of potential snooping by ASes.

In this paper, we seek to address both of the above shortcomings with Tor today by making only client-side modifications. This approach ensures that a user can obtain the resultant benefits in latency and anonymity simply by updating her Tor client, without having to wait for changes to the rest of the Tor network. Therefore, we seek to answer the following question: what latency improvements can a Tor client obtain *today*, without any modifications to the rest of Tor, while also avoiding paths on which an AS could break the client's anonymity by correlating traffic? Towards this end, we design and implement *LASTor*, a new Tor client that differs from the default Tor client only in its path selection algorithm.

In developing *LASTor*, we make three primary contributions. First, we show that significant latency gains are possible by solely accounting for the inferred geographic locations of relays, rather than needing up-to-date latency information of Internet paths (e.g., from network coordinates). We implement the Weighted Shortest Path (WSP) algorithm that probabilistically chooses paths with a preference for shorter paths. However, with a naive implementation of WSP, an adversary can increase the probability of a relay under his control being on the chosen path by simply setting up a large number of relays in the same location, which is close to the direct line between the client and the destination. To preempt this attack, we implement *LASTor* to execute WSP on a graph of the Tor network where nearby relays are clustered together; this increases the onus on an adversary to establish relays in several

DOI 10.1109/SP.2012.35

locations in order to ensure a high probability for the chosen path traversing a relay under his control. A side-effect of clustering relays is that WSP's runtime is significantly reduced.

Second, we make *LASTor* resilient to the attack where an AS can correlate traffic on the entry and exit segments of the chosen path by explicitly avoiding such paths. To do so, we need to equip *LASTor* with the ability to predict Internet routing between relays and end-hosts; we cannot simply measure routes from every relay since we seek a solution that only requires client-side modifications. The use of existing approaches for predicting Internet routes is however impractical since they either require clients to download gigabytes of data daily [11], [12] or have significantly high runtimes [13], which would override the benefits of selecting a low latency path. Therefore, we instead develop a computationally lightweight technique that has a low false-negative rate in failing to identify paths that permit the possibility of "snooping" ASes. Our key insight here is to predict the *set of ASes* through which the Internet *may* route traffic between a pair of IP addresses, rather than predicting the *precise route* between them. Importantly, in order to run this AS set prediction algorithm, clients need download only 13 MB of data initially and 1.5 MB every week thereafter.

Finally, *LASTor* makes path selection tunable. Probabilistic selection of paths with a preference for shorter paths reduces the entropy of path selection, and all users may not wish to trade-off the resulting reduction in anonymity for reduced latency. Therefore, *LASTor* enables a user to choose an appropriate tradeoff between latency and anonymity. By choosing a value between 0 (lowest latency) and 1 (highest anonymity) for a single parameter, a user can configure *LASTor* to appropriately tailor path selection.

We demonstrate *LASTor*'s benefits in improving latency by using it to visit the top 200 websites from 50 geographically distributed PlanetLab nodes. We see that even without any modification to the rest of Tor, *LASTor* provides a median latency improvement of 25% over the default Tor client. We also use measurements of AS-level routes on over 200K Internet paths to evaluate *LASTor*'s ability to preempt the possibility of snooping ASes jeopardizing the anonymity of clients. We see that for the median (client, destination) pair, *LASTor* fails to identify only 11% of the instances in which a snooping AS can exist; in comparison, we observe a false-negative rate of 57% with the default Tor client.

II. Background and motivation

In this section, we provide some background on Tor and discuss results that motivate our work.

A. Tor overview

Tor [14], a low-latency open source application that allows users to use the Internet anonymously, was developed in September of 2002. In Tor, clients download a list of relays and some information about these relays from directory servers. To establish a connection to a destination, a client selects three

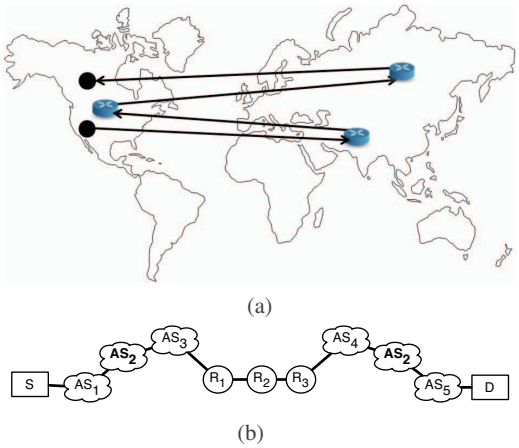


Fig. 1. (a) Random relay selection can inflate end-to-end latencies due to circuitous routing, and (b) an example in which an AS (AS2) can subvert the client's anonymity by correlating traffic across the entry and exit segments.

relays—entry, middle, and exit nodes—and builds a circuit [1] through these three relays. The client appropriately encrypts the data it sends to the entry relay so that each of these three relays only knows the nodes before and after it on the path, i.e., the entry relay knows the source and the middle relay, the middle relay knows only the entry and exit relays, and the exit relay knows only the middle relay and the destination. This form of onion routing [15] preserves the client's anonymity by ensuring that no one other than the client knows that it communicated with the destination.

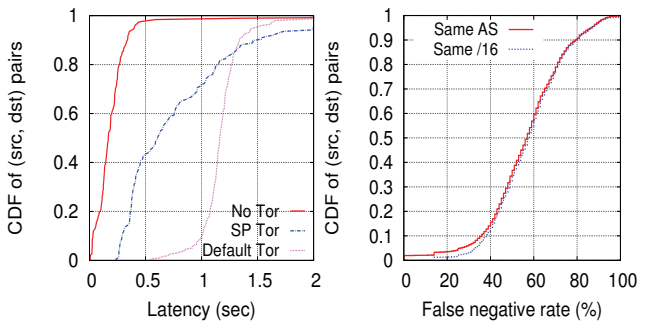
To avoid statistical profiling attacks, the default Tor client restricts its choice of entry nodes to a persistent list of three randomly chosen nodes named "entry guards" [16]. For the middle node, the Tor client sorts Tor relays based on their access link bandwidth and randomly selects a relay, with the probability of selection being higher for relays with higher bandwidth. For the selection of the exit node, clients are constrained by the fact that a large fraction of relays choose to not serve as exit nodes. This is because destination servers see the exit node as the computer that communicates with them; if any malicious activity is detected by the destination, it will assume that the exit relay is responsible. Therefore, when selecting an exit node, a client chooses at random (again with bias for higher bandwidth relays) among those relays willing to serve as an exit node for the particular destination that the client is attempting to contact and the particular service with which this communication is associated.

B. Motivation

The motivation for our work stems from two sources of inefficiency in path selection as above in Tor today—high latency due to circuitous routing and degradation of anonymity because of path selection being oblivious to Internet routing.

Poor latency. First, as discussed above, a client selects entry, middle, and exit nodes in a circuit more or less at random. As a result, the circuit between a client and a destination

[1] We use the terms path, circuit, and tunnel interchangeably in this paper.



(a) (b)

Fig. 2. (a) Comparison of latencies on the direct Internet path, with Shortest Path routing on Tor, and with the default Tor client. (b) False negatives in detecting snooping ASes with default Tor client.

can often be circuitous, causing significant latency overhead compared to latency on the default Internet path between the client and the destination. Since Tor is predominantly used for interactive communication [1], e.g., to visit websites, this increased latency degrades user experience. Fig. 1(a) presents such an example. A client in the US communicates with a server in Canada. The client incurs significant latency overhead due to relay selection inefficiencies because all packets from the client travel around the world two times before they reach their destination.

To quantify the extent of this latency overhead, we measured the latency of visiting the top 200 websites [17] from 50 PlanetLab nodes [18] spread across the globe. We measured the latency between every PlanetLab node and every website as the median latency of 5 HTTP HEAD requests. We first measured latencies by having the PlanetLab nodes contact the websites directly. Next, we repeated the same with the communication happening over the default Tor setup. We finally measured latencies via Tor when choosing entry, middle, and exit nodes that result in the shortest end-to-end path based on the geographical locations (inferred using MaxMind's IP geolocation database [19]) of the client, the destination, and the relays on the path. Fig. 2(a) shows the distribution across (PlanetLab node, website) pairs of the latencies measured in the three cases. First, we see that latencies measured using default Tor are more than 5x greater than via the direct Internet path (*no Tor*) in the median case. Second, latencies over the shortest path on Tor (*SP Tor*) result in a 2x reduction in median latency compared to default Tor.

Circuit establishment in Tor however cannot simply be modified to select the shortest path between the client and the destination; this makes path selection deterministic and enables adversaries to strategically setup relays that can subvert the client's anonymity. Instead, motivated by the latency improvements possible by choosing geographically shorter paths, our goal is to enable probabilistic path selection that can deliver some of these latency benefits without significantly compromising client anonymity.

Lack of AS-awareness. Though Tor's use of onion routing tries to ensure that no one other than the client has knowledge of the destinations with which it communicates, there are a variety of attacks possible (e.g., [20] [21]) from which this information can be inferred. One such attack arises because of Tor's path selection being oblivious to Internet routing. In the case where the routes through the Internet from the client to the entry node and from the exit node to the destination both traverse a common Autonomous System (AS), such an AS can correlate the traffic it observes to infer the (client, destination) pair [22], [23]. Fig. 1(b) shows an example in which AS2, which appears on both the routes from the source S to the entry relay R1 and from the exit relay R2 to the destination D, can potentially infer that S is communicating with D. We hereafter refer to such ASes that have the potential of correlating traffic by snooping as *snooping ASes*. Note that even though traffic between the client and the entry node is encrypted, ASes can observe the client's IP address in the headers of the packets that the client sends to the entry node.

Feamster and Dingledine [9] showed that the probability of existence of snooping ASes is 10–30%. This observation was re-evaluated 5 years later by Edman and Syverson [10]. They observed that while there are many more Tor relays than before, this growth has only a slight effect on mitigating attacks by snooping ASes. This is because Tor relays are not scattered uniformly among ASes, and so the growth of the network does not guarantee path location diversity. Further, the presence of ASes that can snoop is especially likely in cases where the client and destination are in the same location, because the entry and exit segments of the circuit may go through the same ASes with presence in that region.

Therefore, to protect its anonymity, a Tor client needs to ensure that its algorithm for path selection prevents, or at least minimizes, the existence of common ASes across both ends of a circuit. To preempt AS-level attacks and preserve anonymity, Tor's default path selection algorithm ensures that the entry and exit nodes on any particular circuit do not share the same /16 IP address prefix [24].

We however find that this heuristic performs poorly in practice in avoiding snooping ASes. First, in the deployment of Tor as of June 2011, we observe that 60% of ASes that have Tor relays resident in them have at least two relays that are in different /16 subnets. In addition, we evaluated the /16 prefix heuristic on a dataset of measured AS paths (the *PL-BGP-Rand* dataset described later in Section III). For every (client, destination) pair in our dataset, we computed the false negative rate of the /16 heuristic, i.e., of all entry and exit node combinations in which there was a common AS across the entry and exit segments, the fraction that the /16 heuristic deemed as safe from snooping ASes. Fig. 2(b) plots this false negative rate for this heuristic across (client, destination) pairs. The /16 heuristic for avoiding snooping ASes miss over 40% of instances of snooping ASes for more than 80% of (client, destination) pairs. Furthermore, we find that simply accounting for the ASes in which the relays reside (the *"Same AS"* line in Fig. 2(b)) is also insufficient.

To address the shortcomings of these heuristics, Tor clients need to determine the ASes through which the Internet routes traffic between them and entry nodes and between exit nodes

Goal	Technique	Section
Reduce latency of communication on Tor	Weighted Shortest Path (WSP) algorithm for probabilistic selection of paths with preference for low-latency paths	IV-A
Defend against strategic establishment of relays to increase probability of compromised relays on chosen path	Clustering of relays in nearby locations	IV-B
Enable user to choose trade-off between latency and anonymity	Augment WSP with parameter α that can be varied between 0 (lowest latency) and 1 (highest anonymity)	IV-D
Account for distributed destinations	DNS lookup service on PlanetLab nodes	IV-C
Preempt traffic correlation attacks by ASes	Lightweight algorithm to determine set of ASes through which Internet may route traffic between a pair of IP addresses	V

TABLE I
OVERVIEW OF TECHNIQUES DEVELOPED TO BUILD *LASTor*.

and destinations. Since we seek only client-side solutions, modifying relays to measure routes is not an option. Querying a route prediction service (e.g., iPlane [12]) for this information is not an option either since the client and destination will be revealed to the service. On the other hand, having clients download pre-computed AS paths between themselves and all entry guards and between all exit relays and all end-hosts will require clients to download a prohibitively large dataset. For example, even if we aggregate Tor relays and all end-hosts on the Internet into BGP atoms [25] [2], based on the average AS path length of 4 on the Internet, we estimate that clients will have to download on the order of 500 MB of data. Further, this data will have to be continually updated to account for flux in the Internet's routing.

Instead, it is imperative that clients download a snapshot of Internet topology and routing information and make route predictions locally. However, enabling such local route predictions with current techniques poses two problems. First, it is impractical to expect clients to download several gigabytes of data, e.g., iPlane's Internet atlas, to make such predictions. Second, AS path inference techniques that operate on a compact Internet atlas [11], [13], have high computational overhead and take on the order of a second to estimate the AS path between a pair of IP addresses. Since a Tor client has to choose from around 1000 exit relays in setting up a circuit, the use of such computationally-heavy techniques to estimate AS paths can impose high overhead on path selection, rendering the latency benefits of avoiding circuitous routes moot.

III. OVERVIEW

Next, we define the precise problem statement that we target and provide a brief overview of our work. We also discuss the datasets that we use throughout our work to evaluate the techniques that we develop.

A. Problem statement

Our goal in this paper is to address the shortcomings in Tor discussed above with respect to latency and anonymity without requiring a revamp of Tor's design. Leveraging the fact that intelligence in Tor resides at the client, we seek to only modify the client-side path selection algorithm so that clients can benefit today without waiting on updates to relays to be developed and deployed. In doing so, we respect conventional wisdom on how to preserve client anonymity in Tor, e.g., the use of three entry guards to protect against statistical profiling attacks and the need for sufficient randomness in relay selection to protect against colluding relays.

Table I summarizes the techniques that we present in the rest of the paper to address this problem by developing *LASTor*.

[2]All IP addresses in the same atom have identical AS paths from/to them to/from the rest of the Internet.

Dataset	Clients	Relays	Destinations
PL-Tor-Web	50	2423	200
PL-BGP-Rand	50	378	500
PL-PL-Web	50	50	500

TABLE II
SUMMARY OF DATASETS.

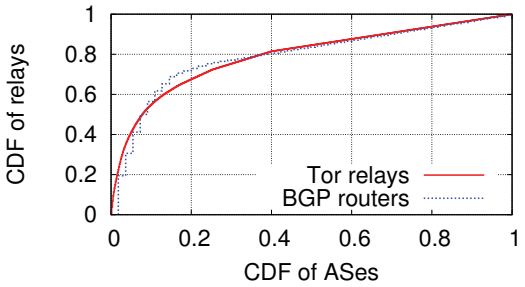


Fig. 3. Distribution of relays across ASes in *PL-BGP-Rand* and *PL-Tor-Web* datasets.

B. Measurement datasets

To evaluate *LASTor*'s components, we make use of three datasets (summarized in Table II), with PlanetLab nodes serving as clients in all three cases; we pick 50 PlanetLab nodes to use as clients, in keeping with the distribution across countries of Tor clients [26]. In our first dataset, *PL-Tor-Web*, we use 200 websites [17] as destinations and the relays in the actual Tor network serve as relays. In this dataset, while we can measure both latencies and AS-level routes from PlanetLab nodes to Tor relays, we do not have access to either information on paths from relays to destinations. Second, we use the *PL-BGP-Rand* dataset, in which BGP routers seen in various BGP feeds [27], [28] serve as relays and the .1 IP address in 500 randomly chosen /24 prefixes serve as destinations. Here again, we can directly measure latencies and AS paths from PlanetLab nodes to BGP routers. In addition, we obtain the AS paths from the BGP routers to the destinations from various BGP feeds, but we do not have latencies along these paths. This dataset enables to evaluate our techniques for AS-awareness in path selection using *measured* AS-level Internet routes, unlike prior work in this area [9], [10] that has relied on

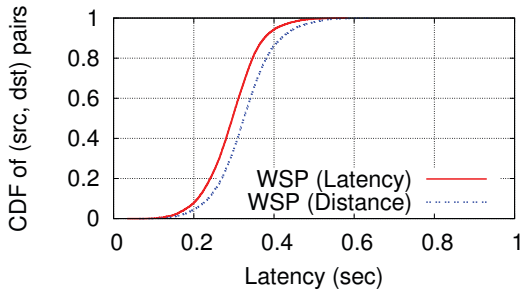


Fig. 4. Comparison of end-to-end latency when using geographical distance versus the use of path latency in the Weighted Shortest Path algorithm.

inferred AS-level routes. Though we have to use BGP routers as proxy for Tor relays for this purpose, Fig. 3 shows that the distribution of relays across ASes in the *PL-BGP-Rand* dataset is similar to that in the case of real Tor relays.

Finally, in the *PL-PL-Web* dataset, we use PlanetLab nodes as both clients and relays and the top 200 websites as destinations. In this case, we can measure latencies and AS paths both from all clients to all relays and from all relays to all destinations. To emulate typical Tor clients, we ensure throughout our evaluation that we do not provide as input to iPlane [12] any Internet topology measurements from the 50 PlanetLab nodes used as clients; as we describe later in Section V, we use AS path length estimates from iPlane for our AS set prediction.

IV. Path Selection

Path latency on the Internet is a sum of three factors—propagation delay (time spent by packets on the wire), queueing delay (time spent by packets enqueued at end-hosts or intermediate routers, waiting to be put onto the wire), and transmission delay (time to put a packet onto the wire). Since access link bandwidths of the client and Tor relays is beyond our control, we cannot reduce transmission delay. On the other hand, as we show later in Section VII, a modification of Tor relays would be necessary to reduce queueing delays. Therefore, we focus here on reducing propagation delay.

A. Preferential selection of low-latency paths

To reduce propagation delays, we need to reduce the probability of selection of circuitous paths. We cannot however simply pick the shortest possible path through three relays between a client and a destination. This would make path selection deterministic and hence, susceptible to strategically placed adversarial Tor relays. Therefore, we implement a Weighted Shortest Path (WSP) algorithm. WSP orders all possible paths between a client and a destination based on the expected latency on each path. The latency along a path is the sum of latencies on each of the four segments of the path—(client, entry relay), (entry relay, middle relay), (middle relay, exit relay), and (exit relay, destination). The probability of a particular path being selected is then inversely proportional to the expected latency on it.

However, in order to estimate the latency along every possible path of three relays between the client and the destination, we would need latencies between the client and all candidate entry relays, between all candidate exit relays and the destination, and between all pairs of relays. As proposed in previous approaches to improve latency in Tor [6], [7], gathering this latency information would require a modification of Tor relays. Instrumenting measurements of the Internet at such a scale is a non-trivial undertaking. As a result, these prior proposals are yet to translate into practice.

Our focus here instead is on a practical implementation of WSP, with changes only at the Tor client. Therefore, we use the end-to-end geographical distance along a path as a proxy for the latency along it. This ensures that we do not need to modify relays to track latencies between them, but we can rely instead on the estimated geographic locations of clients, relays, and destinations. We compute the end-to-end geographical distance along a path by summing up the distance along each segment, which we in turn compute based on the (latitude, longitude) coordinates of the hosts at either end of a segment. We can estimate the geographic locations of end-hosts and relays using an IP geolocation database, such as MaxMind [19]. Out of all candidate paths, WSP then selects one path with the probability of a path's selection being proportional to the weight associated with it; the weight associated with a path is the difference between the maximum end-to-end distance across all paths and the distance along this particular path.

Though the use of geographical distance ignores the effect of routing on latency (the Internet may forward packets along a circuitous route [29]), we confirm empirically that our use of geographical distance as the weight for every edge in the graph when running WSP is a reasonable substitute for the latency of every edge. Since we can compute end-to-end latencies of paths only on the *PL-PL-Web* dataset, we perform this analysis on that data. We perform this analysis first using latency as the edge weight metric for running WSP and then repeat the same using geographic distance for edge weights. Fig. 4 shows that the end-to-end latencies of chosen paths are similar irrespective of whether WSP uses latencies or geographic distances as edge weights. Therefore, we believe that our use of geographic distances delivers most of the benefits of reducing propagation delays without warranting the need for a distributed infrastructure that measures latencies between all pairs of relays, an unarguably arduous undertaking.

B. Clustering of relays

A straightforward implementation of WSP however causes two problems. First, WSP's preference for paths with lower end-to-end geographical distance results in a greater preference for paths through relays that are close to the direct line between the client and the destination. For example, in Fig. 5, WSP will select the path through relay $R1$ with a higher probability than the path through $R2$. As a result, if an adversary wishes to ensure that a relay under his control is on the chosen path between S and D, then the adversary can choose a location that is close to the direct line between S and D and setup a large number of relays at that location. It is

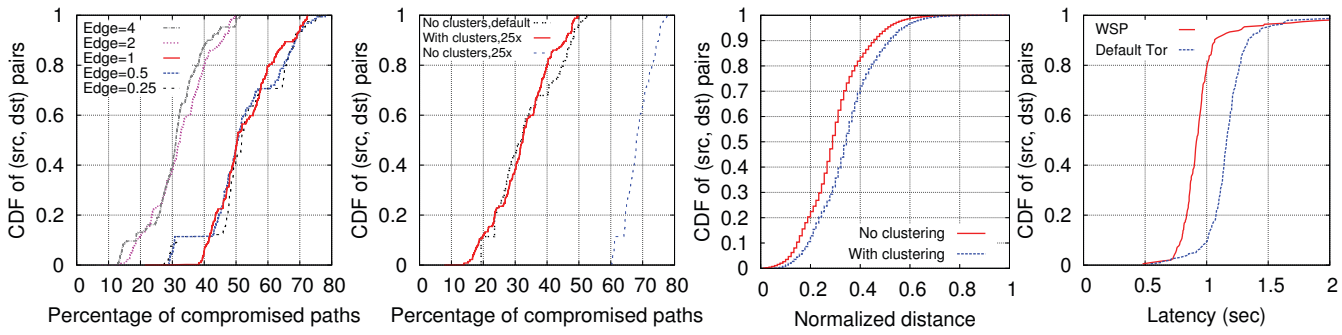


Fig. 6. (a) Clustering with higher cell sizes provides better resilience. Clustering of relays (b) reduces the probability of an adversary compromising a large fraction of paths, but (b) increases the length of the chosen path. (d) WSP yields latencies lower than those obtained with the default Tor client.

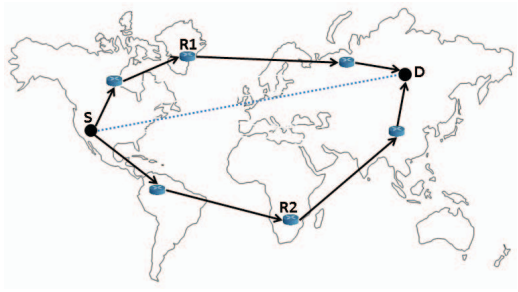


Fig. 5. WSP results in greater preference for paths through relays located close to the direct line between the client and the destination.

relatively easy for an adversary to setup several relays in the same location, for example, by renting several virtual machines in a cloud service. The high probability of at least one of the adversary's relays being on the selected path increases the chances for the adversary to use recent traffic analysis attacks on Tor [30] and infer that S is communicating with D.

The second problem with a strawman implementation of WSP is its runtime. Today, Tor has over 2500 relays with roughly 1000 of these relays willing to serve as exit nodes. The number of candidate paths between a client and a destination is therefore in the order of billions. So, a naive computation of the end-to-end geographical distance on every candidate path is computationally expensive and takes roughly 6.5 seconds to run even on a 2.5 GHz processor. This large runtime—in comparison to Internet path latencies that are of the order of tens or hundreds of milliseconds—to even select a path can render the selection of a low latency path redundant.

To address both of these problems, we cluster Tor relays that are located in geographically nearby locations. We employ a simple clustering algorithm in which we divide the globe into a grid of square cells and cluster all relays within a cell; the edge length of the cells is a configurable parameter. We then execute WSP on the clustered Tor network where every node is a cluster of relays, and each candidate path is through three clusters. WSP computes the end-to-end distance on every cluster-level path and then selects one path with preference to shorter paths as before. We translate the chosen cluster-level path to a path through three Tor relays by picking one relay at random from each of the clusters on the selected path.

This modification of WSP reduces its runtime to select a path between a client and a destination through today's Tor network to 245 milliseconds, in comparison to the runtime of 6.5 seconds with the naive implementation. More importantly, the modified WSP ensures that the establishment of a large number of relays in the same location does not bias the selection of paths through them since WSP considers paths at the granularity of the cluster to which all of them belong; paths through different relays in a cluster are not considered independently. Thus the modified WSP increases the onus on an adversary to establish relays in multiple locations in order to have one of those relays be on the chosen path with a very high probability.

We conduct the following experiment 1) to choose the cell size to be used in clustering of relays, and 2) to demonstrate the improved resilience of WSP to an adversary as discussed above. In the *PL-Tor-Web* dataset, for every (client, destination) pair, we emulate an adversary who controls the 5% of relays that are closest to the direct line between the client and the destination. We then model the adversary increasing the number of relays that he controls by replicating these 5% of closest relays by a factor of 25. We run WSP on this modified Tor network with and without clustering of relays. In either case, given a (source, destination) pair, we compute the probability of the path between them selected by WSP traversing at least one compromised relay. This value represents an upper bound on the fraction of cases in which the chosen path will traverse a relay controlled by the adversary, if the adversary controls at most 5% of relays.

Fig. 6(a) compares the distribution across (source, destination) pairs of this upper bound when clustering relays with different cell sizes. We vary the edge length of every cell from 0.25 to 4—measured in terms of the difference in latitude or longitude—and, in each case, we compute the fraction of paths that traverse a relay controlled by the adversary. We see that using a edge length of 2 for each cell significantly decreases the influence of the adversary compared to the effect when using lower edge lengths, and increasing the edge length further has minimal impact.

Next, we evaluate the resilience offered by running WSP after the clustering of relays. Fig. 6(b) compares the distri-

bution across (source, destination) pairs of the fraction of paths that traverse a compromised relay in the following three cases: 1) when running WSP on the *PL-Tor-Web* dataset without clustering of relays (*No clusters, default*), and when an adversary replicates relays in this dataset as above and WSP is executed 2) after clustering relays (using a cell size of 2x2) (*With clusters, 25x*), or (3) without clustering (*No clusters, 25x*). By comparing the "*No clusters, default*" and "*No clusters, 25x*" lines, we see that, in the absence of clustering, the adversary can increase the fraction of paths that traverse a compromised relay from around 35% to over 65% on average by replicating the relays that he controls by 25x. In contrast, when relays are clustered into cells of size 2x2, the adversary gains nothing by replicating relays.

Clustering of relays however has a negative impact on the latencies along paths chosen by WSP. This is because, in cases where there are several relays in a location close to the direct line between the source and the destination, the basic version of WSP can choose from the several candidate paths through these relays. In contrast, after these relays have been clustered, WSP has only path of choice through these relays. Hence, as shown in Figure 6(c), the geographic distance along the path chosen by WSP increases by roughly 15% in the median case when relays are clustered. This inflation in path length due to relay clustering is a compromise that we have to bear, in exchange for increasing the onus on adversaries to setup relays in several locations to attract traffic through compromised relays with high probability.

Finally, we evaluate the latency improvement obtained with WSP in practice. We modify the default Tor client to implement the WSP path selection algorithm and use the modified client to measure latencies over the Tor network to the top 200 websites from 50 PlanetLab nodes. For each (client, destination) pair, we run WSP 5 times and on each attempt, we measure the median latency of 5 HTTP HEAD requests. We then compute the median latency across the 5 attempts. We repeat the same process using the default Tor client and compute the median latency across 5 paths chosen by it, considering the median latency across 5 HTTP HEAD requests on each path. Fig. 6(d) presents the latency distribution measured across (client, destination) pairs when using WSP as compared to that when using the default Tor client. We see that WSP results in a 25% reduction in latency in the median case.

C. Accounting for distributed destinations

Thus far, our exposition of WSP has assumed that the destination has a single location associated with it. In practice, the destinations associated with interactive communication (e.g., webservers) are often replicated across several geographic locations. In such cases, users specify the destination by its hostname, and upon DNS resolution of the hostname, the webservice provider returns the IP address of the server located closest to the end-host that performs the DNS lookup. This implies that when a client uses a Tor circuit to contact a destination, the particular server with which the client ends up communicating depends on DNS resolution of the destination's hostname at the exit node on that circuit. Therefore, when WSP estimates the end-to-end distance on any candidate path, it must take into account the location of the particular IP address to which the exit node on that path will be redirected.

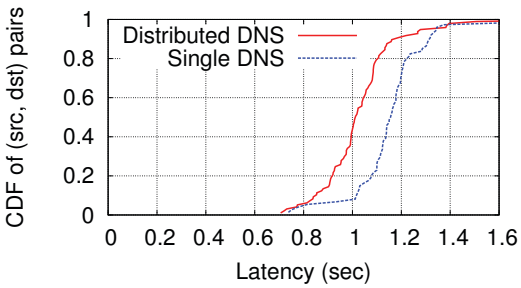
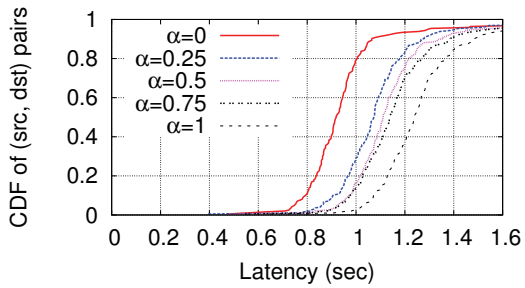


Fig. 7. Lower latencies obtained with WSP when accounting for distributed destinations.

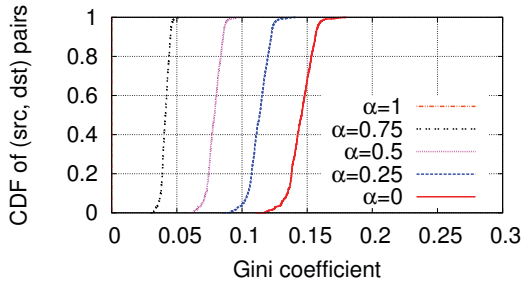
However, at the time of path selection, it is impractical to perform DNS lookups for the destination on *all* candidate exit relays. Doing so would require the client to setup a circuit for every candidate exit relay; the client cannot simply ask a relay to resolve the destination hostname since that would leak the client's anonymity. Establishing one circuit for every candidate exit relay every time a path needs to be selected would not only impose significant overhead on Tor but also take several tens of seconds, thus nullifying the benefits of selecting a low-latency path.

Instead, we setup a DNS lookup service across a set of 15 geographically distributed PlanetLab nodes. When a client needs to run WSP for a destination, it submits a request to resolve the destination's hostname to each of the PlanetLab nodes running the DNS lookup service. The client submits these requests via any one of the circuits that it had previously established, e.g., the default Tor client establishes three circuits when it starts up. The client uses HTTPS to submit these DNS resolution requests to the PlanetLab nodes so that the exit node on the circuit used for communicating with the PlanetLab nodes cannot infer the destination. Once the client receives the set of IP addresses obtained for the destination, we assume any candidate exit relay would be redirected to the IP address that is geographically closest to it amongst this set. Thus, when we subsequently run WSP to pick a path to the destination, we compute the end-to-end distance on each candidate path by using the distance along the exit segment as the distance between the exit node on that path and the destination's IP address to which we believe the exit node will be redirected.

To evaluate the utility of this modification to WSP, we consider the top 1000 websites from Quantcast and focus on those that return IP addresses in multiple locations when resolved from all PlanetLab nodes. We then measure latencies over the Tor network to these websites with 50 PlanetLab nodes as clients. We measure latencies in two cases. In the first case, we run WSP as described above where it uses IP addresses obtained by resolving the destination on 15 geographically distributed PlanetLab nodes. In the second case, we run WSP assuming the destination to have a single



(a)



(b)

Fig. 8. Increasing the value of α when using WSP results in (a) higher latencies and (b) greater entropy of path selection.

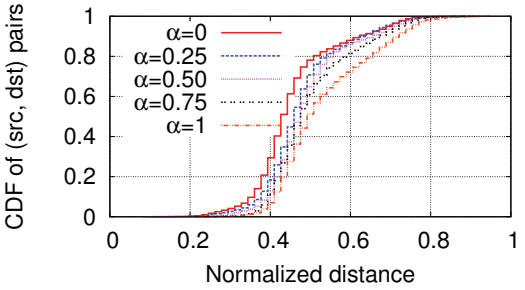


Fig. 9. End-to-end distances on paths chosen with WSP when using α to tailor the set of relays from which we select entry guards.

IP address obtained by DNS resolution at a randomly chosen exit relay. Fig. 7 compares the latencies measured in these two cases. We see that accounting for the fact that destinations could be potentially distributed reduces path latency in the median case by 15%.

D. Latency versus anonymity tradeoff

Though clustering of relays reduces the chances of compromised relays being present on a large fraction of chosen paths, WSP's preference for shorter paths naturally reduces the entropy of path selection. All users may not wish to trade-off this reduction in entropy for lower latencies. Therefore, we make path selection with WSP *tunable* with a parameter α. A user can vary α in the range 0 to 1, with a value of 0 corresponding to lowest latencies and a value of 1 corresponding to highest entropy.

We incorporate this parameter α into WSP as follows. As previously mentioned, after computing the end-to-end distance on every candidate path, WSP associates a weight with every path that is equal to the difference between the maximum end-to-end distance across all paths and the distance on that path. The probability of WSP choosing a particular path is then proportional to its weight. We now modify this weight w for a path to instead be $w^{(1-\alpha)}$. In the case when α is equal to 0, WSP defaults to the original version we presented above, which picks paths with a preference for shorter ones. On the other hand, when α is equal to 1, all paths have a weight of 1 and thus, any particular path is chosen at random. For any other value of α between 0 and 1, path selection is appropriately biased towards low latency or higher entropy.

Fig. 8 shows the effect that varying α has on both latencies and entropy. Figure 8(a) shows latencies measured with α equal to 0, 0.25, 0.5, 0.75, and 1 in the same setting as that used in Section IV-B—median latency from 5 HEAD requests each to the top 200 websites from 50 PlanetLab nodes as clients. Lower values of α result in lower latencies.

To capture the corresponding variance in entropy, we use the Gini coefficient metric [31], which has previously been used to measure anonymity of path selection in Tor, e.g., in [5]. Gini coefficient is a measure of skew in a set of values. A value of 0 for the Gini coefficient indicates perfect equality—that all values in the set are equal, whereas a value of 1 indicates perfect inequality. We use this metric to measure, for each (client, destination) pair in the *PL-Tor-Web* dataset, the skew across candidates paths of the probability of them selected by WSP. Fig. 8(b) shows that higher values of α result in lower values for the Gini coefficient, which corresponds to a lower skew across paths in the probability of their selection.

Finally, we use the parameter α to also guide the selection of entry guards. To avoid statistical profiling attacks, the default Tor client restricts its choice of entry nodes to a persistent list of three randomly chosen nodes selected when the client starts up [16]. All circuits setup by the client thereafter choose entry relays from one of these three entry guards. As one would expect, this constraint on the selection of entry relays, though good for anonymity, hurts the selection of low-latency paths by WSP; the path between a client and a destination may be unavoidably circuitous if all three entry guards chosen happen to be distant from both the client and the destination.

Therefore, in keeping with our goal of making path selection tunable between a preference for low latency or anonymity, we modify the selection of entry guards as follows. After we cluster relays as above, we order all clusters that contain candidate entry relays [3] based on their distance from the client. We then choose three clusters at random from the closest $(g + \alpha \cdot (100 - g))\%$ clusters in this ordering, and pick one relay at random from each of these clusters as the three entry guards, where g is a configurable parameter; in our implementation we use a value of 20 for g. Thus, when α equals 0—a preference for the lowest latencies—we choose the entry guards at random from the closest 20% of relays to the client. This minimizes the probability of circuitous routes when $\alpha = 0$, while still providing good anonymity by selecting entry guards from a fairly large subset (20%) of the candidate

[3]The default Tor client considers a subset of all Tor relays for selection as entry guards based on their stability.

entry relays. On the other hand, when a user chooses a value of 1 for α to get the best level of anonymity, selection of entry guards defaults to the current best practice of choosing from all candidate entry relays at random. Fig. 9 shows the effect that this modified entry guard selection algorithm has on the end-to-end distance of the chosen path in the *PL-Tor-Web* dataset. With increasing α, the randomness of entry guard selection increases and results in longer path lengths.

V. AS Awareness

Next, we address the second limitation of interest in the default Tor client—avoiding paths in which an Autonomous System (AS) can correlate traffic across the routes between the client and entry relay and between the exit relay and the destination. Since our goal is to not require any modifications to Tor relays, we cannot avoid such paths by simply having all relays measure routes from them to the client and to the destination. Therefore, we next discuss how a client can locally make estimations of routing in the Internet in order to identify and ignore paths that present the possibility of snooping ASes.

A. AS set estimation

Precise inference of AS-level routes between arbitrary IP addresses is hard, as seen in the fact that no existing technique for doing so [32], [11], [33], [12], [13] is close to perfect. Therefore, when evaluating whether a particular combination of entry and exit relays offers the possibility of a snooping AS, we preclude the approach of estimating the AS-level route on the entry and exit segments of the circuit. Instead, we take the approach of predicting for either segment, *a set of candidate ASes through which the Internet is highly likely to route traffic* on the segment. We can then determine the potential existence of snooping ASes by checking if the intersection between the AS sets for the paths between the client and the entry relay and between the exit relay and the destination is non-empty.

To enable such inference of AS sets by Tor clients, we require clients to download three inputs. First, we use the Internet's AS-level topology represented as a set of inter-AS links. Second, we need an estimate of the AS path length between every Tor relay and every end-host on the Internet. We need this information as input because the AS path selected by BGP is often longer than the shortest path in the AS topology [13]. As we show later, AS path lengths can be stored much more compactly and are significantly more stable compared to AS paths. Third, we store AS three-tuples as described below to represent routing policies being employed by ASes.

Given this AS-level topology and an estimate L for the AS path length between a source S and destination D, we put together the set of ASes through which traffic may be routed from S to D as comprising any AS that is on any policy-compliant route of L AS hops between S and D in the topology. Here, we stress on policy-compliance because every path in the AS-level topology does not conform to routing policies of ASes. Therefore, to ensure that we only consider the ASes on policy-compliant paths, we borrow the technique of using AS three-tuples from iPlane Nano [13]. From a collection of AS path measurements—obtained from BGP feeds [27], [28] and by mapping traceroute measurements [34], [12] to AS paths—we identify every sequence of three consecutive ASes seen on any AS path and add them to a set of AS three-tuples. For example, if we observe an AS path $AS1 \rightarrow AS2 \rightarrow AS3 \rightarrow AS4 \rightarrow AS5$, then we add $(AS1, AS2, AS3)$, $(AS2, AS3, AS4)$, and $(AS3, AS4, AS5)$ to our set of AS three-tuples. Any such AS three-tuple (A, B, C) represents routing policy by showing that B is willing to transit traffic from A on to C (in other words, B passes along route announcements received from C on to A). We generated such a set of AS three-tuples by aggregating various BGP feeds, and we are able to represent this data in about 1 MB. Note that though Internet routing can be asymmetric in practice, i.e., the route from S to D can differ from the route from D to S, we assume routing asymmetry here and add the three-tuple (C, B, A) to our set of three-tuples for every tuple (A, B, C) discovered from the AS path measurements.

Given an estimate L for the AS path length between a pair of IP addresses S and D, we estimate the set of ASes that are likely to occur on the the route between them using the following two phase algorithm. In the first phase, we run Dijkstra's shortest path algorithm to compute the length of the shortest path from every AS to D's AS. We modify the standard Dijkstra's algorithm to ensure that shortest path

Algorithm 1 Pseudocode of AS set estimation algorithm.

```
Inputs: AS graph G, AS three-tuples set T, source S, destination
    D, AS path length L
Shortest_Path(G, T, D)
Queue Q
List Node PossibleSet
List Node AS_set
S.hops = 0
Add S to Q
while Q is not empty do
    cur ← Q.pop
    cur.added ← 0
    Add cur to PossibleSet if cur ∉ PossibleSet
    for n ∈ cur.neighbors do
        Skip n if (cur.parent, cur, n) ∉ T
        Skip n if ∄ m ∈ n.neighbors such that m.pathLength +
        cur.hops + 2 = L
        if n has ancestor p with p.pathLength <
        p.parent.pathLength then
            Skip n if n.pathLength > cur.pathLength
        end if
        n.hops = cur.hops + 1
        Add n to Q
        cur.added += 1
    end for
    if cur.added = 0 then
        Decrement n.added for every ancestor n of cur
    end if
end while
for n ∈ PossibleSet do
    Add n to AS_set if n.added > 0
end for
return AS_set
```

lengths are computed only across those paths that satisfy the criterion that any three consecutive ASes on a path are in the set of AS three-tuples. Next, we determine for every AS in the topology, the set of path lengths to D available via any of the AS's neighbors.

In the second phase, we determine the output set of ASes by performing a modified breadth-first search (BFS) from S. While performing BFS, we traverse a neighbor B of an AS A that is k hops away from S only if B has a path of length $(L-k-1)$ available to D via one of its neighbors. In addition, we enforce the valley-free nature of Internet routes [35] by ensuring that once the BFS goes from a node A to a neighbor B that has a shorter shortest path to D than from A, thereafter, we never traverse a node's neighbor that has a longer shortest path to D than from that node. Furthermore, we again ensure that the input AS three-tuples are respected; we traverse a neighbor B of A, whose parent in the BFS is C, only if (C, B, A) is in the input set of AS three-tuples. Algorithm 1—which takes as input the AS graph G, the set of AS three-tuples T, the source S, the destination D, and the estimated AS path length between them—summarizes the pseudocode of this algorithm.

B. Avoiding snooping ASes

When selecting a path from itself to a destination, a client needs to use the above procedure to determine AS sets for paths between itself and its 3 entry guards and between all exit relays and the destination. For the latter set of paths, we do not compute the AS sets independently. Instead, we run the first phase of our AS set estimation algorithm once, and thereafter run the BFS in phase two of the algorithm from each exit relay independently. We can then ignore from consideration all paths that potentially have snooping ASes on them by ignoring those combinations of entry and exit relays for which the intersection between the AS sets for the (client, entry relay) and (exit relay, destination) paths is non-empty. This algorithm can prune out paths with snooping ASes in around 3 seconds, even when choosing from 1000 exit relays.

Other than being efficient in terms of computation, our approach also minimizes the data to be downloaded by a client to make local inference of AS sets. First, the set of inter-AS links and the set of AS three-tuples are each roughly about 1 MB in size and changes to these datasets are rare. Second, all Tor relays and all end-hosts on the Internet can be grouped into roughly 600 and 50K BGP atoms [25], [12], respectively. Therefore, we need every client to download AS path lengths for 30M paths—between every (relay, end host) pair.

We evaluate the expected size to store these AS path lengths and the stability of this data using traceroutes gathered daily by iPlane [36] from all PlanetLab nodes to all IP address prefixes at the edge of the Internet. We analyze this data for the period of three weeks in July 2011. On each day, we map all traceroutes to their corresponding AS-level routes and compute the AS path length, i.e., the number of ASes seen on the route. First, we find that less than 0.05% of paths traverse more than 8 AS hops. So, every AS path length can be stored in 3 bits,

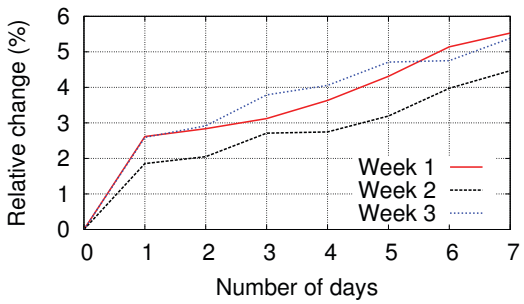


Fig. 10. Relative changes in AS path length data across days.

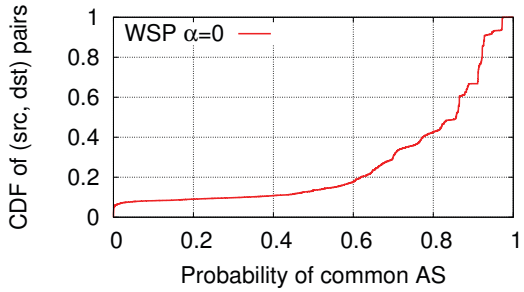


Fig. 12. The probability of existence of snooping ASes across (src, dst) pairs in the *PL-BGP-Rand* dataset.

making the size of the AS path length data to be downloaded initially by a client to be around 11 MB.

For each week in the considered period, we then compare AS path lengths on every day with those measured on the first day in that week. We perform the comparison by computing the fraction of paths that have a different AS path length on day i compared to that on day 0. As shown in Figure 10, AS path lengths changed on a little over 5% of paths even after a week. Therefore, in summary, our design requires clients to initially download 13 MB of data across inter-AS links, AS three-tuples, and AS path lengths—a close to 40x reduction in size compared to pre-computed AS paths between all Tor relays and all end-hosts—and a client need only fetch less than 1.5 MB weekly thereafter to keep the data up-to-date.

C. Evaluation of AS-awareness

Next, we evaluate our technique for AS set estimation in two parts. First, we examine if the estimated AS sets accurately cover actual AS paths. For this, we estimate AS sets for the paths from PlanetLab nodes to Tor relays in the *PL-Tor-Web* dataset. Fig. 11(a) and 11(b) show that the estimated AS sets are typically compact—90^{th} percentile size less than 10 ASes—and at most one AS on the actual AS path is not in the estimated set for over 75% of paths.

Second, we use the *PL-BGP-Rand* dataset to study the accuracy with which AS sets enable prediction of potential snooping ASes; we do not have AS paths from exit nodes to destinations in the *PL-Tor-Web* dataset, and the *PL-PL-Web* dataset is biased for this analysis [4]. For every (client, destination) pair in the *PL-BGP-Rand* dataset, we partition

[4]Paths between PlanetLab nodes typically traverse a different set of ASes, e.g., research and educational ASes, compared to paths from PlanetLab nodes to random destinations on the Internet

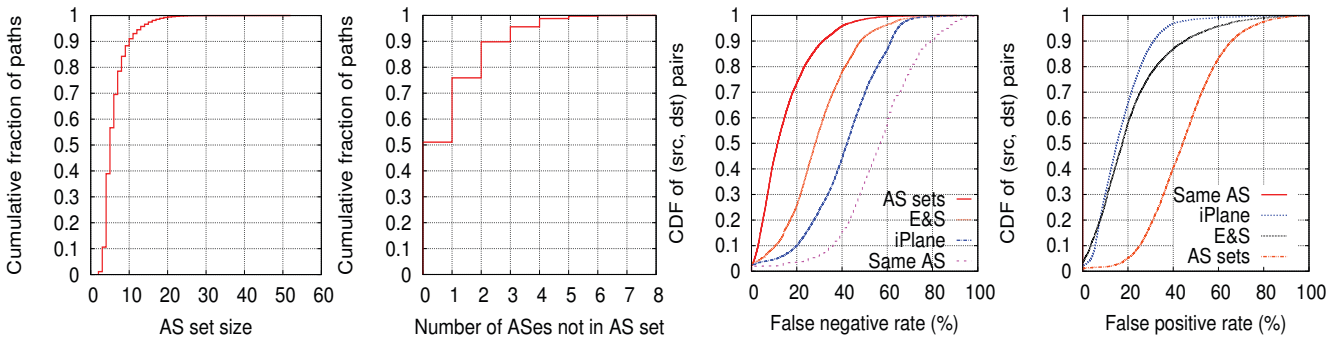


Fig. 11. (a) Distribution of predicted AS set sizes, (b) accuracy of predicted AS sets encompassing actual AS paths, and distribution of (c) false negative and (d) false positive rates in predicting the existence of snooping ASes.

all entry and exit relay combinations into those that have a common AS across the entry and exit segments and those that do not. We compute the false negative rate in predicting the presence of snooping ASes as the fraction of entries in the former partition not caught by our approach of computing intersections between estimated AS sets. Fig. 11(c) shows that our median false negative rate is 11%. This compares to median false negative rates of 28–57% with alternate approaches—using iPlane's predicted AS paths, using the approach proposed in [10] (the "*E&S*" line), or when only accounting for ASes of end-hosts and relays (the "*Same AS*" line). On the flip side, in Fig. 11(d), we see that AS sets produce a much greater false positive rate—fraction of paths that do not have a snooping AS but are declared as having one by our technique—compared to other approaches. However, as we see in Fig. 12, the fraction of paths with potential snooping ASes is low for most (src, dst) pairs. So, pruning out about 45% of candidate paths in the median case still leaves a sizeable set of paths from which WSP can choose.

D. Impact of AS-awareness on path latency

Finally, we evaluate the impact that the incorporation of AS-awareness has on path latencies obtained with WSP. WSP has to now select from a subset of all possible candidate paths, because it has to ignore those detected by our AS set estimation algorithm as potentially traversing an AS capable of inferring the (client, destination) pair by traffic correlation. Though the subset of candidate paths with snooping ASes is typically small in practice, the high false positive rate of our detection procedure significantly reduces the subset of paths considered. Therefore, we again use WSP (with α set to 0) to measure latencies over the Tor network from 50 PlanetLab nodes to the top 200 websites. Fig. 13(a) compares these latencies with those obtained when using WSP without AS-awareness and when using the default Tor client. We see that the pruning of paths to avoid snooping ASes results in a slight increase in latency. Fig. 13(b) shows that this increase in latency is due to an increase in the length of the chosen path when using WSP informed by AS sets. In future work, we plan to pursue a reduction in false positives to further improve latencies when using WSP with AS-awareness.

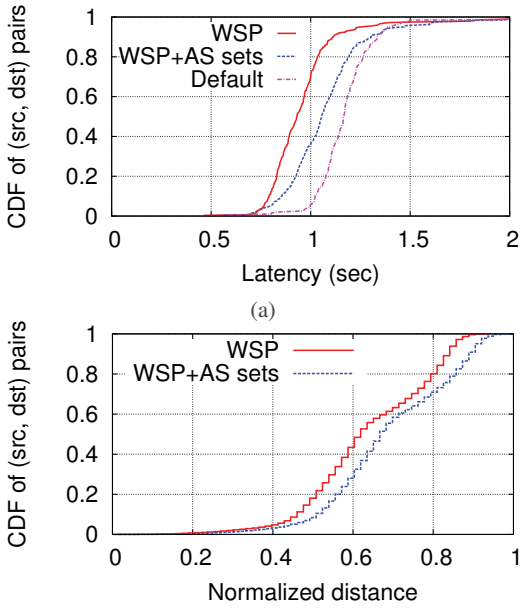


Fig. 13. Comparison of (a) latencies and (b) normalized geographical distance along paths chosen with WSP ($\alpha = 0$) with and without AS-awareness.

VI. Implementation

We implement all of the algorithms developed thus far—to improve path latency, to make path selection tunable, and to incorporate AS-awareness into path selection—in the *LASTor* Tor client. In this section, we summarize *LASTor*'s path selection algorithm and provide an overview of our implementation.

A. Client in action

In the default Tor client, the client sets up a few circuits on startup and thereafter, when the user chooses to communicate with a particular destination via Tor, the client routes the user's traffic over one of the established circuits [24]. *LASTor* mimics the default Tor client in this respect. In addition, once *LASTor* learns the destination that the user wishes to communicate with, it quickly selects a path using AS-aware WSP, sets up a new circuit along the chosen path, and then transitions the user's traffic to the destination to this new circuit. Thus, the

latency obtained with *LASTor* matches that of the default Tor client in the case when the user's communication with the destination is short. In the case when the user's interaction with the destination is prolonged, e.g., when the user visits several web pages on a website, *LASTor* significantly improves latencies for most of the user's interaction, i.e., once *LASTor* switches the user's traffic to the circuit chosen with WSP.

To select a path to the specified destination, *LASTor* executes the tunable AS-aware WSP algorithm with the following sequence of steps.

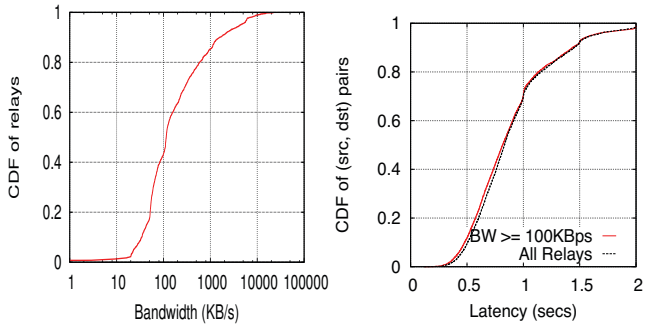
- Upon initialization, the *LASTor* client clusters all available relays, and using the value for α specified in its input configuration, it chooses three entry guards at random from the $(20 + \alpha \cdot 80)\%$ closest relay clusters to the client.
- When required to select a path to a destination, *LASTor* resolves the destination's hostname on a distributed set of nodes that service requests to perform DNS lookups. These requests are submitted via one of the circuits established upon initialization of the client.
- *LASTor* estimates the AS sets for the paths from the client to the entry guards and from all exit relays to the destination, mapping every candidate exit relay to the closest among the IP addresses obtained for the destination.
- *LASTor* then computes the end-to-end distance on every candidate path through three clusters that satisfy the check of the AS sets for the entry and exit segments being disjoint. One cluster-level path is then selected with the probability of a path being chosen dependent on the end-to-end distance on it and the input value of α.
- The circuit to the destination is then established via one relay selected at random from each of the clusters on the chosen cluster-level path.

B. Modification of default Tor client

We implement *LASTor* by building upon the default Tor client. We have implemented a Java application which connects to the default Tor client on its control port. This control port is a port on the Tor client which can be used to manage and monitor the Tor client based on a standard protocol [37]. By issuing commands to the control port, our Java application can either obtain information such as the description of all available relays, or manage the Tor client by establishing or closing a circuit, attaching streams to a circuit, and clearing Tor's DNS cache. To setup a circuit, our program first fetches relevant information through the Tor control port and provides this as input to our tunable path selection algorithm. It then issues commands to the Tor client, again via the control port, to build desired circuits. We implement *LASTor* to take as part of its input configuration 1) a value of α to guide path selection, and 2) a file with a list of nodes that provide the DNS lookup service.

C. Input datasets

To run the tunable AS-aware WSP path selection algorithm, our Java program needs several datasets as input. First, it fetches a IP geolocation database that maps IP addresses to locations from MaxMind [19]. Second, the first time it is executed, the program downloads 1) a AS-level representation of the Internet topology, 2) the set of AS three-tuples used to determine policy-compliant paths, and 3) a snapshot of AS path lengths for paths in either direction between all Tor relays and all end-hosts, grouped at the granularity of BGP atoms. We put together the first two datasets by aggregating AS paths from various sources [27], [28], [12], [34]. To estimate AS path lengths, we issue queries to iPlane [38]. We find that iPlane can process roughly 1000 queries per second, and so, we can re-query iPlane every day for all 60 million IP pairs (600 BGP atoms with Tor relays $\times$ 50K BGP atoms comprising all end-hosts, in either direction) for which we need AS path length information. As mentioned before, all three datasets can be stored in less than 13 MB in size. Since these datasets are the same across all clients and the information of a client having downloaded this data does not hamper its anonymity, clients can download this data from each other via a peer-to-peer file distribution system such as BitTorrent, so as to not overwhelm the bandwidth requirements of any central server. Bandwidth-constrained clients can however download relevant subsets of this data from the central server, e.g., only AS path length information necessary for communication with popular websites. Lastly, every week, the client downloads a roughly 1.5 MB update for AS path length information, and more infrequently, fetches updates for the set of inter-AS links and AS three-tuples. These updates are fetched from a central server since the update depends on the version of the data already on the client. For all datasets required by *LASTor*, we can enable clients to verify integrity of the data they download using an approach similar to that used to guarantee integrity of the default Tor client—by posting a cryptographic hash of the dataset on the Tor website.



(a) (b)

Fig. 14. (a) Distribution of bandwidth across Tor relays, and (b) comparison of end-to-end latencies with and without taking relay bandwidth into account; median latency across 5 paths are shown.

VII. Discussion

In this section, we discuss the extensions to Tor necessary to further reduce latencies and the impact on load balancing if *LASTor* is widely adopted.

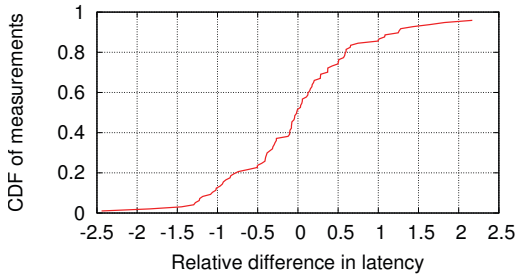


Fig. 15. Variation across time of latencies on paths spanning similar end-to-end geographical distances.

A. Accounting for dynamic load

Though we showed that WSP can significantly reduce latencies for communication on Tor, there remains a significant overhead compared to communication over the default Internet path. Therefore, to reduce latencies further, other than reducing propagation delays with the use of the WSP path selection algorithm, it is necessary to minimize queueing delays by taking into account the load at each relay at the time of path selection. Here, we present some preliminary results from our efforts to do so.

First, we observe that access link bandwidths of Tor relays are spread over a wide range, as shown in Figure 14(a). Therefore, we investigate the potential for reducing queueing delays by restricting the choice of relays among those with high bandwidth. To study this, we measure path latencies on the Tor network when visiting the top 200 websites from 50 PlanetLab nodes in two cases. We measure latencies first when choosing relays at random from those which have bandwidth greater than 100 KBps, and then repeat the same choosing from all Tor relays. To keep propagation delay similar in both settings, for every path that we pick from relays with bandwidth greater than 100 KBps, we pick a corresponding path with the entry, middle, and exit nodes in the same locations, but with no restriction on relay bandwidth. For either path selection strategy, we measure latencies between every (client, destination) pair on five different paths. The lines "*All Relays*" and "*BW* $\geq$ *100 KBps*" in Fig. 14(b) show that the distribution of median latency (across the 5 chosen paths) is identical whether we account for relay bandwidth or not. In this case, we use the "Estimated" bandwidth estimate for each relay—the value used by the default Tor client to perform path selection—but we found the results to be similar when using other estimates of relay access link bandwidth provided by the Tor directory.

Next, we studied the variation in latencies over time on a given path. We selected 20 (client, destination) pairs at random, and for each of them, we considered two different disjoint paths with the same end-to-end geographical distance; either path traversed three Tor relays. For each (client, destination) pair, we measured latencies once every half hour on either path selected for it and noted the relative difference between latencies measured on the first and second path; we randomly order the two paths chosen for every (client, destination) pair and fix that ordering across all measurement rounds. Fig. 15 shows the variation of this difference in measured latencies across the period of a day. We see that, though the pair of paths selected for every (client, destination) pair span identical geographical distances, the path that provides better latencies significantly varies over time.

Therefore, these results seem to indicate that we can reduce queueing delays only by modifying relays—either by having them track and report load at finer granularities of time or by introducing a new queue management algorithm at relays—which is outside the scope of our goal of enabling immediate latency improvements for Tor clients. Given the current implementation of Tor relays, biasing relay selection based on their bandwidth may help improve throughput, but this will not improve latencies for interactive transfers.

B. Load balancing

When choosing a path, the default Tor client currently selects relays with a probability proportional to their access link bandwidth. As a result, the fraction of all of Tor's traffic that traverses any particular relay is roughly proportional to that relay's access link bandwidth, thus balancing the load across relays.

In contrast, load across Tor relays could be significantly skewed if *LASTor* were widely used. If most users choose to use *LASTor* with a value close to 0 for α, paths chosen by each client will be biased towards traversing relays that result in lower end-to-end distances to the destinations with which the client communicates. On the other hand, even if all users use *LASTor* with a value of 1 for α, the consequent selection of relays at random will result in an equal distribution of load across relays, which is undesirable given the significant skew in access link bandwidths across relays (seen earlier in Fig. 14(a)).

Though addressing this issue requires further investigation outside the scope of this work, we present two recommendations that we speculate would enable widespread use of *LASTor* without harming the balance of load across Tor relays. First, we recommend that Tor users who use the network for bulk transfers, such as BitTorrent, should continue to use the default Tor client. Since bulk transfers account for a majority of the traffic on Tor [1], the use of the default Tor client for such traffic will ensure a distribution of load across relays that is reasonably close to the distribution of their access link bandwidths. The loss of anonymity due to protocol-specific path selection requires further investigation. Second, *LASTor*'s path selection algorithm itself will need to be modified to take the access link bandwidths of relays into account. However, to do so, we will need to discover the distribution of the value of α used by Tor users who use the *LASTor* client. Discovering this distribution should be possible by means of an anonymous survey across users. *LASTor*'s path selection algorithm can then be tweaked to not simply have a preference for paths with a lower end-to-end distance but to also account for the access link bandwidths of relays and the distribution of α across users.

VIII. Related Work

We build upon three lines of prior work—1) improving performance in Tor, 2) improving anonymity with Tor, and 3) AS path inference. We discuss related efforts in these areas.

Improving performance in Tor. To improve performance on Tor, Sherr et al. [6], [39] proposed a path selection algorithm based on the concept of link-based relay selection. In this approach, a client computes a cost for each path by aggregating values for the chosen metric (e.g., latency, bandwidth) across segments on the path, and then picks a path with probability based on this cost. With the aid of simulations, they showed that their approach offers better performance on each of the objective functions mentioned above. In order to obtain these performance benefits, they discuss relays disseminating information among themselves using, for example, a network coordinate system. However, modifying relays to build such a distributed system for performing measurements and then disseminating this information is not a trivial task. Therefore, we focus on latency benefits possible without any modification to relays. Furthermore, to evaluate the anonymity of their approach, Sherr et al. count the number of traversed ASes on the path and consider the traversal of a lower number of ASes to provide better anonymity. Instead, we explicitly detect common ASes on the entry and exit segments of a path and avoid such paths.

Panchenko et al. [7] propose two algorithms to improve the performance on Tor. First, to reduce latency, they measure the latency between every pair of relays and choose a path with a probability related to the end-to-end latency on that path. Second, to help throughput-oriented applications, they perform passive measurements to infer the available bandwidth on each relay and pick a path based on the expected end-to-end throughput. However, again, modifications to all Tor relays are necessary to implement these approaches. Also, since most connections on Tor correspond to interactive traffic [1], we focus only on reducing latency and show how to do so with only client-side modifications.

The authors of [40] studied the influence of geographical diversity on the performance of Tor and found a tradeoff between improved performance and anonymity. They found that though low diversity of relays may lower the latencies in setting up circuits, greater geographical diversity of nodes is an important factor to provide strong anonymity guarantees. We similarly illustrate the loss in anonymity when preferring low latency paths, but make path selection tunable to enable latency benefits to be overridden for better anonymity, when desired.

Snader and Borisov [5] showed how a client can trade off between performance and anonymity when selecting paths. However, Snader and Borisov focused on improving throughput on Tor (their evaluation revolved around the download of a 1 MB file), while we focus on latency. We showed that the selection of lower latency paths warrants the need for several techniques not necessary when optimizing throughput, such as the careful selection of entry guards and accounting for destinations that are geographically distributed. DefenestraTor [8] improves latencies in Tor by modifying traffic management in Tor relays to reduce congestion-related queueing delays. We pursue a complementary approach that reduces propagation delays without any modifications necessary to Tor relays.

AS-awareness in path selection. In 2004, Feamster and Dingledine [9] studied the Tor network to investigate the problem of an AS eavesdropping both ends of a circuit. First, they showed that there are Tor relays with different IP addresses that are in the same AS, and that Tor clients should avoid selecting two relays from the same AS. Second, they discovered that the probability of an AS observing both ends of a circuit varies between 10% and 30% across (client, destination) pairs. To reduce this probability, they proposed the passive monitoring of BGP feeds to determine AS paths. However, they did not elaborate on how clients should fetch and maintain up-to-date information from BGP routing tables. Instead, motivated by their observation, we make AS-aware path selection practical by reducing both time and space complexity.

Later, in 2009, Edman and Syverson [10] showed that although the number of Tor relays increased significantly since Feamster and Dingledine's analysis, the probability of an AS being able to observe both ends of a connection did not decrease much. To protect against occurrences of snooping ASes, the authors suggest that all Tor server authorities agree upon a snapshot of ASes based on Routing Information Bases (RIB). Client can then use AS topology snapshots to select a path in which AS-level routes from the client to the entry node and from the exit node to the destination span a disjoint set of ASes. As we showed in our evaluation, our approach of using AS sets significantly reduces the rate of missing snooping ASes compared to that proposed by Edman and Syverson.

AS path inference. Several systems and algorithms have been developed for inference of AS paths between arbitrary IP addresses on the Internet. Approaches for this can be broadly classified into two classes. One set of approaches [12], [32], [33] enable computationally efficient estimation of AS paths but use a large corpus of path measurements as input. Such approaches are ideal for hosting services that can be queried for AS path inferences, but this is not an option in the case of Tor since the queries for AS paths can leak client anonymity. The second set of approaches [11], [13] for AS path inference require much lesser data as input, e.g., only the Internet's PoP-level or AS-level topology, but are computationally prohibitive in processing queries. The use of such techniques to select paths that avoid snooping ASes will render the selection of low latency paths moot. Given these shortcomings of prior approaches for AS path inference, we develop a new technique that *both* has low runtimes and requires compact inputs.

Other related work. Several measurement studies [41], [42], [1] of the Tor network have been performed to determine the location diversity of Tor users and the popularity of different kinds of traffic such as HTTP, BitTorrent, and E-mail. These studies have shown that though HTTP transfers account for a small fraction of the traffic on Tor, they constitute a large

majority of connections. Hence, for most Tor users, latency is more important than throughput. To the best of our knowledge, we are the first to show how to improve latencies on Tor in a practical manner with only client-side modifications.

Hopper et al. [21] studied the loss in a client's anonymity by knowing the latency on the circuit in use by the client. While complementary to our effort, this study needs to be revisited in the light of our tunable AS-aware WSP path selection algorithm. We speculate that the knowledge that a client is using WSP to choose paths probably leaks more information about the client when path latency is known.

IX. Conclusions and Future Work

Though Tor is the most widely used anonymity network today for low latency anonymous communication, poor latencies on it and the fear of traffic correlation attacks by underlying ASes are the biggest problems with Tor's usability today. Prior proposals have either focused on improving the performance on Tor in terms of throughput, which does not help interactive communication, or they mandate significant modifications to Tor relays, which places the onus on developers and thus are yet to be deployed.

In this paper, we developed a new Tor client, called *LASTor*, to demonstrate that both significant latency gains and protection against snooping ASes can be obtained on Tor *today* without requiring any modifications to Tor relays. Based on measurements along paths between 10K (client, destination) pairs, we showed that *LASTor* can deliver a 25% reduction in median path latency. To deliver these latency benefits, we showed that it is important to carefully select entry guards and account for replicated destinations. We also developed a space- and time-efficient technique for enabling *LASTor* to reliably detect the possible presence of snooping ASes on any path. Moreover, we have made path selection in *LASTor* tunable so that a user can easily choose an appropriate trade-off between latency and anonymity.

We plan to make *LASTor* available for public use. We are also investigating the use of latency estimation approaches [13], [12] that do not require measurements from relays to further improve latencies on Tor without necessitating modifications to relays.

References

[1] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in *PETS*, 2008.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*, 2004.

[3] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *IEEE S&P*, 2003.

[4] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman, "IETF draft: Mixmaster protocol version 2," http://www.ietf.org/internet-drafts/draft-sassaman-mixmaster-03.txt, 2005.

[5] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *NDSS*, 2008.

[6] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *PETS*, 2009.

[7] A. Panchenko and J. Renner, "Path selection metrics for performance-improved onion routing," in *SAINT*, 2009.

[8] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. M. Voelker, "DefenestraTor: Throwing out windows in Tor," in *PETS*, 2011.

[9] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *WPES*, 2004.

[10] M. Edman and P. F. Syverson, "AS-awareness in Tor path selection," in *CCS*, 2009.

[11] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang, "On AS-level path inference," in *SIGMETRICS*, 2005.

[12] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *OSDI*, 2006.

[13] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane Nano: Path prediction for peer-to-peer applications," in *NSDI*, 2009.

[14] "The Tor Project, Inc." http://www.torproject.org.

[15] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE JSAC*, 1998.

[16] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," in *IEEE S&P*, 2003.

[17] "Quantcast," http://www.quantcast.com/top-sites-1.

[18] "PlanetLab," http://www.planet-lab.org.

[19] "MaxMind - GeoLite City," http://www.maxmind.com/app/geolitecity.

[20] M. Edman and B. Yener, "On anonymity in an electronic society: A survey of anonymous communication systems," *ACM Computing Surveys*, 2009.

[21] N. Hopper, E. Y. Vasserman, and E. Chan-TIN, "How much anonymity does network latency leak?" *TISSEC*, 2010.

[22] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," in *PETS*, 2004.

[23] S. J. Murdoch and P. Zieliski, "Sampled traffic analysis by internet-exchange-level adversaries," in *PETS*, 2007.

[24] "Tor path specification," https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt, 2011.

[25] A. Broido and kc claffy, "Analysis of RouteViews BGP data: Policy atoms," in *Network Resource Data Management Workshop*, 2001.

[26] "Tor metrics portal: Users," https://metrics.torproject.org/users.html.

[27] D. Meyer, "RouteViews," http://www.routeviews.org.

[28] "RIPE Routing Information Service," http://www.ripe.net/ris/.

[29] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize CDN performance," in *IMC*, 2009.

[30] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *CCS*, 2011.

[31] C. Gini, "Measurement of inequality of incomes," *The Economic Journal*, 1921.

[32] D. Lee, K. Jang, C. Lee, G. Iannaccone, and S. Moon, "Scalable and systematic Internet-wide path and delay estimation from existing measurements," *Computer Networks*, 2011.

[33] J. Qiu and L. Gao, "AS path inference by exploiting known AS paths," in *GLOBECOM*, 2006.

[34] "Archipelago measurement infrastructure," http://www.caida.org/projects/ark/.

[35] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM ToN*, 2001.

[36] "iPlane: Datasets," http://iplane.cs.washington.edu/data/data.html.

[37] "The Tor directory protocol," https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=dir-spec.txt, 2011.

[38] "iPlane: Measurements and query interface," http://iplane.cs.washington.edu/iplane_interface.pdf.

[39] M. Sherr, B. Thau, and L. M. Blaze, "Towards application-aware anonymous routing," in *HotSec*, 2007.

[40] A. Panchenko, L. Pimenidis, and J. Renner, "Performance analysis of anonymous communication channels provided by Tor," *International Conference on Availability, Reliability and Security*, 2008.

[41] S. Le-Blond, P. Manils, C. Abdelberi, M. A. Kâafar, C. Castelluccia, A. Legout, and W. Dabbous, "One bad apple spoils the bunch: Exploiting P2P applications to trace and profile tor users," *CoRR*, 2011.

[42] K. Loesing, S. J. Murdoch, and R. Dingledine, "A case study on measuring statistical data in the Tor anonymity network," in *Workshop on Ethics in Computer Security Research (WECSR)*, 2010.