Florentin Rochet* and Olivier Pereira

Waterfilling: Balancing the Tor network with maximum diversity

Abstract: We present the *Waterfilling* circuit selection method, which we designed in order to mitigate the risks of a successful end-to-end traffic correlation attack. Waterfilling proceeds by balancing the Tor network load as evenly as possible on endpoints of user paths. We simulate the use of Waterfilling thanks to the TorPS and Shadow tools. Applying several security metrics, we show that the adoption of Waterfilling considerably increases the number of nodes that an adversary needs to control in order to be able to mount a successful attack, while somewhat decreasing the minimum amount of bandwidth required to do so. Moreover, we evaluate Waterfilling in Shadow and show that it does not impact significantly the performance of the network. Furthermore, Waterfilling reduces the benefits that an attacker could obtain by hacking into a top bandwidth Tor relay, hence limiting the risks raised by such relays. Waterfilling does not require any major change in Tor, and can co-exist with the current circuit selection algorithm.

Keywords: Tor, Path selection algorithm, anonymity, traffic correlation

1 Introduction

Tor is an implementation of an Onion Routing protocol designed to provide anonymity over the Internet to TCP-based applications. Tor can be seen as a distributed overlay network run by volunteer-operated nodes where the users have an interest in remaining anonymous when surfing on the web. Over the last few years, the Tor network has grown from about 2000 to roughly 7000 relays [4]. It provides anonymity by bouncing the traffic through the network using a path of at

*Corresponding Author: Florentin Rochet: Université catholique de Louvain - ICTEAM - Crypto Group, E-mail: florentin.rochet@uclouvain.be
Olivier Pereira: Université catholique de Louvain - ICTEAM - Crypto Group, E-mail: olivier.pereira@uclouvain.be

least 3 relays among the pool of nodes and rotates the path every 10 minutes [14].

However, anonymity is not a guaranteed property. Tor is known by design to be unable to preserve anonymity in a situation where an adversary observes the traffic entering and leaving the anonymity network. It is indeed possible for an adversary to retrieve the identity of a Tor user and his destination by matching the traffic at both endpoints of the anonymity network. This attack is called end-to-end traffic correlation and is considered in the literature to be a serious threat against anonymity networks. Many works were conducted to demystify the consequences of this threat over the Tor network. Global passive adversaries have been studied with different correlation functions to match streams entering and exiting the anonymity network. From simple packet counting [7, 32] to timing analysis [11, 24], the anonymity provided by the Tor network has showed to be circumvented [29]. However, such adversaries are too powerful and are not part of the Tor original threat model [14]. More realistic passive adversaries have been studied, where either Autonomous Systems (ASes) or internet exchange points (IXPs) are compromised [15, 18, 28]. Experimental results have shown a probability of 20% to encounter the same AS at both ends of a path inside the Tor network. Johnson *et al.* [22] have also shown a probability of 95% to be deanonymized in three months by a single IXP for Tor users located in common places.

Monitoring parts of the Internet is not the only way to perform traffic correlation: since the Tor network is designed to be a volunteer-based network, an attacker can also deploy nodes and use them to perform traffic correlation, waiting for Tor users to pick these nodes at the ends of their path. Johnson *et al.* [22] studied the likelihood of such a comprised path when realistic corrupted nodes are injected into the network. Results shows a probability of 80% to encounter such a compromised path within a 6-month period between 2012 and 2013 for a 100MiB/s relay adversary.

These results show the importance of fighting traffic correlation. One approach is to design counter-measures tailored to the traffic characteristics used to match streams [24, 35]. While this approach has the potential

of completely suppressing the feasibility of one specific traffic correlation attack, it also leaves completely open the possibility to use a different correlation measurement method. A second approach, which we explore in this paper, consists in designing the network in such a way that it minimizes the probability that an adversary could monitor both ends of a path through the Tor network. This second approach cannot fully preclude the feasibility of a correlation attack, but it is more robust than the first one in the sense that it is independent of the correlation detection method that is used. The two approaches can of course be combined and can, in some specific circumstances, interfere with each other [26].

Modifying the path selection algorithm in order to reduce the feasibility of a traffic correlation attack is most appealing if the overall performance of the network stays unchanged. Currently, Tor uses a selection process called Adjusted Bandwidth Weight Random Selection (ABWRS) that combines bandwidth-weights and nodes perceived bandwidth to apply the weighted choice. The perceived bandwidth allows Tor clients to bias toward relays with more available resources while the bandwidth-weights allow to balance the load from one position to another (e.g., if there is too much bandwidth for the entry position, a fraction of each node bandwidth is moved to the middle position).

Our contributions

In this paper, we explore a different way to achieve the balance, providing a more uniform relay selection mechanism to Tor users. Waterfilling replaces the single fraction of the bandwidth attributed to all relays for each (available) position to a fraction defined per relay. While not impacting the total bandwidth of the network, these individual fractions make sure that low-bandwidth relays devote most of their capacity to traffic at the endpoints of circuits, making it possible for high-bandwidth relays to devote a larger part of their capacity to the middle-point of circuits. As a result, top relays become a considerably lower threat to anonymity compared to the current Tor network state.

More precisely, in this paper:
- We suggest a modification of the current Adjusted Bandwidth Weight Random Selection (ABWRS) called Waterfilling Adjusted Bandwidth Weight Random Selection (WFABWRS) or Waterfilling for short. Waterfilling keeps the Tor network balanced and provides either more diversity in both ends or in one end of the Tor network, depending on the network load case.

- In order to evaluate the impact of Waterfilling, we use known metrics and propose a new one, based on guessing entropy [25]. Our metric indicates the expectation on the number of nodes to be compromised before being able to mount a successful correlation attack, and we believe that it provides useful information on the security of the Tor network at a given time.
- We provide a concrete analysis of the benefits of Waterfilling based on simulations executed from consensuses available for 5 months during 2015. Our analysis shows that an attacker would be required to control on average 150 extra nodes in order to run a successful end-to-end correlation attack on a targeted circuit, and that he would need to control 35 nodes in order to obtain the same benefits as if he had compromised the top guard node during the first half of 2015.
- We modified TorPS [5] in order to implement Waterfilling and, on our way, we identified and fixed an issue, which caused the results in previous publications to overestimate the success probability of attacks. Our pull request has been merged in the original github project.
- We developed a prototype of Waterfilling in the Tor 0.2.6.7 base code and we performed concrete performance evaluations in the Shadow simulator, comparing Waterfilling with Tor's ABWRS.

Roadmap

We start by reviewing the related works regarding path selection algorithms and anonymity metrics in Section 2 and provide background information in Section 3. Then, we dive into our subject in Section 4 explaining what is Waterfilling and how we compute our new bandwidth-weights. We then describe in Section 5 our threat model and explain which metrics will be used and why they are suitable. Our Section 6 gives the security analysis of Waterfilling versus unmodified Tor using our metrics and an empirical evaluation against relay adversaries. Our Section 7 gives the performance analysis of Waterfilling in terms of expected circuits latency and time to download a web page or a large file. We discuss why Waterfilling should be easy to deploy in Section 8, and we discuss current limitations and open questions in Section 9. We conclude in Section 10.

2 Related Work

Path selection algorithms

In the original Tor design, the basis of the path selection algorithm was to select uniformly at random the nodes needed to build a circuit. The reason to do this was to meet the theoretical security level of Onion Routing against relay adversaries [39]. However this strategy does not provide a good network performance for Tor users in average, because relays offer very different bandwidths. Therefore, the selection moved to an adjusted bandwidth weight random selection choice, making the selection probability of a node proportional to the bandwidth it offers (see details in Section 3.1). Several other proposals of path selection algorithms have been made during the past few years.

Snader and Borisov studied a tune-up to balance between the anonymity and performance properties [36]. LASTor from Akhoondi *et al.* [6] suggests a path selection algorithm that reduces both the latency and the risk to encounter a circuit where both entry and exit are in the same Autonomous System (AS). Unfortunately, LASTor does not result in a balanced network and suffers from the same throughput issue as uniform selection. Hence, it cannot be deployed as it is suggested [40].

In a complementary way, some methodologies have been developed to improve performance regarding latency on the top of a path selection algorithm. In this line of work, the "normal" path selection algorithm (or another one) is used as a first step, but a second selection step is proposed in order to filter among paths obtained in the first step. Wang *et al.* [41] developed a congestion-aware selection scheme that models the congestion of a circuit as a node-based approach instead of a link-based approach. In this case, the filtering is on the top of the normal Tor adjusted bandwidth weighted random selection. An interesting fact is that the congestion computed for the nodes is not correlated with their bandwidth, hence such a scheme does not unbalance the network. Sherr *et al.* proposed a latency-aware link-based path selection algorithm called Coordinate on the top of Snader and Borisov's scheme [34]. More recently, Wacek *et al.* suggested hybridizing the "normal" path selection algorithm with Coordinate and evaluated its performance [40].

Hybridizing path selections or looking to improve an existing idea have been welcomed and even integrated in Tor in the past few years. In this line of work, we have the Guard flag that is a response to protect against the Predecessor Attack introduced by Wright *et al.* [2, 42]. Researchers evaluated also the well-funded values of some policies. Elahi *et al.* [17] simulated various guard related parameters to assert which approach might be more interesting for Tor users, and Backes *et al.* [9] also explored various path selection strategies.

DistribuTor

The closest proposal of our work is the suggestion by Backes *et al.* at CCS'14 [8]. They present a path selection called DistribuTor that redistributes the bandwidths based on computation performed on the client side (and not by the Tor authorities). In DistribuTor, a bandwidth upper bound is chosen by the client and the choice of an exit node is based on both the bandwidth available on the exit node and on that upper bound: the probability of selecting an exit node is proportional to its bandwidth, except that this bandwidth is considered to be trimmed to the chosen upper bound, effectively bounding the probability of selecting high bandwidth nodes.

Our work differs from that one in several aspects:

– We focus on entry point, which is more important for protection against correlation attacks.
– We push the definition of the probability selection of the various nodes back to the directory authorities, which helps maintaining a balanced network.
– We perform detailed security and performance analysis of our path selection algorithm.

Anonymity models and metrics

The general consensus from the literature regarding modelling anonymity comes from Pfitzmann and Hansen [30] and holds in a set of definitions. First of all, anonymity is defined as the state of not being identifiable within a set of subjects called the anonymity set. The anonymity set stands for the set of all probable subjects. These subjects are linked to anonymous actions which remain anonymous if an adversary cannot distinguish the subjects on which they are executed. The definitions of the subjects and their related anonymous actions are context dependent and defined regarding the anonymity system studied. Surveys over anonymous systems were conducted in [12, 16], including mix networks, Dining Cryptographers networks and Onion Routing.

To quantify anonymity, we use probability distribution about the set of subjects linked to the anonymous actions we study. From these probability distributions, we extract information using the notion of en-

tropy. Entropy provides a quantification of the uncertainty involved in predicting on which subject is linked our anonymous action. Depending on which information we extract, we infer different meaning of the result.

Serjantov and Danezis [31] and Diaz *et al.* [13] are the first to suggest the idea of using information theory metrics such as Shannon's entropy [33] to infer the number of bits of additional information that the attacker needs in order to definitely identify the subject. Moreover, Diaz *et al.* suggested using a normalized version of Shannon's entropy as the degree of anonymity for the system. A degree 0 means that the anonymity system does not provide anonymity at all. A degree 1 means that the uncertainty is maximal. Snader and Borisov [36] suggested using the Gini coefficient as a measure of equality for the subjects. A Gini coefficient of 0 means that we have perfect equality between the subjects. we cannot distinguish one from the other regarding our anonymous action. A coefficient of 1 means a perfect inequality. Those metrics have been extensively used in the literature by the Tor community.

However, even if entropy is widely used, there are some drawbacks to it. For instance, Syverson pointed that Shannon entropy, as an average, does not necessarily capture worst case situations [38]. If we take the following example where we have a set of 1025 potential senders of a message with 1024 of them having an equal probability of 1/2048 to be the origin of the message, and a single sender with probability of 1/2. We end up having a degree of anonymity of 0.6 for this distribution which is quite high regarding the fact that one sender has a strong probability to be the source.

In the Tor community, some researchers started to advocate the use of an empirical anonymity measure based on a well-defined adversary. Hamel *et al.* suggested measuring the probability of path compromise under an adversary with fixed bandwidth capability [19]. Johnson *et al.* built a Tor Path Simulator to mimic the Tor path selection over time and infer statistical confidence about first path compromise under a fixed relay adversary [22]. We also use such metrics in our anonymity evaluation.

3 Background

The Tor network is composed of different types of nodes, *onion proxies* (Tor client), *onion routers* (relays), *directory servers*, *bridges* and *hidden servers*. Di-

rectory servers are responsible for setting the network up by publishing a consensus document every hour that assigns a selection weight for each relay role. Those weights are used to balance the network between the three node positions respectively entry, middle and exit nodes. A relay might have different roles and could act as an exit but also as an entry and a middle node. Thus, the weights allow the resources of a relay to be proportionally distributed among the different roles that it handles. The nodes are divided among the positions according to some status flags assigned by the *directory servers*. We have the *Guard* flag that allows a relay to be picked out by the Tor client as its entry node in the Tor network. Guard nodes must fulfill performance and stability constraints, and we currently have a pool of roughly 1600 Guards among all relays. An *Exit* flag is also assigned to nodes that accept exit policies for range of IP addresses and ports. Exit nodes are picked out by the Tor client to be the node responsible to connect to the requested service. The destination IP and port must match the node exit policy, hence the pool of available exit nodes depends on the service that the user wants to access. All remaining relays are middle nodes.

3.1 Path selection in details

The description above showed that we have two entities involved in the path selection: directory servers and Tor clients.

Directory servers

The Tor Project provides documentation about Tor specifications [1] and explains the responsibilities of directory servers. Among them, the weight computation is our subject of interest. Weights are used to balance the network among the positions, which arise from the solution of a simple system of equations related to the equality of the bandwidths from entry, middle and exit positions. The system of equations from dir-spec.txt is:

$$Wgg.G + Wgd.D = \quad M + Wmd.D + Wme.E + Wmg.G \quad (1)$$
$$Wgg.G + Wgd.D = \quad\quad\quad\quad\quad\quad Wee.E + Wed.D \quad (2)$$
$$D = \quad\quad Wed.D + Wmd.D + Wgd.D \quad (3)$$
$$G = \quad\quad\quad\quad\quad Wmg.G + Wgg.G \quad (4)$$
$$E = \quad\quad\quad\quad\quad Wme.E + Wee.E \quad (5)$$

With:

- G being the total bandwidth for Guard-flagged nodes
- M being the total bandwidth for non-flagged nodes

– E being the total bandwidth for Exit-flagged nodes
– D being the total bandwidth for Guard+Exit-flagged nodes
– Wgd being the weight for choosing a Guard+Exit for the guard position
– Wmd being the weight for choosing a Guard+Exit for the middle position
– Wed being the weight for choosing a Guard+Exit for the exit position
– Wme being the weight for choosing an Exit for the middle position
– Wmg being the weight for choosing a Guard for the middle position
– Wgg being the weight for choosing a Guard for the entry position
– Wee being the weight for choosing an Exit for the exit position

This system constrains a bandwidth equality between entry and middle nodes in equation (1). The same constraint is applied between middle nodes and exit nodes in equation (2). Equations (3), (4) and (5) ensure that the weights are consistent. Of course, these equations cannot always be satisfied, when a resource (e.g., exit nodes) is scarce in the network. In such cases, some of these equations become inequalities, and the bandwidth is allocated on a case-by-case basis: the Tor specification divides all possibilities into 12 cases, and provides constraints for each of them.

It is worth noticing that this system of equations achieves a balanced network condition regardless of the bandwidth of each node independently. It only cares about the sum of each pool (G, M, E, D). It results that if we have to balance bandwidth from one position to another, the same fraction of bandwidth is transferred for each node.

Tor clients
Once the weights have been computed, voted and published in a network status document, the Tor client uses them to assign selection probability to each relay. Each Tor client biases its selection according to the weights received for each position and the consensus weight [1] of the relays. We have:

$$ClientWeight_{(relay_i, position\ p)} = ConsWeight_i * W_{pf}$$

With W_{pf} the weight computed by the directory server, depending on the desired position and the flags of $relay_i$. Then, the Tor client makes a weighted random choice among relays when building circuits, using all computed $ClientWeights$. Consequently, if each Tor user applies this strategy, the Tor network end up having the same bandwidth consumed by relays for entry, middle and exit positions[2]. Also, the ClientWeight of a relay depends directly on its consensus weight which is a value based on the perceived bandwidth of the relays measured by the directory servers. We end up having a selection probability of a relay for a specific position that is directly proportional to its perceived bandwidth:

$$Pr[relay_i, position\ p] = \frac{ConsWeight_i * W_{pf}}{\sum_j ConsWeight_j * W_{pf}} \quad (6)$$

where the sum at the denominator ranges over all nodes and W_{pf} is defined to be 0 for nodes that cannot be placed in position p.

There is however a little extra complexity in the selection process, as some constraints are placed on the structure of a circuit:
– The exit node must have a policy accepting connections to the desired IP address and port;
– The same relay can not be chosen twice for the same circuit;
– Two relays in the same family can not be chosen for the same circuit;
– At most one relay selected in a given /16 subnet for the same circuit.

We will not touch these constraints here.

4 Waterfilling Bandwidth-Weights

We suggest changing the bandwidth weights computation method in order to maximize the diversity selection in guard and exit position. Before diving into the equations and generalizing the concept, we show in the first subsection the intuition behind our Waterfilling strategy with an example.

---

1 The consensus weight is called Bandwidth in the network status document

2 Unless we fall into a scenario where the resource of a position is too scarce

4.1 Looking at the big picture

Figure 1 shows all guard-flagged nodes sorted by decreasing consensus weight (bandwidth). The dashed blue line shows the total capacity of each guard-flagged node received from the network status document. The dotted green line shows the capacity of each guard-flagged node dedicated to the entry position, which is decided by the Wgg value from the network status document. The capacity between those two lines is what is transferred to the middle position to verify the equations previously introduced in Section 3.1. It is a shift of resource from guards to unflagged nodes to obtain the equivalence of total bandwidth consumption from each position.

The plain red line is the result of our Waterfilling scheme. It allows transferring the same amount of capacity to the middle position by moving from a global Wgg to a per-node Wgg_i. Everything above the horizontal red segment (the water level) is transferred to the middle position. That is, we fill the smaller guard-flagged nodes until a level where all above area enclosed between the dashed blue line and the horizontal part of the plain red line is equal to the area enclosed between the dashed blue and dotted green lines. We allocate resources differently but we shift the same amount of capacity as with classic bandwidth-weights, thus leaving the network capacity untouched. However, by using lower-bandwidth guards fully in their guard role, and by capping the use of higher-bandwidth nodes for guard traffic, we obtain a much more uniform probability distribution for guard node selection, which will render correlation attacks more challenging to mount.

The network load case for this particular example corresponds to a situation in which the bandwidth in the exit position is scarce and the total bandwidth of guard position is greater than the total bandwidth of middle position. This situation corresponds to the third case, subcase a in the Tor specification, and we write it *3aE=SG>M*. It is the most representative, and has been observed 97.7% of the time during the first five months of 2015. For the remaining percentage, we are in a single network case, for which we propose a similar Waterfilling strategy.

4.2 Computing Waterfilling bandwidth weights

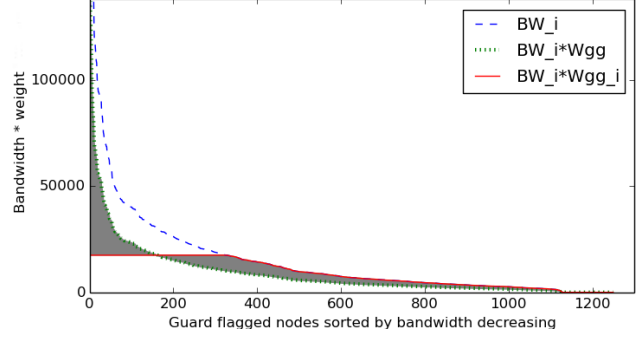Following our example above, we first explain how to compute the Waterfilling bandwidth weights through

Fig. 1. Waterfilling on guard nodes for the 10th consensus from 25th May 2015

the network load case *3aE=SG>M*. For this network load case, the Tor specification indicates to compute the weights from the following equations:

$$Wee = Wed = 1; \tag{7}$$
$$Wmd = Wgd = Wme = 0; \tag{8}$$
$$Wgg = (G + M)/(2 * G) \tag{9}$$
$$Wmg = 1 - Wgg; \tag{10}$$

We can apply Waterfilling to Wgg when Wgg is neither 0 nor 1. Generally speaking, with another network load case, we may also end up with Wee or Wed or Wgd not equal to 0 or 1. If this is the case, the following introduced constraints can be symmetrically applied to Wee and with a small modification to Wed and Wgd.

Now, based on these values, we compute the individual weights Wgg_i of each guard node, from which we can also derive the index N of the *pivot* node, which the last guard to be "above the water level", or the last guard that is going to be used both as guard and middle node. It is the point where the dashed blue line and the horizontal part of the plain red line meet in Figure 1.

Suppose there are K nodes with the guard flag, and let BW_i be the bandwidth of the i-th guard node with the highest bandwidth. Then the new constraints are:

$$Wgg_i * BW_i = Wgg_{i+1} * BW_{i+1} \ \forall i \in (1, N) \tag{11}$$

$$Wgg_i = 1 \ \forall i \in (N + 1, K) \tag{12}$$
$$0 \le Wgg_i \le 1 \ \forall i \in (1, K) \tag{13}$$
$$\sum_{i=1}^{K} Wgg_i BW_i = Wgg * G \tag{14}$$

Equation 11 expresses that all nodes before the pivot must devote the same amount of bandwidth to the guard position. Equation 12 expresses that all nodes after the pivot position will fully play their role as guards.
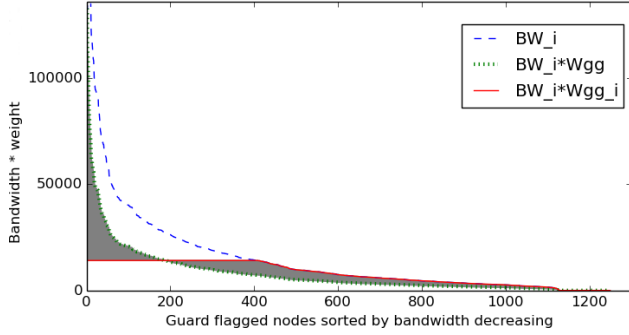
Fig. 2. Better Waterfilling obtained from a recalculation of the bandwidth-weights. Waterfilling is applied on guard nodes for the 10th consensus from 25th May 2015

Equation 13 guarantees that no node will be required to offer more bandwidth than available, and Equation 14 guarantees that the total amount of bandwidth available for the guard position remains unchanged compared to the original Tor strategy, based on the unique Wgg. From a visual point of view, this equation guarantees that both grey areas have equal surface in Figure 1.

Solving with these constraints gives the weight Wgg_i of each node, from which we compute $Wmg_i = 1 - Wgg_i$ for each guard-flagged node.

4.3 Going further with Waterfilling

Waterfilling balances the network on the top of classical bandwidth-weights. We may wonder if those bandwidth-weights are always computed in a suitable way for Waterfilling. In some scenarios, the equations 1 and 2 from Section 3 cannot be verified together: when either $(G + D) < T/3$ or $(E + D) < T/3$.[3] In the previous section, we discuss a consensus where we have the second scenario: $(E + D) < T/3$. In this scenario, we cannot achieve a balance between the three positions. The way Tor specifications suggests computing the weights for this particular scenarios (Equations 7, 8, 9, 10) shows that they verify the equation 1 but release equation 2 to an inequality. We have:

$$Wgg.G + Wgd.D = \quad M + Wmd.D + Wme.E + Wmg.G \tag{15}$$

$$Wgg.G + Wgd.D > \quad\quad\quad Wee.E + Wed.D \tag{16}$$

Therefore, they equalize the bandwidth between guards and middles. We may achieve a better balance for Waterfilling: equalizing the bandwidth between exits and guards and pushing what remains to the middle posi-

---
3 T being the total bandwidth: T=G+M+E+D

tion. The equations become:

$$Wgg.G + Wgd.D < \quad M + Wmd.D + Wme.E + Wmg.G \tag{17}$$

$$Wgg.G + Wgd.D = \quad\quad\quad Wee.E + Wed.D \tag{18}$$

Achieving the balance this way reduces the Wgg value compared to the first approach. It becomes:

$$Wgg = (E + D)/G \tag{19}$$

We can compare Figure 1 and Figure 2 when Waterfilling is applied on these two approaches. On Figure 1, the pivot is around the 340th node while on Figure 2, it is around the 400th node. Consequently, the water level is smaller than with the approach from Tor specifications. Hence, the probability selection on guard nodes are closer to the uniform distribution. We may also wonder if this approach reduces the performance of the network. Intuitively, we have now two positions where congestion might occur instead of one. We answer this concern in our performance analysis explained in Section 7.

5 Security Models and Metrics

5.1 Threat Model

Since its initial design, Tor has been known to lack efficient protection against end-to-end traffic correlation. Solving this problem might be impossible or at least extremely difficult for any low-latency anonymity network. Existing techniques such as padding [35] have shown to be too costly to be deployable while other techniques such as delaying cells or defensive dropping [24] have shown to be ineffective in practice. One way to prevent end-to-end traffic correlation is to accept it and to strive to minimize its impact while guaranteeing a high quality of service.

In this threat model, an adversary controls a bunch of nodes in order to perform end-to-end traffic correlation when a Tor user is passing through one of its guards and exits nodes. Following Murdoch *et al.* [27], a realistic view of a relay adversary considers both IP addresses and bandwidth to contribute to a cost that the attacker tries to minimize. We consider two variants of this model:

– Budget relay adversary: An adversary deploying its own relays into the network. The budget is fulfilled by the cost of the bandwidth and the IP addresses.

In order to avoid obvious detection, a relay adversary would take different /16 if he chooses to deploy several nodes, and distribute them into different data centers, hence increasing its costs proportionally to the number of nodes and total bandwidth.

– Intruder relay adversary: An adversary hacking into existing Tor relays, servers or private computers. In this case, the bandwidth has little direct influence on the cost, since the attacker does not pay for it. The cost will instead be related to the number of machines that need to be corrupted, and to their level of protection. Here, a generic botnet is unlikely to offer a good strategy for an adversary, due to the regular downtime of the corrupted machines [37], and more efforts will then be needed. At the other extreme of the spectrum, big existing relays are likely to be top targets for such an adversary, at least in the current Tor path selection algorithm: these relays already handle a major part of the traffic due to the current path selection algorithm. Besides, the visible effects of a corruption will be lower, since this corruption does not add a new relay traffic on the network. In all cases, due to the relatively high profile of the machines to be corrupted (they need to be able to obtain the guard flag), the number of machines to corrupt appears to be an important measure of the adversary effort.

Based on these observations, we will evaluate the cost of a successful correlation attack both in terms of relays that need to be corrupted and total bandwidth that is required, the relative weight of these depending on the exact setting and willingness of the adversary to hack into other computers.

We make the assumption that a correlation is instantaneous and perfect. Notice that network adversaries are not taken into account in this model.

5.2 Metrics

In order to evaluate the impact of our Waterfilling strategy, we use three anonymity metrics.

Our first metric, used in related works, is the *uniformity degree of circuit selection*, computed as Diaz's degree of anonymity [13] over the probability distribution of selected guard-exit pairs in the Tor network. This metric gives an indication of how well the network is exploited at a given point in time, and is normalized (that is, independent of the network size).

Our second metric is based on the notion of *guessing entropy* [25], and indicates the expected number of nodes that an adversary should control in order to mount a successful end-to-end traffic correlation attack. This metric is related to the previous one in the sense that it also focuses on the state of the network at a single point in time, but it provides a more concrete information to the user.

The last metric we use is the *probability distribution on time until first path compromise*, which is an empirical metrics from Johnson *et al.* [22]. Contrary to the other two, this metric adopts a dynamic perspective by considering, across a certain amount of time, the success probability of an adversary who controls a predefined number of nodes and offers a predefined bandwidth on these nodes.

5.2.1 Uniformity degree of circuit selection

The uniformity degree of circuit selection measures how close to uniform is the selection process of a guard and an exit node for a circuit. It shows a first interesting indication of the resistance of the network to end-to-end correlation attacks.

This uniformity degree is computed by evaluating the probabilities $p_{i,j}$, which indicate, for all i and j, the probability of picking the guard i with exit j in a circuit. Then the Shannon entropy of this distribution is computed as: $H(Y) = -\sum_{i=1}^{N} \sum_{j=1}^{K} p_{i,j} \log_2(p_{i,j})$.

This quantity is then normalized to the maximum entropy H_M that this distribution could have, which is computed as $log_2(N * K)$, where N and K are the size of the set of guard and exit nodes respectively.

So, the uniformity degree d of a circuit selection process is computed as: $d = \frac{H(Y)}{H_M}$.

Note that it may not be desirable to obtain a uniformity degree of 1, as it would not take into account any topological aspect of the circuit, for which policies are in place (e.g., we cannot have two same /16 addresses in the circuit).

The uniformity degree of circuit selection is interesting to compare the quality of a path selection algorithm on different states of the Tor network that do not necessarily contain the same number of nodes.

We evaluate this metric by using the TorPS tool to run simulations of the original and Waterfilling based circuit selection processes, based on various consensuses from the first five months of 2015: these simulations offer estimations of the $p_{i,j}$ values, from which the uniformity degree can be evaluated as defined above.

5.2.2 Guessing entropy

The uniformity degree provides an interesting indication on the distribution of the circuit selection process, but does not provide a measure that can easily be interpreted in terms of success probability of an end-to-end correlation attack: Shannon entropy is an average measure, not a worst-case one, and the normalization aspect makes it possible to obtain a very high degree of uniformity even for a Tor network that could contain 3 nodes.

In order to address these questions, we propose using the notion of *guessing entropy*, as a measure of the number of nodes that an adversary must expect to control or compromise before being able to deanonymize a specific circuit.

The guessing entropy is computed from the same probabilities $p_{i,j}$ as above, by ranking the relays in decreasing order of their contribution to the success probability of a successful end-to-end correlation attack.

We then define q_i as the marginal probability of a successful attack (that is, the increase of success probability of the attack resulting of the compromise of the i-th node), and evaluate the guessing entropy of the resulting distribution:

$$g = \sum_{i=1}^{N+K} i.q_i$$

Of course, we always have $q_1 = 0$ because we cannot mount a correlation attack from a single compromised node, but we choose the first node in order to maximize the impact of the compromise of a second node. In order to compute the remaining elements of our vector q, we keep selecting nodes in a monotonic way such that q_i is maximal with respect to the $i - 1$ already selected nodes.

$$Pr_{G=x} = max\left(\sum_{y \in q} Pr(G = x, E = y) \forall x \notin q\right) \quad (20)$$

$$Pr_{E=y} = max\left(\sum_{x \in q} Pr(G = x, E = y) \forall y \notin q\right) \quad (21)$$

$$q_i = max(Pr_{G=x}, Pr_{E=y}) \quad (22)$$

An example is available in Appendix A.1.

This selection process corresponds to a monotonic strategy in which the adversary compromises relays one after the other, looks for the best choice based on the relays that he already compromised, and does so until his attack succeeds.

This strategy is definitely more effective than simply looking for individual nodes from the two sets and constructing the product of the distributions rather than

our joint distribution, as shown by Johnson et al. [21]. A more effective attacker strategy would be to select sets of relays to be jointly compromised instead of picking them one by one, but such a strategy would also be considerably more difficult to translate into a simple metric due to combinatorial explosion.

5.2.3 Time until first compromise

This measure gives an estimate of the evolution over time of the probability until a first path compromise happens, for an adversary controlling a specific set of nodes.

To evaluate this measure, we repeatedly simulated the circuit selection process of a client during a period of 5 months using TorPS, and used these simulations to estimate the probability of building a compromised circuit over time. We applied this strategy with the current Tor selection scheme and with our modified Tor Waterfilling selection scheme, for comparison. More details about our relay adversary are given in Section 6.

This measure has the advantage of giving an interesting insight regarding a concrete threat, as far as TorPS correctly mimics Tor circuit selections. And, as a complement information to the metrics discussed above, this one integrates a dynamic aspect of circuit compromise over time, and not just at a given point in time, hence taking into account path rotation and relay instabilities.

However, it is also a very specific measure, that is essentially valid for a particular choice of relay adversary and the time period that is chosen.

6 Security Analysis

We now evaluate our Waterfilling scheme by comparison with the current Tor path selection process, based on the three metrics discussed in the previous section.

6.1 Methodology

To evaluate the security of the Tor network with respect to our threat model, we need to compute the probability distribution of node selection. There are multiple ways to do it. A first one would have been to use the equations from Sections 3.1 and derive probabilities accordingly. However, the resulting distribution would not account

for the extra path construction policies (e.g., no pair of nodes from the same /16 range on a single circuit) or for realistic user behaviors (e.g., it does not take into account exit policies that can also shape path selection).

In order to obtain more realistic results, we used the TorPS tool with the objective to evaluate the probability distribution considering the typical user model from [22]. This user model has been designed to mimic average Tor use with simulated connections to Gmail / Google Chat, Google Calendar/Docs, Facebook and web search.

While working with TorPS, we noticed and fixed an issue where the addition of relay adversary bandwidth was not considered in the computation of the bandwidth-weights. Thus, adding nodes in the simulated network made TorPS inaccurately reflect Tor's path selection: the higher the injected bandwidth, the further the TorPS results diverged from the reality. As a result, our simulation results should not be directly compared to those obtained by Johnson *et al.* [22].

We studied the anonymity of the Tor network over the first 5 months of the year 2015 [4]. Inside this range, we pick 20 given moments for each type of network load/resource scarcity appearing in this period. From these simulations we evaluate our various metrics.

We also compare our Waterfilling scheme and Tor's ABWRS against a relay adversary. For both schemes, we compute the time until the first compromise path and we discuss different relay adversary strategies.

6.2 Analysis

During the first 5 months of 2015, only two of 12 possible network load cases have been observed: in both cases pure exit nodes were scarce ($E < T/3$); most of the time adding the nodes with the guard and exit flags was not enough to remove scarcity ($E + D < T/3$, case 3aE=SG>M) but, at a few moments, they were sufficient ($E + D \geq T/3$, case 3bE=S). Figures 3 and 4 show how the uniformity degree of circuit selection and guessing entropy evolve between various points in time for these two network load cases. Our analysis focuses on guards rather than on exit nodes: when exit nodes are scarce, no Waterfilling strategy can be applied (or, in other words, the water level reaches the consensus

---

4 We wanted to update with first semester of 2016 but the Stem library used by TorPS is outputting many errors when reading descriptors. We got then many missing descriptors.

weight of the fastest exit node). Figure 3 uses equations from Section 4.3 recomputing the bandwidth-weights and applying Waterfilling on the top of them using the constraints 11, 12, 13 and 14 from Section 4.2. For Waterfilling simulations on Figure 4, we use equations from Section 4.2 because, for this network load case, a balance between the three positions is achieved. For this network scenario, we still have too much bandwidth in the guard position. We apply Waterfilling constraints 11, 12, 13 and 14 to obtain a per-node Wgg_i. We also apply Waterfilling on the D set since $(E + D) \geq T/3$ as depicted in the appendix.

In all cases, we observe that these metrics favor the Waterfilling scheme over Tor's ABWRS: we have an improvement of around 150 nodes for network case 3aE=SG>M and about 130 nodes for network case 3bE=S. It is an increase of about $\approx 25\%$ for the guessing entropy and an increase of about $\approx 2\%$ for degree of uniformity (which is harder to interpret). The difference that appears in the metrics between those two network cases can be explained. For 3bE=S, the network achieves a balance for the three positions but not for the network case 3aE=SG>M. It results that the network should achieve better performance for case 3bE=S than case 3aE=SG>M but less diversity because, for 3aE=SG>M we can put in practice the idea introduced in Section 4.3.

We also observe an overall correlation between these two metrics: both are impacted by the uniformity of the probability distribution. But, divergences also happen, as the guessing entropy takes also into account the number of guards and exits: the more we have guards and exits in the network, the higher would be the guessing entropy (except if the distribution is highly non uniform), while such a change may not impact the degree of uniformity. As a result, a variation of nodes in the network between two given moments like the loss of a bunch of guards may result in a drop of the guessing entropy while the degree of uniformity could stay identical.

Entropy measures are a good estimation of anonymity at a given moment for overall usage but they might also be suitable to compare particular user behaviour. Indeed, the guessing entropy would have been smaller in Figures 3 and 4 if we had considered bittorent download as user model instead of simple HTTP requests, due to the smaller number of exit nodes that allow bittorent ports. On the other hand, the degree of uniformity could fail to show that bittorent users are less protected against end-to-end traffic correlation, simply because the probability distribution of exit nodes allow-
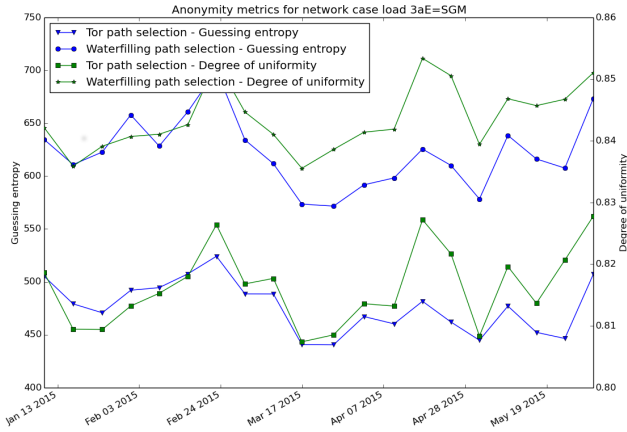
Fig. 3. Anonymity metrics over 20 given moments where network load case is 3a exit nodes are scarce and there is more bandwidth in Guard position than in middle position (3aE=SG>M) during 01-2015 to 05-2015. Waterfilling simulations use equations from Section 4.3

Fig. 4. Anonymity metrics over 20 given moments where network load case is 3b and exit nodes are scarce (3bE=S) during 01-2015 to 05-2015

ing bittorent might have the same shape as the probability distribution of exit nodes allowing HTTP requests.

We saw from [38] and [22] that measuring anonymity at a static point in time might not be sufficient: an adversary may keep monitoring the network for some time and, depending on the way users update their circuits, have a lower or higher probability of winning end-to-end correlation attack at *some* point in time. But, as Waterfilling does not impact the circuit evolution strategy, the time evolution is the same for both approaches.

In the spirit of the work of Johnson *et al.* [22], we may still wonder about the concrete impact of a specific adversary over time. Figure 5 shows the time until first compromise circuit obtained from TorPS with a relay adversary owning a guard with a consensus weight value of $480,310$ and an exit with consensus weight value of $282,607$. Both values have been chosen by computing the consensus weight average of the top-1 guard and exit among the 5-month time period. In the current design of relay selection, top guards are a threat to anonymity since they handle the major part of the traffic in the entry position. Our Waterfilling scheme largely mitigates this issue: Figure 5 shows a drop from $\approx 24\%$ probability to use a compromised path after 5 months to $\approx 2\%$. This results from the fact that, with Waterfilling, most of the capacity of the adversary guard is used for the middle position, and no single node runs a very significant part of the guard traffic.
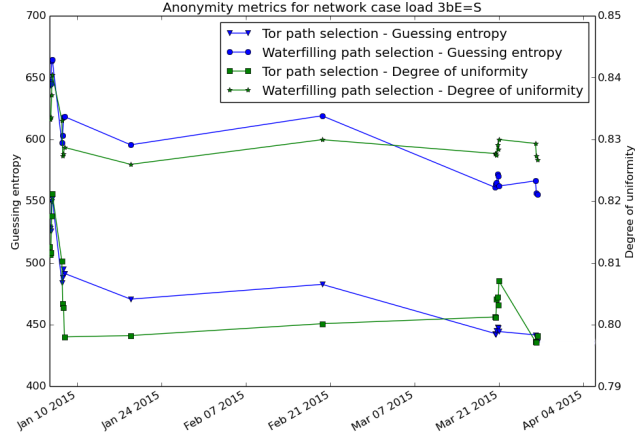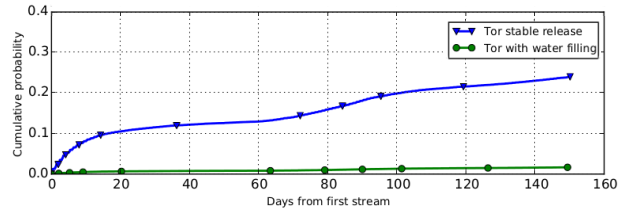

Fig. 5. Empirical evaluation of time to first compromise path - With numEntryGuards=3 - We add 1 adversary guard with 480,310 cons weight, 1 adversary exit with 282,607 cons weight

Effect of Waterfilling on guard rotation

We may also consider the potential interaction that Waterfilling applied on guards could have on guard rotation. In theory, the guard rotation period is randomized between 2 and 3 months. However, in practice, clients rotate guards more often on average due to guard nodes losing their flag or vanishing from the network. Using Waterfilling, we may consider an eventual impact on the average guard rotation period. Indeed, using more smaller guards may affect the expected amount that clients must choose additional guards due to a difference in reachability/uptime with respect to bigger guard nodes. We used TorPS to investigate this phenomenon. Figure 6 shows a comparison between two simulations. We compare results for classical Tor with 1 adversarial guard of $480,310$ consensus weight with the Waterfilling experiment given 35 adversarial guards cumulating $298,752$ consensus weight (62.2% of $480,310$). We use 62.2% of the consensus weight because at average, guards were used at 62.2% of their bandwidth in the entry position (and 37.8% in the middle position). While
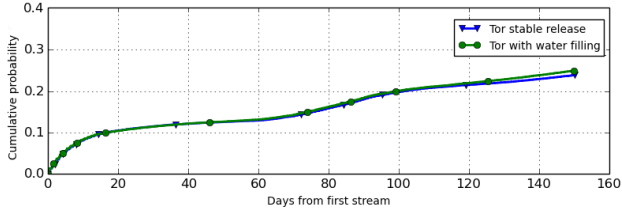
Fig. 6. Empirical evaluation of time to first compromise path - 35 adversarial guards cumulating 298,752 cons weight (Waterfilling) versus 1 guard of 480,310 cons weight (Tor ABWRS).
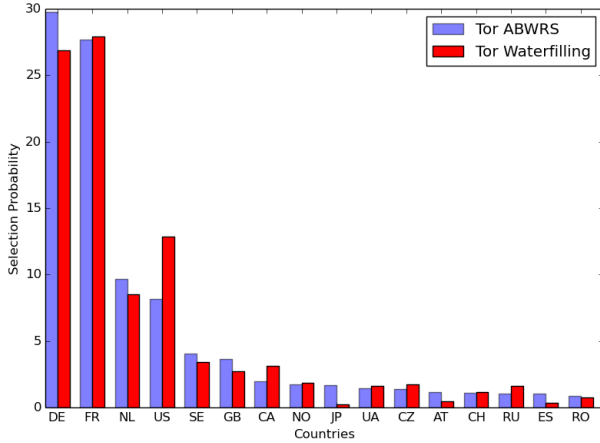
ABWRS	Waterfilling
AS12876:**15.73%**	AS16276:**18.21%**
AS16276:**13.26%**	AS12876:**11.24%**
AS24940:**10.87%**	AS24940:**10.82%**
AS24961:**4.42%**	AS24961:**3.54%**
AS8972: **3.71%**	AS200130:**2.08%**

Table 1. Probability to end up using a guard from the Top-5 AS

Fig. 7. Probability to end up using a guard from the top-16 countries. Like in Section 4, the 10th consensus from 25th May 2015 is used.

with Waterfilling, smaller guards below the water level are used at 100% of their bandwidth in the entry position. If we were in an ideal situation, where guard nodes never disappears, both curves would perfectly match. Using historical data of the Tor network, Figure 6 shows that using Waterfilling, the average guard rotation is slightly faster over time, leading to a higher probability of compromise ($+ \approx 1\%$). It means that smaller guards tend to be slightly less stable than top guards, but hopefully not that much.

Security against network adversaries
Despite the fact that network adversaries are not considered by our security model, we may still be curious regarding the impact that Waterfilling has against them. We use equations 6 from Section 3 to compute the selection probabilities of guard nodes. We group them by country and autonomous system, which allows us to compare the network diversity provided by Waterfilling and Tor ABWRS against these two types of network adversary. Figure 7 shows an overall similarity between

the two schemes. An overall diversity improvement or degradation compared to Tor ABWRS would be a sign of correlation between the size of relays and their localization. Here, we see in Figure 7 some changes to the few top countries where the probability to end up in the 2-top countries is a little bit decreased, from 57.41% to 54.01%. However, US jumps from 8.14% to 13.64%. It means that among the few top countries, the US hosts the highest number of guards with a lot of them below the water level. Table 1 shows AS16276 (OVH Hosting located in US) jumping from 13.26% to 18.21%, which explains the US jump on Figure 7. In the meantime, AS12867 (Online S.A.S located in France) being the most interesting AS to tap in (containing top guards) is loosing interest. In any case, it seems hard to draw any clear conclusions from these numbers, as they are going both ways, and could change with the evolution of the Tor relay localization and bandwidth. This is a consequence of our choice to focus the application of Waterfilling at the relay level rather than at a higher aggregation level (country, AS, . . .) (see also discussion in Section 9.1).

7 Performance Analysis

We now evaluate the performance impact of Waterfilling on a simulated Tor network run in Shadow [20].

7.1 Methodology

To evaluate the performance, we implemented Waterfilling in the Tor source code and ran several simulations in a virtual network. We constructed a topology scaled down to 796 relays from a consensus in May 2016[5] (195 guard-flagged nodes, 501 unflagged nodes, 56 exit-flagged nodes, 43 guardexit-flagged nodes, 3 directory authorities and 1 bandwidth authority). The virtual

---

5 2016-05-28-01-00-00 precisely

network topology is built to mimic the current internet structure with geographic clustering by country as explained in Jansen and Hopper [20]. We generate two classes of experiments. The first one shows simulations in a low network load where we simulated 1125 web clients with 3% of them performing only bulk transfer. Moreover, we have 75 perf clients for each of 50KiB, 1MiB and 5MiB file size. The overall throughput of this setup is on average ≈ 120 MiB/s, which is proportionally small compared to the traffic load handled by the current Tor network. The second class of experiments shows simulations in a heavy loaded network proportionally close to the throughput of the real Tor network if we don't take into account internal traffic. We run 3000 web clients with 10% performing bulk transfer. Moreover, we still have 75 perf clients for each of the file size mentioned above. Altogether, those clients push ≈ 550 MiB/s, which is $\approx 63\%$ of the total capacity dedicated to the exit position (E+D) in our virtual network.

In both class of experiments, the geographical location of the clients are set up according to Tor's directly connecting user statistics [3]. These clients are performing HTTP GET requests to 130 servers with geographic locations assigned using the Alexa Top Sites data set. Therefore, the Tor circuits built by those clients and connecting to the servers should be representative of the real Tor circuits built over the Internet.

7.2 Analysis

We study the impact of Waterfilling with two performance metrics: the time to first byte (ttfb) and the time to last byte (ttlb). The first metric shows responsiveness of the network while the second one captures the overall performance. The first class of experiments is under light load, Figure 8 shows no performance discrepancy when using Waterfilling. We may explain latency results (Figure 8a) in the following way: regarding responsiveness of the network, we have two phenomena to consider. relay-based latency and link-based latency. Relay-based latency has shown to be uncorrelated with relay bandwidth in a previous work [41], thus it does not interfere. Link-based latency is more interesting. To explain why we observe no difference, we may use the following example. Let's suppose that top-bandwidth guards are connected to better links compared to low-bandwidth guards. Therefore, at average, the part of the Tor circuits surrounding guards would suffer from higher latency when using Waterfilling because the clients use guards more uniformly. However, the reverse is applied

with middle nodes because we use them less uniformly than current Tor's ABWRS. In average, the part of the Tor circuit surrounding middle nodes would have less latency due to a higher utilization of good links surrounding top nodes. So, with these hypotheses, we should expect more latency through guards and less latency through middles. Because ttfb only cares about the global latency of Tor circuits, these differences tie summed up on average, and we obtain the same result.

The ttlb metrics (Figures 8b, 8c) confirms our intuition that Waterfilling does not modify the overall performance of the network because it keeps the same total amount of bandwidth for each position, as classical bandwidth-weights. Moreover, we also ran a simulation with half of the clients using Waterfilling weights while the other half are using classical bandwidth-weights. As expected, both techniques can be used simultaneously in the network without impacting the performance. This result allows a smooth transition from Tor ABWRS to Waterfilling or a coexistence if it does not reduce the anonymity set. More details are discussed in Section 8.

Section 4.3 discussed that the classical bandwidth weights can be calculated in a way that improves Waterfilling even more, achieving what we wanted: balancing the Tor network with maximum diversity. The new bandwidth-weights induce the same scarcity between end positions and push what remains to the middle position. Hence, instead of having only one position with low capacity (exit), we now have two positions with low but equal capacity (entry and exit). In Figure 9, we investigate if such strategy lowers performance. Under light load, the result show that such strategy does not raise the latency or lower the overall performance of the network. The *Bwweights modified* curves are the bandwidth-weights recalculated using the strategy described in Section 4.3.

The second class of results are experiments under a loaded network. Figure 10 shows that congestion is increased when using Waterfilling. About 3% of the worst circuits become a little bit worse, meaning that a few circuits are more likely to suffer from congestion. However, Figure 10c shows that this degradation seems not to appear for large bulk transfers. A surprising and welcoming result comes from the strategy where we apply Waterfilling on the top of recalculated bandwidth-weights. We expected to have more congestion with two positions scarce instead of one. However, the results show that congestion is reduced compared to Waterfilling on the top of classical bandwidth-weights. It seems that pushing as much bandwidth as we can to the middle position has a good impact for the network performance.
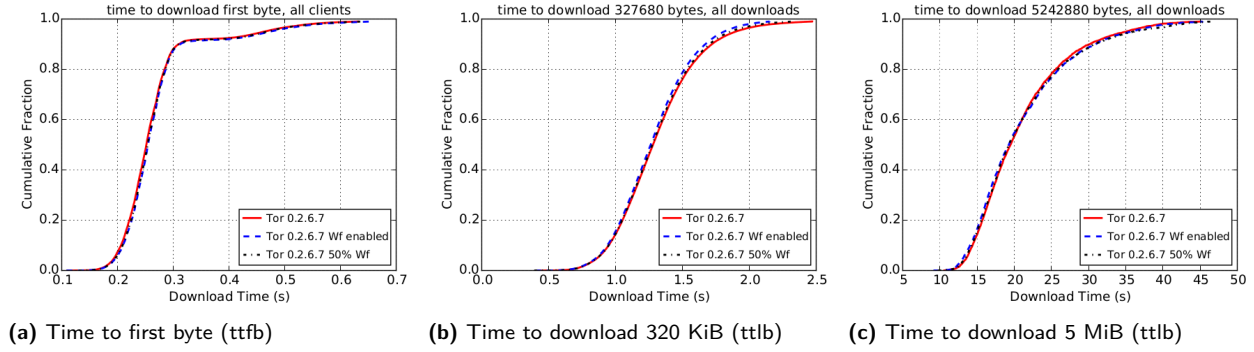
(a) Time to first byte (ttfb) **(b)** Time to download 320 KiB (ttlb) **(c)** Time to download 5 MiB (ttlb)

Fig. 8. Network performance under light load (796 relays, 1125 web clients with 30 bulk clients, 75 perf clients for each of 50KiB, 1MiB and 5MiB file size performing an average throughput of ≈ 120 MiB/s altogether). Comparison of Waterfilling fully used on all clients versus Tor's ABWRS used on all clients versus mix 50% of clients doing Waterfilling and 50% of clients doing Tor's ABWRS – The topology generated is scaled down from the first consensus from May 28, 2016. Graphs are plotted until percentile 0.99.
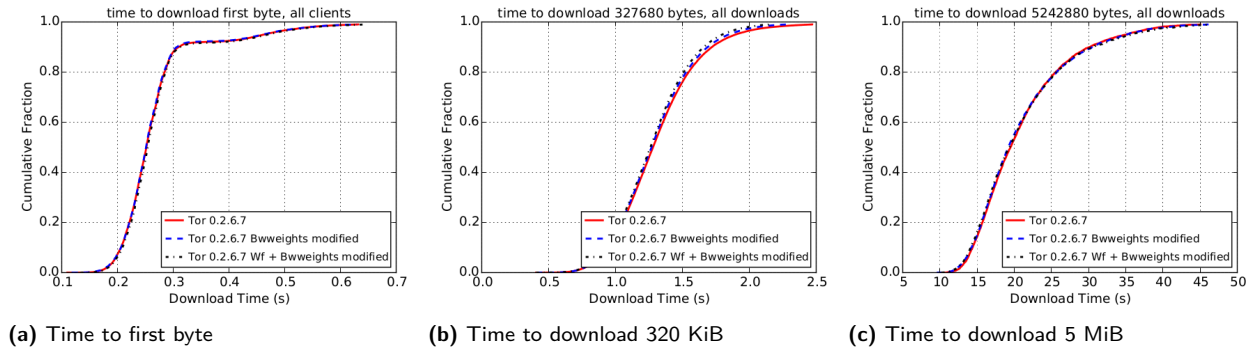

(a) Time to first byte **(b)** Time to download 320 KiB **(c)** Time to download 5 MiB

Fig. 9. Going further with Waterfilling: network performance under light load. Tor with bwweights modified means that we have used equation 19 on Tor's ABWRS to obtain the same capacity between entry en exit while pushing what remains to the middle position. Applying Waterfilling on the top of this weight computation gives a more uniform selection probability compared to Waterfilling on the top of classical bandwidth weights. Graphs are plotted until percentile 0.99.

Since applying Waterfilling on the top this choice improves also anonymity (Section 6), this result is very welcome. Moreover, pushing as most as we can to the middle position is probably the right thing to do because the balance performed by the bandwidth-weights assumes 3-hop Tor circuits even thought it is not always true. Indeed, Tor circuits carrying hidden service traffic are composed of 2 guards and 4 middles. Moreover, there are also some weird users configurations (such as more than 1 middle node) in the wild. Therefore, we may expect that pushing every exceeding resource to the middle position would give even better performance results on the real Tor network compared to our simulations.

8 Deployment

Deploying Waterfilling would not raise performance issues and allows a smooth transition between the current path selection and Waterfilling. In theory, the equations from Sections 3.1 and 4.2 explain this claim because they still hold in the same way. However, in practice, some other issues might be captured. We conducted the performance analysis using Shadow in Section 7 and showed classical bandwidth-weights and Waterfilling can be used together by different Tor clients in the network without disturbing it. The only thing that could be observed from the relay operator's viewpoint is either a slow transition of their relay bandwidth consumption from entry to middle or exit to middle if their relay capacity dedicated to this position is higher than the water level or a slow transition from middle to entry or middle to exit if their dedicated relay capacity is smaller than the water level. The time needed for the tran-
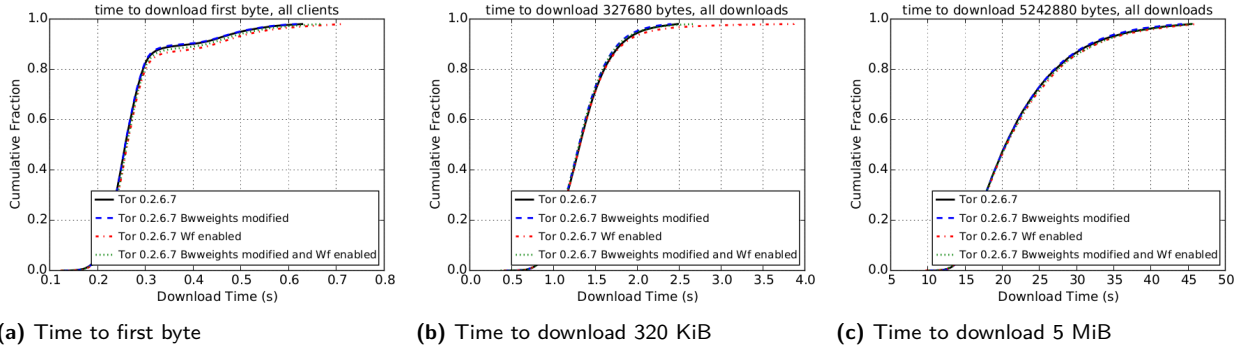
(a) Time to first byte   **(b)** Time to download 320 KiB   **(c)** Time to download 5 MiB

Fig. 10. Network performance under loaded traffic ($\approx$550 MiB/s, 63% of E+D advertised bandwidth). Comparison of Waterfilling fully used on all clients versus Tor's ABWRS fully used on all clients. Graphs are plotted until percentile 0.98.

sition would depend on Tor users updating their Tor client. Hopefully, since Tor 5.x.x, the Tor browser auto-updates, which would bring up to date with Waterfilling a major fraction of Tor users.

The deployment security risk

As explained by Backes *et al.* [9], the transition phase or a co-existence of path selection algorithms might be a threat to the anonymity. A bunch of compromised middle nodes might distinguish Waterfilling users from ABWRS users. Moreover, they can also have an idea of what type of service they are accessing, looking at the exit policies. In the case of a transition phase, the first few users switching to the new path selection algorithm are vulnerable. In the case of a co-existence of path selection algorithms, the users are not vulnerable if the transition is finished and both anonymity sets are considered large enough. Hopefully, we could manage to achieve a secure transition phase with a consensus parameter that would be turned on to authorize Waterfilling when enough users have updated.

We specify here a proposal of which type of change should be made in the directory protocol in order for it to be deployed. We suggest modifying the network status document [1] and add a parameter in *params* called «UseWaterfilling »with possible values 0 or 1. When 0 is set, classical bandwidth-weights are used. When 1 is set, the network status document has to give per-node bandwidth-weight values with at most 7 new weights depending on which set(s) Waterfilling is applied to (guards, exits, guards+exits are 3 disjoint sets leading to a possible modification of Wgg, Wmg, Wee, Wme, Wgd, Wed, Wmd). We suggest adding an item called "wfbw" to each router status entry followed by the optional weights in the form $W_{fp} = <value>$.

9 Limitations and Future Work

The previous sections discussed the impact of Waterfilling on the design of end-to-end correlation attacks and its expected impact on the efficiency of the Tor network. We now discuss some open questions and possible adversarial reactions to Waterfilling.

9.1 Adversary position and cost structure

Our analysis focused on an adversary running an end-to-end correlation attack by controlling network relay, with a primary focus on the number of relays that would need to be controlled in order to mount a successful attack. This captures an adversary who would mount his attack by hacking into existing relays, assuming that the cost is proportional to the number of machines to compromise, or paying an infrastructure for running relays, in which the number of relays to run would be an important part of the cost (matching the cost structure of many cloud service providers, for instance).

Other adversary positions and cost structure could be considered, though.

From one to many relays threat

The analysis in Section 6 shows that the Waterfilling strategy is very effective against adversaries targeting top relays: it forces an adversary to monitor a number of relays in order to achieve what he could obtain from a single relay right now. This situation holds if we consider relays as individual threats. This is a reasonable assumption in the current path selection since dividing the bandwidth into multiple relays increases the cost

without increasing the utility in term of attack effectiveness.

With Waterfilling, the situation changes. Adversaries could react by taking control (or creating) a large number of relays with a more limited amount of bandwidth. Assuming that the adversary focuses on relays that offer an amount of bandwidth close to the "water level", we can evaluate the number of relays that need to be controlled in order to achieve the same effect as controlling the top guard in the current network.

As explained in Section 6, we observed that the top guard achieved an average consensus weight of 480,310 during the first half of 2015. We also observe that the average value of Wgg during the same time period is 62.2%, meaning that $480,310 * 0.622 \approx 298,752$ of the consensus weight of the top guard is devoted to its guard role. During that same period, the water level was as low as 8710. So, in order to achieve the same consensus weight in total, we see that $298,752/8710 \approx 35$ nodes are needed.

We suspect (and leave for further research) that moving from an individual threat model to a multiple-relays threat model is beneficial to the Tor network: running such a strategy would be more difficult for the adversary:

- Regarding budget adversaries, the attacker would have to launch and run his botnet of guard nodes in such a way that would not raise suspicion. The proper way would be to imitate the appearance distribution of other guard nodes, in time and localization. The attacker would need a higher amount of time to be fully operational and would face the difficulty to find proper datacenters to put his nodes. We suspect that Waterfilling increases the effort to set up an effective botnet, hence increasing the budget.
- Regarding Intruder adversaries, we see that the adversary now needs to hack into 35 computers instead of one, which must be quite more complex. Using Waterfilling saves the top-bandwidth relays from also being top targets.

A potential downside of Waterfilling could appear if we focus on the bandwidth required to mount an attack rather than on the number of nodes that need to be controlled. Indeed, thanks to the unique value Wgg in the current Tor network, only 62.2% of the bandwidth controlled by the attacker is used for the guard role, and this is the useful part for running an end-to-end correlation attack. Waterfilling provides additional utility in terms of attack effectiveness if the attacker splits the bandwidth in multiple relays at the water level. When using Waterfilling, an adversary aiming for nodes below or up to the water level can hope to have 100% of its bandwidth allocated to its guard role. As a result, even if the adversary needs to control 35 nodes with Waterfilling instead of one, he only needs to provide 62.2% of the bandwidth that he would need to offer in the current Tor network. An open question is to evaluate the cost of running many nodes to benefit from this additional utility versus the cost of the same utility in the current Tor network.

An interesting adversary reaction against Waterfilling is to take advantage of a botnet structure: among many compromised computers, a few of them might have the required bandwidth and be stable enough to acquire the Guard flag. Such adversary would obtain the additional utility provided by Waterfilling at an additional cost of none, compared to the current path selection. An open question would be to evaluate the number of such compatible botnets in the wild and what would be the cost of running one of them.

One mitigation of this many-relays threat would be to make a new use of the *NodeFamily*: we could apply Waterfilling by considering nodes with the same position and within the same family as a single node. As a result, n guards belonging to the same family and all offering a bandwidth equal to the water level would be selected to offer a *total* guard bandwidth equal to the water level (and not n times that level).

Network adversaries

Analysis in the paper did not show any significant impact of Waterfilling against AS adversaries, given recent localizations of the Tor network relays. But this could change, for the better or for the worse, as new relays appear or disappear. If network adversaries are decided to be the major threat compared to relay-level adversaries, then an interesting open question would be to explore the impact of applying Waterfilling at the AS or country level rather than at the relay level. One contribution to the security analysis in this field [10, 22, 23] would be to use our guessing entropy to calculate the number of ASes that an adversary must expect to control or compromise before being able to deanonymize a specific circuit. Network adversary reactions should be taken into account in the analysis. Regarding the performance aspect, we conjecture that its marginal impact would be the same if we preserve the total amount of bandwidth per position. Nevertheless, a proper analysis should be conducted.

9.2 Interaction with other path selection algorithms

Waterfilling, like the current path selection algorithm, is used to perform preemptive built of circuits (circuits are built before the user needs them). Some other path selection algorithms have been designed to choose the path between built circuits that maximizes an objective function [10, 34, 41]. An interesting research direction is to look for a combination of different approaches. In this spirit, a combination of destination naive AS-awareness [10] and Waterfilling on relays might lead to better security against relay adversaries and network adversaries with minimal loss of performance. We may even compare this strategy against a mix of Waterfilling on relays and on ASes.

10 Conclusion

In this paper, we present a solution to both balance the network and optimize the diversity in endpoints of Tor circuits. This method is called Waterfilling and could be applied in most of the network load cases detailed in the directory server specifications.

We carry out a security analysis of our scheme with the help of information theoretic metrics and with an empirical estimation over time of the probability until a first path compromise. This estimation is obtained from a modified version of TorPS that implements Waterfilling. We suggest using guessing entropy as a new metric to indicate the strength of the Tor network against relay adversaries at a static point of time. Indeed, guessing entropy captures more aspects than previously used theoretic-information metrics and its meaning may be more informative about the hardness of breaking anonymity, compared to previous notions, like diversity. We show that the guessing entropy increases by about 130 to 150 nodes, or about 25% for any moment throughout the period we consider; meaning that, on average, an adversary would have to corrupt around 130 more nodes before being able to complete a traffic correlation attack at a given time.

Our security analysis shows also that, when using Waterfilling, a relay adversary needs to control 35 guards in order to obtain the same attack success probability as an adversary controlling the top guard in the current Tor network. We conjecture that taking control of 35 nodes while remaining undetected is a considerably higher challenge than controlling a single one.

We also perform a performance analysis of Waterfilling. Using Shadow, we set up a private network of ≈ 800 nodes. Our results show a small degradation of performance that might be considered negligible. Indeed, about 97% of Tor circuits constructed with Waterfilling gives the same performance as the circuits constructed by Tor ABWRS.

Acknowledgement

References

[1] Directory authorities specifications from the tor project. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. Accessed: 2016-01-04.

[2] Entry guard design discussions on the tor blog. https://blog.torproject.org/category/tags/entry-guards. Accessed: 2016-01-04.

[3] Estimated number of clients in the tor network. https://metrics.torproject.org/clients-data.html. Accessed: 2016-07-26.

[4] Number of relays in the tor network. https://metrics.torproject.org/networksize.html. Accessed: 2016-01-04.

[5] Torps's github project. https://github.com/torps/. Accessed: 2016-01-04.

[6] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, May 2012.

[7] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[8] M. Backes, A. Kate, S. Meiser, and E. Mohammadi. (nothing else) MATor(s): Monitoring the anonymity of tor's path selection. In *Proceedings of the 21th ACM conference on Computer and Communications Security (CCS 2014)*, November 2014.

[9] M. Backes, S. Meiser, and M. Slowik. Your choice mator (s): large-scale quantitative anonymity assessment of tor path selection algorithms against structural attacks. *Proceedings on Privacy Enhancing Technologies*, 2016(2):40–60, 2015.

[10] A. Barton and M. Wright. Denasa: Destination-naive as-awareness in anonymous communications. *Proceedings on Privacy Enhancing Technologies*, 4:356–372, 2016.

[11] G. D. Bissias, M. Liberatore, and B. N. Levine. Privacy vulnerabilities in encrypted http streams. In *Proceedings*

of *Privacy Enhancing Technologies workshop (PET 2005)*, pages 1–11, May 2005.

[12] G. Danezis, C. Diaz, and P. F. Syverson. Systems for Anonymous Communication. In B. Rosenberg and D. Stinson, editors, *CRC Handbook of Financial Cryptography and Security*, CRC Cryptography and Network Security Series, pages 341–390. Chapman & Hall, August 2010.

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[14] R. Dingledine, N. Mathewson, S. Murdock, and P. Syverson. Tor: The second-generation onion router (2014 draft v1), 2014.

[15] M. Edman and P. F. Syverson. AS-awareness in Tor path selection. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, pages 380–389. ACM, November 2009.

[16] M. Edman and B. Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys*, 42(1):1–35, 2009.

[17] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2012)*. ACM, October 2012.

[18] N. Feamster and R. Dingledine. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, October 2004.

[19] A. Hamel, J.-C. Grégoire, and I. Goldberg. The misentropists: New approaches to measures in tor. Technical report, Technical Report 2011-18, Cheriton School of Computer Science, University of Waterloo, 2011.

[20] R. Jansen and N. Hopper. 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, February 2012.

[21] A. Johnson and P. Syverson. More anonymous onion routing through trust. In *2009 22nd IEEE Computer Security Foundations Symposium*, pages 3–12. IEEE, 2009.

[22] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 20th ACM conference on Computer and Communications Security (CCS 2013)*, November 2013.

[23] J. Juen, A. Johnson, A. Das, N. Borisov, and M. Caesar. Defending tor from network adversaries: A case study of network path prediction. *Proceedings on Privacy Enhancing Technologies*, 2015(2):171–187, 2015.

[24] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix-based systems. In A. Juels, editor, *Proceedings of Financial Cryptography (FC '04)*, pages 251–265. Springer-Verlag, LNCS 3110, February 2004.

[25] J. L. Massey. Guessing and entropy. In *In Proceedings of the 1994 IEEE International Symposium on Information Theory*, page 204, 1994.

[26] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011.

[27] S. J. Murdoch and R. N. M. Watson. Metrics for security and performance in low-latency anonymity networks. In N. Borisov and I. Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 115–132. Springer, July 2008.

[28] S. J. Murdoch and P. Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In N. Borisov and P. Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*. Springer, June 2007.

[29] G. O'Gorman and S. Blott. Improving stream correlation attacks on anonymous networks. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 2024–2028. ACM, 2009.

[30] A. Pfitzmann and M. Hansen. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, July 2000.

[31] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[32] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003*, October 2003.

[33] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[34] M. Sherr, M. Blaze, and B. T. Loo. Scalable link-based relay selection for anonymous routing. In I. Goldberg and M. J. Atallah, editors, *Proceedings of Privacy Enhancing Technologies, 9th International Symposium (PETS 2009)*, volume 5672 of *Lecture Notes in Computer Science*, pages 73–93. Springer, August 2009.

[35] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of ESORICS 2006*, September 2006.

[36] R. Snader and N. Borisov. A tune-up for Tor: Improving security and performance in the Tor network. In *Proceedings of the Network and Distributed Security Symposium - NDSS '08*. Internet Society, February 2008.

[37] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

[38] P. Syverson. Why i'm not an entropist. In *In Seventeenth International Workshop on Security Protocols. Springer-Verlag, LNCS, 2009. Forthcoming*.

[39] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009,

July 2000.

[40] C. Wacek, H. Tan, K. Bauer, and M. Sherr. An Empirical Evaluation of Relay Selection in Tor. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'13*. Internet Society, February 2013.

[41] T. Wang, K. Bauer, C. Forero, and I. Goldberg. Congestion-aware Path Selection for Tor. In *Proceedings of Financial Cryptography and Data Security (FC'12)*, February 2012.

[42] M. Wright, M. Adler, B. N. Levine, and C. Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004.

A Security Models and Metrics

A.1 Guessing entropy - example

Suppose we have a Tor network with 3 guard nodes and 2 exit nodes. From the selection process, suppose we have the following matrix with $p_{i,j}$ the probability to selection guard i with exit j

$$P = \begin{pmatrix} 1/6 & 1/18 \\ 5/18 & 1/3 \\ 1/24 & 1/8 \end{pmatrix}$$

Let $G = \emptyset$ the set of prioritized guards and $E = \emptyset$ the set of prioritized exits. q is computed in the following way:

– $q_1 = 0$
– $q_2 = max(P) = 1/3, \{g_2\} \cup G, \{e_2\} \cup E$
– $q_3 = \text{max\_marg\_prob(P)}$
  $= max(Pr_{G=x}, Pr_{E=y})$
  $= max[max(\sum_{y \in E} Pr(G = x, E = y) \forall x \notin G),$
  $\qquad max(\sum_{x \in G} Pr(G = x, E = y) \forall y \notin E)]$
  $= max(1/8, 5/18) = 5/18, \{e_1\} \cup E$
– $q_4 = \text{max\_marg\_prob(P)}$
  $= max(1/6 + 1/18, \emptyset) = 2/9, \{g_1\} \cup G$

– $q_5 = 1/24 + 1/8, \{g_3\} \cup G$

Then:
$$\sum_{i=1}^{N+K} i.q_i = 3.22$$

B Waterfilling Bandwidth-Weights

B.1 Waterfilling on the guard+exit flagged nodes

Waterfilling can also be applied on Guard+Exit flagged nodes when a part of the bandwidth of this pool has to be given to the middle one. In the time period we took to evaluate Waterfilling, it happened when the network load case was 3bE=S.

Suppose there are K nodes with the guard+exit flag, and let BW_i be the bandwidth of the i-th guard+exit flagged node with the highest bandwidth. Then we had the following constraint on the top of classical bandwidth-weights:

$$Wd_i * BW_i = Wd_{i+1} * BW_{i+1} \ \forall i \in (0, N) \qquad (23)$$

$$0 \le Wd_i \le 1 \ \forall i \in (0, K) \qquad (24)$$
$$Wd_i = 1 \ \forall i \in (N+1, K) \qquad (25)$$
$$\sum_{i=0}^{K} Wd_i BW_i = (Wgd + Wed) * D \qquad (26)$$

Solving with these constraints gives NWd_i, then we can compute:
$$Wmd_i = 1 - Wd_i$$
$$Wgd_i = Wd_i * \frac{Wgd}{Wgd + Wed}$$
$$Wed_i = Wd_i * \frac{Wed}{Wgd + Wed}$$