PRIVACY-PRESERVING NETWORK CONGESTION CONTROL

BY

HUSSEIN DARIR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Geir E. Dullerud

ABSTRACT

Cyber-physical technology is applied in various domains and connect computation with physical processes. Distributed cyber-physical networks have had a major impact in the area of networking and communications, more specifically on the Internet.

The Internet nowadays is an important tool of communication, however one of the main challenges facing users on the Internet is maintaining privacy, especially when facing widespread surveillance.

Anonymity networks have emerged as a solution to this problem by allowing users to conceal their identities online. The most successful anonymous communications network to date is currently the Tor network. It is operated by volunteers around the world and has many users worldwide. However, the trade off between performance and anonymity has always been a major problem for this type of networks.

Users' traffic in Tor is routed across a series of servers; each user's path going through the network transits three of them. This process of path selection creates a load balancing problem that could lead to network congestion. Congestion may deteriorate the network performance and service quality resulting in queuing delay, data packet loss and the blocking of new connections.

In this work, we study the problem of load-balancing in path selection in anonymous networks such as Tor. We first find that the current Tor path selection strategy can create significant imbalances. We then develop a (locally) optimal algorithm for selecting paths and show, using flow-level simulation, that it results in much better balancing of load across the network.

Our initial algorithm uses the complete state of the network, which is impractical in a distributed setting and can compromise users' privacy. We therefore develop a revised algorithm that relies on a periodic and differentially private summary of the network state to approximate the optimal

assignment. Our simulations show that the revised algorithm significantly outperforms the current strategy while maintaining provable privacy guarantees [1].

To my parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost, the utmost gratitude goes to my parents. Nothing would have been possible without their love and support. They worked hard and risked everything unconditionally to ensure my well-being and my success. I would like to thank my brother Ali for being the kind-hearted and caring brother, and Dalia for loving and supporting me along my journey. Without my family, I would have never fulfilled this accomplishment.

I would like also to thank my advisor Professor Geir Dullerud for offering me the chance to work in his research group. His continuous support and help as well as his guidance and advice during the past years were of utmost importance to me. This thesis would not have been accomplished without him. I want to thank him for being the ideal academic advisor as well as a life mentor. Also, I would like to thank Professors Nikita Borisov and Sayan Mitra for their plentiful collaboration which had a major impact on the ideas and methods presented in this thesis.

Moreover, I would like to thank my friend and colleague Hussein Sibai for being the go-to friend and supportive mentor in my research, without his contribution and experience, this work would not have been achieved. I also want to thank my friends: Nabil Ramlawi, Hasan Dbouk, Patrick Birbarah and many more from the Lebanese community at UIUC for their invaluable companionship.

Finally, I want to thank National Science Foundation (NSF) for supporting this work through a research grant (Grant No. 1739966).

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Cyber-physical systems connect the physical world to the world of communication and computation [2]. This integration must be done efficiently, securely and safely. Cyber-physical technology is applied in different domains such as critical infrastructure control, alternative energy, manufacturing and social networking [3]. Of particular importance is the application of cyber-physical technology in the Internet where the anonymity of users is an important property that should be addressed.

Throughout human history, methods of communication seem to be equipped with different tools guaranteeing anonymity. In fact, the Internet is no exception. By nature, users of the Internet are shielded behind their computers and are able to communicate without being limited by geographical boundaries [4]. Users around the world take advantage of this features for various purposes.

However, the Internet does not provide the level of anonymity that it might appear to guarantee. Users' messages sent from a computer are tagged with an address that enables the recipient to respond. Combining this address with some information from the service provider can allow the identification of the source. As an example, communications happening over higher-level protocols, such as email, usually contain information allowing the tracking of a message back to the sender.

Nowadays, users are increasingly turning to anonymous communication networks to protect themselves from surveillance, online tracking, or government censorship.

Anonymizing services, such as the Tor network (fig. 1.1), tackle the issue of tracking. In order to achieve anonymity, users' traffic in the network are

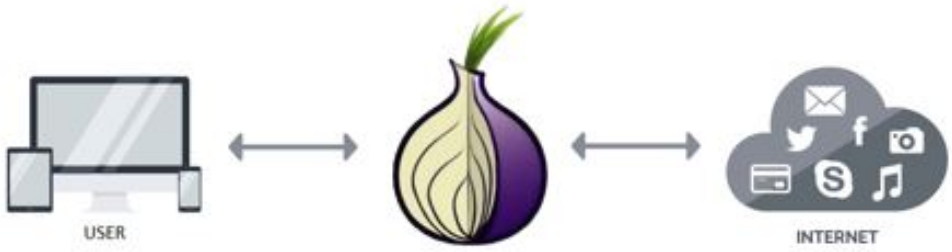

Figure 1.1: Anonymous communication networks: Tor.

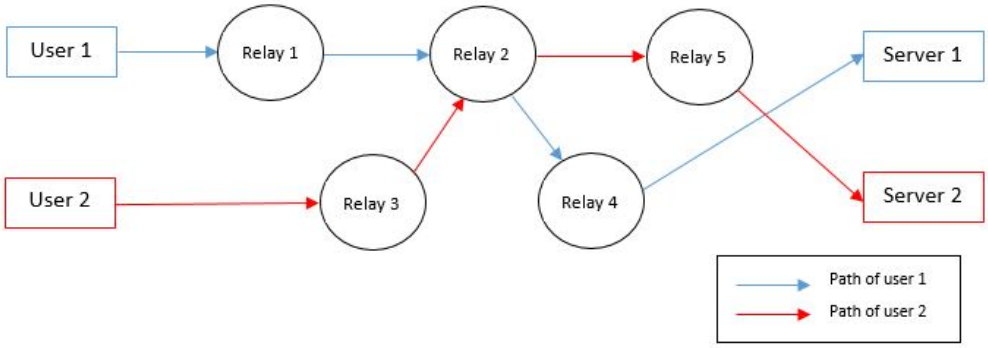

Figure 1.2: Users paths and servers in Tor.

encrypted within multiple layers of encryption.

The Tor network has several million daily active users [5] and has recently been integrated into the privacy-focused Brave browser [6].

To achieve anonymity in Tor, users' traffic is routed across a series of servers, called *relays*. There are several thousand relays, run by volunteers; each user's path through the network, called a *circuit*, typically transits three of them, as depicted in fig. 1.2. Each relay sees the immediate previous relay as the origin and the immediate next relay as the destination. Thus, no relay knows both the true origin and destination of the packet. However, this creates a load-balancing problem of assigning circuits to relays while ensuring no relay gets overloaded and all circuits receive good performance. Complicating this problem are the highly heterogeneous relay capacities—spanning some five orders of magnitude—and the privacy requirements of circuit construction. In particular, no one except the user must know the entirety of the circuit, precluding any centralized load-balancing solution.

Currently, the Tor network uses a randomized assignment of flows cir-

cuits to relays, where each user chooses the relays for their circuits randomly weighted by their measured capacity (with some other constraints, see section 2.1). This ensures that each relay has the same *average* load; however, as we will demonstrate in section 3.1, this can create significant imbalances at any given point in time. We therefore consider the question of whether it is possible to provide better load balancing while satisfying the privacy requirements.

In this thesis, we first study a non-private load-balancing algorithm. We adapt an algorithm that calculates the max-min-fair allocation of bandwidth to circuits to select an optimal set of relays for a new path. We show that this results in significantly better load-balancing. Running this algorithm for each new flow would be impractical, therefore we create a batch version of the algorithm, which speculatively generates new circuits using the optimal algorithm and uses these circuits to induce a distribution over the relays, which is then sampled from to generate new circuits. Note that, unlike the Tor algorithm, the distribution reflects the *current* state of the network and is periodically refreshed; we show that this results in significantly better load-balancing performance than the Tor algorithm.

We then design a private version of this algorithm. Rather than working with a list of circuits, we break each hop into two 2-hop *segments*. We can then summarize the state of the network by creating a histogram of segments with one entry for each pair of relays. Our design is motivated by the fact that each entry in this histogram can be filled in by a single relay; moreover, each relay can *locally* add noise to the entry resulting in a differentially private histogram. We show that using a private histogram, we can implement a modified batch algorithm that approximates the optimal load-balancing. Our experiments show that the algorithm results in significantly better balanced circuits than the Tor randomized approach, while preserving privacy.

1.2 Contributions

In this thesis, our main contributions are as follows:

- We first show through flow-level simulation, that the current Tor path selection strategy can create significant imbalances.

- We develop a (locally) optimal algorithm for selecting paths and show, using flow-level simulation, that it results in much better balancing of load across the network.
- We develop a revised algorithm that relies on a periodic, differentially private summary of the network state to approximate the optimal assignment.
- We show through flow-level simulation, that the revised algorithm significantly outperforms the current strategy while maintaining provable privacy guarantees.

1.3 Related Work

There is a significant amount of work dealing with the different aspects of Tor performance. Hundreds of thousands of daily users currently use the Tor [7] anonymity network in order to ensure and enhance the privacy of their communications [8]. Significant effort has been initiated in order to improve different aspects of performance of the Tor network, which suffers from high congestion and latency [9]. Those efforts tackle different features of the network; specifically improving Tor's circuit processing [10, 11], relay recruitment [12, 13], transport mechanism [14, 15], and relay selection [16, 17, 18, 19, 1]. This section surveys previous research papers that investigated the relay selection problem in Tor and aimed to improve it by trading off its performance with anonymity.

The survey by AlSabah and Golberg [20] covers different aspects of Tor performance and examines the design of the Tor network. The survey identifies some of the shortcomings of Tor's design, and based on the outlined weaknesses presents classification of research directions and ongoing work. One of the main problems outlined in the survey is the inconvenient performance that manifests itself in the form of large and highly variable delays and download times experienced during web surfing activities.

As stated in section 1.1, users of Tor select a source routed circuit formed by three relays by querying any one of several authoritative directories. After choosing the relays and constructing the circuit, the user's traffic is forwarded

through this circuit using a layered encryption scheme based on onion routing [21].

Snader and Borisov tackled the problem of path selection in Tor and suggested biasing selection towards higher bandwidth relays, showing that it improved performance in both simulation and real-world Tor measurements [18, 22]. Herbert et al.proposed a relay selection algorithm for optimizing the queuing latency while modeling Tor traffic as an M/D/1 queuing network [23]. Both these papers assert that the selection of relays weighted by their bandwidth results in suboptimal path selection, however their solutions did not include any feedback mechanisms.

Wang et al. presented a congestion-aware path selection algorithm [24]. Their algorithm proposed performing latency measurements by users on circuits in the network, in order to identify overloaded relays and avoid selecting them during the path selection process. Even though each user will have a partial view of the network, the experiments conducted have shown that significant improvements can be realized. Likewise, Conflux [25] aims to reduce network congestion by multiplexing traffic across two paths through the Tor network. Although performance can be improved by using this technique however the last relay in the circuit cannot be mitigated since this is where the traffic must converge. Both of the aforementioned scheme enable the user to have a partial view of the network in order to detect and avoid congestion rather than balancing load across the entire Tor network, as in the scheme that will be presented in this thesis.

Sherr et al. [26] suggest replacing the relay-based path selection process with a link-based one. According to their observations, choosing paths based on link characteristics such as the number of traversed Autonomous Systems (ASes) or latency, can enhance the performance of the system.

LASTor [16], proposed by Akhoondi et al., presents a new approach for selecting path in the Tor network. Instead of basing the selection process on relays' capacity or links' latency, LASTor is a weighted shortest path algorithm, where routing decisions are based on the geographical distance between the source and the destination. One main advantage of this algorithm over the others is that only user-side updates are required instead of the expensive router updates.

Wacek et al. [27] assess the performance and security of the aforementioned path selection algorithms. To realistically model the live Tor network,

an emulated scaled-down Tor network is used. Their results show that the congestion-aware algorithm, proposed by Wang et al. has the best performance when compared to the others, while also preserving anonymity. On the other hand, LASTor has the poorest performance but induces the highest anonymity guarantees.

Chen et al. [28] also studied the problem of path selection by studying the performance impact of changing the number of relays in a circuit. Their observations show that reducing the number of relays in a path and the geographic distance between relays helped improve performance and reliability. However those changes caused a reduction in the anonymity guarantees of the network.

Despite the growing interest in the problem of path selection techniques in the Tor network, none of the proposals presented have been evaluated under more realistic conditions that reflect the ones encountered in a live anonymity network. In fact, it is shown [18, 11] that the performance improvements attained under simulations and modeling are not usually manifested when the algorithms are tested under more realistic conditions [29, 30]. Although advantageous effects of a given algorithm may be experienced when tested for a small number of clients or relays, however unexpected negative consequences may arise when applied on a larger scale network.

In order to achieve load balancing in Tor, an accurate measure of relay capacity is required, which is currently performed by TorFlow [31]. TorFlow uses bandwidth authorities to proactively measure relay performance, and incorporates a long-term feedback mechanism: relays that are assigned too high a bandwidth value and become overloaded will perform worse in subsequent measurements and thus have their consensus weight reduced, and vice versa. This feedback, however, occurs over a period of days and does not deal with more transient congestion and load imbalances. EigenSpeed [32] proposed an alternate measurement approach that relied on opportunistic measurements of relays by other relays. In EigenSpeed, each Tor relay reports the speed of its connections to other relays. The authority receiving those measurements, applies Principal Component Analysis (PCA) in order to generate bandwidth estimates. Johnson et al. identified several attacks on both TorFlow and EigenSpeed and proposed an improved peer measurement scheme called PeerFlow [33].

Capacity estimation techniques are susceptible to attacks, Karame et al.

[34] describe those attacks and propose relying on a trusted network hardware to secure these measurements. Suselbeck et al. [35] suggest estimating the capacities based on active traffic injection and passive measurements. Haeberlen et al. propose the PeerReview system [36], that detects misbehavior in a distributed system while using information related to relays' activities and actions.

Anonymity is the principle objective of this type of networks. However, as most anonymous communication networks, Tor is vulnerable to traffic correlation attacks. An adversary controlling a certain number of relays in the network, can apply any one of many known traffic analysis attacks [37, 38] in order to correlate the source and destination.

Chapter 2

PRELIMINARIES

In this chapter, we review the key properties of anonymous communication networks relevant to load balancing. We then present the max-min fair bandwidth allocation algorithm that we will use to model the load-balancing performance, and introduce the differential privacy framework that will be used to maintain users' anonymity.

2.1 Anonymous Networks and Tor

Anonymity networks provide users a way to communicate without revealing their identity, and without revealing their relationships to third parties. Starting with Chaum's seminal mix network design [39], anonymity has been most frequently achieved by forwarding traffic through a series of servers in order to disguise its origin. In onion routing networks [40], each packet is multiply encrypted, with a layer of encryption being removed by each server in the path. This makes it impossible for any server to learn the entire path; rather, it knows only the preceding and following hops.

In Tor [41], paths (called *circuits*) typically take three hops to transit the network. These hops are chosen from a collection of volunteer-run servers, called *relays*. These relays have vastly varying bandwidth capacity; in order to balance the load among them, their bandwidth is measured using TorFlow [31]. Relays are then allocated to circuits randomly, weighted by their measured capacity (also known as the consensus weight) (Revisited in section 3.1).

An important feature of the Tor network is that the relays, due to being supplied by volunteers, vary wildly in their bandwidth capacity. Figure 2.1 shows the relays and their measured capacity from two consensus documents take about five months apart. The distribution is highly skewed, with mea-

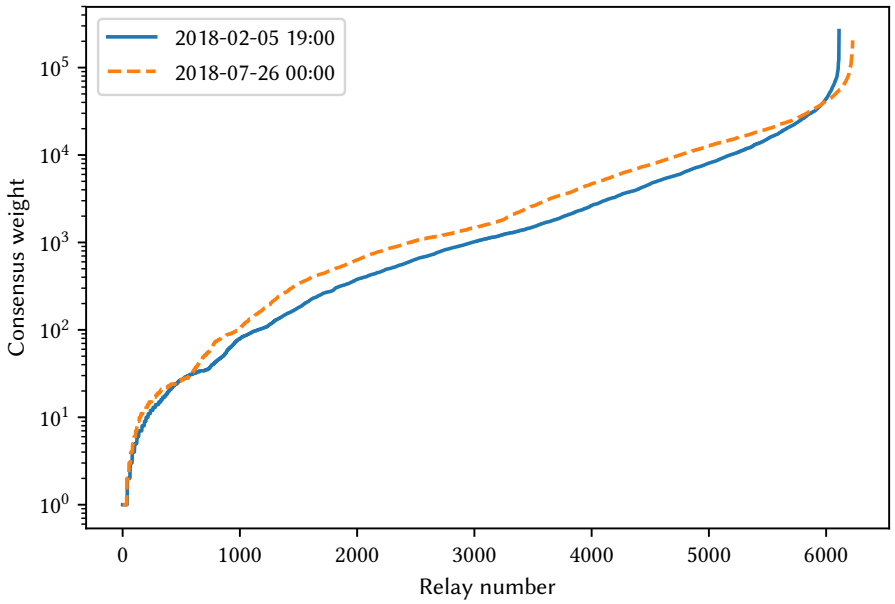

Figure 2.1: Measured relay bandwidths from two Tor consensus documents: February 5, 2018 at 19:00 (UTC) and July 26, 2018 at 00:00 (UTC). Note the logarithmic scale of the y-axis.

sured capacities spanning over five orders of magnitude. Note that the distribution in the two consensus documents follows a similar pattern; we therefore use the February 5, 2018 19:00 (UTC) consensus as representative for our experiments in this thesis.

2.2 Max-min Fairness

Each user's path in the network has a *bandwidth* limit that dictates the rate at which a packet is transferred from the source to the destination. This limit is imposed by the capacities of the relays forming each path. In order to compute paths' bandwidths an accurate model of the Tor network must be adopted.

In this section we introduce our model of Tor performance using max-min fair bandwidth allocation. Our model of the Tor network includes two simplifying assumptions: (a) each user holds a single path through relays and (b) path capacities are constrained only by the relays, and not by the

links between relays. The former can be easily adjusted by creating virtual users; the latter assumption is standard in analyzing Tor, and indeed central to the Tor bandwidth measurement and allocation architecture. We use the max-min fair allocation as a model because Tor schedules circuits in a round-robin fashion, which has been shown to achieve max-min fairness [42]. One further assumption is that each path is in simultaneous active use. We discuss some relaxations of this assumption in section 5.5, and defer more complex modeling and simulation of circuit usage to future work.

Notation. We introduce some notation for the rest of the thesis. For any positive integer k, $[k]$ denotes the set $\{1, \ldots, k\}$. We denote the number of relays by n and the number of users (and therefore the number of paths) by m. We assign integer identifiers to the relays and users, and thus, the sets of relays and users are $[n]$ and $[m]$, respectively. The *capacity* of relay $r \in [n]$ is the positive constant $C[r]$. The *path* assigned to user $p \in [m]$ is a sequence of three relays and is denoted by $P[p]$. We identify this sequence with the the p^{th} *path*. Given an allocation of paths to all users, for any relay $r \in [n]$ we define $R[r] = \{p \in [m] \mid r \in P[p]\}$ to be the set of identifiers of the paths to which the relay r belongs.

Each relay r allocates some bandwidth to each of the paths in $R[r]$. The bandwidth allocated to the p^{th} path is the minimum bandwidth allocated for it by its three relays and is denoted by $band[p]$. For any relay r, the total allocated bandwidth to all paths in $R[r]$, must be less than the capacity of r:

$$\forall\ r \in [n],\ \sum_{p \in R[r]} band[p] \leq C[r]. \tag{2.1}$$

Allocations satisfying eq. (2.1) are said to be *feasible*.

A feasible allocation *band* is *max-min fair* if and only if an increase of bandwidth allocation to any path (within the set of feasible allocations), must be at the cost of a decrease in allocation of another path with an already lower bandwidth in *band* (See Section 6.5.2 in [43]). That is, for any other feasible allocation $band'$ and any path $p_1 \in [m]$, if $band'[p_1] > band[p_1]$, then there exists $p_2 \in [m]$ such that $band'[p_2] < band[p_2]$ and $band[p_2] \leq band[p_1]$.

It is well-known that the allocation algorithm shown below (algorithm 1) achieves max-min fairness. It takes as input a network of relays and paths,

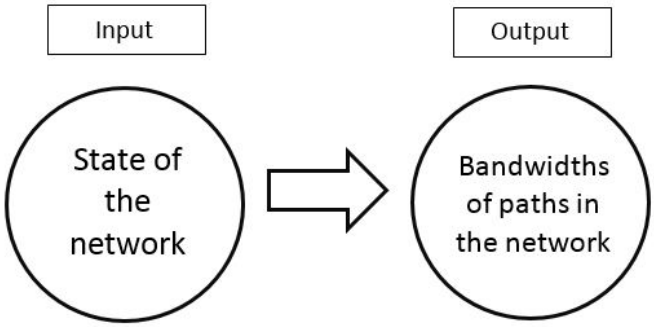

Figure 2.2: Max-min fairness allocation algorithm input and output.

and allocates a bandwidth to each path in an iterative fashion, see fig. 2.2.

Specifically, the inputs are the array or map C of all the relay capacities, the array of user paths P, and the array R. The algorithm keeps track of the *residual capacity*, *Cres*, of each relay after subtracting the bandwidths of the paths passing through it. It also keeps track of the *residual paths*, *Rres*, that is, the set of paths passing through each relay after removing those paths whose bandwidths are already allocated. At each iteration, one relay $r^* \in [n]$ is chosen and each path in $Rres[r^*]$ is allocated a bandwidth. The chosen relay r^* is the one that has the smallest ratio $Rat[r] := Cres[r]/|Rres[r]|$ at the corresponding iteration (line 7). After it is chosen, each of the paths in $Rres[r^*]$ is assigned a bandwidth of $Rat[r^*]$ (line 9). Relay r^* is called the *bottleneck relay* of these paths. Then, these paths are removed from their corresponding relays (line 12) and the capacities of these relays get subtracted by $Rat[r^*]$ (line 11). This is repeated until all paths are allocated bandwidth.

Remark 1. *Suppose the relay r^* chosen at line 7 of algorithm 1 belongs to a path p. Then, path p is allocated bandwidth of $Rat[r^*]$, that is $band[p] = Rat[r^*]$. Further, this allocation is not changed in subsequent iterations.*

2.3 Differential Privacy

To perform load-balancing, we would like to incorporate feedback about the state of the network into the path selection process. However, as discussed above, the state of the network is explicitly required to be private, as this is key to preserving users' anonymity. We will use differential privacy to ensure

Algorithm 1 Max-min Bandwidth Allocation Algorithm

input: C, R, P
$Cres[r] \leftarrow C[r], \forall r \in [n]$
$Rres[r] \leftarrow R[r], \forall r \in [n]$
$band[p] \leftarrow 0, \forall p \in [m]$
while $\exists\, p \mid band[p] = 0$ **do**
    $Rat[r] \leftarrow \begin{cases} \frac{Cres[r]}{|Rres[r]|} & \forall r \in [n] \mid |Rres[r]| \neq 0 \\ \infty & otherwise \end{cases}$
    $r^* \leftarrow \underset{r \in [n]}{\operatorname{argmin}} \; Rat[r]$
    for $p \in Rres[r^*]$ **do**
        $band[p] \leftarrow Rat[r^*]$
        for $r \in P[p]$ **do**
            $Cres[r] \leftarrow Cres[r] - Rat(r^*)$
            $Rres[r] \leftarrow Rres[r] \setminus \{p\}$
        end for
    end for
end while
return $band$

our feedback mechanism does not result in privacy loss.

Differential privacy was first proposed by Dwork [44]. It formalizes the notion that a mechanism operating over a private data set must produce an output that depends only minimally on each item in the data set. We will use the formulation given by Vadhan [45]:

Definition 1 ((Approximate) differential privacy)**.** *[45, Definition 1.4] For $\epsilon \geq 0, \delta \in [0, 1]$ we say that a randomized mechanism $\mathcal{M} : \chi^n \times \Omega \to \mathcal{Y}$ is (ϵ, δ)-differentially private if for every pair of neighboring datasets $x \sim x' \in \chi^n$ (i.e. x and x' differ in one row), and every query $q \in \Omega$, we have:*

$$\forall T \subseteq \mathcal{Y}, Pr[\mathcal{M}(x, q) \in T] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(x', q) \in T] + \delta,$$

where Ω is the set of possible queries. Moreover, δ should typically satisfy $\delta \leq n^{-\omega(1)}$ for this definition to be meaningful.

In our case, the dataset in question will be the complete list of circuits in the Tor network, with each circuit representing a row. As a result, differential privacy will guarantee the privacy of each individual circuit while providing aggregate traffic statistics. We note that differential private mechanisms have previously been used to study traffic properties of Tor [46].

Chapter 3

RANDOM ALLOCATION ALGORITHM

In this chapter, we will present the current method used to create paths in the Tor network. Using flow-level simulations, we will show that this method creates significant imbalances between users, more specifically at the level of the bandwidth allocated to each user's path.

3.1 Random Allocation Algorithm

As stated in section 2.1, relays are allocated to circuits randomly, weighted by their measured capacity (also known as the consensus weight). The full Tor path selection algorithm [47] is somewhat complex because it must account for some relays not being usable in certain positions of the circuit as well as other constraints. For the purposes of this work, we will approximate the algorithm as picking three random relays, without replacement, from the distribution induced by the measured capacities, leaving simulations of the full Tor algorithm for future work.

Inputs and outputs of the random allocation algorithm are shown in fig. 3.1.

3.2 Implementation of the Random Allocation Algorithm

In order to simulate the random allocation algorithm, we wrote a code using Python programming language (algorithm 2). This code generates paths by sampling without replacement three relays using the distribution over relays where each relay is weighed by its capacity.

We evaluate the performance of this algorithm with respect to the parameters of the Tor network. Jansen and Johnson [46] estimated that there were

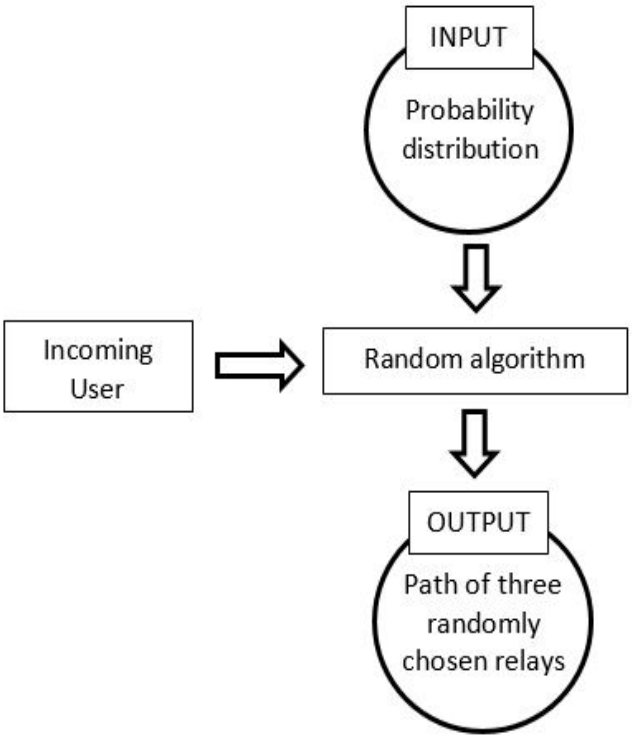

Figure 3.1: Random allocation algorithm inputs and outputs.

Algorithm 2 Random Path Allocation Algorithm

1: **input:** C
2: Sample three relays $\{r_1, r_2, r_3\}$ without replacement from the set of relay where each relay is weighed by its capacity.
3: **return:** $\{r_1, r_2, r_3\}$

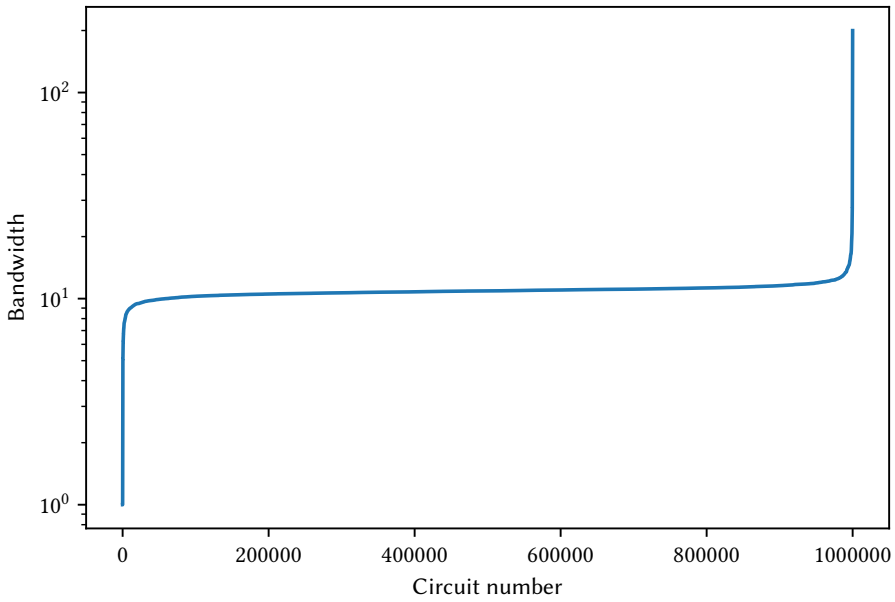

Figure 3.2: Bandwidth allocation of 1 million paths using the random algorithm 2.

approximately 1.2 million active circuits (95% CI +/- 500,000) in the network. We therefore created 1 million paths using the random algorithm 2 and then computed their bandwidth allocation using max-min fair algorithm 1. The results are in fig. 3.2. We can see that the majority of circuits receive an allocation close to the average of 10.9, but there are significant tails at both ends of the distribution: the minimum circuit has allocation of 1.0 and the maximum of 190. (The standard deviation is 1.28.)

Jansen and Johnson define an active circuit as one that has *ever* been used to forward data; however, at any given moment, most circuits are idle. This can be seen by comparing the data about the aggregate Tor network traffic of approximately 100 Gbps [48] to the performance of a sustained download, which is roughly 10 s for 5 MiB [49]. Given that each circuit gets carried by 3 relays, this suggests that $100\,\text{Gbps}/3/(5\,\text{MiB}/10\,\text{s}) \approx 10\,000$ circuits are active at any given time. Figure 3.3 shows the bandwidth allocation of 10 000 circuits generated by algorithm 2. Observe that the imbalances in this case are much more significant: the minimum allocation is 14 and the maximum is 6419, with a standard deviation of 475.8. 932 out of 10 000 flows receive less than half of the average bandwidth of 980, and 37 receive less than 10%.

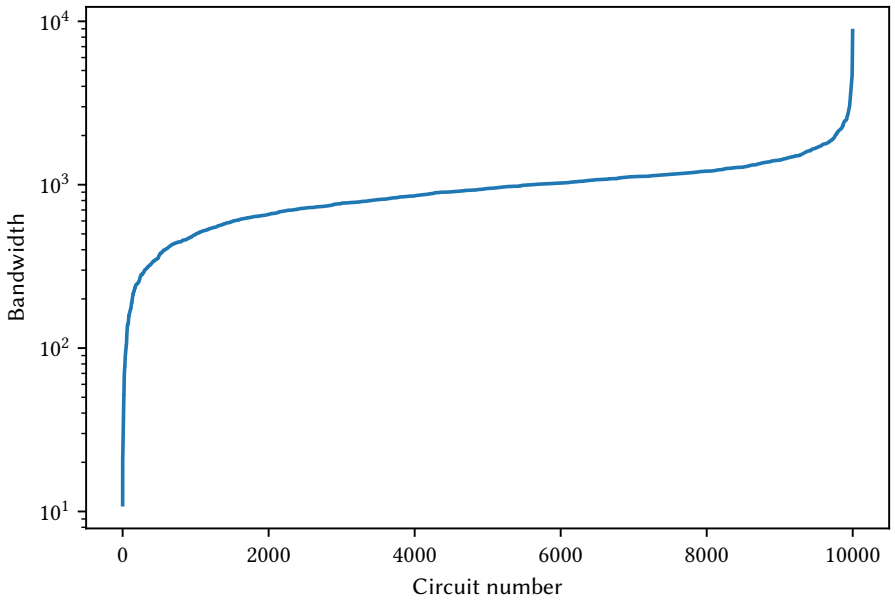

Figure 3.3: Bandwidth allocation of 10 000 paths using the random algorithm 2.

3.3 Conclusion

We have presented the current path selection algorithm, along with the code used to implement it (algorithm 2). We then generated 1 million and 10000 paths using algorithm 2 and computed their bandwidth allocation using algorithm 1. The results show the large imbalance between the bandwidths allocated to different paths in the network, depicted by the significant tails at both ends of the distributions obtained.

Chapter 4

LOCALLY OPTIMAL INCREMENTAL PATH ALLOCATION

In this chapter, we modify the max-min fair allocation algorithm 1 to design an algorithm that, given the state of the Tor network, returns three relays that would result in an optimal allocated bandwidth for a new path. We will then show, using flow-level simulations, that this algorithm results in a much better load-balancing between paths across the network than the previously presented algorithm 2.

4.1 Algorithm Description

The developed algorithm aims to find three relays in the set of relays $[n]$, forming the path with the highest possible bandwidth when added to the given network. We first develop an algorithm that takes as input the state of the network, i.e. the whole set of paths in the network along with the relays forming the paths (see fig. 4.1). The result is algorithm 3.

The developed algorithm assumes that the bandwidths of the existing paths in the network are allocated using max-min fairness (algorithm 1).

The algorithm iteratively creates a list B of $(bandwidth, relay)$-pairs. This list determines how much bandwidth would be allocated to a newly added path to the network. That is, of the relays appearing in a new path, the one that appears earliest in B determines the bandwidth of the path.

The idea of algorithm 3 is to simulate the behaviour of the max-min fairness algorithm 1 on the network when an arbitrary new path is added. This simulation allows us to know how much bandwidth it would get allocated. A trivial but key observation is that when a new path is added, the number of paths passing through each of its three relays will be incremented by one. For all other relays, the number of paths will remain unchanged. Moreover, as per remark 1, the relay that is chosen first from a path determines the

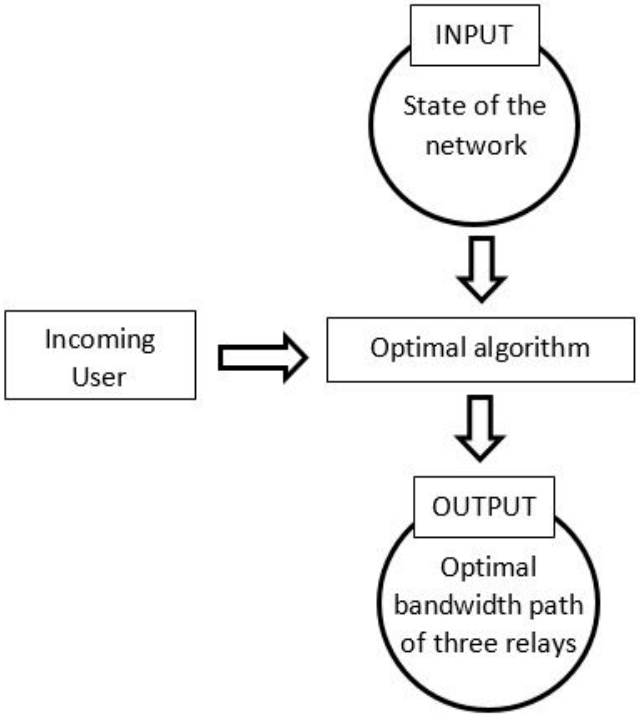

Figure 4.1: Optimal allocation algorithm inputs and outputs.

path's bandwidth allocation. Since we are searching for the relays that would maximize the bandwidth allocation for a newly added path, algorithm 3 computes the different possible allocations based on the different possible bottlenecks.

In addition to the ratio $Cres[r]/|Rres[r]|$ tracked in algorithm 1 (line 7), algorithm 3 also tracks $Cres[r]/\left(|Rres[r]|+1\right)$, for each relay r (line 8). In other words, Rat is now a $2 \times n$ matrix: row 1 stores $Cres[r]/|Rres[r]|$ and row 2 stores $Cres[r]/\left(|Rres[r]|+1\right)$.

At each iteration, a minimum of all the $2n$ ratios is chosen. We denote the minimizing row by $t^* \in \{1,2\}$ and the corresponding relay by $r^* \in [n]$. If the minimizing row $t^* = 1$, the algorithm proceeds as max-min fair allocation algorithm 1 by allocating bandwidth of $Rat[1,r^*]$ to each of the paths in $Rres[r^*]$. Then, it removes them from the other relays in which they pass (lines 17 and 18). Both ratios in the same column of Rat, i.e., ratios corresponding to the same relay, are updated in the same way unless the ratio in the second row is already added to B (line 8). That is because both ratios use the same arrays $Cres$ and $Rres$ for their numerator and denominator.

Otherwise, if $t^* = 2$, the pair $(Rat[2, r^*], r^*)$ is added to the end of B. It is added at the end since the r^* will be the bottleneck of the added path only when its other relays are not in already in B at this iteration. If one of its relays is already in B at this iteration, that would be its bottleneck instead of r^*. This will be proved in the section 4.2. The algorithm iterates until all n relays have entries in B.

Algorithm 3 Locally Optimal Path Allocation Algorithm

input: C, R, P
$B \leftarrow \emptyset$
$Cres[r] \leftarrow C[r], \forall r \in [n]$
$Rres[r] \leftarrow R[r], \forall r \in [n]$
$band[p] \leftarrow 0, \forall p \in [m]$
while $\exists\, i \notin B$ **do**
    $Rat[1, r] \leftarrow \begin{cases} \frac{Cres[r]}{|Rres[r]|} & \text{if } |Rres[r]| \neq 0 \\ \infty & \text{otherwise} \end{cases}$
    $Rat[2, r] \leftarrow \begin{cases} \frac{Cres[r]}{|Rres[r]|+1} & \text{if } r \notin B \\ \infty & \text{otherwise} \end{cases}$
    $(t^*, r^*) \leftarrow \underset{t \in \{1,2\}, r \in [n]}{\text{argmin}} \; Rat[t, r]$
    if $t^* == 2$ **then**
        $B.push([Rat[t^*, r^*], r^*])$
        $Rat[t^*, r^*] \leftarrow \infty$
    else
        for $p \in Rres[r^*]$ **do**
            $band[p] \leftarrow Rat[t^*, r^*]$
            for $r \in P[p]$ **do**
                $Cres[r] \leftarrow Cres[r] - Rat[t*, r*]$
                $Rres[r] \leftarrow Rres[r] \setminus \{p\}$
            end for
        end for
    end if
end while
Let $((b_1, r_1), (b_2, r_2), (b_3, r_3))$ be the last three elements of B
return: $\{r_1, r_2, r_3\}$

4.2 Algorithm Correctness

In this section, we will prove that the output of algorithm 3 is a path with maximum bandwidth allocation possible, for a new path that is to be added to the given network. In this analysis, we will compare the state of algorithm 3 with the state of max-min fair allocation algorithm 1, in the same iteration.

Notation. We will add bars on top of the variable names of algorithm 3 to distinguish them from the variables with the same names in algorithm 1. A subscript of zero refers to the initial values of the variables. A subscript $l > 0$ denotes the value of the variable at the end of the l^{th} while-loop iteration, for the corresponding program[1]. For example, $\overline{Cres}_2$ is the value of *Cres* of algorithm 3 at the end of the second iteration of its while loop. We will use $\oplus$ to represent sum of sets.

The following key lemma is used to prove an equivalence between the behaviors of algorithm 1 and algorithm 3: given any new path p, the bandwidth allocated to p by algorithm 1 equals the bandwidth associated with the relay in p that appears earliest in $\overline{B}$ (computed by algorithm 3).

Lemma 1. *Let the new path be $H = \{h_1, h_2, h_3\} \in [n]^3$. Assume (a) $C = \overline{C}$, (b) $R[r] = \overline{R}[r]$ for all $r \in [n] \setminus H$ and $R[r] = \overline{R}[r] \cup \{m+1\}$ for $r \in H$, and (c) $P[p] = \overline{P}[p]$ for all $p \in [m]$, and $P[m+1] = H$. Assume w.l.o.g that h_1 is the first relay to be added by algorithm 3 to $\overline{B}$. Let $l_1, l_2, \ldots l_k$ be the iterations in algorithm 3 at which $\overline{t}^* = 1$ before h_1 is added to $\overline{B}$. Then, for all $s \leq k$, $Cres_s = \overline{Cres}_{l_s}$ and $Rres_s[r] = \overline{Rres}_{l_s}[r]$ for all $r \in [n] \setminus H$ and $Rres_s[r] = \overline{Rres}_{l_s}[r] \cup \{m+1\}$ for $r \in H$.*

Proof. First, $Cres_0 = C_1 = C_2 = \overline{Cres}_0$. Second, note that $Rat_1[r] = \overline{Rat}_1[1, r]$ for all $r \in [n] \setminus H$ and $Rat_1[r] = \overline{Rat}_1[2, r]$ for all $r \in H$. Hence, if the minimum ratio at the first iteration of algorithm 3 exists in Rat_1, that same ratio will be chosen by algorithm 1 at its first iteration. Thus, if $\overline{t}_1^* = 2$ and $\overline{r}_1^* = h_1$ ($\overline{r}_1^*$ cannot be h_2, h_3 as we assumed that h_1 is chosen first), then the lemma would hold with $k = 0$ and $r_1^* = h_1$.

[1]For algorithm 1, it is the value of the variable after the execution of line 15 and for algorithm 3 it is the value after the execution of line 22

If $\overline{t}_1^* = 2$ and $\overline{r}_1^* \neq h_1$, then the minimum ratio of algorithm 3 does not belong to Rat_1 and neither $\overline{Cres}$ nor $\overline{Rres}$ would be changed in this iteration, i.e. $\overline{Cres}_1 = \overline{Cres}_0$ and $\overline{Rres}_1 = \overline{Rres}_0$. In that case, Rat_2 would still be a subset of $\overline{Rat}_2$.

The case where $\overline{t}_1^* = 1$ and $\overline{r}_1^* \in H$ cannot happen since $\overline{Rat}_1[2, r] \leq \overline{Rat}_1[1, r]$ for all $r \in [n]$.

Finally, if $\overline{t}_1^* = 1$ and $\overline{r}_1^* \notin H$, then $l_1 = 1$ and both algorithms will have the same minimum ratio, the else branch would be taken in algorithm 3 and $\overline{Cres}$ and $\overline{Rres}$ would be updated in the same manner as those *Cres* and *Rres*. Thus, the property in the lemma would be preserved.

Hence, the above analogy can be repeated to shown that the l^{th} iteration of algorithm 3 at which $\overline{t}^* = 1$ will run the same updates as that of the l^{th} iteration of algorithm 1 till h_1 is added to $\overline{B}$. Iterations of the while loop in algorithm 3 at which $\overline{t}^* = 2$ does not affect $\overline{Cres}$, $\overline{Rres}$, and the ratios common with algorithm 1 until h_1 is chosen. Once h_1 is added to $\overline{B}$, that corresponds to the $k + 1^{th}$ iteration of algorithm 1 where h_1 will be chosen as the minimizing relay too and the bandwidth of the new path would be determined. □

Corollary 1. *For any new path $H = \{h_1, h_2, h_3\}$. The bandwidth associated with h_1 in $\overline{B}$ (algorithm 3) equals the bandwidth allocated for H by algorithm 1.*

Since in the following lemma we will be only analyzing algorithm 3, there will be no confusion with the variables of algorithm 1 so we drop the bars.

Lemma 2. *The ratios in B appear in increasing order.*

Proof. Consider the l^{th} iteration of algorithm 3 at which a ratio-relay pair (b, r_l^*) is added to B, i.e. $t_l^* = 2$. If in the preceding iteration $l - 1$, $t_{l-1}^* = 2$, another ratio-relay (b_1, r_{l-1}^*) would have been added to B before it. Moreover, both *Cres* and *Rres* would not have changed and since b_1 was chosen first means it is smaller than b_2 and thus in this case the B would be increasing. Otherwise, if $t_{l-1}^* = 1$, the "else" branch would be taken. Denote the number of paths r_{l-1}^* and r_l^* share at the $(l-1)^{th}$ iteration be s. Then, $Rat_l[2, r_l^*] = \frac{Cres_{l-1}[r_l^*] - Rat_{l-1}[1, r_{l-1}^*]}{|Rres_{l-1}[r_l^*]| + 1 - s}$. We will show that it is larger than $Rat_{l-1}[1, r_{l-1}^*] = \frac{Cres_{l-1}[r_{l-1}^*]}{|Rres_{l-1}[r_{l-1}^*]|}$.

We know that $Rat_{l-1}[2, r_l^*] := \frac{Cres_{l-1}[r_l^*]}{|Rres_{l-1}[r_l^*]|+1} > \frac{Cres_{l-1}[r_{l-1}^*]}{|Rres_{l-1}[r_{l-1}^*]|}$. Hence,

$$
\begin{aligned}
& Cres_{l-1}[r_l^*]|Rres_{l-1}[r_{l-1}^*]| > Cres_{l-1}[r_{l-1}^*](|Rres_{l-1}[r_l^*]| + 1) \\
\implies & Cres_{l-1}[r_l^*]|Rres_{l-1}[r_{l-1}^*]| - s(Cres_{l-1}[r_{l-1}^*]) \\
& \qquad > Cres_{l-1}[r_{l-1}^*](|Rres_{l-1}[r_l^*]| + 1 - s) \\
\implies & |Rres_{l-1}[r_{l-1}^*]|(Cres_{l-1}[r_l^*] - s\frac{Cres_{l-1}[r_{l-1}^*]}{|Rres_{l-1}[r_{l-1}^*]|}) \\
& \qquad > Cres_{l-1}[r_{l-1}^*](|Rres_{l-1}[r_l^*]| + 1 - s) \\
\implies & \frac{Cres_{l-1}[r_l^*] - s\frac{Cres_{l-1}[r_{l-1}^*]}{|Rres_{l-1}[r_{l-1}^*]|}}{|Rres_{l-1}[r_l^*]| + 1 - s} > \frac{Cres_{l-1}[r_{l-1}^*]}{|Rres_{l-1}[r_{l-1}^*]|}.
\end{aligned}
$$

Therefore, the ratios are non decreasing in the iterations at which $t^* = 1$ and constant (other than the one added to B) when $t^* = 2$ which means that every time a ratio is added to B, it will be equal or larger than all previously added ones. □

Corollary 2. *Choosing the last three relays of the resulting B for the new path would result in maximum bandwidth allocated for it by algorithm 1. That bandwidth is the one associated with the third relay from the end of B.*

4.3 Implementation of the locally optimal algorithm

In order to simulate algorithm 3, we wrote a code using Python programming language. Starting with an empty set, one million paths were created by repetitively running our locally optimal algorithm 3 while adding the output path to the network. The bandwidth allocation of the one million paths generated is found using the max-min fair algorithm 1. Those results were then compared to the bandwidth allocations of one million paths generated using the random algorithm 2. The results are shown in fig. 4.2.

The locally optimal algorithm 3 produces a much more well-balanced set of paths: the minimum allocation if 10.96 is nearly the same as the average of 10.97, and the maximum is 21. In contrast, the paths created using the random algorithm 2, while having a similar average bandwidth allocation of 10.94, span a much broader range, with a minimum allocation of 1 and a maximum of 210.14.

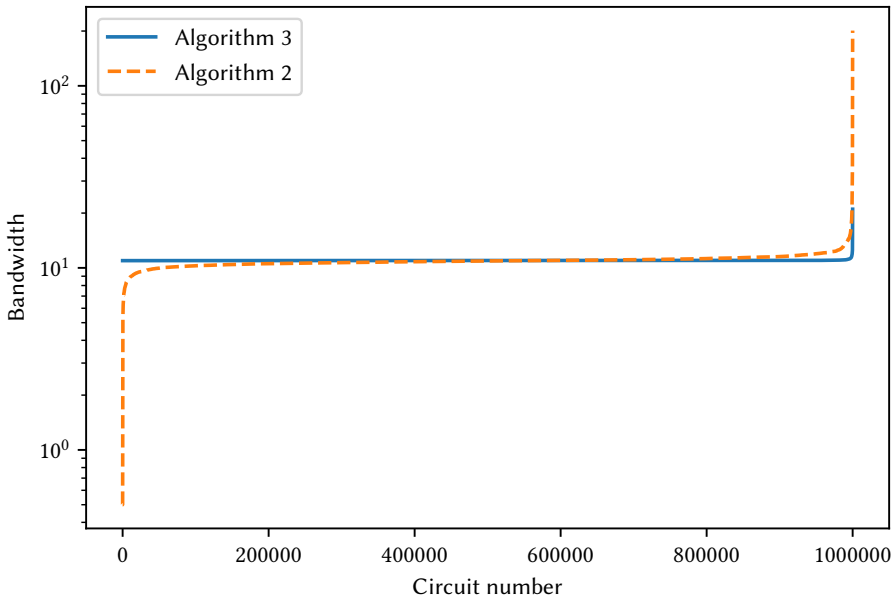

Figure 4.2: Graph comparing the bandwidth allocation of one million paths generated using the locally optimal algorithm 2 and the random algorithm 3. Note the logarithmic scale of the y-axis.

4.4 Conclusion

We developed a locally optimal path selection algorithm that, given the state of the network, generates a path with maximum bandwidth allocation possible. We implemented the algorithm using Python programming language. We then showed, using flow-level simulations, that the new algorithm 3 results in a more load-balanced path allocation between users than the random algorithm 2.

Chapter 5

DIFFERENCIALLY PRIVATE ALGORITHM

The locally optimal algorithm 3 requires the knowledge of the state of the network as input. Thus, if it is going to be used, every user will need to know the state of the network, i.e., the paths of every other user, in order to be able to construct a path for herself. This will defeat the purpose of onion routing as it will then be possible to deanonymize users. In this chapter, we will be discussing how to implement a differentially private version of algorithm 3.

5.1 Overview

In this section, we present an overview of the different steps required in order to implement the differentially private algorithm. In the private version of the algorithm, we first decompose each circuit (r_1, r_2, r_3) into two circuit segments, (r_2, r_1) and (r_2, r_3). We then adapt the locally optimal algorithm 3 to use these segments, rather than complete circuits, in creating an optimal new path. To add privacy, we summarize the list of segments as a histogram, indexed by pairs of relays, of the number of circuit segments (r_i, r_j) that are present. We then create a differentially private version of this histogram by using a threshold-based differentially private count, to account for the sparse nature of the histogram.

One feature of the private count algorithm is that each histogram entry can be processed individually. Therefore, a relay r_i can apply it to the count of each histogram entry (r_i, r_j), since it knows the (actual, non-private) number of such flow segments. The private counts can then be aggregated and distributed using a modification of the existing Tor directory mechanism or another peer-to-peer broadcast scheme.

Since the private histogram can only be updated periodically, we use this histogram to generate the next K near-optimal circuits. We cannot assign

these circuits directly to each new user; instead, we count the number of times each relay appears in these K circuits and use it to induce a distribution over relays that the users sample from for their circuit. This approach is similar to the random algorithm 2, except that the weights reflect relays that are underloaded in the current state of the network, rather than the static relay capacities.

Thus in order to implement a differentially private algorithm, we will tackle each of the aforementioned steps in different sections of this chapter by developing a suitable algorithm for each step:

- We first decompose the three relays paths in the network into ordered pairs of relays and summarize the list of segments in a histogram.
- We create a private version of the histogram using (ϵ, δ)-differential privacy in section 5.4.
- We will adapt the optimal path allocation algorithm 3 to take as input a pairs of relays in section 5.2 .
- We will then generate K near-optimal circuits using the algorithm developed in the previous step in order to induce a distribution over relays that the next L users sample from for their circuits in section 5.3.

We will combine the algorithms developed in each section and simulate our differentially private algorithm in section 5.5.

5.2 Pair-based Algorithm

5.2.1 Description

As stated previously, we first adapt algorithm 3 to take as input ordered pair of relays instead of three relays paths.

Notation. As in the locally optimal algorithm 3, we denote the capacity of the r^{th} relay by $C[r]$. As discussed before, the paths are now ordered pairs of relays. Being ordered is essential for the correctness of the pair-based algorithm 4 as will be discussed later. We denote by P the map from

these paths to the ordered pairs of relays to which they belong. For example, those corresponding to the p^{th} path are $P[p] = (r_{p,1}, r_{p,2})$. Also, as before, we define $R[r]$ to be the set $\{p \mid r \in P[p]\}$ of paths to which relay r belongs. However, we decompose R into two maps: R_c mapping relays to the paths in which they appear in the first component of the ordered pair, i.e. the central relay, and R_e mapping relays to the paths in which they appear in the second component of the ordered pair, i.e. the end relay.

We finally define $Nres[r]$ as the number of actual paths (paths with three relays) containing relay r. We can compute it as: $Nres[r] = |Rres_c[r]|/2 + |Rres_e[r]|$, since for an actual path in the Tor network where r is the central relay, it will appear as two pairs in $Rres_c[r]$ while it should be counted once. It will only appear once in $Rres_e[r]$ if r is one of its end relays.

In algorithm 4, (see algorithm 4 below), as in the previous two algorithms, in the iterations at which $t^* = 1$, $Rat[1, r^*] = \frac{Cres[r]}{Nres[r]}$ of bandwidth would be allocated to each of the paths passing through it. After that, the paths in $Rres_e[r^*]$, $Rat[1, r^*]/2$ would be deducted from the residual capacity of the central relay in the path. That is because the bandwidth of the other path that corresponds to the same actual path will be subtracted from the residual capacity of the central relay in that path too. For the paths in $Rres_c[r^*]$, $Rat[1, r^*]$ would be deducted entirely from the residual capacity of the end relay of the path. That is because the end relay of an actual path of the Tor network only appears once in the two-relay paths corresponding to that path. Using the same analogy, for the paths in $Rres_c[r^*]$, $Nres[r]$, r being the end relay in the path, is deducted by one and for paths in $Rres_e[r^*]$, $Nres[r]$, r being the central relay in the path, is deducted by half. The rest of the algorithm follows the same steps of algorithm 3.

Hence, algorithm 4 takes as an input a path matrix of ordered pairs of relays along with the capacity matrix. The output is the path of three relays that when added to the system gives the highest bandwidth allocation compared to any other path that can be added to the system.

Algorithm 4 Pair-based Locally Optimal Path Allocation Algorithm

input: C, R, R_c, R_e, P
$B \leftarrow \emptyset$
$Cres[r] \leftarrow C[r], \forall r \in [n]$
$Rres[r] \leftarrow R[r], \forall r \in [n]$
$band[p] \leftarrow 0, \forall p \in [m]$
$Nres[r] \leftarrow \frac{|Rres_c[r]|}{2} + |Rres_e[r]|, \forall r \in [n]$
while $\exists\, i \notin B$ **do**
    $Rat[1,r] \leftarrow \begin{cases} \frac{Cres[r]}{Nres[r]} & \forall r \mid Nres[r] \neq 0 \\ \infty & otherwise \end{cases}$
    $Rat[2,r] \leftarrow \begin{cases} \frac{Cres[r]}{Nres[r]+1} & \text{if } r \notin B \\ \infty & otherwise \end{cases}$
    $(t^*, r^*) \leftarrow \underset{t \in \{1,2\}, r \in [n]}{\text{argmin}} \; Rat[t,r]$
    if $t^* == 2$ **then**
        $B.push([Rat[t^*, r^*], r^*])$
        $Rat[t^*, r^*] \leftarrow \infty$
    else
        for p $\in Rres[r*]$ **do**
            $band[p] \leftarrow Rat[t^*, r^*]$
            for $r \in P[p]$ **do**
                if $p \in Rres_c[r]$ **then**
                    $Cres[r] \leftarrow Cres[r] - Rat[t*, r*]/2$
                    $Nres[r] \leftarrow Nres[r] - 1/2$
                    $Rres[r] \leftarrow Rres[r] \setminus \{p\}$
                    $Rres_c[r] \leftarrow Rres_c[r] \setminus \{p\}$
                else
                    $Cres[r] \leftarrow Cres[r] - Rat[t*, r*]$
                    $Nres[r] \leftarrow Nres[r] - 1$
                    $Rres[r] \leftarrow Rres[r] \setminus \{p\}$
                    $Rres_e[r] \leftarrow Rres_e[r] \setminus \{p\}$
                end if
            end for
        end for
    end if
end while
return: $\{r_1, r_2, r_3\}$

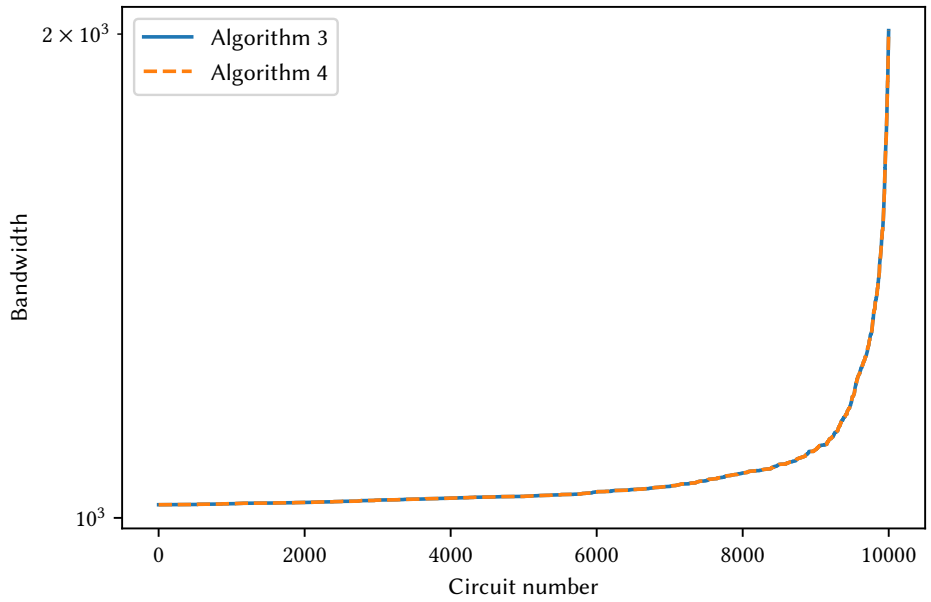

Figure 5.1: Graph comparing the bandwidth allocation of 10 000 paths generated using the pair-based algorithm 4 and the locally optimal algorithm 3.

5.2.2 Implementation of the Pair-based Algorithm

Starting from an empty set, the pair-based algorithm 4 was used repetitively to generate a set of 10 000 paths of three relays. At each run, a path matrix representing the decomposition of the paths in the network into segments of two relays is constructed, and then used as input to algorithm 4 along with the capacity matrix. The output, the path of three relays, is then added to the network. The bandwidth allocations for these paths are then computed using the max-min fairness algorithm 1.

The results were then compared to the bandwidth allocations of 10 000 paths generated by repetitive application of locally optimal algorithm 3 starting from an empty set, in order to know how much accuracy we lost by the decomposition of paths into pairs. The results are shown in fig. 5.1. The two curves coincide which shows there is zero loss of accuracy.

The same experiment is repeated but instead 1 million paths were created using both algorithms. The results are shown in fig. 5.2. The maximum difference between the two allocations was 0.918.

As the results of the two simulations show, the bandwidth allocation of

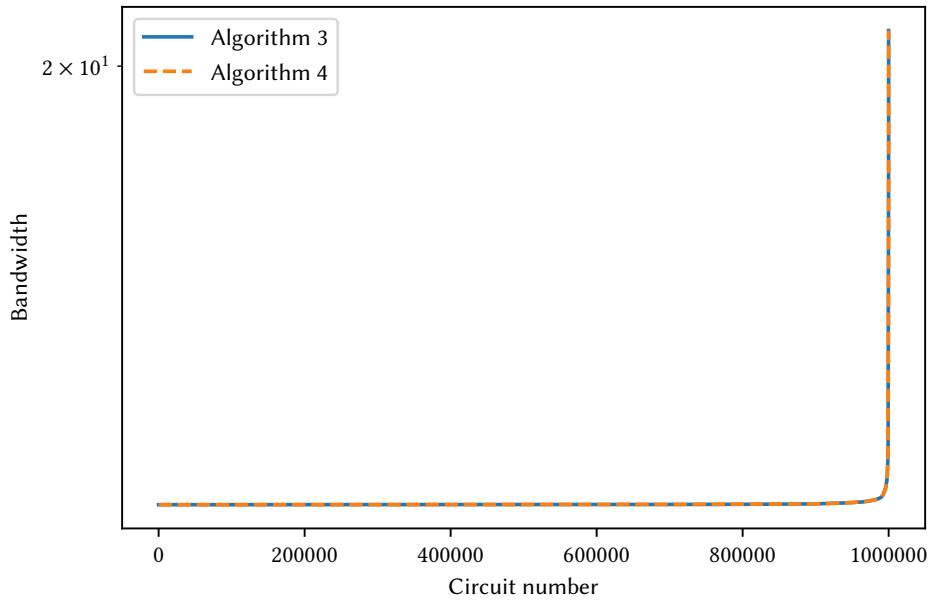

Figure 5.2: Graph comparing the bandwidth allocation of one million paths generated using pair-based algorithm 4 and locally optimal algorithm 3 (in blue). Note the logarithmic scale of the y-axis.

the paths generated using both algorithms coincides completely, showing that there is no loss of accuracy when adapting the locally optimal algorithm 3 to take as input ordered two-relay paths (pair-based algorithm 4).

5.3 Batch Path Allocation Algorithm

As we discussed earlier, the server would periodically collect data from the relays and create a histogram mapping each ordered pair of relays to the number of paths passing through them. It would then create a differentially private version of this histogram (we will deal with this step in the next section 5.4). After that, given the histogram, it runs the pair-based algorithm 4 (section 5.2) repetitively to generate K additional paths. These paths are not added to the actual network but to a virtual copy of the network. The number of times each relay appeared in these K paths is counted to generate a probability distribution over the relays. The distribution is then released to the public. Incoming users would sample that distribution (without re-

placement) to get three different relays which would constitute their paths.

The batch algorithm 5 creates, given a histogram mapping each pair of relays to the number of path going through them, a batch of L random paths using the procedure we just described. L represents the expected number of paths (i.e. users) that would be created in a single period before the distribution gets updated.

Algorithm 5 Batch Path Allocation Algorithm

1: **input:** C, R, R_c, R_e, P, K, L
2: Repeat K times:
3: Input $C, Rres, R_c, R_e, P$ to algorithm 4.
4: Add the returned path to the network.
5: Update $Rres, R_c, R_e, P$ accordingly.
6: Compute the distribution of the relays in the added K paths.
7: Sample L paths with three relays from that distribution
8: **return:** Generated L paths

To evaluate the behaviour of the batch algorithm 5, K is set to 10 000 and L to 200. Then, starting from a network with no paths, we ran it repetitively to create a network with 600 000 paths, 200 at a time. At each run, a path matrix is constructed by decomposing the paths in the network to paths of two ordered relays as previously discussed and taken as input for the next run.

After that, the bandwidth allocations of the created paths were computed using the max-min fair algorithm 1 and compared to the bandwidth allocations of 600 000 paths generated using the the random algorithm 2. The results are shown in fig. 5.3.

The minimum bandwidth of a path generated using the batch algorithm 5 was 6.33, the maximum was 37.23 and the average was 17.29. On the other hand, for the set of paths generated randomly, the minimum bandwidth of a path was 1, the maximum was 287.99 and the average was 17.23.

The simulation demonstrates the fairness property of algorithm 5. The range of the bandwidth allocations of the paths is [6.33, 37.23] compared to a range of [1, 287.99] for the random paths, while the average is being conserved. The algorithm hence avoids the use of paths with very low bandwidth and guarantees a more fair distribution of the bandwidth between users.

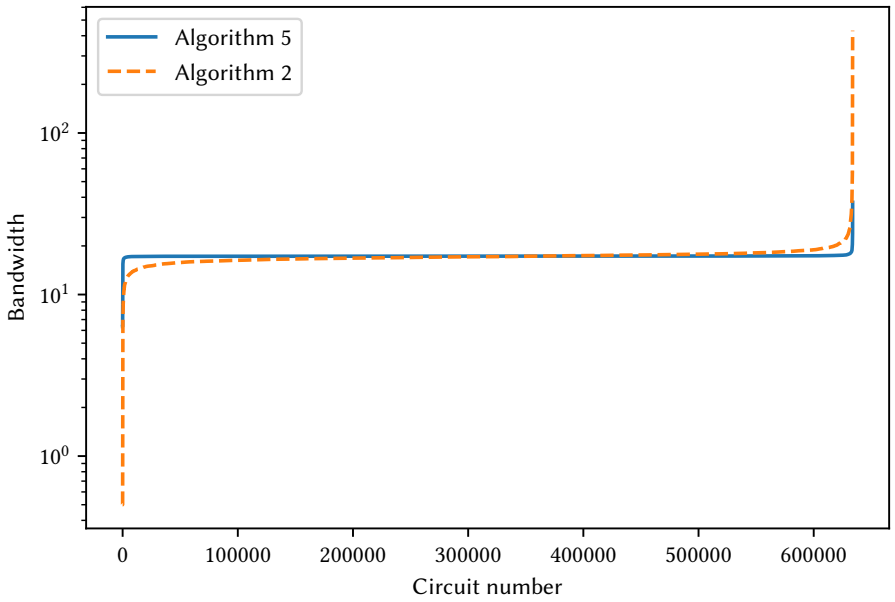

Figure 5.3: Graph comparing the bandwidth allocation of 600 000 paths generated using the random algorithm 2 and the batch algorithm 5. Note the logarithmic scale of the y-axis.

5.4 Adding Differential Privacy

In this section, we show how the server ensures differential privacy of the statistics it releases about the network.

As discussed earlier, at the end of each period, the server generates a differentially private version of a histogram (matrix) of size n^2. In our work, we use the notion (ϵ, δ)-differential privacy defined in section 2.3. We did not use the usual ϵ-differential privacy since the histogram that we are making private, is very large ($n^2 \approx 36\,000\,000$) and is very sparse. Using (ϵ, δ)-differential privacy allows us to operate on the sparse histogram, as described below, rather than producing a noisy version of each 0 value in the histogram, which would overwhelm the algorithm. We replace n with n^2 in the original definition since in our case the dataset is of size n^2.

The following theorem shows that there is a mechanism that ensures the differential privacy of the histogram against queries consisting of point functions while guaranteeing an acceptable level of accuracy of query responses. Before stating the theorem, let us define formally the set of queries for which

the mechanism ensures privacy.

Definition 2 (page 6 in [45]). *Point Functions (Histograms): Let $\mathcal{X}$ be an arbitrary set and for each $y \in \mathcal{X}$, we consider the predicate $q_y : \mathcal{X} \to \{0,1\}$ that evaluates to 1 only on input y. The family $Q^{pt} = Q^{pt}(\mathcal{X})$ consists of the counting queries corresponding to all point functions on data universe $\mathcal{X}$. (Approximately) answering all of the counting queries in Q^{pt} amounts to (approximately) computing the histogram of the dataset.*

As before we substitute n by n^2 from the original theorem.

Theorem 1. *(stability-based histograms [50]). For every finite data universe $\chi, n \in \mathbb{N}, \epsilon \in (0, 2\ln n)$, and $\delta \in (0, 1/n^2)$ there is an (ϵ, δ)-differentially private mechanism $\mathcal{M} : \chi^{n^2} \to \mathbb{R}^{\chi}$ that on every dataset $x \in \chi^{n^2}$, with high probability $\mathcal{M}(x)$ answers all of the counting queries in $Q^{pt}(\chi)$ to within error*

$$O\left(\frac{\log(1/\delta)}{\epsilon n^2}\right)$$

In the proof of the theorem, the authors provided such mechanism which takes a dataset (histogram) $x \in \chi^{n^2}$ as input and returns a privatized version of it. The mechanism is shown in algorithm 6. It iterates over the elements of the histogram, those that are zero are kept zero. A positive entry would be added to an independent Laplace variable with $\lambda = 2/(\epsilon n^2)$. If the result is below the threshold $((2\ln(2/\delta))/(\epsilon)) + (1)$, the entry would be set to zero, otherwise it is set to the result. Note that we multiplied the threshold that is in the mechanism by n^2 since we do not need the result of the query to be normalized, i.e. between 0 and 1, so we do not need the n^2 factor.

Algorithm 6 (ϵ, δ)-differentially Private Histogram Mechanism

input: x, χ
For every $y \in \chi$:
  If $q_y(x) = 0$ then:
    Set $a_y = 0$
  If $q_y(x) > 0$ then:
    Set $a_y \leftarrow q_y(x) + Lap(2/(\epsilon n^2))$.
    If $a_y < 2\ln(2/\delta)/\epsilon n^2 + 1/n^2$ then:
      Set $a_y = 0$
return: $(a_y)_{y \in \chi}$

As discussed earlier, the server releases a distribution over the relays at the beginning of every period based on the generated private histogram. Thus, information about the paths that remain over multiple periods in the network will be released several times. This will deteriorate the level of privacy they are guaranteed. To bound this deterioration we consult the following composition theorem of differential privacy.

Theorem 2. *(Composition of (ϵ, δ)-differentially-private algorithms, Theorem 16 in [51]). Let $T_1 : D \to T_1(D)$ be (ϵ, δ)-differentially-private, and for all $J \geq 2, T_j : (D, s_1, ..., s_{J-1}) \to T_J(D, s_1, ..., s_{J-1}) \in \zeta_J$ be (ϵ, δ)-differentially-private, for all given $(s_1, ..., s_{J-1}) \in \otimes_{j=1}^{J-1} \zeta_j$, where "$\otimes$" denotes direct product of spaces. Then for all neighboring D, D' and all $S \subseteq \otimes_{j=1}^{J-1} \zeta_j$:*

$$P((T_1, ..., T_J) \in S) \leq e^{J\epsilon} P'((T_1, ..., T_J) \in S) + J\delta$$

We can conclude that the parameters of privacy ϵ and δ that we can guarantee for a certain path increase linearly in the number of periods it stays in the network.

5.5 Implementation of the Differentially Private Optimal Path Allocation Algorithm

In this section we will combine the different algorithms presented in the different sections of this chapter in order to implement the differentially private optimal path allocation algorithm.

As previously stated, the server would create a histogram mapping each ordered pair of relays to the number of paths passing through them in the network. We use algorithm 6 in order to create a private version of this histogram. We chose the parameters ϵ and δ to be 0.3 and 0.001, respectively. This private version of the histogram is then inputted to algorithm 5, in order to generate a distribution over the relays from which we sample L three relays paths corresponding to the next L users.

To evaluate the behaviour of algorithm 5 after we privatize the histogram before using it to generate the distribution over relays, we conducted the following experiment: starting from an empty network, we generated $N = 1\,000\,000$ paths using repetitive application of batch algorithm 5 with $K =$

10 000 and $L = 200$ while using a private version of the histogram before each run. Those steps are shown in algorithm 7 and fig. 5.4.

Algorithm 7 (ϵ, δ)-differentially Private Optimal Path Allocation Algorithm

1: **input:** $C, R, R_c, R_e, P, N, K, L$
2: **for** $i \in [\lceil \frac{N}{L} \rceil]$ **do**
3: A histogram mapping the pairs of relays in the network to the number of paths between each pair is constructed.
4: A private version of the histogram is generated using the mechanism described in algorithm 6.
5: The private histogram is input to the batch algorithm 5 with parameters K and L.
6: The generated L paths are added to the network.
7: Update R, R_c, R_e, P.
8: **end for**

The bandwidth allocations of those paths are then computed using max-min fair allocation algorithm 1 and compared to the bandwidth allocations of one million paths generated using the random algorithm 2. The sorted (in bandwidth) results are shown in fig. 5.5.

The minimum bandwidth allocation of a path of differentially private algorithm 7 was 8.2, the maximum was 9603.5 and the average was 10.54. While for those generated randomly using algorithm 2, the minimum was 1, the maximum was 210.14 and the average was 10.94. Although the average was lower for our algorithm, it has no paths with low bandwidth as the random one. The tail on the left is much shorter while the tail on the right is much longer than the random one. The difference in most paths is negligible.

Finally we evaluate the scenario where only a subset of the generated paths are active, as discussed in section 3.2. We take the million circuits generated using the privacy-preserving algorithm 7, as above, and choose a random sample of 10 000 circuits as being active. We then compare the bandwidth allocated to those circuits with 10 000 circuits chosen by the random algorithm 2 in fig. 5.6. Note that the randomized algorithm has a significantly longer tail on the left side of the graph, representing circuits with a pathological bandwidth allocation. The minimum bandwidth in algorithm 2 is 7, whereas the minimum for the sampled algorithm 7 is 171. The randomized algorithm produces 953 flows with a bandwidth less than half of the average, 980. The sampled circuits from the private algorithm have a slightly lower

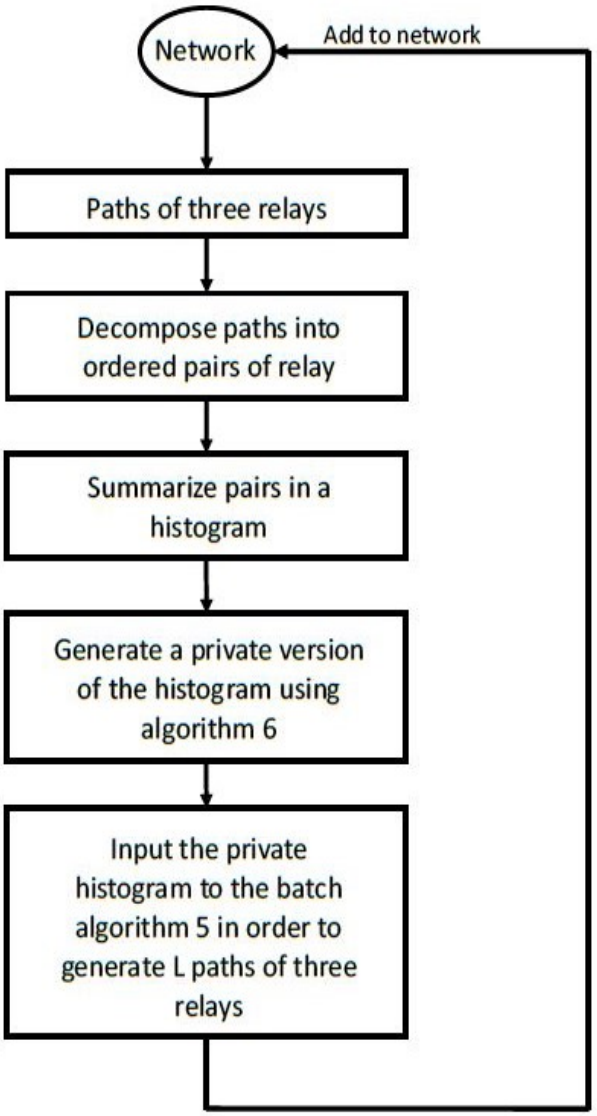

Figure 5.4: Differentially private path allocation algorithm implementation

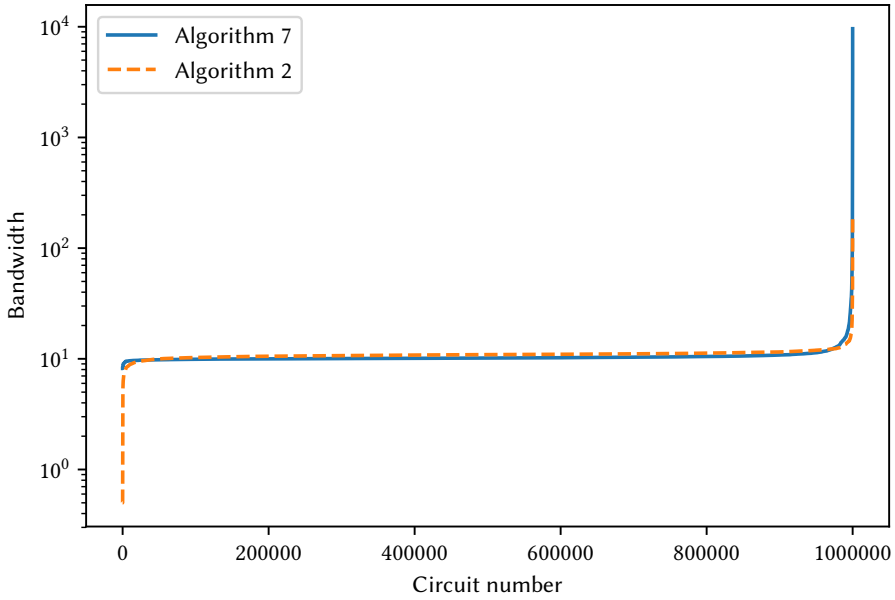

Figure 5.5: Graph comparing the bandwidth allocation of one million paths generated using the differentially private algorithm 7 and using the random algorithm 2. Note the logarithmic scale of the y-axis.

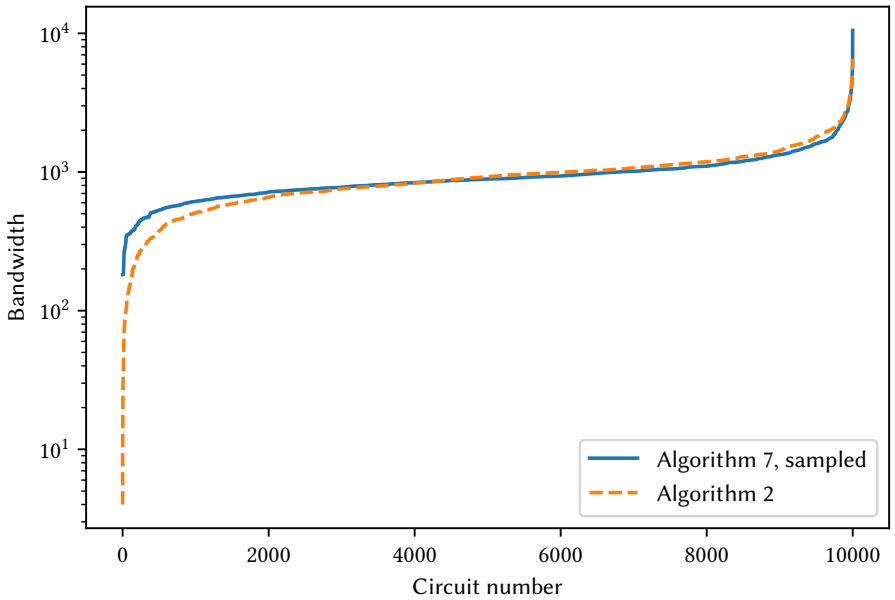

Figure 5.6: Comparing random allocation in algorithm 2 to a sample of 10 000 circuits out of 1 million that were generated by the differentially private algorithm 7.

average of 960, but only 326 flows have a bandwidth of less than 980/2. The lower average performance is due to some relays being underutilized, as evidenced by the longer tail on the right-hand side of the graph; in our ongoing work we are investigating adjustments to the algorithm to make better use of these relays.

5.6 Conclusion

We have developed a private version of the optimal path allocation algorithm 3. We first introduced the concept of decomposing three relays paths into ordered pairs of relays that can be summarized in a histogram. We then adapted algorithm 3 to take as input the histogram generated instead of a set of three relays paths (Pair-based algorithm 4). The simulations of algorithm 4 showed that no loss of accuracy is induced when adapting the new algorithm. We then introduced the batch algorithm 5 that runs the Pair-based algorithm 4 repetitively to generate K optimal path without actually adding them to the network. Instead, algorithm 5 generates a probability distribution over the relays by counting the number of times each relay appears in those K paths. Then the algorithm samples from this distribution L paths to be used by users. The simulations of algorithm 5 demonstrated the fairness property of our algorithm. Finally, we added a layer of privacy to the histogram generated by creating a private version of it using (ϵ, δ)-differential privacy. This private version is inputted to algorithm 5 instead of a non-private histogram; the result is algorithm 7. By running algorithm 7 repetitively to generate one million paths, we have shown that the bandwidth allocation of the generated paths have a more fair distribution among users by avoiding the use of paths with very low bandwidth as shown by the absence of the tail on the left-hand side of the graph in fig. 5.5.

Chapter 6

CONCLUSION AND FUTURE WORK

We have presented an algorithm that approximates locally optimal load-balancing of circuits in the Tor network while preserving user privacy [1]. In chapter 4, we introduced an optimal path selection algorithm that takes as input the whole state of the network and generates a path with the highest possible bandwidth when added to the network. In Chapter 5, we further developed the above algorithm by taking into account the privacy preserving aspect of our problem. We thus developed the differentially-private optimal path allocation algorithm. We demonstrated that this algorithm significantly improves on the randomized relay assignment in Tor using flow-level simulations of max-min fair bandwidth allocation.

Our promising results encourage the further exploration of using privacy-preserving feedback for load balancing in anonymity networks. Several important challenges remain. First, load imbalance occurs over short time scales, thus the private summary of the network state must be quickly distributed to all users. We note that this is a similar problem to that of distributing blocks in cryptocurrencies; in the Bitcoin network, which is similarly sized to Tor, measurements have shown that blocks reach the median node in 6.5s and the 90th percentile node in 26s [52]. Improving this latency while maintaining resilience to attack is an area of active research.

A second problem is that malicious nodes may misreport their contributions to the histogram to direct circuits away from honest nodes and towards malicious ones. We note that this problem is somewhat similar to the peer bandwidth measurement problem in EigenSpeed [32] and PeerFlow [33] and thus some of the defenses used in those systems may be adaptable to this setting. Furthermore, all our work is based on the consensus document containing the measured capacities of the relays. However as stated before, those measured capacities can vary wildly in time and are also the target of many adversaries attacks. Developing the procedure used to measure the capacities

of the relays remains one major challenge that needs to be addressed.

Finally, flow-level simulation is a coarse-grained approximation of Tor traffic; web browsing is a dominant use of Tor and web traffic is known to be quite bursty. Further evaluation of the load-balancing mechanism using queuing-based traffic models or full network simulation [53] is needed.

BIBLIOGRAPHY

[1] H. Darir, H. Sibai, N. Borisov, G. Dullerud, and S. Mitra, "Tightrope: Towards optimal load-balancing of paths in anonymous networks," in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, ser. WPES'18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3267323.3268953 pp. 76–85.

[2] M. N. O. Sadiku, Y. Wang, S. Cui, and S. M. Musa, "Cyber-physical systems: A literature review," *European Scientific Journal*, vol. 13, 12 2017.

[3] T. Sanislav and L. Miclea, "Cyber-physical systems - concept, challenges and research areas," *Control Engineering and Applied Informatics*, vol. 14, pp. 28–33, 01 2012.

[4] A. Johnson, "Design and analysis of efficient anonymous-communication protocols," Ph.D. dissertation, Yale University, 2009.

[5] T. T. Project, "Tor metrics: Users," https://metrics.torproject.org/userstats-relay-country.html, 2018.

[6] Brave, "Brave introduces beta of private tabs with Tor for enhanced privacy while browsing," https://brave.com/tor-tabs-beta, 2018.

[7] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," *Paul Syverson*, vol. 13, 06 2004.

[8] K. Loesing, "Measuring the Tor network, evaluation of client requests to the directories to determine total numbers and countries of users," The Tor Project, Tech. Rep. 2009-06-002, June 2009. [Online]. Available: https://research.torproject.org/techreports/directory-requests-2009-06-25.pdf

[9] R. Dingledine and S. J. Murdoch, "Performance improvements on tor or, why tor is slow and what we're going to do about it," 06 2019.

[10] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. M. Voelker, "Defenestrator: Throwing out windows in tor," in *Privacy Enhancing Technologies*, S. Fischer-Hübner and N. Hopper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 134–154.

[11] C. Tang and I. Goldberg, "An improved algorithm for tor circuit scheduling," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866345 pp. 329–339.

[12] R. Jansen, N. Hopper, and Y. Kim, "Recruiting new tor relays with braids," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866344 pp. 319–328.

[13] W. B. Moore, C. Wacek, and M. Sherr, "Exploring the potential benefits of expanded rate limiting in tor: Slow and steady wins the race with tortoise," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076762 pp. 207–216.

[14] N. Mathewson, "Evaluating sctp for tor," http://archives.seul.org/or/dev/Sep-2004/msg00002.html, 2004.

[15] J. Reardon and I. Goldberg, "Improving tor using a tcp-over-dtls tunnel," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09. Berkeley, CA, USA: USENIX Association, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855768.1855776 pp. 119–134.

[16] M. Akhoondi, C. Yu, and H. V. Madhyastha, "Lastor: A low-latency as-aware tor client," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 476–490.

[17] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. T. Loo, and M. Blaze, "A3: An extensible platform for application-aware anonymity," in *NDSS*, 2010.

[18] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *15th Network and Distributed System Security Symposium (NDSS)*, C. Cowan and G. Vigna, Eds. Reston, VA, USA: Internet Society, Feb. 2008.

[19] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for tor," in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 98–113.

[20] M. AlSabah and I. Goldberg, "Performance and security improvements for Tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 32, 2016.

[21] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Proceedings of the First International Workshop on Information Hiding*. London, UK, UK: Springer-Verlag, 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=647594.731526 pp. 137–150.

[22] R. Snader and N. Borisov, "Improving security and performance in the Tor network through tunable path selection," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 5, pp. 728–741, 2011.

[23] S. Herbert, S. J. Murdoch, and E. Punskaya, "Optimising node selection probabilities in multi-hop M/D/1 queuing networks to reduce latency of tor," *Electronics Letters*, vol. 50, no. 17, pp. 1205–1207, 2014.

[24] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for Tor," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 98–113.

[25] M. AlSabah, K. Bauer, T. Elahi, and I. Goldberg, "The path less travelled: Overcoming Tor's bottlenecks with traffic splitting," in *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*. Springer, 2013, pp. 143–163.

[26] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Privacy Enhancing Technologies*, I. Goldberg and M. J. Atallah, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 73–93.

[27] C. Wacek, H. Tan, K. S. Bauer, and M. Sherr, "An empirical evaluation of relay selection in tor," in *NDSS*, 2013.

[28] F. Chen and J. Pasquale, "Toward improving path selection in tor," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec 2010, pp. 1–6.

[29] R. Jansen and N. J Hopper, "Shadow: Running tor in a box for accurate and efficient experimentation shadow: Running tor in a box for accurate and efficient experimentation," 06 2019.

[30] S. J. Murdoch and R. N. Watson, "Metrics for security and performance in low-latency anonymity systems," in *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, ser. PETS '08. Berlin, Heidelberg: Springer-Verlag, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70630-4_8 pp. 115–132.

[31] M. Perry, "TorFlow: Tor network analysis," in *Proceedings of the 2nd Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs)*, 2009, pp. 1–14.

[32] R. Snader and N. Borisov, "EigenSpeed: Secure peer-to-peer bandwidth evaluation," in *8th International Workshop on Peer-To-Peer Systems*, R. Rodrigues and K. Ross, Eds. Berkeley, CA, USA: USENIX Association, Apr. 2009.

[33] A. Johnson, R. Jansen, N. Hopper, A. Segal, and P. Syverson, "PeerFlow: Secure load balancing in Tor," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 74–94, 2017.

[34] G. Karame, D. Gubler, and S. Čapkun, "On the security of bottleneck bandwidth estimation techniques," in *Security and Privacy in Communication Networks*, Y. Chen, T. D. Dimitriou, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 121–141.

[35] R. Süselbeck, G. Schiele, P. Komarnicki, and C. Becker, "Efficient bandwidth estimation for peer-to-peer systems," in *2011 IEEE International Conference on Peer-to-Peer Computing*, Aug 2011, pp. 10–19.

[36] A. Haeberlen, P. Kuznetsov, and P. Druschel, "Peerreview: practical accountability for distributed systems," in *SOSP*, 2007.

[37] A. Serjantov and P. Sewell, "Passive-attack analysis for connection-based anonymity systems," *International Journal of Information Security*, vol. 4, no. 3, pp. 172–180, Jun 2005. [Online]. Available: https://doi.org/10.1007/s10207-004-0059-3

[38] A. Back, U. Möller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Information Hiding*, I. S. Moskowitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 245–257.

[39] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, February 1981.

[40] P. F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr, "Towards an analysis of onion routing security," in *Workshop on Design Issues in Anonymity and Unobservability*, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Springer, 2000, pp. 96–114.

[41] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*. USENIX, 2004, pp. 303–320.

[42] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1024–1039, 1991.

[43] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[44] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer, 2006, pp. 1–12.

[45] S. Vadhan, "The complexity of differential privacy," https://privacytools.seas.harvard.edu/publications/complexity-differential-privacy, 2017.

[46] R. Jansen and A. Johnson, "Safely measuring tor," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978310 pp. 1553–1567.

[47] R. Dingledine and N. Mathewson, "Tor path specification," https://gitweb.torproject.org/torspec.git/plain/path-spec.txt, 2017.

[48] T. T. Project, "Tor metrics: Traffic," https://metrics.torproject.org/bandwidth.html, 2018.

[49] T. T. Project, "Tor metrics: Performance," https://metrics.torproject.org/torperf.html, 2018.

[50] M. Bun, K. Nissim, and U. Stemmer, "Simultaneous private learning of multiple concepts," *CoRR*, vol. abs/1511.08552, 2015. [Online]. Available: http://arxiv.org/abs/1511.08552

[51] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536466 pp. 371–380.

[52] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *13th International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013, pp. 1–10.

[53] R. Jansen, K. Bauer, N. Hopper, and R. Dingledine, "Methodically modeling the tor network," in *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET 2012)*, August 2012.