CLAPS: Client-Location-Aware Path Selection in Tor

Florentin Rochet*
UCLouvain
florentin.rochet@uclouvain.be

Ryan Wails
U.S. Naval Research Laboratory
ryan.wails@nrl.navy.mil

Aaron Johnson
U.S. Naval Research Laboratory
aaron.m.johnson@nrl.navy.mil

Prateek Mittal
Princeton University
pmittal@princeton.edu

Olivier Pereira
UCLouvain
olivier.pereira@uclouvain.be

ABSTRACT

Much research has investigated improving the security and performance of Tor by having Tor clients choose paths through the network in a way that depends on the client's *location*. However, this approach has been demonstrated to lead to serious deanonymization attacks. Moreover, we show how in some scenarios it can lead to significant performance degradation. For example, we demonstrate that using the recently-proposed Counter-RAPTOR [38] system when guard bandwidth isn't abundant could increase median download times by 28.7%. We propose the CLAPS system for performing client-location-aware path selection, which fixes the known security and performance issues of existing designs. We experimentally compare the security and performance of CLAPS to Counter-RAPTOR and DeNASA [5]. CLAPS puts a strict bound on the leakage of information about the client's location, where the other systems could completely reveal it after just a few connections. It also guarantees a limit on the advantage that an adversary can obtain by strategic relay placement, which we demonstrate to be overwhelming against the other systems. Finally, due to a powerful formalization of path selection as an optimization problem, CLAPS is approaching or even *exceeding* the original goals of algorithms to which it is applied, while solving their known deficiencies.

KEYWORDS

Tor; onion routing; anonymity

ACM Reference Format:
Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. 2020. CLAPS: Client-Location-Aware Path Selection in Tor. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3372297.3417279

1 INTRODUCTION

Tor enables anonymous communications on the Internet by offering a network of relays through which users can route their TCP traffic. The most effective way of defeating the anonymity goal of Tor is to run a traffic correlation attack [19, 25, 26, 32], in which an adversary is able to observe a connection between a user and its *guard* (i.e., the first relay it uses) and to detect that same connection between the destination and the *exit* (i.e., the last relay). This attack is successful despite Tor's use of encryption, as the low latency of Tor, which is a core element of the design, makes it effective to simply time sequences of packets and correlate those timings.

As a result, Tor's relay selection strategy is critical to limit the probability that an adversary could observe both the user-guard traffic and the destination-exit traffic. One common approach to improving relay selection is to consider the client's *location* when choosing relays [1, 4, 5, 10, 13, 18, 27, 31, 38], which allows a client to avoid sending traffic over distant or dangerous paths. However, recent work has shown two major security vulnerabilities of this general approach: (1) it potentially leaks unbounded amounts of information about the client's location to an adversary that can make and link partial observations of a user's paths [18, 46], and (2) it allows an adversary to strategically place relays in locations that yield a higher chance of being selected by a user [47].

To add to these security problems, we show how existing proposals share a performance problem caused by not fully considering the impact that location-awareness can have on Tor's load balancing. Tor currently computes relay weights that bias relay selection so that relays expect a load proportional to their capacity. All location-aware path-selection proposals effectively change these weights such that load balancing is no longer guaranteed, which can lead to some relays being overloaded (e.g., by being located in a location preferred by users). We demonstrate this problem in the Counter-RAPTOR [38] system, where our experiments show that the good performance reported for Counter-RAPTOR depends on a fortuitous distribution of relays and users across locations. For example, when guard bandwidth is not abundant, (a scenario that has happened many times over Tor's existence), we show that downloads of moderate size (2 MiB) increase for the median client from 9.43s in current Tor to 12.14s.

In order to address the performance issues of previous works and to solve their known security issues, we introduce CLAPS, a generic framework for the design of client-location-aware path-selection algorithms in Tor. CLAPS makes it possible to optimize path selection to achieve a primary location-aware goal, while still satisfying other critical security and performance criteria. CLAPS uses a powerful linear-programming framework that can yield a solution improving on prior work even with respect to that work's own primary location-aware goal. Moreover, CLAPS introduces several new tools for path selection, including (1) a method for location-aware load balancing; (2) a generic technique to prevent location-aware schemes from leaking user locations to a long-term adversary; and (3) a new method to bound the risk of relay-placement attacks [47].

[*] Florentin Rochet and Ryan Wails share first authorship.

CCS '20, November 9–13, 2020, Virtual Event, USA
2020. ACM ISBN 978-1-4503-7089-9/20/11.
https://doi.org/10.1145/3372297.3417279

In order to demonstrate the effectiveness of CLAPS, we apply it to two recent proposals: Counter-RAPTOR [38] and DeNASA [5]. Through experiments and analysis, we show that CLAPS leads to improvements in *both* performance and security. Detailed Tor simulations show that CLAPS *eliminates* the losses of performance these prior systems experience in realistic network scenarios. Our analysis shows that CLAPS can be configured to achieve any desired limit on the amount of information leaked about the client's location, setting up a trade-off between this leakage and the main location-aware goal. CLAPS similarly guarantees a configurable maximum advantage from maliciously placing relays. Our analysis shows that an adversary can obtain up to a 7× advantage in Counter-RAPTOR and a 40× advantage in DeNASA, while we configure CLAPS with a maximum advantage of just 2×.

We achieve these improvements while targeting the goals of the original algorithm to which CLAPS is applied. Compared to Counter-RAPTOR, CLAPS *increases* the median expected resilience by 18%. When applied to DeNASA, CLAPS decreases the median expected number of "Suspect ASes" (i.e., certain large and highly-connected ASes) able to use traffic correlation to deanonymize the client by a factor 2.3 compared to Vanilla Tor, where DeNASA reduced the number by a factor 7.

Finally, we discovered performance and security problems in the current entry guard design. We wrote a Tor proposal [30], further discussed in Section 6, implemented it, and our code has been released in Tor-0.4.4.1-alpha.

2 BACKGROUND AND MOTIVATION

The Tor network is comprised of over 6,000 relays and 9 Directory Authorities, distributed around the world and run by individual volunteers. The relays mainly forward user traffic. The Directory Authorities establish an hourly network *consensus*, which is a document containing the authoritative list of all relays in the network. All relays and clients in the network maintain a copy of the current consensus. To create an anonymous connection, a Tor client first chooses a *path*, which consists of a sequence of three relays: *guard*, *middle*, and *exit*. Then the client constructs a cryptographic *circuit* over that path, encrypting messages once for each hop on the path such that each one only can identify the previous and next hop.

In Tor's current path-selection algorithm, which we call *Vanilla Tor*, a client chooses relays for a path on the basis of per-relay *flags* and *weights* determined from the consensus. The client will only choose for the guard position those relays with the Guard flag, which indicates that the relay has suitable bandwidth and stability to serve as a guard. The client also will only choose for the exit position a relay that allows connections to be made to the destination port and IP, which for many purposes, Tor approximates with the Exit flag, indicating that the relay allows connections to common Web ports (80 and 443). For the middle position, any relay might be selected depending on a positional factor computed by the Directory Authorities. The client chooses relays based on their consensus weights, which are Tor's estimates for the relay's bandwidth and enables overall load balancing. To choose a relay for position p, the client scales the consensus weights by a positional factor that depends on p and the relays' flags, yielding a vanilla weight v_r^p, and then selects relay r with probability $v_r^p / \sum_s v_s^p$. The positional factors are determined to maximize network throughput by maximizing the minimum total vanilla weight in any position. Making selection probabilities proportional to bandwidth balances the traffic load such that all relays in the most bandwidth-constrained positions are expected to use the same fraction of their bandwidth. We will design CLAPS to obtain the same load-balancing property.

2.1 Selected Algorithms

In our evaluation (Section 4), we will apply CLAPS to two proposed path-selection algorithms: Counter-RAPTOR [38] and DeNASA [5]. These algorithms are designed to defend against different threats, and we briefly describe them.

Counter-RAPTOR [38]. This algorithm's goal is to reduce the likelihood to select a guard that would be vulnerable to a BGP hijack/interception attack. For a client in the ith Autonomous System (AS) and the jth guard, it uses a pre-computed *resilience* value $R_{ij} \in [0, 1]$ that is the fraction of Internet ASes that cannot perform a same-prefix BGP hijack or interception on the guard (for details, see p. 3–5 of [38]). To provide some load balancing, the resilience is combined with the guard's bandwidth B_j. B_j is normalized to $\bar{B}_j = B_j / \max_k B_k$. Then the weight of the guard is computed as $w_j = \alpha R_{ij} + (1 - \alpha)\bar{B}(j)$, where $\alpha \in [0, 1]$ is a blending parameter ($\alpha = 0.5$ is suggested). The selection probability is derived from the weights of all guards: $w_j / \sum_k w_j$. Selecting middle relays and exit relays are the same as in compared to Vanilla Tor.

DeNASA [5]. This algorithm's goal is to defend Tor users against passive end-to-end correlation attacks by an adversary that compromises some Autonomous Systems (ASes). DeNASA only attempts to avoid observation by certain "Suspect Ases", which are the k most common ASes on paths into and out of the Tor network. Qiu and Gao AS-path inference [29] is used to determine the ASes between clients and guards and between exits and destinations. DeNASA is "destination-naive" on the exit side, and for each exit the system estimates the likelihood that a given Suspect AS will be between it and the destination, where common Internet destinations are used for this estimation. DeNASA defines a strategy for guard selection called *g-select* and a strategy for exit selection called *e-select*. In *g-select*, the client repeatedly applies Vanilla Tor guard selection until none of the top $k = 2$ Suspect ASes appear between the client and guard. In *e-select*, after selecting a guard, the client applies Vanilla Tor exit selection until an exit is selected such that for all top $k = 8$ Suspect ASes the exit has a score of less than a parameter τ (the authors suggest $\tau = 0.1$).

2.2 Vulnerabilities in Path Selection

Client-location Leakage. Location-aware path-selection algorithms choose a path based on the client location. Thus, simply observing which relays a client selects can leak information about the client's location, reducing its anonymity. Prior work has demonstrated that this leakage worsens when the adversary is able to partially observe *multiple* circuits and identify when they are created by the same user [13, 18, 46]. These attacks can be efficient: in Astoria [27] (a Tor path-selection algorithm designed to avoid AS-level traffic correlation), an adversary running destination servers and some relays can learn over 6 bits of entropy about the client's origin AS [18]; against DeNASA, an adversary running relays can learn over 14 bits of entropy about the client's AS after observing just 2 guard selections [46]; and against Counter-RAPTOR, an adversary running relays can learn over 5 bits of entropy about the client's AS within 10 guard selections [13].

Guard Placement Attacks. In the guard placement attack [5, 31, 47], the adversary places guard relays into the Tor network in positions that maximize their selection probability. Location-aware algorithms are vulnerable to this attack, as clients prefer relays in certain advantageous locations. Obtaining the guard relay enables powerful deanonymization attacks such as website fingerprinting [14, 36, 49] and end-to-end correlation [19]. Wan *et al.* [47] demonstrate this attack on several algorithms: against LASTor [1] (a path-selection algorithm to reduce latency), an adversary can increase a guard's selection probability by as much as 158× its Vanilla Tor probability; against DeNASA, the attack can increase selection probability by up to 964×; and against Counter-RAPTOR, an adversary can increase a guard's selection probability by over 13.6×. They then propose the θ-GP secure security definition, which requires that an adversary's selection probability is increased by at most a factor θ over Vanilla Tor (or relative to some abstract "cost"). In our work, we consider and defend against a generalized version of this attack that applies to all relays, not just guards. We will require that relay selection in each position satisfies the θ-GP secure notion, which we enforce via a bound on selection probability of θ times the vanilla probability.

2.3 Performance Challenges in Path Selections

We observe that all existing location-aware path selection algorithms disrupt the load balancing that Tor currently provides. Tor carefully chooses positional and relay weights to ensure that network throughput is maximized and that relays of a given type all use the same fraction of their bandwidth in a given position. However, location-aware path-selection algorithms all use the locations of clients and relays as another relay-weighting criterion, and the re-weighting does not take into consideration how many clients are coming from each location. Therefore, if it occurs that there are a large number of clients from a given location and relatively little relay capacity in locations they prefer, the preferred relays will be overloaded, slowing down those clients and making overall network throughput sub-optimal.

As a demonstration of this problem, we show via simulation that this problem indeed exists in Counter-RAPTOR [38] (see Figure 11 in Appendix E.1). A reason that this problem has not previously been acknowledged is that prior work experimentally evaluated performance using the current distribution of relays and clients as well as current load levels. While it may be that Tor performance at the moment would not be impacted by these algorithms, that could change at any time as client demand and relay supply change. Moreover, the problem may be hidden due to a current excess of relay bandwidth [44], a situation that could also change as client demand continues to grow. Our experiments shows that load imbalance can cause reduced performance in Counter-RAPTOR when the Tor network changes, specifically when demand increases in a given location or when relay supply reduces in a given position. Another example of the importance of considering multiple network loads is offered by incentive research for anonymous communication, in which a change of the network load has made state-of-the-art prioritization techniques inefficient [8].

We argue that path-selection algorithms should be guaranteed to provide good performance regardless of the network state, as both will inevitably and unpredictably change. A path-selection algorithm should be *versatile* to network changes and always reach a state in which overall throughput is maximized and all relays in the most-constrained positions are using the same fraction of their bandwidth capacity.

Vanilla Tor's strategy is versatile and modifies in which position some relays are used upon network state changes [9]. In our design, we show how to incorporate performance versatility into location-aware path selection goal.

3 DESIGN

As we have described, existing proposals for location-aware path selection do not maintain the load balancing necessary for reasonable Tor performance. In addition, prior work has demonstrated that these proposals in general suffer from two severe security vulnerabilities: (1) each path selection leaks more information about the client's location, and the adversary eventually learns the client's location [46]; and (2) malicious relays can be adversarially placed to observe a disproportionately large fraction of network traffic [47].

We present the CLAPS system to solve all of these deficiencies. CLAPS includes two high-level, novel components: (1) a generic optimization framework that uses linear programs (LPs) to compute relay weights that balance multiple security and performance criteria, and (2) a location masking technique that guarantees only limited information leaks about a client's true location over time. These strategies can be used independently; however, CLAPS uses them both as they solve different and critical problems with location-aware path selection.

CLAPS can be viewed as a framework to improve existing systems for client-location-aware path selection. It solves the known problems with these systems while still achieving their original goals, thereby making them suitable candidates to replace Tor's existing path-selection algorithm. CLAPS is flexible, which we demonstrate by showing how it can be used to achieve the goals of two proposed systems: Counter-RAPTOR [38] and DeNASA [5].

We first give on overview of the design of CLAPS before discussing details of its key assumptions and components. Notation is introduced throughout and is also summarized in Appendix A.

3.1 Overview

CLAPS modifies Tor path selection by changing how relay weights are computed and used. Location-awareness is achieved by computing different sets of weights for different locations. Moreover, clients mask their locations when using these weights to prevent leaking their true location through path selections. Weights are thus only needed for a set of mask locations. As in Vanilla Tor, the Directory Authorities compute the weights and distribute them to relays. Similarly, the Directory Authorities produce the masking data and distribute them to the relays. Given a set of weights, a client uses them to select relays for a circuit and then constructs the circuit in the same way as in Vanilla Tor.

The Directory Authorities each compute weights and mask information on the basis of public data and the rest of the consensus data. Mask data is recomputed at a low frequency (e.g., every year) to limit information leakage across masks changes while allowing for some changes in the underlying location data. Weights are recomputed more frequently (e.g., every day) to accommodate churn among the Tor network relays.

3.2 Locations and Densities

We suppose that there exists a public set of locations $\mathcal{L}$. Depending on the path-selection goals, these locations could, for example, be the set of ASes on the Internet or a set of geographic regions. We assume that each client exists in one location, and that each client can determine its own location.

To make load balancing possible, we require that the Tor network measure the *density* of clients in each location. That is, the network must publish the relative amount of client traffic coming from each location in $\mathcal{L}$. As demonstrated in Sec. 2.3, without some information about how much client demand exists in each location, we cannot guarantee a load-balanced network while allowing path selection to depend on the client location. We do not prescribe a specific way to obtain these densities; instead, we observe that Tor Metrics [44] already provides daily estimates of total users per country and suggest that either these techniques or more sophisticated methods [23] can be used.

3.3 Weight Computation

CLAPS uses Tor's consensus weights as the key mechanism through which it achieves performance and security goals. Clients use weights to select relays for circuits by randomly choosing from among the relays with probabilities proportional to their weights. As in Tor currently, the Directory Authorities compute a new set of weights for each consensus (i.e., hourly).

CLAPS uses a novel optimization approach to compute these weights that takes into account multiple goals. It uses linear programs (LPs) to formulate the weight computation as an optimization problem. LPs provide a general framework for determining optimal weights. A single objective function allow multiple performance and security criteria to be balanced, while hard requirements can be expressed using optimization constraints.

The LPs determine the relay weights for each position: guard, middle, and exit. The positions are given some order, and then weights for each position are computed in that order. Weights are greedily optimized at each position to keep the LPs manageable in size and ensure the constraints and objective remain linear.

For a given position, a different set of weights may be produced for each combination of a client location and relay choices in the previous circuit positions. This design allows us in principle to produce an arbitrary path-selection distribution over circuits for every client location, although for efficiency we generally only consider a smaller number of locations and limited number of previous positions. For example, as we describe in Section 4, we adapt the DeNASA algorithm by computing a set of guard weights for each client location, a single set of middle weights, and a set of exit weights for each pair of client and guard location.

The optimization objective is to minimize the total *penalty* over all clients. Penalties are an abstraction of the soft goals of path selection, that is, goals that are not hard constraints. Penalty scores may indicate a target threat model (e.g., susceptability to a BGP hijack [38]) or a key performance goal (e.g., minimal latency [1]). Penalty scores for relays in a given position are provided as inputs to the optimization, where the scores may be different for each combination of a location and the relays in previous positions.

Requirements for path selection are specified as LP constraints. These requirements include those goals that take higher precedence than minimizing penalties. CLAPS uses constraints to ensure that (1) Tor remains load-balanced, (2) Tor is secure against relay placement attacks (i.e., is θ-GP secure [47]), and (3) no client obtains an expected penalty worse than it would under Vanilla Tor. These additional criteria allow us to minimize penalties without reducing Tor throughput, becoming vulnerable to malicious relays, or making Tor penalties worse for any individual client, respectively.

As a first step in ensuring load balancing, we use an initial LP, LP_1, that maximizes the minimum bandwidth across all positions. We will require that the final weights provide this bandwidth in each position to ensure maximum total throughput. Let $\mathcal{P} = \{\text{g}, \text{m}, \text{e}\}$ be the set of circuit positions (i.e., guard, middle, and exit). Let $\mathcal{R}$ be the set of all relays, and let $\mathcal{R}_p$ be the subset of relays that can be used in position $p \in \mathcal{P}$ (e.g., relays in $\mathcal{R}_\text{g}$ must have the Guard flag). Let B_r be the bandwidth of relay $r \in \mathcal{R}$. LP_1 outputs the maximum bandwidth β that can be provided in all $p \in \mathcal{P}$ simultaneously. LP_1 identifies a weight w_r^p for each relay r and position p such that $\sum_{p \in \mathcal{P}} w_r^p \leq B_r$ and $\sum_{r \in \mathcal{R}_p} w_r^p \geq \beta$. Note that LP_1 computes relay weights, but these are just used to determine β and are not be by clients. For details, see Figure 7 in Appendix A.

Next, we determine weights for each position in the same order π in which clients will choose relays for a circuit. We allow relay weights for a given position to depend on the client's location and the relays chosen in previous positions. Doing so enables algorithms such as choosing guards and exits that are in dissimilar locations to avoid network-level adversaries. Generally, guards should be selected independently of other positions (i.e., $\pi_1 = \text{g}$) because they are used long-term, but we do not require this (notably, Vanilla Tor uses order $\pi = (\text{e}, \text{g}, \text{m})$). To express the weights, we define an extended set of *positional locations* $\mathcal{L}_{\pi_i}$ for the ith position that contains a potentially different element for each client location and sequence of $(i-1)$ relays. The first position simply uses the client locations: $\mathcal{L}_{\pi_1} = \mathcal{L}$. For example, if the choice of exit should depend on both the client and guard location, $\mathcal{L}_\text{e}$ should contain all pairs of client and guard locations. For convenience, define $\Lambda(\ell, r) : \mathcal{L}_{\pi_i} \times \mathcal{R} \rightarrow \mathcal{L}_{\pi_{i+1}}$ as the map from the current positional location ℓ to the next one after choosing r for the ith position.

The optimization problem in each position depends on several external values. Let $\theta_p \geq 1$ be the maximum factor by which any relay's selection probability in position p can increase over Vanilla Tor. This value limits the susceptability to a relay-placement attack. θ_p should be set to the largest acceptable advantage factor for an adversary in position p. Let $D_\ell \geq 0$ be the client density in location $\ell \in \mathcal{L}$, with $\sum_\ell D_\ell = 1$. D_ℓ is obtained from network measurements. Let $P_{\ell r}^{\pi_i}$ be the penalty for choosing relay r in position π_i from $\ell \in \mathcal{L}_{\pi_i}$. Higher penalty values indicate worse relays to choose. Penalty values can be set to achieve differing performance or security goals, but they must depend on public data.

For convenience, we also define several values to help us compute weights for the ith position. These values depend on weights already

Minimize

$$\sum_{\ell \in \mathcal{L}_{\pi_i}} \sum_{r \in \mathcal{R}_{\pi_i}} D_\ell^{\pi_i} P_{\ell r}^{\pi_i} w_{\ell r}^{\pi_i} \tag{1}$$

Minimize the average penalty in position π_i.

Subject to

$$\forall j \geq i, \ell \in \mathcal{L}_{\pi_i}: \quad \sum_{r \in \mathcal{R}} w_{\ell r}^{\pi_j} = \beta \tag{2}$$

Set weight in each position to value β obtained from LP$_1$.

$$\forall j \geq i, \ell \in \mathcal{L}_{\pi_i}, r \in \mathcal{R}: \quad \frac{w_{\ell r}^{\pi_j}}{\beta} \leq \frac{\theta_{\pi_j} v_r^{\pi_j}}{\sum_{s \in \mathcal{R}} v_s^{\pi_j}} \tag{3}$$

Limit relay-placement advantage to θ_{π_j}.

$$\forall \ell \in \mathcal{L}_{\pi_1}: \quad \sum_{\ell' \in \mathcal{L}_{\pi_i}} \lambda_{\ell \ell'} \sum_{r \in \mathcal{R}} \frac{w_{\ell' r}^{\pi_i} P_{\ell' r}^{\pi_i}}{\beta} \leq V_\ell^{\pi_i} \tag{4}$$

Limit per-location average penalty to that of Vanilla Tor.

$$\forall r \in \mathcal{R}: \quad \sum_{j<i} \omega_r^{\pi_j} + \sum_{j \geq i} \sum_{\ell \in \mathcal{L}_{\pi_i}} \delta_\ell^{\pi_i} w_{\ell r}^{\pi_j} \leq B_r \tag{5}$$

Limit relay's total client weight to its bandwidth.

$$\forall j \geq i, \ell \in \mathcal{L}_{\pi_i}, r \in \mathcal{R}: \quad w_{\ell r}^{\pi_j} \geq 0 \tag{6}$$

Require weights to be non-negative.

$$\forall j \geq i, r \in \mathcal{R} \backslash \mathcal{R}_{\pi_j}: \quad w_r^p = 0 \tag{7}$$

Set weight to zero when inadmissible in the position.

Figure 1: Linear program LP$_2$ that determines weights for the ith position (π_i).

having been computed for the previous $i-1$ positions. Let $\lambda_{\ell\ell'}$ be the probability that a client in $\ell \in \mathcal{L}_{\pi_j}$ chooses relays for the jth to $(i-1)$st positions that yield positional location $\ell' \in \mathcal{L}_{\pi_i}$. Let $\delta_\ell^{\pi_i}$ be the client density over positional locations $\mathcal{L}_{\pi_i}$, computed as the expected density in a location after choosing the first $i-1$ relays. Let $\omega_r^{\pi_i}$ be the sum over positional locations of the weights of relay r in position π_i weighted by densities $\delta_\ell^{\pi_i}$. Finally, let $V_\ell^{\pi_i}$ be the expected penalty in π_i from $\ell \in \mathcal{L}_{\pi_j}$ when using the vanilla weights to choose positions from j to i according to π. See Appendix A for precise definitions of these values.

The linear program LP$_2$ (Figure 1) is solved to determine the weights in position π_i. It outputs a weight $w_{\ell r}^{\pi_i}$ for each $\ell \in \mathcal{L}_{\pi_i}$ and $r \in \mathcal{R}$, which is the weight used by a client in location ℓ for relay r. The Directory Authorities solve an instance of LP$_2$ for each circuit position in $\mathcal{P}$ in the order they appear in π.

LP$_2$ minimizes the expected penalty of the average client in position π_i, subject to several constraints. Constraint 2 maximizes network throughput by ensuring that the total bandwidth allocated to the position is equal to the amount β determined in LP$_1$. The intuition behind this constraint is to see the cumulative bandwidth of relays in each position of the path as a pipe through which data flows. We want the three pipes to have the same size, similar to Vanilla Tor's bandwidth-weighed design.

Constraint 3 requires that the selection probability of any relay is at most θ_{π_i} times its probability in Vanilla Tor (recall that v_r^p is the vanilla weight of relay r in position p). We note that doing so guarantees that CLAPS satisfies θ-GP-security [47] in *all* positions, not just at the guard. Constraint 4 guarantees that the expected penalty from *any* location is at most the expected penalty in Vanilla Tor. The LP objective does minimize the penalty for the average client, but this constraint provides a worst-case guarantee that no client will be worse off than under current Tor. Worst-case guarantees of this sort are important to provide fairness and minimize adoption concerns. Constraint 5 enforces relay bandwidth limits. In combination with Constraint 2, it ensures that traffic is load balanced in the sense that no relay is fully loaded until maximum total network throughput is achieved, thus preventing the performance problems discussed in Section 2. Finally, Constraints 6 and 7 guarantee that weights are non-negative and respect position flags.

Relay weight $w_{\ell r}^p$ can be interpreted as the amount of r's bandwidth that is allocated to position p from clients in ℓ. More precisely, for total traffic from clients in ℓ of $\beta_\ell \leq \beta$, $w_{\ell r}^p \beta_\ell / \beta$ of that traffic will be forwarded by r in position p, in expectation. Thus we can see that CLAPS allows *per-relay* allocation of bandwidth across positions, in contrast to all previous systems, which follow Vanilla Tor's positional allocation based only the four relay classes defined by the Guard and Exit flags. This flexibility gives CLAPS the capability to improve even on the original goals of existing designs. For example, CLAPS may achieve better guard resilience than Counter-RAPTOR itself, as it can use more high-resilience relays in the guard position and prefer low-resilience relays in the less-critical middle position.

LP$_2$ also includes weights $w_{\ell r}^{\pi_j}$ for positions $j > i$. Including these weights ensures that greedily optimizing the current position does not prevent weights for future positions from existing that satisfy the LP constraints. Note that these variables are defined for $\ell \in \mathcal{L}_{\pi_i}$ and not $\ell \in \mathcal{L}_{\pi_j}$. This is not a limitation for Constraints 2, 3, 5, 6, or 7, because any satisfying weights for $\mathcal{L}_{\pi_i}$ can be turned into one for $\mathcal{L}_{\pi_j}$ simply by taking their weighted sum with the $\lambda_{\ell\ell'}$ probabilities as weights, and vice versa. However, this equivalence is not necessarily true for Constraint 4 because it contains penalty values that depend on the location. Therefore, we initially seek a solution satisfying Constraint 4 only for the given position i. If one is not found, we backtrack to previous positions (trying one at a time) and solve the LPs again after adding extra constraints for positions $j > i$ that their weights are proportional to the vanilla weights (Appendix A). These extra constraints only enforce that such weights could be a solution, not that they will be the output of the later LPs. This addition is unnecessarily restrictive, which is why we only add it if an initial solution is not found (in our experiments, we never fall back to this case). This algorithm is guaranteed to find a solution as the vanilla weights always satisfy the constraints in all positions (after normalizing to sum to β).

3.4 Location Masking

We describe how to use *location masking* with any location-aware path-selection algorithm to limit leakage of the client location over time. In this technique, each client chooses a *location mask* and then performs path selection as if that mask were its true location. Thus its choices of paths can at most reveal its location mask, even if the adversary can observe many such choices over time. As long as the choice of mask does not leak much information about the true client location, the client obtains a long-term defense against the deanonymization demonstrated in the Tempest attacks [46].

Masks are chosen that balance several goals: (1) the mask should be similar to the true location so that the client continues to benefit from the location-awareness of path selection, (2) the mask should not reveal too much about the true location, and (3) the mask should be used by many users and thus not serve as a pseudonym. We abstract the first goal using a distance function d that takes a pair of locations and returns a number indicating how similar they are for purposes of a given location-aware path-selection algorithm. For example, in Counter-RAPTOR, a reasonable distance function is the average difference between the resilience values of the available guards. We abstract the second goal with an anonymity function α that takes a set of locations and returns a vector of scores indicating the degree of anonymity along several different dimensions. For example, anonymity dimensions may include the number of locations and the number of users. For the third goal, we will ensure a minimum density of clients using each mask.

We further introduce the notion of a *location guard*, which is a mask chosen by a client and maintained for a long time (e.g., several years). Location guards are needed because location masks must get updated as distances between locations vary over time. For example, distances based on BGP routes will change as the Internet topology changes. Thus, clients need to update their masks as the system discards old ones, but each new mask selection would allow more information to leak about the true location. To solve this, each client maintains a location guard and chooses a mask based on its guard rather than its true location. Thus, even as changing conditions cause mask updates, clients still limit leakage about their locations to only what is implied by their guards.

We cluster locations to define masks. Clusters are chosen to minimize intra-cluster distances as measured by d while ensuring that each cluster provides a minimal level of anonymity in every dimension, including a minimum client density. All clients with a location guard in a given cluster use as their mask the member that is the closest on average to the entire cluster. Let L be the set of client locations (e.g., all ASes on the Internet). Let $A \in \mathbb{R}^a$ be the set of anonymity thresholds for the clusters. The Cluster algorithm is detailed in Figure 2. It modifies a standard greedy algorithm for clustering by requiring that the cluster exceed the anonymity thresholds, which it finds by gradually tightening a limit ϕ on the amount of anonymity increase needed in each greedy selection. Moreover, it attempts to maximize the number of clusters, as in the larger CLAPS system, more masks creates a larger space of possible weights when trying to minimize the location-aware objective. Cluster starts with one cluster and thus always finds a solution as long as the anonymity constraints are satisfied by the whole set of locations.

3.5 Circuit Construction

Clients in CLAPS maintain Directory Guards as in Vanilla Tor. These guards are just used to distribute network information and thus need not be selected using location information. To obtain its location mask, a client requests the mask corresponding to its location guard from one of its Directory Guards, which can already observe the client's location. Similarly, a client can obtain from a Directory Guard all relay weights for its location mask. However, we can improve the efficiency when the first position is the guard (i.e., π_1 = g) by obtaining any weights that depend on the entry guard from the guard itself; e.g., in DeNASA, the exit weights depend on the guard AS, and the client can obtain exactly the exit weights needed for its guards by requesting one set through each guard.

```
Input: $L$, a set of locations that will be clustered.
Output: $C$, a clustering of $L$; and $M$, the mask of each cluster
Procedure:
$C \leftarrow \emptyset, M \leftarrow \emptyset$
for $n \leftarrow 1$ to $|L|$ do
    for $\phi \in (1, 1 - 1/|L|, \ldots, 1/|L|)$ do
        (1) Initialize $C'$ with $n$ clusters and its masks to arbitrary
            distinct locations $M' \leftarrow (\ell_1, \ldots \ell_n)$.
        (2) Until all locations have been assigned: Select cluster
            $C_i \in C'$ with lowest anonymity score $\alpha(C)_j$. Take the
            top fraction $\phi$ of unassigned locations, ranked by the
            amount that they would increase $C_i$'s minimum
            anonymity score, and from them select the location $\ell$ that
            minimizes distance $d(M'_i, \ell)$. Assign $\ell$ to $C_i$.
        (3) For each cluster, if the mask doesn't minimize the
            average distance between the mask and all cluster
            locations, update it to a minimizing location.
        (4) If a cluster mask changed in Step (3) and the iteration
            limit hasn't been exceeded, return to Step (2) with the
            new masks.
        if $\forall C \in C', 1 \le i \le a$: $\alpha(C)_i \ge A_i$ then break
    if $\exists C \in C', 1 \le i \le a$: $\alpha(C)_i < A_i$ then break
    else $C \leftarrow C', M \leftarrow M'$
return $(C, M)$
```

Figure 2: Cluster algorithm

Circuits are constructed in CLAPS using the Tor's current cryptographic protocol. CLAPS only changes the way that the relays for the circuit are chosen. Relays are chosen for each position in a circuit in the order they appear in π. After choosing the first $i - 1$ positions, the client is in some positional location $\ell \in \mathcal{L}_{\pi_i}$, and relay r is chosen for the ith position with probability $w^{\pi_i}_{\ell r}/\beta$. If π_i = g, then the relay chooses a new guard if it hasn't selected them all yet. Once the relay has chosen its guards, it selects one for its circuit from among its existing guards with uniform probability. Uniform selection prevents double-weighting guards when clients maintain multiple guards (the default number is currently one). However, it is not suitable with location-aware guard selection if the guard is not selected first, as guards are held long-term and may have high penalties with the current partially-constructed circuit. Therefore, it should be that π_1 = g unless guard selection is location-unaware (e.g., Vanilla Tor). After choosing relays for all positions, CLAPS returns to Tor's existing algorithm, which applies some restrictions, such as the disallowing a circuit to contain multiple relays in the same /16. Path selection is retried if the current choice violates a restriction.

4 EVALUATION

In this section, we empirically evaluate the security and performance properties of location-aware path selection algorithms generated by the CLAPS framework. We present two algorithms: (1) CLAPS-CR, a path selection algorithm designed to maximize chosen guards' hijack resilience, and (2) CLAPS-DN, a path selection algorithm designed to minimize the probability that a given "suspect AS" is simultaneously present on both the client-guard network path and exit-destination network path. These algorithms goals' mirror those of Counter-RAPTOR [38] and DeNASA [5]; as such, we can use these systems as reference points for comparison.

4.1 Implementation

We implemented all of the CLAPS algorithms in a approximately ten-thousand lines of C, C++, and Python. The clustering algorithm was implemented in C++ for efficiency. We use Python to define CLAPS LPs and use the CLP linear program solver to solve the generated LPs [7]. We implemented CLAPS in Tor v0.3.5.8 so that it could be simulated in the Shadow simulator. Our source code is available online.[1]

4.2 CLAPS-CR

A few algorithm inputs must be defined in order to run CLAPS and obtain a path selection distribution. Cluster (Figure 2) requires a set of locations to cluster $\mathcal{L}$, the choice of a distance function d, an anonymity function α, and a minimum anonymity score A. The LP described in Section 3.3 requires the definition of positional locations $\mathcal{L}_{\pi_i}$, a parameter for relay placement θ, and penalty matrices P^{π_i} where each entry $P^{\pi_i}_{\ell r}$ corresponds to the penalty of a client in location ℓ choosing relay r for position π_i in a circuit. Below we discuss how we chose inputs to instantiate CLAPS-CR.

4.2.1 Path-Selection Goal.

In CLAPS-CR, our objective is to have clients choose guard relays with high hijack resilience. Recall from Section 2 that a guard's hijack resilience depends both on the client's and guard's autonomous system of origin. Let $R_{\ell,r}$ be the hijack resilience of relay r when used from client location ℓ.

4.2.2 Clustering Inputs.

First, we must decide on a set of locations that are input into the clustering algorithm. Following Sun et al., we consider a client's autonomous system to be its location [38].

Next, we must choose d_{CR}, the distance function used in the location clustering algorithm. Recall that all locations within a cluster are restricted to use the same relay selection distribution; the clustering algorithm will attempt to minimize the distance between locations in the same cluster. Thus, a reasonable distance function will assign low distances to pairs of client locations that have similar preferences for relays. Accordingly, we define d_{CR} to be sum, across all guard relays, of the bandwidth-weighted resilience disagreement between two locations. This is expressed as $d_{CR}(\ell_1, \ell_2) = \Sigma_{r \in \mathcal{R}_g} B_r |R_{\ell_1,r} - R_{\ell_2,r}|$.

This distance is minimized when $R_{\ell_1,r} \simeq R_{\ell_2,r}$ for all guard relays, i.e., when ℓ_1 and ℓ_2 agree on which relays are hijack resilient. We weight the magnitude of the disagreement by the relay's bandwidth. With this weighting, disagreements about the resilience of high-bandwidth guard relays are more significant than disagreements about low-bandwidth relays.

Then, we must define an anonymity function α_{CR} that maps a set of locations to an anonymity score. We consider three different anonymity criteria when defining α_{CR}: (1) the total client density in C, defined as $\text{Size}(C) = \Sigma_{\ell \in C} D_\ell$, (2) the entropy of the distribution of users across locations in the cluster, and (3) the entropy of the distribution of users across countries in the cluster. The entropy of the distribution of users across locations is computed as $EL(C) = -\Sigma_{\ell \in C} \left[D_\ell/\text{Size}(C) \cdot \log_2(D_\ell/\text{Size}(C))\right]$; $EC(C)$, the entropy of the distribution of users across countries, is defined similarly. All together, we define $\alpha_{CR}(C) = (\text{Size}(C), EL(C), EC(C))$. The cluster size criterion allows us to concretely specify the minimum number of clients per cluster. The entropy-based criteria allow us to quantify anonymity with respect to an adversary who has no a priori knowledge of a given client's location/country and ensures that we provide sufficient anonymity over these sensitive attributes [34, 46].

Finally, we must define $A_{CR} \in \mathbb{R}^3$, the minimum anonymity score that must be achieved by each cluster in a valid clustering. We set the components of A_{CR} relative to the current Tor network. In this work, we allow each cluster to reduce Tor's anonymity by a factor up to 20; we will show that the clusters produced at this reduction-level still provide strong anonymity. Let L be the set of all client locations present in the Tor network today. We set $A_{CR} = \left(\text{Size}(L)/20, EL(L) - \log_2(20), EC(L) - \log_2(20)\right)$.

After running the LP, we obtain a clustering $C = \{C_i\}_{i=1}^n$.

4.2.3 LP Inputs.

Sequentially, the LPs are defined and executed after the clustering is complete, so we can use outputs from the clustering algorithm as inputs to our LPs. In CLAPS-CR, we will be obtaining an optimized guard-selection distribution for each cluster.

For $\mathcal{L}_g$, the set of positional locations input into the guard-relay LP, we will use $\{C_i\}_{i=1}^n$, the set of clusters. This will define one guard-selection distribution for each cluster.

Next, we must define $P^g_{C_i r}$, the penalty of cluster C_i choosing relay r as a guard. Note that a penalty matrix must be defined only for each positional location in $\mathcal{L}_g$ and not the total set of locations L. We define $P^g_{C_i r} = \Sigma_{\ell \in C_i} \left[D_\ell (1 - R_{\ell,r})\right]$; in other words, we define the penalty for relay r to be the weighted linear combination of the complement of hijack resilience for each client location in the cluster. Since the LP minimizes penalty, hijack resilience will be maximized. Weighting each penalty value by client density allows us to provide more benefit to client locations with many Tor users; note that, by our constraint, we will not make security worse for any low-density client-locations whose penalties are not as heavily reflected in this sum.

Finally, we must choose a value θ_g to provide guard placement attack security. We find that θ has a large influence on the quality of solutions we obtain. We explore the values recommended by Wan et el[47]: $\theta_g \in \{1.25, 2, 5\}$.

When defining middle and exit relay distributions for CLAPS-CR, we set $\mathcal{L}_m = \mathcal{L}_e = \{\perp\}$, $P^m_{\perp r} = P^e_{\perp r} = 1$, and $\theta_m = \theta_e = \infty$. These trivial settings indicate that all CLAPS-CR clients will use middle and exit weights proportional to Vanilla Tor middle and exit weights while maintaining the network's load balance.

4.3 CLAPS-DN

4.3.1 Path-Selection Goal.

In CLAPS-DN, our objective is to have clients choose relays for circuits that avoid a set of suspect ASes who are likely to be on the client-guard and exit-destination paths when building circuits in Vanilla Tor. We use the same set of eight suspect ASes $\mathcal{S}_8$ = {AS1299, AS3356, AS6939, AS174, AS2914, AS3257,

[1] https://github.com/orgs/CLAPS-CCS2020/

AS6453, AS9002} identified by Barton et al. [5]. Let $\mathcal{S}_2 = \{$AS1299, AS3356$\}$ (the two ASes avoided by DeNASA's g-select algorithm).

4.3.2 Clustering Inputs. Similar to CLAPS-CR, we use clients' autonomous systems as the set of locations that will be clustered; in part, a client's autonomous system determines the suspects on the client-guard paths, as was the case for CLAPS-CR.

We define $d_{DN}(\ell_1, \ell_2) = \Sigma_{r \in \mathcal{R}_{\mathrm{g}}} \Sigma_{s \in \mathcal{S}_8} \left[B_r | I^s_{\ell_1, r} - I^s_{\ell_2, r} | \right]$ where $I^s_{\ell, r}$ is an indicator variable such that $I^s_{\ell, r} = 1$ if suspect AS s in on the link between client location ℓ and relays r; otherwise, $I^s_{\ell, r} = 0$. Similar to the distance function defined for CLAPS-CR, this distance function captures the magnitude of disagreement about which suspects are present to the guards in the network.

We set $\alpha_{DN} = \alpha_{CR}$ and $A_{DN} = A_{CR}$. Both algorithms use autonomous systems for the set of client locations being clustered, and so we can use the same anonymity criterion for both algorithms.

4.3.3 LP Inputs. Unlike CLAPS-CR, in CLAPS-DN we will define non-trivial exit relay penalties that will mimic the design of DeNASA. First, to obtain the guard selections distributions for each cluster, we set the initial positional locations to $\mathcal{L}_{\mathrm{g}} = \{C_i\}_{i=1}^n$.

We define $P^{\mathrm{g}}_{C_i r} = \Sigma_{\ell \in C_i} \left[D_\ell \cdot \max(\{I^s_{\ell, r}\}_{s \in \mathcal{S}_2}) \right]$; i.e., the penalty for relay r is the density-weighted combination of $\max(\{I^s_{\ell, r}\}_{s \in \mathcal{S}_2})$ for each location ℓ in the cluster. This penalty matches DeNASA's g-select algorithm, which selects guards only if neither suspect AS in $\mathcal{S}_2$ is on the client-guard link.

We also use a CLAPS LP to generate exit selection distributions that avoid suspect ASes. One exit-selection distribution is generated for each cluster-guard relay pair. So, for this LP, we set the set of positional locations to $\mathcal{L}_{\mathrm{e}} = \{(C_i, g)\}$ for $1 \le i \le n$ and $g \in \mathcal{R}_{\mathrm{g}}$.

Let $\Pr[s \in (e \leftrightarrow D)]$ be probability that suspect AS s is on the link between exit e and destination D, where the randomness is taken over uniform choice of D from among the Top 200 Alexa websites [5]. To match DeNASA's e-select algorithm, we define $P^{\mathrm{e}}_{(C_i, g) r} = \Sigma_{\ell \in C_i} \Sigma_{s \in \mathcal{S}_8} \left[D_\ell \cdot I^s_{\ell, g} \cdot \Pr[s \in (r \leftrightarrow D)] \right]$. In other words, for each client location and guard ℓ, g, we compute the probability that a suspect on the client-guard link is present on an exit-destination link. The penalty for a relay r in the exit position is the weighted combination of this value for each $\ell \in C_i$.

For simplicity, we set $\theta_{\mathrm{g}} = \theta_{\mathrm{e}}$ and run simulations for the recommended values of θ (1.25, 2, and 5).

Similar to CLAPS-CR, $\mathcal{L}_{\mathrm{m}} = \{\perp\}$, $P^{\mathrm{m}}_{\perp r} = 1$, and $\theta_{\mathrm{m}} = \infty$. These settings indicate the CLAPS-DN clients should select middle relays with weights proportional to Vanilla Tor while respecting the network load balance.

4.4 Security Evaluation

4.4.1 Datasets & Methodology. We empirically evaluated the security of CLAPS-CR and CLAPS-DN through a series of analyses and simulations on archived Tor and Internet data. We used Tor consensus documents and server descriptors from 15 January 2019 [42]; at this time, there were approximately 6,600 relays in the Tor network. We used Route Views Prefix-to-AS mappings from 15 January 2019 to map relays to their autonomous systems [33]. CAIDA's AS Organization data from 1 January 2019 was used to map autonomous systems to their organization and country [40]. We used CAIDA's AS relationship topology from January 2019 in order to infer AS paths between Internet hosts [39].[2] Inferred AS paths are used to compute hijack resilience (for Counter-RAPTOR and CLAPS-CR) and the presence of suspect ASes (for DeNASA and CLAPS-DN). The programs for computing relay hijack resilience were obtained directly from Sun et al[38]. We compute AS paths using shortest, valley-free path inference [24], a technique that is consistent with prior work [5, 46, 47].

	Min	Q_1	Q_2	Q_3	Max
CLAPS-CR (14 Clusters)					
Density	5.2%	5.2%	5.3%	8.7%	14%
Num. ASes	2267	3348	4381	5196	6898
Num. Countries	74	89	107	119	137
CLAPS-DN (16 Clusters)					
Density	5.1%	5.1%	5.2%	6.2%	16%
Num. ASes	2974	3251	3627	4559	6481
Num. Countries	130	136	143	155	159

Table 1: Summary statistics describing the clusters obtained for CLAPS-CR and CLAPS-DN. Q_1, Q_2, and Q_3 denote the 25th, 50th, and 75th percentile respectively.

We used autonomous systems as the client locations in our algorithms. For our analyses, we consider any autonomous system in the CAIDA's AS topology that advertises at least one IPv4 address as a client location; in total, there are 62,891 such ASes (many prior works consider only tens or hundreds of ASes when analyzing system security [5, 13, 38, 47]). CLAPS requires a measured density (i.e., Tor user count) for each of these ASes. To estimate AS density, we take Tor's measured user-per-country statistics and distribute users into ASes within their country proportional to the number of IPv4 addresses each AS originates. These estimates can be improved if measurement systems are put in place to directly measure clients' ASes. The top five highest-density ASes according to this assignment are (1) AS12389, Rostelecom, RU; (2) AS3320, Deutsche Telekom, DE; (3) AS23693, Telekomunikasi Selular, ID; (4) AS7018, AT&T, US; and (5) AS15557, SFR SA, FR.

4.4.2 Clusters. In CLAPS, it is possible for a client to leak its cluster to an adversary who observes path-selection behavior over time [46]; therefore, it is crucial that each cluster contains a diverse set of clients to provide anonymity. This is why we quantify cluster anonymity (α) and ensure that the clusters meet a number of anonymity criteria.

We ran our clustering algorithm in our simulated network. For CLAPS-CR, we obtained a clustering with 14 clusters. For CLAPS-DN, we obtained a clustering with 16 clusters. Table 1 contains summary statistics describing the composition of the clusters. Note that the minimum fraction of density in any cluster is 100%/20 = 5%, since we allow a 20× reduction in anonymity set size for the anonymity criteria. Even the smallest cluster in our clustering contains 74 distinct countries, thousands of ASes, and 5% of Tor's millions of users. We find that our clustering procedure does well at producing balanced clusters—the largest clusters obtained are only 3× denser than the least dense clusters in the clustering.

4.4.3 CLAP-CR Hijack Resilience. We examine CLAP-CR's ability to maximize client-guard hijack resilience, its primary objective. For

[2] Our analysis of DeNASA and CLAPS-DN uses an older AS topology from February 2018. App. C shows our main security evaluation when instead using the 2019 topology.

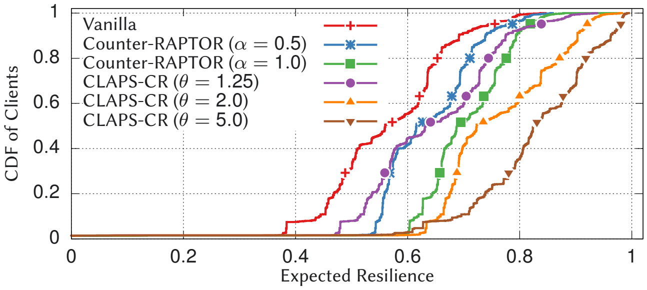


Figure 3: Expected hijack resilience when choosing guards according to Tor (Vanilla), C.R., and CLAPS-CR.

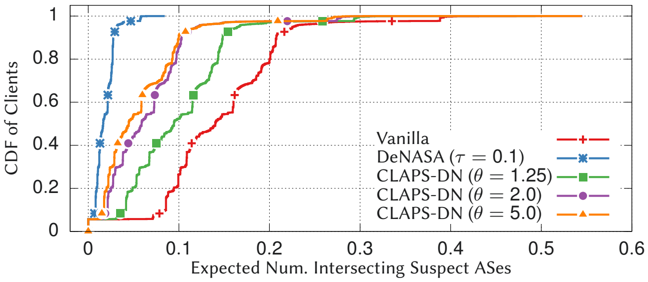


Figure 4: Expected number of suspects present on both the client-guard and exit-destination path when building a random circuit.

each client location, we calculated a client's expected client-guard hijack resilience $\Sigma_{r \in \mathcal{R}_g} \Pr[r] R_{\ell,r}$ where the selection probability Pr is defined under a particular path-selection algorithm. We considered three algorithms: (1) Vanilla, (2) Counter-RAPTOR, and (3) CLAPS-CR. We considered Counter-RAPTOR when $\alpha = 0.5$ (i.e., each guard's selection probability is proportional to a 50/50 blend of its resilience and bandwidth) and when $\alpha = 1.0$ (i.e., each guard's selection probability is proportional to its resilience). Figure 3 shows the results of this analysis. Expected hijack resilience under an algorithm is plotted on the x-axis; the y-axis plots the cumulative fraction of Tor clients that experience at most a given hijack resilience. When $\theta = 1.25$, a conservative setting for θ, the CLAPS-CR path selection algorithm is competitive with Counter-RAPTOR at $\alpha = 0.5$ which is the α-value recommended by Sun et al.; the median hijack resilience for Counter-RAPTOR ($\alpha = 0.5$) is 0.62, whereas the median hijack resilience for CLAPS-CR ($\theta = 1.25$) is 0.63. By $\theta = 2$, the CLAPS-CR strategy dominates Counter-RAPTOR ($\alpha = 1.0$) for nearly all client locations, while simultaneously improving on other aspects of Counter-RAPTOR, such as relay-placement attack vulnerability (which will be shown) and information leakage over time [46]. It offers a 13% improvement in hijack resilience at the 80th percentile (from 0.77 to 0.87). These improvements highlight the benefit of using our optimization framework instead of relying on heuristic approaches to achieve security.

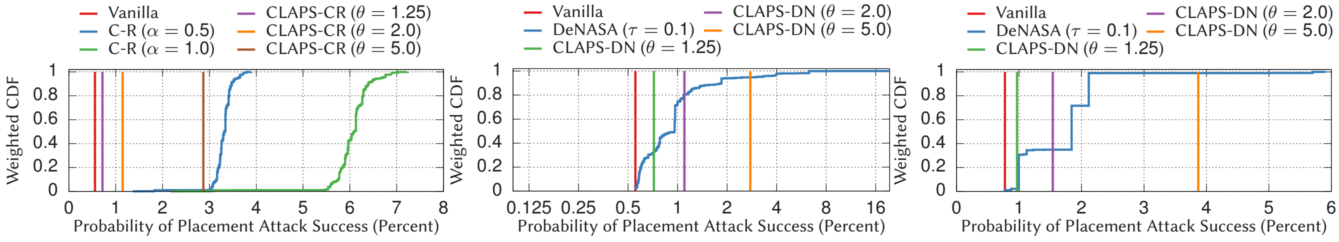
4.4.4 CLAP-DN Suspect Avoidance. Here, we analyze CLAPS-DN's ability to avoid suspect ASes. We compute the expected number of suspect ASes that will be on both the client-guard network path and exit-destination network path where randomness is taken over the choice of guard relay, the choice of exit relay, and the uniform choice of web destination to visit. Following Barton *et al.*[5], we used the Alexa Top 200 destinations as the set of destinations that clients visit. We resolved these destinations to IP addresses and ASes at a network vantage point in New York, USA. Figure 4 presents our results; similar to the previous analysis, the expected number of intersecting suspect ASes appears on the x-axis, and the y-axis plots a CDF of clients.

Unlike Counter-RAPTOR, we are unable to obtain a path-selection algorithm that dominates DeNASA. CLAPS-DN does offer substantial protections over Vanilla Tor, and at $\theta = 2$ offers comparable protections in the median case (0.02 vs. 0.06 expected suspects). We will see that DeNASA offers poor relay-placement protection and prior work has shown it leaks location information quickly [46]; CLAPS-DN provides a way to achieve similar security without these critical vulnerabilities.

4.4.5 Relay-Placement Attacks. In addition to analyzing how well these algorithms achieve their objectives, we also analyzed how well these algorithms defend against relay placement attacks. In this analysis, we consider an adversary who places some malicious relays into the Tor network with the goal of having Tor clients choose the malicious relays for circuits. We consider an adversary who targets a particular cluster of users in CLAPS and the same subset of users under Counter-RAPTOR and DeNASA. We present results for the most vulnerable cluster which gives an upper-bound on the adversary's success probability.

For Counter-RAPTOR and CLAPS-CR, we identified an AS with maximal hijack resilience that the adversary places relays into. We ensured that this AS allows Tor relays, as it already contains guard relays. The adversary in our attack places 100 malicious relays into the identified AS, each relay with bandwidth approximately equal to 7 Mbps (Wan *et al.* showed that effectively attacking Counter-RAPTOR requires running many small relays instead of a few large ones). We compute the likelihood that clients choose these malicious relays and present the results in Figure 5a. In all three subfigures, the x-axis denotes the adversary's probability of succeeding in the relay placement attack, and the y-axis denotes the CDF over *targeted* clients. Because all clients in a given cluster choose relays according to the same distribution, the CDFs for CLAPS-CR appear as vertical lines (this is also true for Vanilla Tor, where all clients choose relays according to a single distribution). Indeed, our constraints provide a concrete bound for the adversary's success probability; for example, in Vanilla Tor, the adversary achieves a success probability of 0.56%—for $\theta = 2$ in CLAPS-CR, the adversary's success probability is 1.12%, a 2× increase. However, for Counter-RAPTOR, this attack success is much higher; for some client locations, the adversary succeeds with probability 3.89% ($\alpha = 0.5$) and 7.23% ($\alpha = 1.0$), a 7× / 13× increase in attack success probability relative to Vanilla Tor.

DeNASA suffers worse from relay placement attacks. We simulated an adversary who places a single malicious guard relay and single malicious exit relay into an AS that already contains Tor relays. We let the relays have bandwidth approximately equal to 700 Mbps, which corresponds to the highest-bandwidth relays running in the Tor network. The adversary's strategy remains the same: the adversary targets the most vulnerable cluster and places relays in ASes that maximize the likelihood that the malicious relays are chosen. In this attack, the adversary maximizes the exit relay selection probability conditioned on the malicious guard relay being selected by a targeted client (in both CLAPS-DN and DeNASA, exit-relay selection distributions are determined by a



(a) **(b)** **(c)**

Figure 5: Probability of success when the adversary runs a relay placement attack. Figure (a) shows the probability of succeeding in a guard placement attack against Counter-RAPTOR and CLAPS-CR. Figure (b) shows the probability of succeeding in a guard placement attack against DeNASA and CLAPS-DN. Figure (c) shows the probability of succeeding in an exit placement attack against DeNASA and CLAPS-DN.

client's guard relay). Figure 5b shows the probability that the adversary's guard placement attack succeeds, and Figure 5c shows the probability that the adversary's exit placement attack succeeds, given that the guard-placement attack succeeded. As was the case in CLAPS-CR, CLAPS-DN places a concrete bound on the adversary's success probability (the success probability against Vanilla Tor is 0.55% in the guard position and 0.77% in the exit position). Against DeNASA, the adversary succeeds in the guard-placement attack against some client locations with ∼20% probability—*a* 40× *increase over Vanilla Tor.* In the exit-placement attack, the adversary achieves 5% probability (a 10× increase) for many locations. These unbounded increases highlight the importance of CLAP's constrained approach to relay placement security.

4.5 Performance Evaluation

We evaluate CLAPS on three different Tor topologies and demonstrate the performance issues of previous location-aware schemes. All our experiments contain our Tor performance and security fixes (our Tor Proposal 310 [30], summarized in Section 6).

4.5.1 Methodology. We ran our experiments using the Shadow discrete-event network simulator [15, 16]. Our simulations are scaled down to 2,400 Tor clients and 250 relays. We used the recommended value for setting the Web/Bulk fraction of clients [17]. The first topology, scaled down from a consensus in June 2017, is representative of current Tor topologies[3], and contains a large amount of guard bandwidth relative to the other positions. Moreover, after balancing bandwidth between path positions, Vanilla Tor exit bandwidth remains scarce; there is roughly half of the bandwidth compared to what is available in middle and entry positions. The second Tor topology is similar to previous Tor network states (e.g., March 2015) and contains a large amount of guard bandwidth, yet not disproportionately high compared to the other positions (157% more than all exit and exit+guard bandwidth). After balancing the bandwidth between positions using the bandwidth-weight equations [43], the three positions have the potential to offer the same total bandwidth to paths (which both Vanilla Tor and CLAPS can achieve with a total of 245145 Consensus Weight for each position). The third topology focuses on the client distribution, and assumes a sudden increase of users from a chosen region of the world within our network built from a June 2017 Tor consensus.

4.5.2 Analysis. Figure 6 shows the measured results from the Shadow simulations. Each plot is a CDF of client performance, where the x-axis shows the median time to download a 2 MiB Web page.

In the first column, Figure 6a and Figure 6d compare CLAPS-CR and CLAPS-DN against Counter-RAPTOR/DeNASA and Vanilla Tor. One goal of CLAPS stated in Section 3 is to offer similar performance to Vanilla Tor from the load-balancing constraint defined in the linear program. *Surprisingly with CLAPS-CR in Figure 6a, about 20% of Tor clients perform even better than Vanilla Tor up to* ≈ 5 *seconds in their median download time for our 2MiB page transfer.* We observed that this performance improvement increases as we relax the guard placement attack constraint. When we relax the placement constraint (i.e., we increase θ), we increase the probability for CLAPS-CR clients to select guard relays with high resilience values. We might expect this behavior to impact the quality of the path regarding performance metrics that are not part of the optimization constraint, such as path latency. Intuitively, from a given location, *highly resilient relays should be closer to the client* hence shortening the overall paths that these clients make. These shorter paths may cause an overall increase in client performance compared to Vanilla Tor while the load-balancing property is similar. Appendix D displays an analysis of path latency and path loss with respect to each path selection and θ value, and the results are consistent with our hypothesis.

Counter-RAPTOR and DeNASA perform worse than Vanilla Tor in the first column (Figure 6a and Figure 6d). Yet, the performance is not disproportionately worse, even when Counter-RAPTOR uses the resilience values as the only selection criterion ($\alpha = 1$), which does not account for any performance factor. For both figures, the distribution of bandwidth gives twice the quantity of bandwidth in the entry and middle positions compared to the exit position after applying bandwidth-weight equations. As previously observed in Section 2.3, *the impact of unbalanced path selection is partially absorbed by the spare bandwidth capacity at the entry position.* If the condition of the network is not favorable to Counter-RAPTOR and DeNASA (i.e., scarce guard bandwidth), performance can degrade significantly.

Indeed, when removing this excess bandwidth (second column, Figure 6b and Figure 6e), *the performance of Counter-RAPTOR and DeNASA degrades significantly* while Vanilla Tor and CLAPS keep performing almost as well as in the original topology. Given the reduction of entry bandwidth, CLAPS's most significant constraint

[3] From 2016 onward, Tor exit bandwidth is scarce

becomes the load-balancing; hence, a given location does not focus on the most secure guards as it does in Figure 6a. For CLAPS-CR, this explains why CLAPS does not outperform Vanilla Tor in this network.

Figure 6c and Figure 6f in the third column show how path selection performs in the event of many Tor users connecting from the same location in the world, such as a city or a country. Sudden spikes of users connecting or disconnecting from the Tor network happened several times in Tor's history in various countries, often due to censorship and societal events. Sudden spikes of users can happen again, and a location-aware path selection algorithm should be able to provide similar performance during these events, similar to how Vanilla Tor performs. Previous works do not account for heterogeneity in user density, which explains why they slow down on this particular network state. In both Figures, we simulate a 20% increase in users from Michigan (a well-connected location in our topology) which is causing *poor load-balancing in Counter-RAPTOR and DeNASA.* CLAPS adapts by using weights subject to our load-balancing constraint that accounts for the increased usage of the network from this location. Note that Michigan was chosen because it is a location close to the Tor relays in expectation, and well connected to them (having low latency and low link loss). Hence, performance loss might be caused by poor usage of the available relays from the path selection algorithm.

Regarding CLAPS, note that the results slightly differ from Vanilla Tor. We observed in this network configuration that CLAPS clients were using paths with slightly better latency in the case of CLAPS-CR, and slightly worse in the case of CLAPS-DN. Regarding CLAPS-DN, by trying to avoid well-connected suspect ASes, network paths chosen become significantly longer. Performance degradation increases with θ as CLAPS-DN avoids more suspect ASes. Appendix D shows path loss and latency results in this setup, showing CLAPS-DN having higher latency and packet-loss as θ increases.

Recap. We conducted a performance analysis under three significantly different Tor network topologies: one similar to the current state of the Tor network and two others reminiscent of previous existing topologies or events. Our results confirm the efficiency of our load-balancing constraint codified in our linear program and the performance versatility of our scheme. *Our results also show that previous works in location-aware path selection algorithms underestimate the importance of network diversity in their approaches.* We conjecture that the same result would apply to other works we did not explore here but which could be implemented with CLAPS as well. Moreover, our results show that we can obtain a more secure location-aware Tor network without undermining performance.

Using the CLAPS approach, *Tor path selection is not limited by a performance/security trade-off, but rather by a security/security trade-off between different threat models* when the goal is to achieve a path selection algorithm that maintains similar performance to Vanilla Tor.

4.6 Efficiency

In this section, we briefly discuss the practical costs of CLAPS. In this analysis, we assume a Tor network with 6,700 relays. In CLAPS-CR, clients must download an additional set of guard, middle, and exit weights for the relays in the network. Assuming a 4 byte encoding is used to store weights, clients will download approximately 80 KB additional data. Tor's directory authorities will send approximately 1.60 MB additional data to each relay serving Tor directory information (for 20 clusters).

The client cost in CLAPS-DN is similar—each client still downloads a set of CLAPS weights which costs 80 KB; however, the client must download a new exit weight distribution each time a new guard is selected, which happens infrequently. A single exit weight distribution costs at most 27 KB. To each directory relay, Tor Directory Authorities need to send (1) one guard weight distribution per cluster, (2) a single middle weight distribution, and (3) one exit weight distribution per cluster-guard-relay pair. (1) costs approximately 536 KB. (2) costs approximately 27 KB. Without compression, sending (3) can be expensive—approximately 175 MB. However, by de-duplicating weight distributions for guards and exits in common ASes, this cost can be reduced to about 10 MB. Applying compression may further reduce this cost.

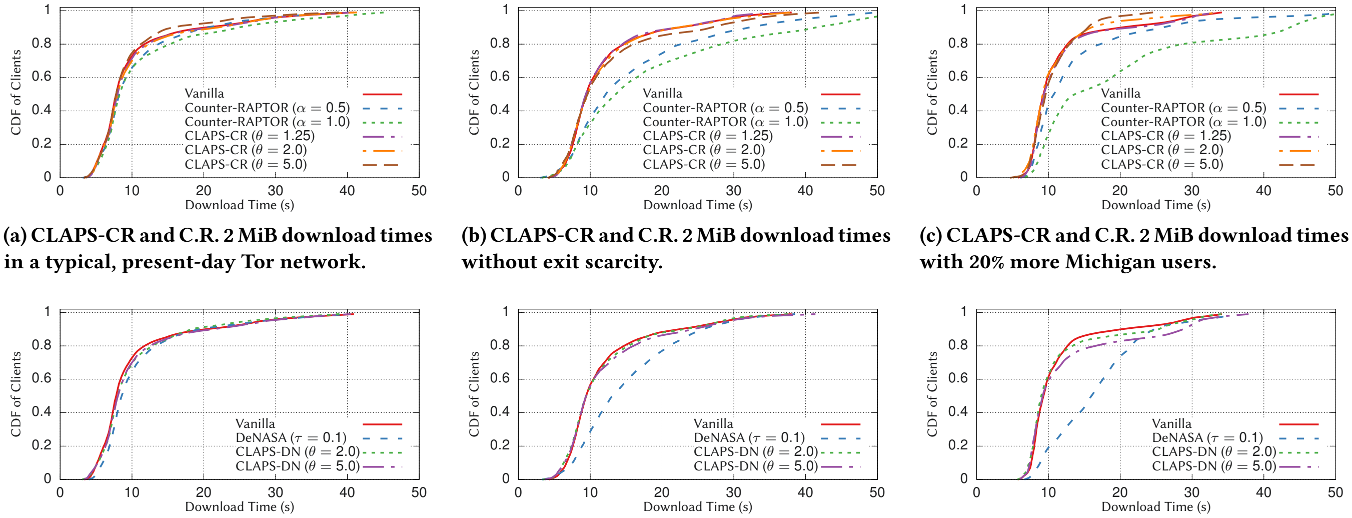
The linear programs formulated and solved in our evaluations of CLAPS-CR and CLAPS-DN were manageable in complexity. The largest LPs had approximately 100,000 rows (constraints) and 30,000 columns (variables) in MPS format. No LP took more than 5 minutes to solve using CLP on a machine with an Intel Xeon E7-8891 CPU.

5 RELATED WORK

A significant number of research works have targeted performance improvements of the Tor network [1–3, 6, 11, 12, 22, 28, 35, 37, 45, 48]. Several of them had their claims disproved with more recent research tools. For example, Snader and Borisov's trade-off [37] of security/performance does not improve performance when the tune-up focuses on high bandwidth relays [6]. We also demonstrated degraded performance of proposed path selection algorithms relative to analyses performed in the original works [5, 38] by considering an analysis involving network topology changes.

Regarding security, we can classify path-selection algorithms within two broad categories: first, the ones that are not location-aware: see [4, 31, 37]. For example, Waterfilling rebalances relay weights to better defend against an adversary owning or compromising relays to perform end-to-end correlation. [31]

Then, there have been many proposed location-aware path selection algorithms [5, 10, 13, 18, 20, 21, 27, 38] (see Table 3 in the appendix). Astoria, which focuses on AS-level adversaries, does already take advantage of a LP formulation to build a distribution that would minimize the probability to encounter an adversary. However, their model lacks performance versatility, load-balancing, and consideration for threats such as the relay placement and Tempest attacks. By using CLAPS and modeling Astoria as a destination-naive algorithm, Astoria could achieve the same load-balance as Vanilla Tor with potentially as good compromised paths avoidance. TAPS [18], another location-aware path selection algorithm, uses a notion of trust to choose paths not likely to be observed be the adversary. TAPS uses a clustering mechanism similar to location masking to prevent leaking location information. However, this mechanism is (1) specific to the TAPS algorithm, (2) includes a limited notion of anonymity, and (3) suffers from a deanonymization attack [46]. We solve the attack with the addition of location guards.



(a) CLAPS-CR and C.R. 2 MiB download times in a typical, present-day Tor network.

(b) CLAPS-CR and C.R. 2 MiB download times without exit scarcity.

(c) CLAPS-CR and C.R. 2 MiB download times with 20% more Michigan users.

(d) CLAPS-DN and DeNASA 2 MiB download times in a typical, present-day Tor network.

(e) CLAPS-DN and DeNASA 2 MiB download times without exit scarcity.

(f) CLAPS-DN and DeNASA 2 MiB download times with 20% more Michigan users.

Figure 6: Simulation results comparing CLAPS-CR/DN to Counter-RAPTOR, DeNASA, and Vanilla Tor. (a) and (d) are scaled down networks from June 2017 containing 250 relays and 2400 clients. The networks in (b) and (e) have 51% reduced guard bandwidth from the topologies used in (a) and (d). Figures (c) and (f) show the performance of clients connecting from Michigan assuming a sudden increase in those clients.

A recent work has showed the impact of information leak over time of location-aware path selections [46]. In response to those problems, DPSelect [13] applies a differential-private guard selection mechanism to Counter-RAPTOR to reduce the maximum divergence between client distributions, yet it does not address performance issues nor deal with relay placement attacks. CLAPS supports a different idea, called *location masking* to make the leak of information eventually only reveal the location mask. This design also applicable to previous location-aware schemes. Techniques from DPSelect may be applied to CLAPS in order to slow the rate at which location masks are potentially leaked.

A summary of each path selection goal which we could consider for CLAPS is given in Table 3 in Appendix B. An instance of CLAPS might be created for any these systems that take advantage of pre-computed data.

6 DISCUSSION

There is no single path-selection algorithm that we recommend with CLAPS. Instead, we aim to demonstrate how the common obstacles to the use of existing proposals can all be solved with CLAPS, making any one of them suitable for Tor. CLAPS can be used with any client-location-aware path-selection algorithm that does not take into account the destination's location and take advantage of pre-computed data. Using the destination location is more difficult, as it is highly variable and is not known before the connection is needed. However, due to these issues, most of the existing location-aware path-selection algorithms avoid using the destination, and many can be usefully modified to only consider the client's location. For example, a version of LASTor [1] can be used that minimizes latency from the client to the exit, no longer considering the destination. Moreover, the various proposed algorithms have several different and important goals, and our general framework provides a powerful methodology to combine and balance the various criteria of these proposals within a single path-selection algorithm.

Finally, in the course of investigating how to improve Tor's weight computations, we have observed a significant flaw in the way that guards are selected. In Tor, guards are first sampled according to their weights and then chosen uniformly at random from the sampled set [41]. This process causes the effective weights for guard to skew towards uniform weighting, where the skew increases with the ratio of sampled to total guards. It especially can cause problems in client-location-aware algorithms, as clients in certain locations may choose from relatively few well-placed guards. It also causes a serious problem with the accuracy of research using Tor simulations, as simulation networks are typically much smaller than the actual Tor network [15]. We have proposed an efficient solution to this issue, described in Tor Proposal 310. We have implemented it as a patch, which we have used in our experiments. Our code has been merged and tagged to be released in June 2020 with Tor 0.4.4.x.

7 CONCLUSION

We improve the security and performance of location-aware path-selection algorithms for Tor by designing a generic framework for location-aware path selection algorithms. CLAPS can be applied to many of them to achieve their primary purpose while solving performance and security problems. We describe how to apply CLAPS to two specific client–location-aware algorithms: counter-RAPTOR [38] and DeNASA [5]. Through experiments, we show how CLAPS eliminates their performance deficits, solves their security flaws, and obtains results towards their primary goal that are competitive or better than the original systems.

ACKNOWLEDGEMENTS

This work has been supported by the Office of Naval Research. This work was supported in part by the National Science Foundation under grants CNS-1553437 and CNS-1704105. This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-19-C-0079. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, or any other sponsoring agency.

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

REFERENCES

[1] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. 2012. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*. 476–490.

[2] Mashael AlSabah, Kevin S. Bauer, Tariq Elahi, and Ian Goldberg. 2013. The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. In *Privacy Enhancing Technologies - 13th International Symposium, PETS*.

[3] Robert Annessi and Martin Schmiedecker. 2016. Navigator: Finding faster paths to anonymity. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 214–226.

[4] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. 2014. (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*.

[5] Armon Barton and Matthew Wright. 2016. DeNASA: Destination-Naive AS-Awareness in Anonymous Communications. In *Proceedings on Privacy Enhancing Technologies*.

[6] Armon Barton, Matthew Wright, Jiang Ming, and Mohsen Imani. 2018. Towards Predicting Efficient and Anonymous Tor Circuits. In *27th USENIX Security Symposium, USENIX Security*.

[7] COIN-OR Linear Program Solver [n. d.]. COIN-OR Linear Program Solver. https://www.coin-or.org/Clp/index.html. ([n. d.]).

[8] Thien-Nam Dinh, Florentin Rochet, Olivier Pereira, and Dan S Wallach. 2020. Scaling Up Anonymous Communication with Efficient Nanopayment Channels. *Proceedings on Privacy Enhancing Technologies* 3 (2020), 175–203.

[9] Directory Specifications [n. d.]. Bandwidth-Weights Specifications, Section 3.8.3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. ([n. d.]).

[10] Matthew Edman and Paul Syverson. 2009. AS-awareness in Tor Path Selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*. 380–389.

[11] John Geddes, Rob Jansen, and Nicholas Hopper. 2013. How low can you go: Balancing performance with anonymity in Tor. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 164–184.

[12] John Geddes, Mike Schliep, and Nicholas Hopper. 2016. ABRA CADABRA: Magically Increasing Network Utilization in Tor by Avoiding Bottlenecks. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society (WPES)*.

[13] Hans Hanley, Yixin Sun, Sameer Wagh, Mung Chiang, and Prateek Mittal. 2019. DPSelect: A Differential Privacy Based Guard Selection Algorithm for Tor. *Proceedings on Privacy Enhancing Technologies* 2019, 2 (2019).

[14] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium (USENIX Security 16)*.

[15] Rob Jansen, Kevin S. Bauer, Nicholas Hopper, and Roger Dingledine. 2012. Methodically Modeling the Tor Network. In *5th Workshop on Cyber Security Experimentation and Test, CSET '12*.

[16] Rob Jansen and Nicholas Hopper. 2012. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'12*. Internet Society.

[17] Rob Jansen, Matthew Traudt, and Nicholas Hopper. 2018. Privacy-Preserving Dynamic Learning of Tor Network Traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1944–1961.

[18] Aaron Johnson, Rob Jansen, Aaron D. Jaggard, Joan Feigenbaum, and Paul Syverson. 2017. Avoiding The Man on the Wire: Improving Tor's Security with Trust-Aware Path Selection. In *24th Annual Network and Distributed System Security Symposium, NDSS*.

[19] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. 2013. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries *(CCS '13)*. 337–348.

[20] Katharina Kohls, Kai Jansen, David Rupprecht, Thorsten Holz, and Christina Pöpper. [n. d.]. On the Challenges of Geographical Avoidance for Tor. In *Proceedings of 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*.

[21] Zhihao Li, Stephen Herwig, and Dave Levin. 2017. DeTor: Provably Avoiding Geographic Regions in Tor. In *26th USENIX Security Symposium (USENIX Security 17)*.

[22] Dong Lin, Micah Sherr, and Boon Thau Loo. 2016. Scalable and Anonymous Group Communication with MTor. In *Proceedings on Privacy Enhancing Technologies*.

[23] Akshaya Mani, T Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. 2018. Understanding Tor Usage with Privacy-Preserving Measurement. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. ACM.

[24] Zhuoqing Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. 2005. On AS-Level Path Inference. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*. ACM.

[25] Steven J. Murdoch and George Danezis. 2005. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (SP '05)*. 183–195.

[26] Steven J. Murdoch and Piotr Zielinski. 2007. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Privacy Enhancing Technologies, 7th International Symposium, PET*.

[27] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. 2016. Measuring and mitigating AS-level adversaries against Tor.

[28] Andriy Panchenko, Fabian Lanze, and Thomas Engel. 2012. Improving Performance and Anonymity in the Tor Network. In *31st IEEE International Performance Computing and Communications Conference (IPCCC)*.

[29] Jian Qiu and Lixin Gao. 2005. AS path inference by exploiting known AS paths. In *Proceedings of IEEE GLOBECOM*.

[30] Florentin Rochet, Aaron Johnson, Ryan Wails, Prateek Mittal, and Olivier Pereira. [n. d.]. Bandaid on entry guard selection. https://github.com/frochet/prop271_towards_loadbalancing/blob/master/xxx-bandaid-on-guard-selection.txt. ([n. d.]).

[31] Florentin Rochet and Olivier Pereira. 2017. Waterfilling: Balancing the Tor network with maximum diversity. *Proceedings on Privacy Enhancing Technologies*.

[32] Florentin Rochet and Olivier Pereira. 2018. Dropping on the edge: Flexibility and traffic confirmation in onion routing protocols. *Proceedings on Privacy Enhancing Technologies* 2018, 2 (2018), 27–46.

[33] Route Views [n. d.]. Routeviews Prefix to AS mappings Dataset for IPv4 and IPv6. https://www.caida.org/data/routing/routeviews-prefix2as.xml. ([n. d.]).

[34] Andrei Serjantov and George Danezis. 2002. Towards an Information Theoretic Metric for Anonymity. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002 (Lecture Notes in Computer Science)*, Vol. 2482.

[35] Micah Sherr, Matt Blaze, and Boon Thau Loo. 2009. Scalable Link-Based Relay Selection for Anonymous Routing. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*.

[36] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.

[37] Robin Snader and Nikita Borisov. 2008. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of 16th Annual Network and Distributed System Security Symposium*.

[38] Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. 2017. Counter-RAPTOR: Safeguarding Tor Against Active Routing Attacks. In *IEEE Symposium on Security and Privacy*. 977–992.

[39] The CAIDA AS Relationships Dataset, 2018-02-01 [n. d.]. The CAIDA AS Relationships Dataset, 2018-02-01. https://www.caida.org/data/as-relationships/. ([n. d.]).

[40] The CAIDA UCSD AS to Organization Mapping Dataset, 2019-01-01 [n. d.]. The CAIDA UCSD AS to Organization Mapping Dataset, 2019-01-01. https://www.caida.org/data/as_organizations.xml. ([n. d.]).

[41] The Tor Project [n. d.]. Another algorithm for guard selection. https://gitweb.torproject.org/torspec.git/tree/proposals/271-another-guard-selection.txt. ([n. d.]).

[42] The Tor Project [n. d.]. CollecTor - Tor Project. https://metrics.torproject.org/collector.html. ([n. d.]).

[43] The Tor Project [n. d.]. Tor Directory Protocol. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. ([n. d.]).

[44] The Tor Project [n. d.]. Tor Metrics Portal. https://metrics.torproject.org/. ([n. d.]).

[45] Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. 2013. An Empirical Evaluation of Relay Selection in Tor. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013*.

[46] Ryan Wails, Yixin Sun, Aaron Johnson, Mung Chiang, and Prateek Mittal. 2018. Tempest: Temporal Dynamics in Anonymity Systems. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 22–42.

[47] Gerry Wan, Aaron Johnson, Ryan Wails, Sameer Wagh, and Prateek Mittal. 2019. Guard Placement Attacks on Path Selection Algorithms for Tor. *Proceedings on Privacy Enhancing Technologies* 2019, 4 (2019).

[48] Tao Wang, Kevin S. Bauer, Clara Forero, and Ian Goldberg. 2012. Congestion-Aware Path Selection for Tor. In *Financial Cryptography and Data Security - 16th International Conference, FC.*

[49] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies* 2016 (2016).

Notation	Description
A	anonymity scores
α (C.R.)	Percent resilience incorporated into relay weight
α (CLAPS)	Allowable anonymity reduction factor
B_r	Bandwidth of relay r
β	Maximum bandwidth in all positions at once
γ	Allowed cluster distance increase
$d(\ell_1, \ell_2)$	Distance between ℓ_1 and ℓ_2 when clustering
D_ℓ	Client density in location ℓ
$\delta_\ell^{\pi_i}$	Positional client density
$EC(C)$	Entropy of the distribution of users in countries in C
$EL(C)$	Entropy of the distribution of users in locations in C
$\mathcal{L}$	Set of client locations
$\mathcal{L}_p$	Locations when choosing relay in position p
$\Lambda(\ell, r)$	Positional location after choosing relay r from ℓ
$\lambda_{\ell\ell'}$	Probability to choose positional location ℓ' from ℓ
$\omega_r^{\pi_i}$	Density weight at relay r in position π_i
$\mathcal{P}$	Circuit positions: g, m, and e
$P_{\ell r}^p$	Penalty for relay r in position p from location ℓ
π	Sequence of positions during circuit relay selection
$R_{\ell,r}$	BGP Hijack resilience of relay r from location ℓ
$\mathcal{R}$	Set of relays
$\mathcal{R}_p$	Relays admissible in position p
Size(C)	Total density contained in set of locations C
τ	DeNASA e-select parameter
θ_p	Weight increase limit in position p over Vanilla Tor
v_r^p	Vanilla weight for relay r in position p
V_ℓ^p	Expected penalty of p from ℓ with vanilla weights
$w_{\ell r}^p$	Weight for relay r in position p from position ℓ

Table 2: Common notation used in this work.

A DESIGN DETAILS

A.1 Notation

Table 2 summarizes common notation used in this work.

A.2 Weight computation

LP$_1$ in Figure 7 computes the maximum bandwidth β achievable in all positions simultaneously. β takes the output value of b from LP$_1$.

We also give some detailed definitions for certain variables.

For $\ell' \in \mathcal{L}_{\pi_i}$, $\lambda_{\ell\ell'}$ is the probability that a client in $\ell \in \mathcal{L}_{\pi_j}$ chooses relays for the jth to $(i-1)$st positions that yield positional location $\ell' \in \mathcal{L}_{\pi_i}$. This can be expressed recursively as

$$\lambda_{\ell\ell'} = \sum_{\ell'' \in \mathcal{L}_{\pi_{i-1}}} \lambda_{\ell\ell''} \sum_{r \in \mathcal{R}:\, \Lambda(\ell'', r) = \ell'} \frac{w_{\ell'' r}^{\pi_i}}{\beta}, \qquad (13)$$

where, for $i = j$, $\lambda_{\ell\ell'} = 1$ if $\ell = \ell'$ and $\lambda_{\ell\ell'} = 0$ otherwise.

$V_\ell^{\pi_i}$ is the expected penalty in π_i from positional location $\ell \in \mathcal{L}_{\pi_j}$, $j \leq i$, when using the vanilla weights to choose the jth to ith positions in π order. Recall that v_r^p is the vanilla weight for relay r in position p. Then let $\lambda_{\ell\ell'}^v$ be the probability to choose positional

Maximize $$b \quad (8)$$

Maximize the total bandwidth in any position.

Subject to

$$\forall r \in \mathcal{R}: \qquad \sum_{p\in\mathcal{P}} w_r^p \le B_r \quad (9)$$

Limit each relay's total weight to its bandwidth.

$$\forall p \in \mathcal{P}: \qquad \sum_{r\in\mathcal{R}} w_r^p \ge b \quad (10)$$

Allocate at least b weight to each position.

$$\forall p \in \mathcal{P}, r \in \mathcal{R}: \qquad w_r^p \ge 0 \quad (11)$$

Require weights to be non-negative.

$$\forall p \in \mathcal{P}, r \in \mathcal{R}\backslash\mathcal{R}_p: \qquad w_r^p = 0 \quad (12)$$

Set weight to zero when inadmissible in the position.

Figure 7: Linear program LP$_1$, used to maximize the minimum positional bandwidth. Its variable are (1) $\{w_r^p\}_{p\in\mathcal{P}, r\in\mathcal{R}}$, where w_r^p is the bandwidth relay r allocates to position p; and (2) b, which is the minimum bandwidth in any position.

location ℓ' from ℓ when using vanilla weights (i.e., with $v_r^{\pi_i}/\sum_s v_s^{\pi_i}$ in place of $w^{\pi_i}_{\ell'' r}/\beta$ in the definition of $\lambda_{\ell\ell'}$). Then

$$V_\ell^{\pi_i} = \sum_{\ell'\in\mathcal{L}_{\pi_i}} \lambda^v_{\ell\ell'} \sum_{r\in\mathcal{R}} \frac{v_r^p p^{\pi_i}_{\ell' r}}{\sum_{s\in\mathcal{R}} v_r^p}. \quad (14)$$

$\delta_\ell^{\pi_i}$ is the client density of positional location $\ell \in \mathcal{L}_{\pi_i}$. Given weights $w_{\ell r}^{\pi_{i-1}}$ for the $(i-1)$st position (e.g., as obtained from LP$_2$), it can be expressed recursively as

$$\delta_\ell^{\pi_i} = \sum_{\ell'\in\mathcal{L}_{\pi_{i-1}}} \sum_{r\in\mathcal{R}:\ \Lambda(\ell',r)=\ell} \frac{\delta_{\ell'}^{\pi_{i-1}} w_{\ell' r}^{\pi_{i-1}}}{\beta}. \quad (15)$$

$\omega_r^{\pi_i}$ is the sum of the weights of relay r in position π_i weighted by the positional location densities. It can be expressed as

$$\omega_r^{\pi_i} = \sum_{\ell\in\mathcal{L}_{\pi_i}} \delta_\ell^{\pi_i} w_{\ell r}^{\pi_i}. \quad (16)$$

Constraints 17 and 18 are added to LP$_2$ if a solution is not initially found for all positions.

$$\forall j > i, \ell \in \mathcal{L}_{\pi_1}: \qquad \sum_{\ell'\in\mathcal{L}_{\pi_i}} \lambda_{\ell\ell'} \sum_{r\in\mathcal{R}} \frac{w^{\pi_i}_{\ell' r}}{\beta} V^{\pi_j}_{\Lambda(\ell',r)} \le V_\ell^{\pi_j} \quad (17)$$

$$\forall j > i, \ell \in \mathcal{L}_{\pi_i}: \qquad \frac{w^{\pi_j}_{\ell r}}{\beta} = \frac{v_r^{\pi_j}}{\sum_{s\in\mathcal{R}} v_s^{\pi_j}} \quad (18)$$

B PATH SELECTION ALGORITHMS

Table 3 presents a summary of many proposed Tor path-selection algorithms. We organize the algorithms into two broad categories: (1) *security focused*, and (2) *performance focused*. The security-focused algorithms primarily aim to improve an aspect of Tor's security (e.g., by preventing BGP hijack attacks), while the security-focused algorithms aim to improve Tor's performance (e.g., by having clients choose short paths through the network). Many of these algorithms are location-aware, denoted by the check mark in the rightmost column of the table.

C DENASA & CLAPS-DN

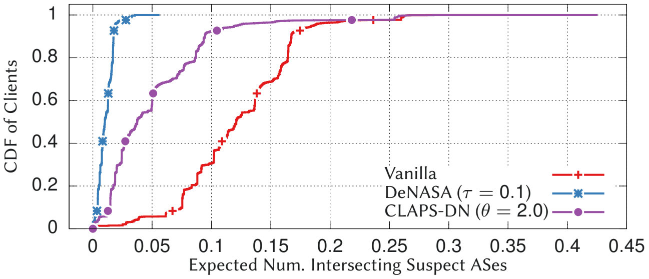


Figure 8: Expected number of suspects present on both the client-guard and exit-destination path when building a random circuit.

In this appendix section we provide a brief analysis of Vanilla Tor, DeNASA, and CLAPS-DN when AS paths are inferred with CAIDA's AS relationship topology from January 2019. We run security experiments in accordance with the methodology set out in Section 4 to estimate the likelihood of a suspect AS appearing on a client's client-guard path and exit-destination path. (For this analysis, our clustering algorithm chose $n = 14$ clusters.

Figure 8 shows the expected number of suspects present on both the client-guard path and exit-destination path for Vanilla Tor, DeNASA, and CLAPS-DN (analogous to Figure 4). We see a similar relationship presented in Section 4: DeNASA offers the best protection against the suspect ASes, but CLAPS-DN offers a significant improvement over Vanilla Tor and is not vulnerable to Tempest and relay placement attacks.

D NOTES ON EVALUATING PERFORMANCE

D.1 Changing Shadow Defaults

Default options for Tor simulation had to be changed to make more circuit selections and make a more statistically relevant use of the Tor network, including a reduction of the error rate. We did not change the client model; we changed how would Tor react when receiving new streams to handle. We made it closer to how Tor behaves with Tor Browser: using a different circuit for each destination or destination port. In effect, the simulation sampled and used 10× more circuits with the same user model. Intuitively, it should make the sampled circuit distribution closer to the theoretical true distribution offering load-balancing with proportional use of each relay.

To validate our approach we compared our Shadow configuration with the default configuration. Shown in Figure 9, we observed a meaningful impact on the performance results, much greater than the discrepancy from two different instances of a same experiment. Then, we compared the empirical distribution of relay selection to

Table 3: Proposed Tor path selection algorithms

Algorithm	Goal	Location Aware
Security Focused		
Edman-Syverson [10]	Prevent traffic analysis	✓
DistribuTor [4]	Prevent traffic analysis	
DeNASA [5]	Prevent traffic analysis	✓
Astoria [27]	Prevent traffic analysis	✓
TAPS [18]	Prevent traffic analysis	✓
DeTor [18]	Avoid geolocations	✓
Waterfilling [31]	Prevent traffic analysis	
Counter-RAPTOR [38]	Prevent BGP hijacks	✓
DPSelect [13]	Limit path selection info leakage	✓
TrilateraTor [20]	Avoid geolocations	✓
Performance Focused		
Coordinate [35]	Improve e2e connection throuput	
LASTor [1]	Improve latency, prevent traffic analysis	✓
Panchenko'12 [28]	Rebalance Tor	✓
Conflux [2]	aggregated bw via multipath	
ABRA [12]	Avoid congested relays	
CAR [48]	Avoid congested relays	✓
PredicTor [6]	Predict fast paths	
NavigaTor [3]	Measurement feedbacks to discard slow circs	✓
Mixed		
Snader-Borisov [37]	Improve load balance	

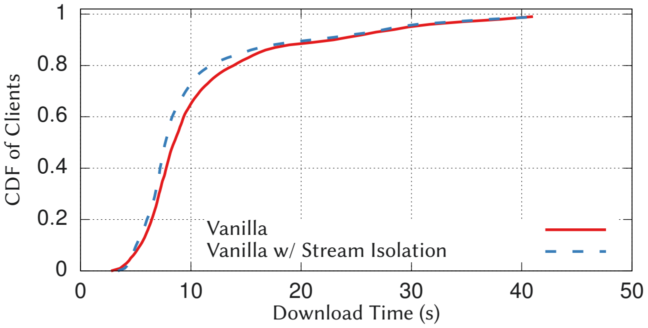


Figure 9: Shadow Tor experimentation with 250 relays and 2400 clients. The only difference is the SocksPort option: one simulation turns on stream isolation by Addr and Port (similar to what Tor browser does. Tor browser also multiplexes stream by same-origin policy)

the ideal (relaxed from Family and /16 constraints) distribution of relay selection. We observed our configuration changes to make the usage of relays closer to the theoretical (and ideal) distribution.

D.2 Details on Path Quality

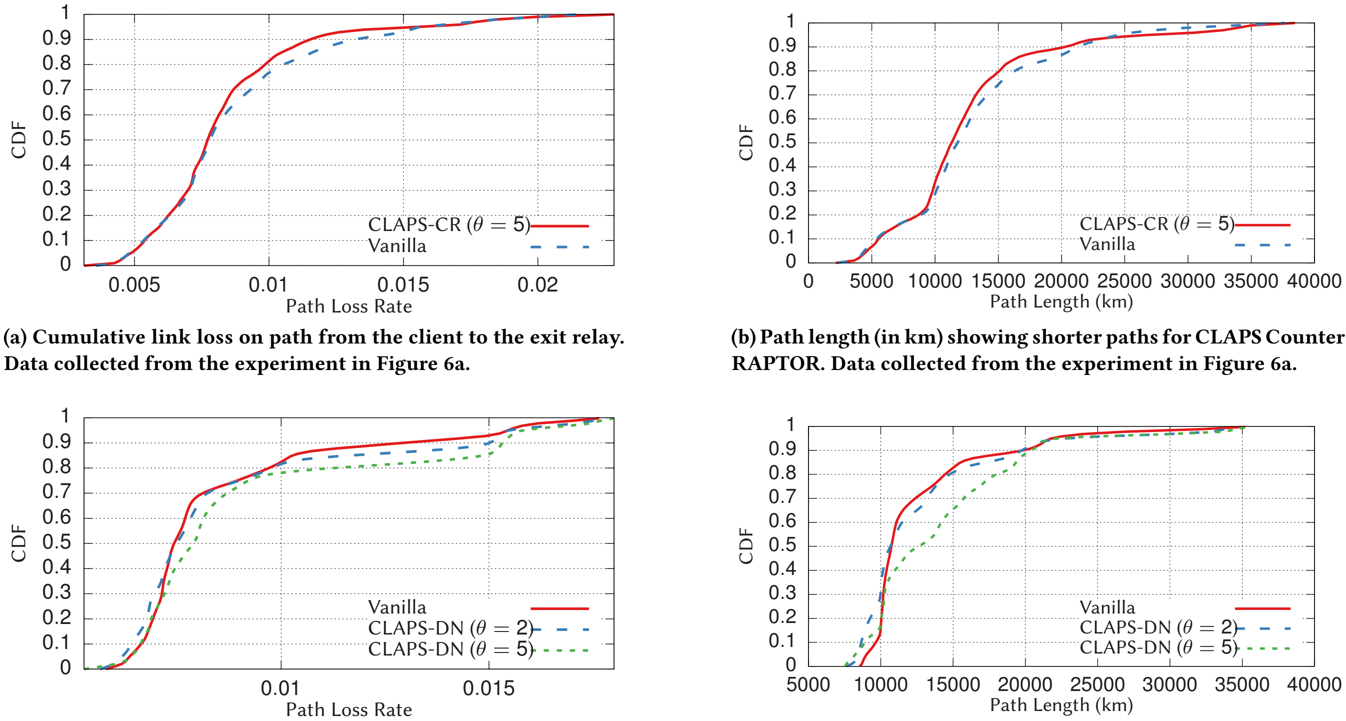
During our experiments, we observed that CLAPS could deviate from the expected performance results (i.e., similar to Vanilla Tor) despite the load-balancing constraint being satisfied. Recall that load-balancing in Vanilla Tor and in CLAPS is only expressed as a function of the bandwidth and the overall bandwidth for each path position. Our hypothesis for explaining these differences (sometimes better, sometimes worse) lays into the quality of paths that the algorithm is selecting regarding metrics that aren't part of the load-balancing logic but could influence performance results. In Shadow, all relays and clients are connected (it is a complete topology), and each edge has a latency and a loss rate associated with it. We measured those values from the paths built in the simulation to observe whether a discrepancy into those values could explain our performance results. Figure 10 shows that it is possibly the case. Note that we do not show the path latency figures because loss and latency are directly proportional in Shadow, hence the graphs are similar. Loss is more meaningful to show for the context of time to download, since this value influences TCP's performance.

E NOTES ON PATH SELECTION

E.1 Performance Challenges

We claim that previous location-aware schemes (e.g., Counter-RAPTOR and DeNASA[5, 38]) can display arbitrarily worse performance compared to Vanilla Tor; however, many of these schemes have been demonstrated in previous works to have satisfactory performance characteristics. To understand our claim and our results, we illustrate with the following example

Suppose that the network's capacity is composed of 10 units of guard-flagged relays, 2 unit of unflagged relays, 1 unit of exit-flagged relays, and 2 units of guard-and-exit flagged relays. The bandwidth-weight equations compute the position weights such that multi-roles relays' bandwidth are affected most where needed to reach an equilibrium: 4 units of guard-flagged relays are used in the middle position, and all units of guard-exit relays are used in



(a) Cumulative link loss on path from the client to the exit relay. Data collected from the experiment in Figure 6a.

(b) Path length (in km) showing shorter paths for CLAPS Counter-RAPTOR. Data collected from the experiment in Figure 6a.

(c) Cumulative link loss on path from the client to the exit relay. Data collected from the experiment in Figure 6f.

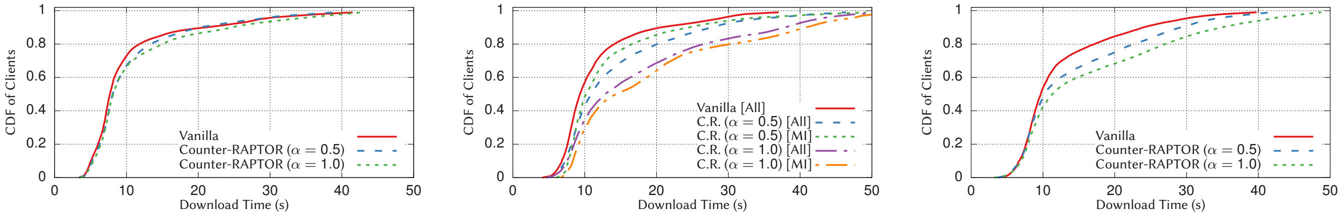
(d) Path length (in km) showing longer paths for CLAPS DeNASA. Data collected from the experiment in Figure 6f.

Figure 10: Path information collected from experiments conducted during CLAPS's evaluation. Those results show the impact of the guard-placement attack constraint θ on the length, loss, and latency of the paths built during the experiments.

exit position. The network resource ends-up to be probabilistically distributed to (6, 6, 3) for (entry, middle, exit) positions of. This proportion is happening for our simulated network from June 2017 (i.e., Figure 11a) and on more recent consensuses as well. The state of the Tor network is what explains Figure 11a, where the network has a very constrained exit position with half of the capacity in total compared to the entry and middle positions, generating congestion at the exits. Therefore, the entry and middle relays have spare bandwidth, or in other words, have a lower load than the exit relays. As a consequence, it is likely that relays involved in entry or middle positions (or both) can be unbalanced (to some extent) without impacting the overall performance too much, leading to Counter-RAPTOR's ostensible good results, similar to other algorithms and analysis in the literature [5, 13, 21].

A first example of non-versatility performance problem of previous location-aware path selection is linked to the distribution of user locations. Previous works ignore the density of Tor usage per client location which can lead to arbitrary performance for the overall Tor users, and arbitrarily low performance from a location in which a high density of users appears. Figure 11b shows a 20% increase of Tor users from a given location and displays unusable download times for a large fraction of them. Note that such a spike of usage already happened several times in the Tor network.

We give in Figure 11c a second example of the non-versatility issue based on a different network state that existed during several years in the past, and that may again exist in the future. Figure 11c shows a simulation scaled down from a consensus of March 2015, where the network resources can be potentially distributed equally between each position (which is the bandwidth-weights equations objective of Vanilla Tor). In such a network state, a path selection algorithm that unbalances one of the path positions (the entry, in the case of Counter-RAPTOR) would yield increased congestion and reduce the expected bandwidth, as depicted in Figure 11c. That is, Counter-RAPTOR performs worse because there is no spare resources in this network state to absorb the impact of the blending.



(a) June 2017 network state simulated with 2400 Tor clients and 250 relays.

(b) June 2017 network state simulated with 2400 Tor clients for which 20% of them are assigned to a particular location, and the others are distributed according to available online metrics. Uses the same 250 relays as in (a).

(c) March 2015 network state simulated with 2400 Tor clients and 250 relays. Compared to (a), this network state can be perfectly load-balanced, hence has no spare bandwidth in guard and middle positions.

Figure 11: These figures show the impact of congestion for Counter-RAPTOR path selection given different network states.