UCLouvain

epl

École polytechnique de Louvain

Improving latency in Tor network using client location-aware path selection algorithm (CLAPS)

Author: **Maxime Wattiaux**
Supervisors: **Olivier Pereira, Jean-Charles Delvenne**
Reader: **Florentin Rochet**
Academic year 2022–2023
Master [120] in Electrical Engineering

Everybody should learn to program a computer, because it teaches you how to think

Steve Jobs. 2005.

Abstract

Tor is currently the most widely used software for anonymous access to the Internet, but a lot of research is being carried out to improve the conditions of this anonymous browsing. Indeed, Tor enables its users to be masked (IP level) at the price of a slower connection than using a traditional browser. Various studies have shown that Tor is approximately five times slower than the regular network and several experiments have already been conducted to improve latency in the network. Recently, a client location-aware path selection algorithm (more briefly called CLAPS) was designed to be applied to any version of Tor with the goal of improving its performance. This work demonstrates a comprehensive methodology for applying CLAPS to the current release of Tor and especially the positive impacts of such an application in terms of latency.

Acknowledgements

First of all, I would like to thank Florentin Rochet, the assistant who supervised me during this thesis. His valuable advice and increased knowledge of the Tor network were very helpful in overcoming some of the technical and conceptual shortcomings I encountered.

I also thank my two supervisors, Pr. Olivier Pereira and Pr. Jean-Charles Delvenne for their patience when I had understanding difficulties and their valuable advice through questions and leads during our meetings.

Finally, I would like to thank my parents for their support and encouragement during the (admittedly long) duration of this master thesis. They supported me even when I had lost my motivation and I sincerely thank them for that.

Contents

Glossary

ABWRS Adjusted Bandwidth Weight Random Selection.

API Application Programming Interface.

AS Autonomous System.

ASN Autonomous System Number.

BGP Border Gateway Protocol.

BWRS Bandwidth Weight Random Selection.

CDF Cumulative Distributive Function.

CLAPS Client Location-Aware Path Selection.

IANA Internet Assigned Numbers Authority.

IP Internet Protocol.

LP1 Linear Program 1.

LP2 Linear Program 2.

NCC Network Coordination Center.

PDF Probability Density Function.

RIR Regional Internet Registry.

RTT Round-Trip Time.

SRS Simple Random Selection.

SSH Secure Shell.

TCP Transmission Control Protocol.

TOR The Onion Router.

USA United States of America.

VM Virtual Machine.

VPN Virtual Private Network.

List of Figures

.

Introduction

Nowadays, it can sometimes be useful to obtain anonymity when browsing the web. For instance, it avoids being monitored by malicious individuals or businesses that analyze traffic. It also allows us to circumvent some forms of control by accessing blocked websites from a specific location[9], such as in China, where social networks are prohibited. We are even able to counteract a kind of censorship, like recently in Russia, where some digital rights activists have managed to share information about the war using the Tor network despite Russian measures to block it.

Many ways are available to hide our identity on the net, so we start by looking at some of them and come to a conclusion : anonymity and latency on the network are linked. If you want to maximize anonymity, latency will be negatively impacted, the reverse is also true. This duality is discussed briefly in the next section.

Secondly, we focus on the Tor network itself, its origins and how it works. As we see in first section, consequence of anonymity using Tor is that its latency is much worse than accessing the Internet through a traditional browser. This latency is what we try to optimize in this master thesis.

Thus, we present a general framework called CLAPS developed a few years ago by some researchers from *U.S. Naval Research Laboratory* (Ryan Wails, Aaron Johnson), *Princeton University* (Prateek Mittal) and *UCLouvain* (Florentin Rochet and Olivier Pereira, two of my supervisors during this work). This algorithm considers the location of customers in the network and thus potentially improves the overall latency.

Next, we show a complete methodology for the application of the CLAPS algorithm to the current version of Tor. For this purpose, we talk about the data we need and how we can collect, calculate and estimate these data. The focus is on performance to obtain these results, considering the need to run this algorithm several times as the topology of the Tor network evolves continuously.

Then, we present the tests we performed and under which conditions and assumptions. This allows us to evaluate our methodology and see if it makes sense or not. Finally, the analysis of the collected results permits us to visualize the impact on latency in the Tor network. Before concluding this master thesis, we suggest some improvements for future work.

1 The latency-anonymity duality

As mentioned in the introduction, anonymity has become necessary nowadays when performing daily tasks such as browsing the web. We observe several ways to be anonymous on the Internet : the basic principle is to mask our IP address. Indeed, the IP address is like our online identity card and it is the one that can betray us.

Among the means implemented to hide an identity, one can note the use of proxies where the user connects to a web server through which the different requests are sent. Then, it is the address of this server that is displayed instead of ours, only the server being able to see it. Making requests through a VPN may also be recommended in order to access private networks from a public network. In this case, the displayed address is that of the VPN provider. Besides hiding our IP, these ways let us hide our location[20].

We may also use systems implemented specifically for anonymization for which we distinguish two types. On the one hand, systems that want to give priority to anonymity as much as possible at the expense of high latency. On the other hand, systems that wish to provide acceptable latency while guaranteeing a minimum of anonymity. These systems are known as high and low latency systems, respectively.

High latency systems are not suitable for any kind of application. A video chat system, for instance, would not be viable because the transmission time would be way too long. However, a mail-type application is entirely appropriate since it can be tolerated that messages take a while to be received in this case. Examples of such systems include MixMinion[6] and Babel[13].

At the opposite, we have the so-called low-latency systems, which can be used in interactive applications such as SSH because of their lower latency. Most of these systems are based on proxies with a system of intermediary nodes. Messages are typically sent through a circuit composed of a series of nodes (usually operated by volunteers) within the network. As each node knows only the identity of its predecessor and successor, anonymity is ensured, although there are currently attacks aimed at constraining such systems, such as the end-to-end correlation attack (in which an opponent tries to analyze incoming and outgoing packets in the network and tries to find templates for depersonalizing the user, instead of deciphering the messages themselves). Examples of systems using such nodes principle are MorphMix[25], Crowds[24], Tarzan[10], and TOR[8]. The Tor low-latency system especially uses onion routing[23] from which it derives its name. We present it in the following section now that we have seen the existence of the duality between anonymity and latency.

2 The Tor network : state of the art

In this section, we describe the Tor system in more detail by examining how the network is composed and how it works. We also present the actual algorithm behind Tor's ability to select the circuits it will use to connect to the Internet. Lastly, we present some of the most frequent types of attacks on the network.

2.1 Tor : The Onion Router

Tor is a low-latency anonymization system which enables you to access the Internet while keeping your identity hidden. It is a decentralized network which currently consists of about 7000 nodes called relays and has approximately 2 million of users per day [33]. It is the most popular system of its kind.

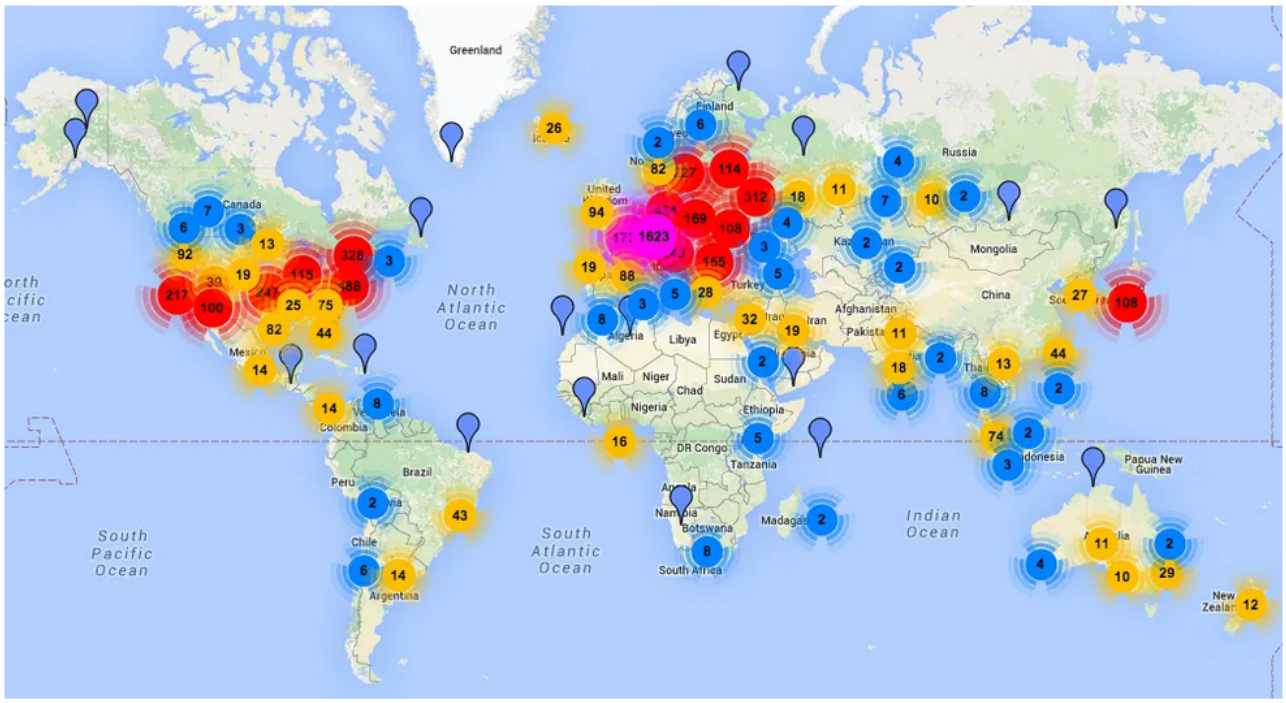

Figure 1: World map snapshot containing all Tor relays[12]

Tor uses a simple TCP layer through an onion routing policy. This type of routing allows TCP traffic to transit through several nodes from the network in order to counteract traffic analysis on the network (e.g. man-in-the-middle attacks). Onion routing makes it possible for Tor to guarantee the anonymity of its users.

2.2 The onion routing

At first glance, onion routing may seem particular. The principle is that when a client wishes to connect to a given destination, Tor creates a circuit composed of three relays : an input relay (guard node), a transient relay (middle node) and an exit relay (exit node) (See Fig. 2). The same circuit is used upstream and downstream. On top of that, messages are encrypted three times (one time for each node). Message is then sent through the circuit where it will be decrypted once every hop like peeling an "onion" (see next section).

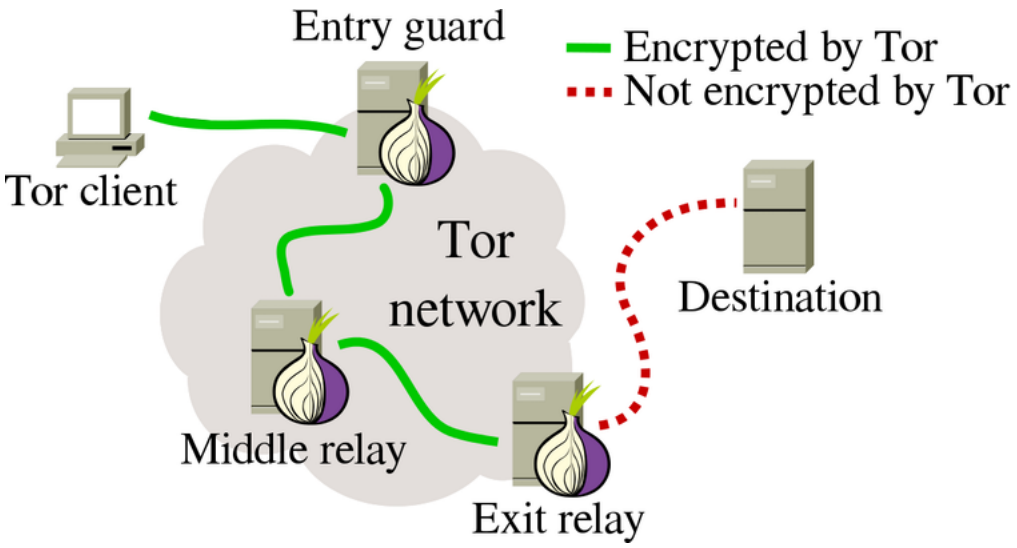

Figure 2: Tor onion routing circuit scheme

Each of the nodes involved in the circuit is only aware of the identity of its predecessor and of its successor. This is why the receiver believes he is communicating with the exit node and not with the client. The client is thus masked, only the guard node knows the identity of the client.

2.3 Circuit construction

To create a circuit, the Tor network must first retrieve a list of all available relays. This can be done through special relays called Directory Authorities which are nodes whose purpose is to maintain an up-to-date list of relays in the network. These Directory Authorities are currently 10 and publish an hourly file called Consensus that contains information on all relays (bandwidth, flags, weight of consensus).

It is important to note that nodes are not necessarily acceptable for each position. For instance, some relays can be used as a guard node but not as an exit node. This is partly because relays must respect some conditions to be admissible in some positions (for example, a minimal bandwidth is required for a guard node). The positions where each relay is eligible are stored in the so-called relay flags. These flags are then saved in a consensus file, allowing Tor to know which relay it can select. Regarding this selection, a weight is assigned to each node for each possible position: guard, middle, node. A zero weight is assigned to

an inadmissible position. The greater the weight, the greater the probability of selecting the corresponding relay. Tor therefore proceeds with a weighted random choice and selects a relay for each position. Once again, the Directory Authorities publish the weights computed in the hourly file.

The construction of the circuit after having chosen the relays is done in such a way that a secret key encrypted with a public key dedicated to each node of the circuit is distributed. The key is specific to the node and it only knows its predecessor and successor.

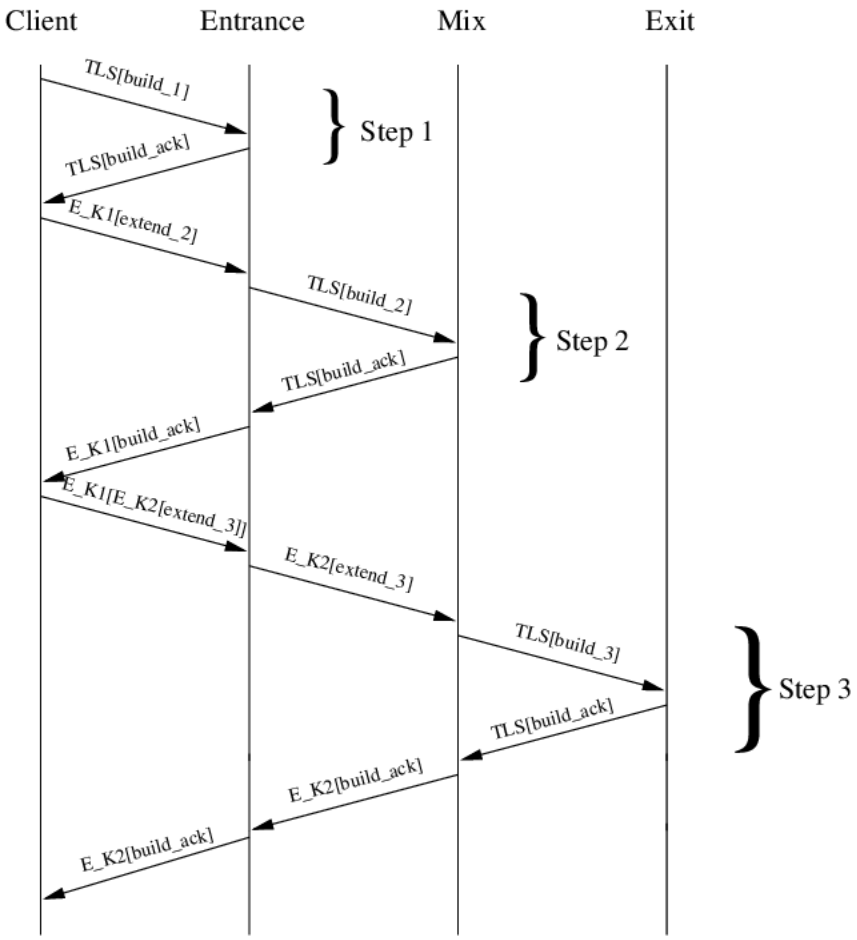

Figure 3: Establishement of a Tor circuit using handshake

In the end, the client encrypts his packet several times in order to end up with this layer configuration, hence the name onion routing. We can see in Fig. 4 that the message is encrypted first with key C (exit node), then with key B (middle node) and finally with key A (guard node). When the packet is finally sent, the guard node receives it, decrypts it and sends it to the middle node which decrypts this message and sends it to the exit node which performs the same operation to finally deserve the message to the destination.

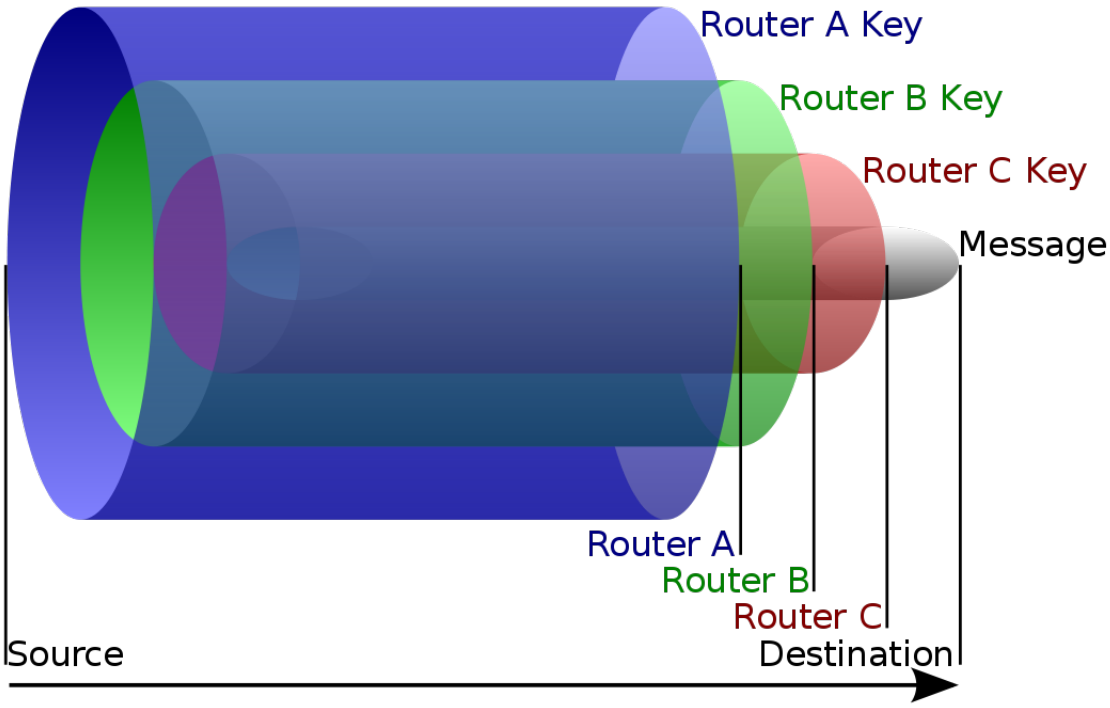

Figure 4: Onion message ciphered with 3 layers corresponding to the three nodes invloved in circuit

2.4 Path selection

2.4.1 Original path selection

Originally, Tor just used a Simple Random Selection (SRS) [30]. A circuit was then only made up of two nodes (middle and exit node) which were chosen completely randomly. This way of proceeding guaranteed a high level of anonymity but had very poor performance because the nodes in the Tor network have bandwidths that are very different and there is therefore no balanced network. Some nodes were overloaded and others had no traffic

2.4.2 Second attempt : Bandwidth Weight Random Selection

The second algorithm used was the Bandwidth Weight Random Selection (BWRS) where the relays were chosen with a probability proportional to their bandwidth. This again caused network balancing problems between nodes.

2.5 Actual algorithm : Adjusted Bandwidth Weight Random Selection

We now use the Adjusted Bandwidth Weight Random Selection (ABWRS) which balances the bandwidths available at each node between the different possible positions (guard, middle, exit) in order to maximize throughput. This can be done by solving the following system [7]:

$$\begin{cases} W_{gg} * G + W_{gd} * D & = M + W_{md} * D + W_{me} * E + W_{mg} * G \\ W_{gg} * G + W_{gd} * D & = W_{ee} * E + W_{ed} * D \\ W_{ed} * D + W_{md} * D + W_{gd} * D & = D \\ W_{mg} * G + W_{gg} * G & = G \\ W_{me} * E + W_{ee} * E & = E \end{cases}$$

with :

- **G** be the total bandwidth for Guard-flagged nodes.
- **M** be the total bandwidth for non-flagged nodes.
- **E** be the total bandwidth for Exit-flagged nodes.
- **D** be the total bandwidth for Guard+Exit-flagged nodes.
- **T** = G + M + E + D

and :

- W_{gd} be the weight for choosing a Guard+Exit for the guard position
- W_{md} be the weight for choosing a Guard+Exit for the middle position
- W_{ed} be the weight for choosing a Guard+Exit for the exit position
- W_{me} be the weight for choosing an Exit for the middle position
- W_{mg} be the weight for choosing a Guard for the middle position
- W_{gg} be the weight for choosing a Guard for the guard position
- W_{ee} be the weight for choosing an Exit for the exit position

The two first equations imply that guard bandwidth = middle bandwidth = exit bandwidth. We can see it as a bandwidth pipeline (see Fig. 5), the middle nodes must be ready to receive all the bandwidth from the guard nodes and the exit nodes must be ready to receive all the bandwidth from the exit nodes.

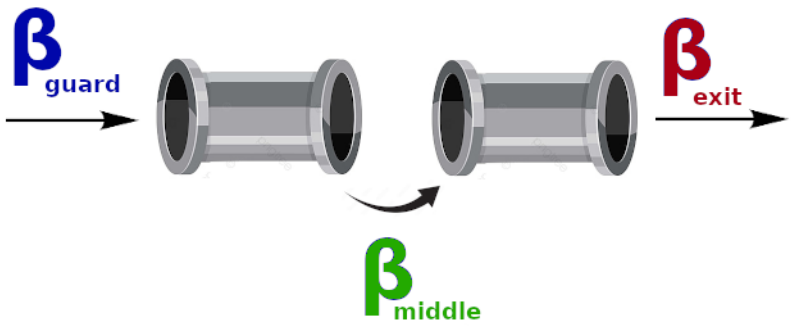

Figure 5: Guard bandwidth = Middle bandwidth = Exit bandwidth

The three last equations verify that weights are consistent. But we must have two more equations to be able to solve the previous system. Those two constraints depend from the state of network load.

CASE 1 : $E \geq \frac{T}{3}$ **and** $G \geq \frac{T}{3}$

In this case, nor exit nor guard is scarce. The two additional constraints are :

$$W_{mg} = W_{md}$$
$$W_{ed} = \frac{1}{3}$$

CASE 2 : $E < \frac{T}{3}$ **and** $G < \frac{T}{3}$

Let :

- R denote the more scarce class between G and E
- S denote the less scarce class

SUBCASE A : $R + D < S$

In this case, we devote all D bandwidth to S. So we have :

$$W_{gg} = W_{ee} = 1$$
$$W_{mg} = W_{me} = W_{md} = 0$$

If $E < G$:

$$W_{ed} = 1$$
$$W_{gd} = 0$$

else :

$$W_{ed} = 0$$
$$W_{gd} = 1$$

SUBCASE B : $R + D \geq S$

In this subcase, if $M \leq \frac{T}{3}$, we have enough bandwidth to try to achieve a balancing condition.

We maximize bandwidth in guard position while still allowing exits to be used as middle nodes by adding constraints :

$$W_{gg} = 1$$
$$W_{md} = W_{gd}$$

CASE 3 : $E < \frac{T}{3}$ **or** $G < \frac{T}{3}$

Let :

- S denote the less scarce class between G and E

SUBCASE A : $S + D < \frac{T}{3}$

If S=G, we have (if E is more scarce than M, keep its bandwidth in place) :

$$\begin{aligned} W_{gg} = W_{gd} &= 1 \\ W_{md} = W_{ed} = W_{mg} &= 0 \\ \text{If}(E < M) : W_{me} &= 0 \\ \text{else} : W_{me} &= \frac{G - M}{2G} \\ W_{ee} &= 1 - W_{me} \end{aligned}$$

If S=E, we have (if G is more scarce than M, keep its bandwidth in place) :

$$\begin{aligned} W_{ee} = W_{ed} &= 1 \\ W_{md} = W_{gd} = W_{me} &= 0 \\ \text{If}(G < M) : W_{mg} &= 0 \\ \text{else} : W_{mg} &= \frac{G - M}{2G} \\ W_{gg} &= 1 - W_{mg} \end{aligned}$$

SUBCASE B : $S + D \geq \frac{T}{3}$

If $S = G$, we add the following constraints to maximize bandwidth in the guard position while still allowing exits to be used as middle nodes :

$$W_{gg} = 1$$
$$W_{md} = W_{ed}$$

This system of equations with the additional constraints is solved and voted on every hour by the Directory Authorities and then published in the corresponding consensus file. Tor then chooses its relays during the circuit creation according to the weights present in this consensus.

Note that for security reasons Tor adds the following constraints :

- Tor cannot select two times the same node in a circuit
- Exit policy of last node must authorize to connect to the destination (IP/port)
- Tor cannot select two nodes with addresses from the same /16 prefix subnet in a circuit

2.6 Known attacks

We are aware of the existence of attacks in the Tor network, we cite some of them for the state of the art but we do not focus on security within this work.

The most well-known attack in the Tor network is the end-to-end attack where an adversary can attempt to de-anonymize a user by analyzing the traffic. To do this, the adversary will try to analyze incoming and outgoing data packets from the network and perform time and volume correlations. It can achieve this by controlling a number of nodes in the network, for example.

A recent study has shown that an adversary could also perform a kind of end-to-end correlation attack using autonomous systems by trying to increase the traffic through this AS thanks to its BGP routing policy and then perform the same traffic analyzes to de-anonymize the client. Proposals have been made to counter this type of attack, notably Counter-Raptor[29].

The guard placement attack [34], where an adversary will operate a relay guard in a location with the aim of making certain clients transit more than others within this relay. These customers therefore see their anonymity threatened.

3 CLAPS : Client location-aware path algorithm

The CLAPS[27] algorithm essentially makes possible to change the way in which the weights are calculated by the Directory Authorities. It makes it possible to take into account the client locations in the computation of weights. This helps improve latency performance as demonstrated in the next results. Indeed, taking into account the location of the customers in the weights allows Tor to choose a sequence of relays which are not located on either side of the globe (imagine for example a circuit passing through the USA then through Russia before to return to the USA, CLAPS tends to avoid this kind of scenario)

The CLAPS algorithm allow us to express weight calculations using an optimization problem and to solve it with two linear programs. First, we must ensure that the network is load-balanced as the actual Vanilla Tor do. For that purpose, we use the initial linear program called LP1 which allow us to compute the minimum bandwidth across all positions

Let :

- $\mathcal{P} = \{g = guard, m = middle, e = exit\}$ be the set of circuit positions
- $\mathcal{R}$ the set of all relays and $\mathcal{R}_p$ the subset of relays that can be used in position p $\in \mathcal{P}$
- B_r the bandwidth of relay r $\in \mathcal{R}$

The linear program LP1 can be formulated as :

$$
\begin{array}{ll}
\textbf{Maximize} & b \\
\textbf{Subject to} & \\
\forall r \in \mathcal{R} : & \sum_{p \in \mathcal{P}} w_r^p \leq B_r \\
\forall p \in \mathcal{P} : & \sum_{r \in \mathcal{R}} w_r^p \geq b \\
\forall p \in \mathcal{P}, r \in \mathcal{R} : & w_r^p \geq 0 \\
\forall p \in \mathcal{P}, r \in \mathcal{R} \backslash \mathcal{R}_p : & w_r^p = 0
\end{array}
$$

Figure 6: Linear program LP1

with the constraints meaning :

- First constraint limit total weight of each relay to its bandwidth
- Second constraint allocate a minimum of b weight for each position
- Third constraint says that weights cannot be negative
- Fourth constraint set weight to a null value when the corresponding relay is inadmissible in this position

Then, LP1 outputs the minimum bandwidth β that can be provided in all p $\in \mathcal{P}$ simultaneously (β takes the output value of b). Weights computed in this linear program are just used to obtain this value of β and then are not used by clients after that.

Note that solving this optimization problem is the same as solving the linear equations of Section 1.5 which also aimed to load-balance the network through the three positions. We will therefore prefer this second technique which is easier to implement.

Next, a linear program LP2 is solved to determine the weights in position $\pi_i \in \mathcal{P} = \{g = guard, m = middle, e = exit\}$. It outputs a weight $w_{lr}^{\pi_i}$ for each $l \in \mathcal{L}_{\pi_i}$ and $r \in \mathcal{R}$ ($\mathcal{L}_{\pi_i}$ is called a positional location and initial location is just $\mathcal{L}_{\pi_1} = \mathcal{L}$), which is the weight used by a client in location l for relay r. A instance of LP2 is solved for each circuit position in $\mathcal{P}$ in the order they appear in π. Here is this LP2 :

Minimize	$\sum_{l \in \mathcal{L}_{\pi_i}} \sum_{r \in \mathcal{R}_{\pi_i}} D_l^{\pi_i} P_{lr}^{\pi_i} w_{lr}^{\pi_i}$
Subject to	
$\forall j \geq i, l \in \mathcal{L}_{\pi_i}$:	$\sum_{r \in \mathcal{R}} w_{lr}^{\pi_j} = \beta$
$\forall j \geq i, l \in \mathcal{L}_{\pi_i}, r \in \mathcal{R}$:	$\frac{w_{lr}^{\pi_j}}{\beta} \leq \frac{\theta_{\pi_j} v_r^{\pi_j}}{\sum_{s \in \mathcal{R}} v_s^{\pi_j}}$
$\forall l \in \mathcal{L}_{\pi_1}$:	$\sum_{l' \in \mathcal{L}_{\pi_i}} \lambda_{ll'} \sum_{r \in \mathcal{R}} \frac{w_{l'r}^{\pi_i} P_{l'r}^{\pi_i}}{\beta} \leq V_l^{\pi_i}$
$\forall r \in \mathcal{R}$:	$\sum_{j<i} \omega_r^{\pi_j} + \sum_{j \geq i} \sum_{l \in \mathcal{L}_{\pi_i}} \delta_l^{\pi_i} w_{lr}^{\pi_j} \leq B_r$
$\forall j \geq i, l \in \mathcal{L}_{\pi_i}, r \in \mathcal{R}$:	$w_{lr}^{\pi_j} \geq 0$
$\forall j \geq i, r \in \mathcal{R} \backslash \mathcal{R}_{\pi_j}$:	$w_r^p = 0$

Figure 7: Linear program LP2

with the constraints meaning :

- First constraint set weight in each position to the value β obtained from LP1
- Second constraint limit relay-placement advantage to θ[34]
- Third constraint limite average penalty per location to penalty of Vanilla Tor
- Fourth constraint limit total weight of each relay to its bandwidth
- Fifth constraint constraint says that weights cannot be negative
- Sixth constraint set weight to a null value when the corresponding relay is inadmissible in this position

We also define :

- $D_l \geq 0$ the client density in location $l \in \mathcal{L}$ with $\sum_l D_l = 1$
- $\theta_p \geq 1$ the maximum factor by which any relay's selection probability in position p can increase over Vanilla tor.
- $P_{lr}^{\pi_i}$ the penalty for choosing relay r in position π_i from $l \in \mathcal{L}_{\pi_i}$.
- $\Lambda(l, r) : \mathcal{L}_{\pi_i} \times \mathcal{R} \to \mathcal{L}_{\pi_{i+1}}$ the map from the current positional location l to the next one after choosing r for the ith position.
- $\lambda_{ll'} = \sum_{l'' \in \mathcal{L}_{\pi_{i-1}}} \lambda_{ll''} \sum_{r \in \mathcal{R}: \Lambda(l'', r) = l'} \frac{w_{l''r}^{\pi_i}}{\beta}$ the probability that a client in $l \in \mathcal{L}_{\pi_j}$ chooses relays for the jth to (i-1)st positions that yield positional location l' $\in \mathcal{L}_{\pi_i}$ (for i=j, $\lambda_{ll'} = 1$ if $l = l'$ and $\lambda_{ll'} = 0$ otherwise).
- v_r^p the Vanilla weight for relay r in position p
- $V_l^{\pi_i}$ the expected penalty in π_i from positional location $l \in \mathcal{L}_{\pi_j}, j \leq i$, when using the vanilla weights to choose the jth to ith positions in π order.
- $\delta_l^{\pi_i} = \sum_{l' \in \mathcal{L}_{\pi_{i-1}}} \sum_{r \in \mathcal{R}: \Lambda(l', r) = l} \frac{\lambda_{l'}^{\pi_{i-1}} w_{l'r}^{\pi_{i-1}}}{\beta}$ the client density of positional location $l \in \mathcal{L}_{\pi_i}$ (given weights $w_{lr}^{\pi_{i-1}}$ for the (i-1)st position obtained from LP2.
- $\omega_r^{\pi_i} = \sum_{l \in \mathcal{L}_{\pi_i}} \delta_l^{\pi_i} w_{lr}^{\pi_i}$ the sum of the weights of relay r in position π_i weighted by the positional location densities.

So we see that it is our LP2 that will allow us to take into account our rental customers in the calculation of weights. Indeed, this linear program tends to minimize an objective function which represents an average penalty per location. It is up to us to choose which element to use as a penalty.

The reading of these equations may seem surprising at first sight for the reader, a development has been carried out for a small network of 5 relays and are available in Appendix A if necessary.

4 Methodology

In order to derive the weights needed to build circuits in the Tor network, we need to give some input data to our CLAPS algorithm. This section describes a complete methodology for gathering and computing those data. An analysis of the relevance and reliability of the collected data is also conducted.

We see in the previous section that our linear program take penalty values as input. The goal of this work is to improve the overall latency of the Tor network. It may therefore be appropriate to take as penalties an estimate/measure of the real latencies in the network. It is the main idea to be used in our methodology. Some works have already tried to improve the general latency of the Tor network using similar estimates. For instance, we cite LasTor[2] which took the inverse of distance between the different nodes in the network to compute new weights for relay selection. We also use this idea for comparative purposes.

Thus, we present a method to obtain CLAPS required inputs :

- All metrics needed to implement our algorithm containing the necessary data/statistics about the tor network and its relays
- An estimation of client density per location
- A clusterization of nodes in the network in order to reduce the complexity of weight computation
- A penalty database using latencies between each pair of nodes involved in the path selection algorithm (clients and relays)
- Vanilla expected penalties

4.1 Metrics

We have seen in the state-of-the-art that the Tor network has several relays (called directory authorities) with a special-purpose to maintain a list of running relays. They publish a consensus file every hour (compiled and voted by each of these relays to ensure that all customers have the same information on the network). We are able to retrieve these consensus files thanks to the CollecTor[5] website. This website periodically collects a set of data on the public Tor network. It is a really friendly-user service for who want to start doing research on the network.

CollecTor directly downloads consensus from the directory authorities and then fetches data from the different nodes present in the consensus before storing them in node descriptor and node extradescriptor files. A huge amount of data and statistics are available ranging from relay and bridge[35] information to network performance measurements obtained from TorPerf[31] or OnionPerf[21]. We only need relay and consensus information and thus we download following files :

- **Relay Server Descriptors** : containing information that relays publish about themselves and that are used by Tor clients in order to ensure their functioning (bandwidth, IP, ...)

- **Relay Extra-info Descriptors** : containing additional information that relays publish about themselves and that are not required for Tor clients to ensure their functioning (bytes reads and writes, number of requests,...)
- **Network Status Consensuses** : containing a list of all relays and the corresponding consensus weights for path selection in the hourly consensus. Some other information like flags, tor relays version, validity periods are contained in this file.

Thus, we use a python script to process (based on a script from TorPs the Tor path simulator[32]) all this information and keep only the ones we need later. We save all this in files that we call **Network State Files** and that contain, as the name indicates, the state of the network at a given time (one file per hour since we Directory Authorities publish a new consensus at this rate)

The resulting network state files are 3-level deep binary files containing the following information :

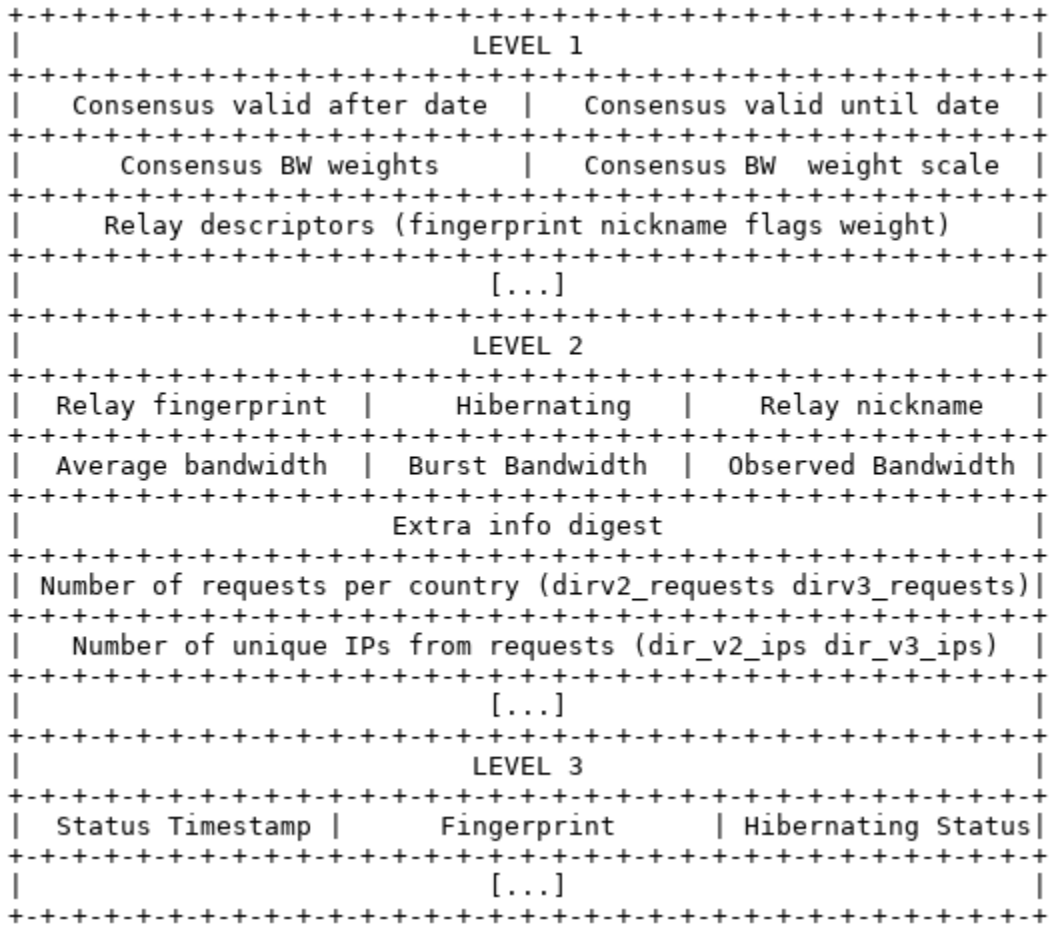

Figure 8: Network state file data architecture

The content of the different fields seems quite obvious for the most part according to their names. We provide more details as we use these data in the rest of our implementation. Extra details are also available for the reader in the *Server descriptor format* section of the Tor specifications[7].

4.2 Estimation of client density per location

Just as Vanilla Tor cannot load-balance users among relays without knowing the bandwidth of each relay, it is impossible to load-balance the network using our location-aware path selection algorithm without knowing not only the available bandwidth for each relay but also the density of clients per location. However, this density is difficult to get as it constantly varies and depends on the choice of locations we make. Furthermore, the live data stored by the relays are not sufficient to estimate them.

The idea used in our methodology is to consider as locations the different autonomous systems (ASes[4]) around the world. This avoids having to consider every IP address in the world as a potential customer in our linear program and thereby reduces the number of equations and the complexity of calculation. Once we have our set of locations, we need to figure out the density of customers for each of them. To do this, we take advantage of a metric present in the network state files obtained previously which tells us the number of unique IP addresses that made at least one request to an entry relay.

4.2.1 Set of client locations : Autonomous systems

In order to obtain the list of ASes around the world, we use RipeStats [26] the large scale data service of Ripe NCC which is one of the 5 RIR in the world. Their service contains a lot of data on ASes around the world, blocks of IP addresses and other statistics sorted by country for example. Their database seems like a good idea because it is filled with their own data that they collect and from other data from RIRs around the world. In a nutshell, it is the service we need for everything related to routing data.

First, we request to RipeStats the list of ASN of running ASes per country. Then, we ask some routing information service to determine which of these ASN correspond to transiting ASes (transiting ASes are just ASes that doesn't anounce origin for IPS and just acts as a router between two other ASes). We remove these ASes from our list of ASN because they don't represent any client. We also check whether there are any reserved ASes in our defined ASN set by checking in the list published by IANA[14].

In our remaining set of ASN, there are probably ASes that belong to hosting companies (hosting ASes). In order to eliminate some of these, one idea is to delete the ASes which correspond to the 1000 top websites in the Alexa ranking[3][1]. This ranking provides a list of the most visited sites worldwide. The high-ranking sites are all major names like Google, Facebook, YouTube, etc... One can easily suppose that the ASes corresponding to these sites are managed directly by the companies. Thus, no customers come from them and we delete these ASes from our ASN list.

[1]Since 15th December 2022, Alexa Top Rank API/Website has been retired by Amazon, some alternatives are Ahrefs[1] or Semrush[28] for example

Finally, after all these steps we still have about 80,000 ASes left. In order to reduce this number to have a reduced set of locations, we decide to remove the ASes having a lower number of IPs than a choosen value. To select this value, we test several powers of two (beginning at 2^{14} and we check which proportion of ASes we delete compared to the total number of ASes in the world. Indeed, by proceeding like this there are customers that we no longer consider in our computations. We make the assumption that there is a weak probability a user comes from one of these ASes.

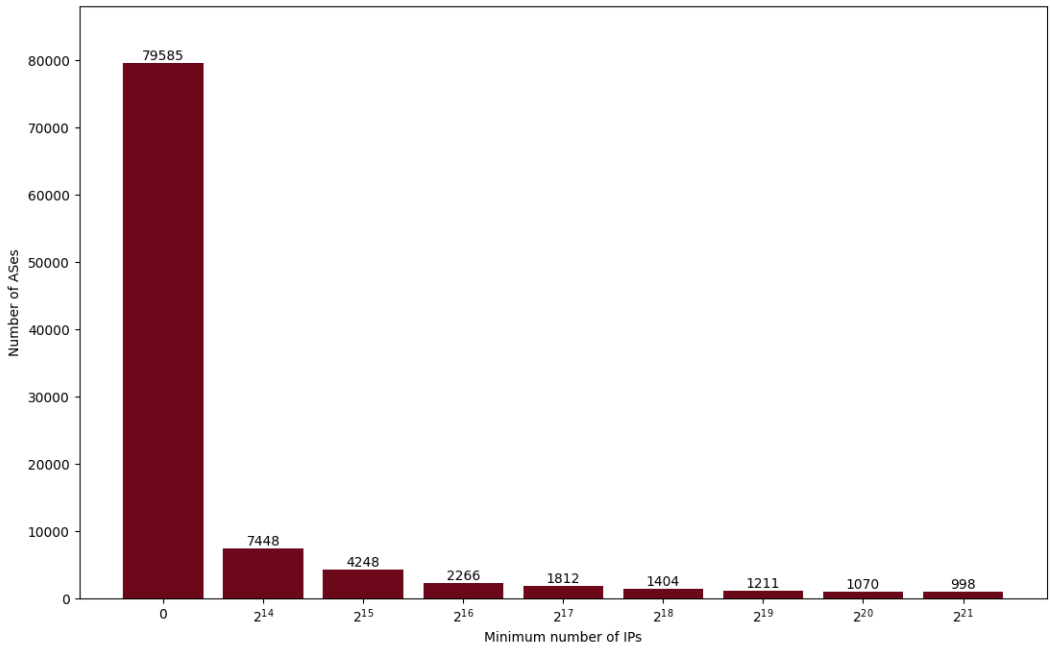

Figure 9: Remaining ASes

We see on Fig. 9 that we delete about 70000 ASes from 80000 if we consider only ASes with a minimum of $2^{14} = 16384$ IPs. This is interesting because it would allow us to consider less locations in our linear programs. But it is especially interesting to look at the proportion of IPs that have been removed compared to the total number of IPs in the world (and then of customers) to see if it is reasonable. Indeed, if we only have a small number of ASes left but besides that we have removed half of the IPs in the world (and thus customers), that no longer makes sense.

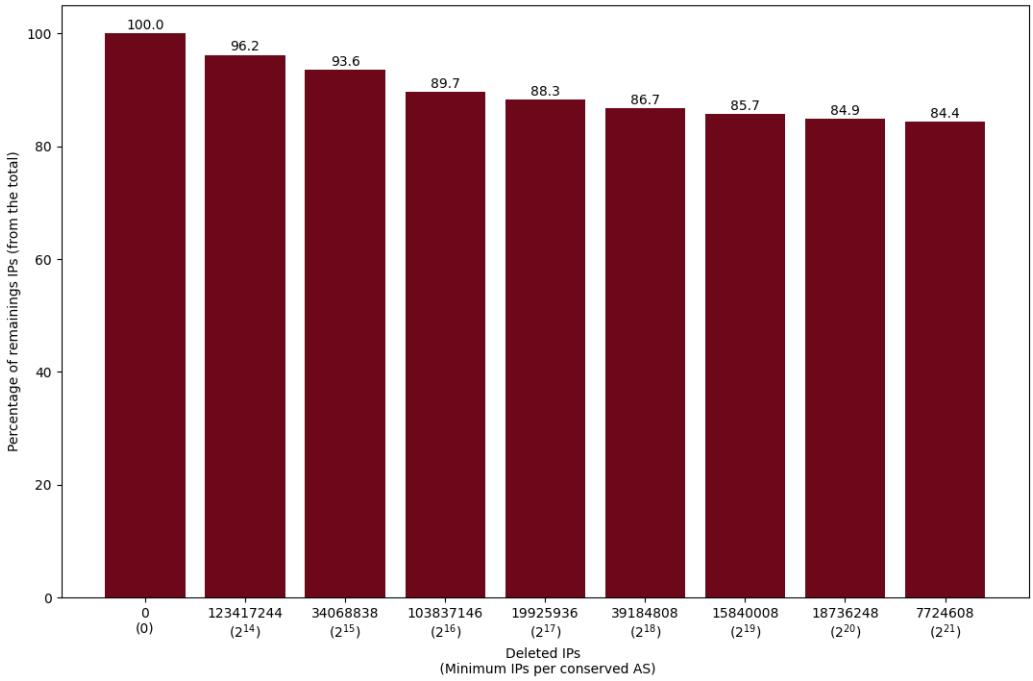

Figure 10: Percentage of remaining IPs (client)

In Fig. 10, we can see the percentage of ASes left after each step. We start to delete ASes with less than $2^{14} = 16384$ IPs and we increment in power of two. We still take 85% of IPs into account after removing all ASes with less than $2^{21} = 2097152$ IPs. It corresponds to 998 ASes and thus to the number of locations to consider in our calculations. We pick this value as it allows us to keep only a subset of locations and the percentage of remaining IPs appears reasonable.

4.2.2 Density of clients

Next, we have to get an estimate of the number of customers per location. We use a specific data stored in the network state files we generated using the metrics. Indeed, for each of the relays we have the following pair of data : (dir_v2_ips, dir_v3_ips). They actually correspond to the number of unique IPs that have made requests (rounded up to the nearest multiple of 8) through the relay to which they refer. Moreover, numbers of requests are sorted by country. We are now able compute the total percentage of IPs coming from a country.

First, we estimate total IPs per country as :

$$\mathrm{IPs}_{\mathrm{country}} = \sum_{\mathrm{i} \in \mathrm{Guard\ Relays}} \mathrm{IPs}_{\mathrm{i,country}}$$

We just take into account IPs requests registered by guard nodes as we are evaluating density of customers from all locations (ASes). Of course, a customer is not authorized to directly make a request to a middle or exit node as he has to transit by a guard node before.

Then, we find the percentage of IPs per country (equivalent to the probability a client comes from a country):

$$\mathrm{P(client\ comes\ from\ country)} = \mathrm{IPs}_{\mathrm{country}}[\%] = \frac{\mathrm{IPs}_{\mathrm{country}}}{\sum_{\mathrm{j} \in \mathrm{Countries}} \mathrm{IPs}_{\mathrm{j}}}$$

Unfortunately this is not enough because we don't know how many customers are issued from each location. However, we know that each IP is considered as a client and we have the number of IPs per AS location ($\mathrm{IPs}_{\mathrm{AS}}$). Therefore, we can compute the probability that a customer comes from a specific AS given that this customer come from a specific country :

$$\mathrm{P(client\ comes\ from\ AS \mid client\ comes\ from\ country)} = \mathrm{IPs}_{\mathrm{AS|country}}[\%] = \frac{\mathrm{IPs}_{\mathrm{AS}}}{\sum_{\mathrm{k} \in \mathrm{ASes\ from\ country}} \mathrm{IPs}_{\mathrm{k}}}$$

Finally, we can estimate the density of clients per location by using a joint distribution. Indeed :

$$\mathrm{P(client\ comes\ from\ AS)} = \mathrm{P(client\ comes\ from\ AS} \cap \mathrm{client\ comes\ from\ country)}$$

Using Bayes' Theorem :

$$\mathrm{P(client\ comes\ from\ AS)} = \mathrm{P(client\ from\ country)}*\mathrm{P(client\ from\ AS \mid client\ from\ country)}$$

$$\mathrm{P}(\text{client comes from AS}) = \mathrm{IPs}_{\text{country}}[\%] * \mathrm{IPs}_{\text{AS|country}}[\%]$$

Now we just have to multiply the previous results for each AS by the total number of clients in world (total number of IPs) and we get a distribution of clients per AS (at least a fair estimate). By normalizing distribution, we get desired density.

4.3 Clusterization of nodes

In order to reduce our number of nodes in our calculations and therefore also of locations, we will cluster the different nodes for each position (guard, middle, exit). To do this, they are clustered in two ways: according to the AS to which the relay belongs but also according to the IP prefix to which the relay belongs. We then consider the nodes of the same cluster as being a single node in the network, which will give us a single weight per cluster each time (each node into this cluster has the same weight). We randomly choose a relay in the cluster as the representative of the cluster.

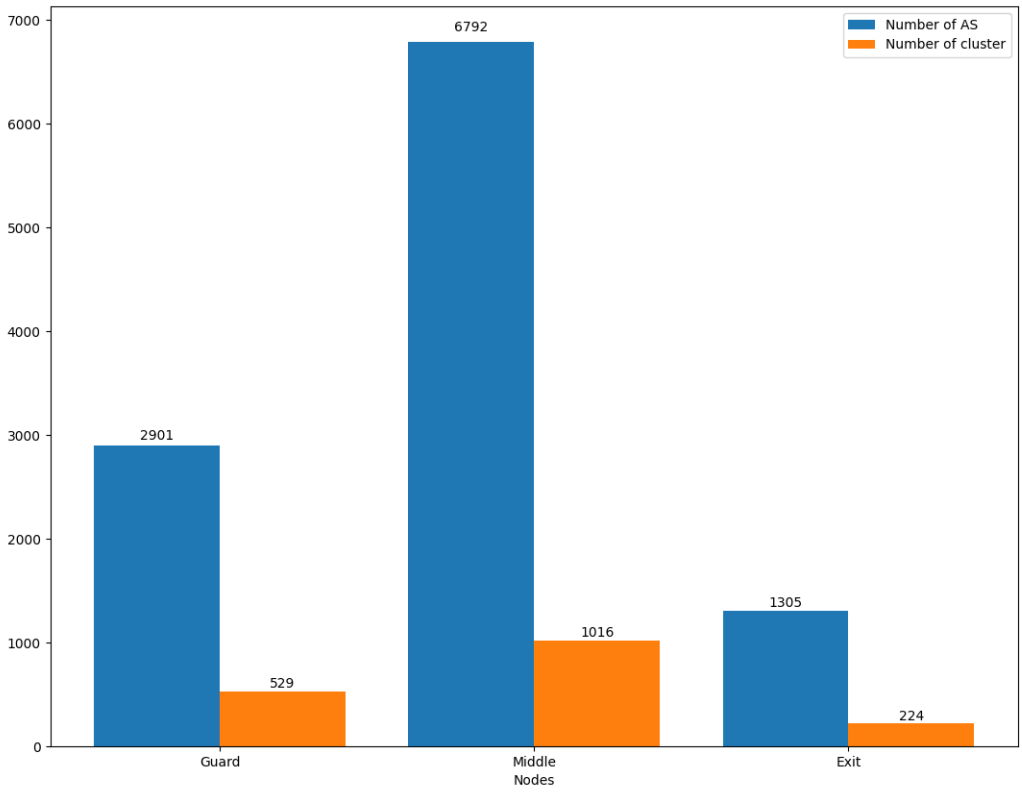

Figure 11: Nodes clustered by AS

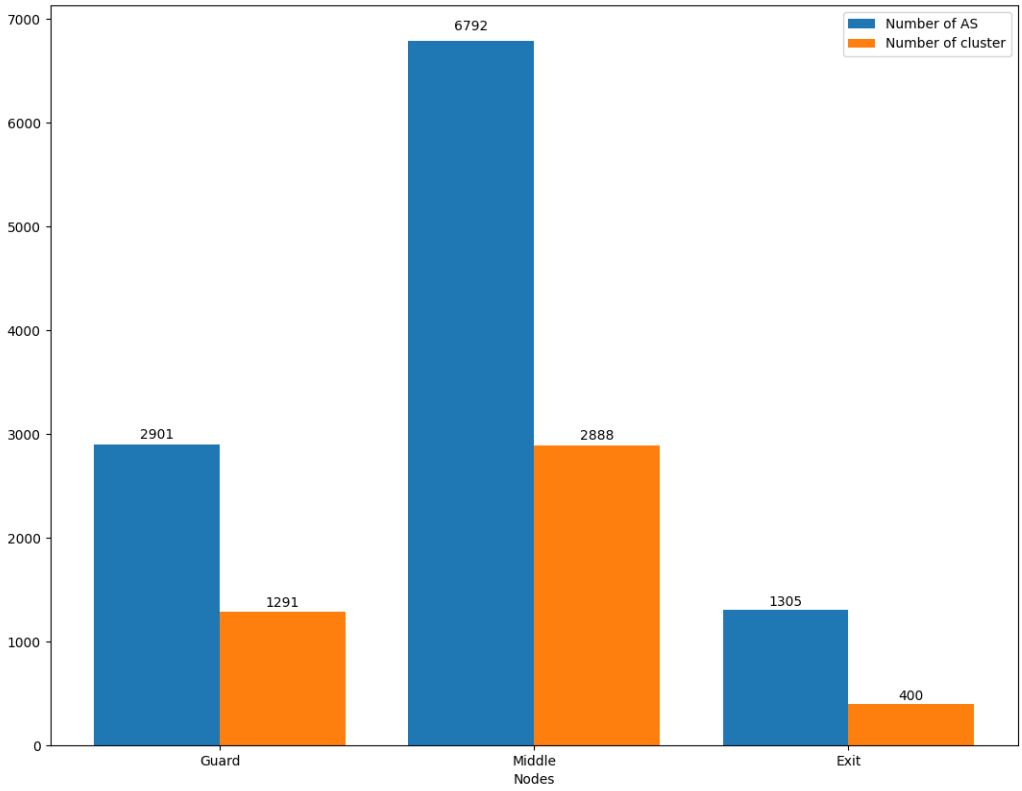

Figure 12: Nodes clustered by IP prefix

We observe in Fig. 11 and Fig. 12 as expected that clustering by prefix gives us more clusters than clustering by AS. However, clustering by prefix is more accurate in terms of location. For instance, if we take the Microsoft website domain and make an IP lookup, we see that there are more than one IP associated to the website and all of theses IP originates from the same AS. Then clustering by AS is less precise because clusters can contain IP blocks located far from each other. Obviously, when we cluster by prefix, we don't have such problem, all IPs are close to each other.

Microsoft.com IPs (AS 8075)	
20.53.203.50	Australia
20.81.111.85	United States of America
20.84.181.62	United States of America
20.103.85.33	Netherlands
20.112.52.29	United States of America

Table 1: Microsoft.com IP Lookup

4.4 Penalty database

As said above, we want to use as penalties an estimate of the latencies in the current Tor network. To do this, we will have to proceed in two steps. First, find an estimate of latencies between all clients and guard nodes. It seems it is a very complicated task because we don't have access to all the computers of the customers in the world. Then we don't know how to calculate the latency because we cannot create requests on the guard relays from client's computer. Then, we have to calculate latencies between all the pairs of nodes (guard, middle) and (middle, exit), which is a little bit easier since we can launch requests through the Tor network.

4.4.1 Client-Guard latencies

Given the infeasibility of actually measuring latencies because we do not have direct access to clients, we will use results obtained by Rob Jansen (a computer security research scientist from U.S. Naval Research Laboratory) in one of his previous publications.

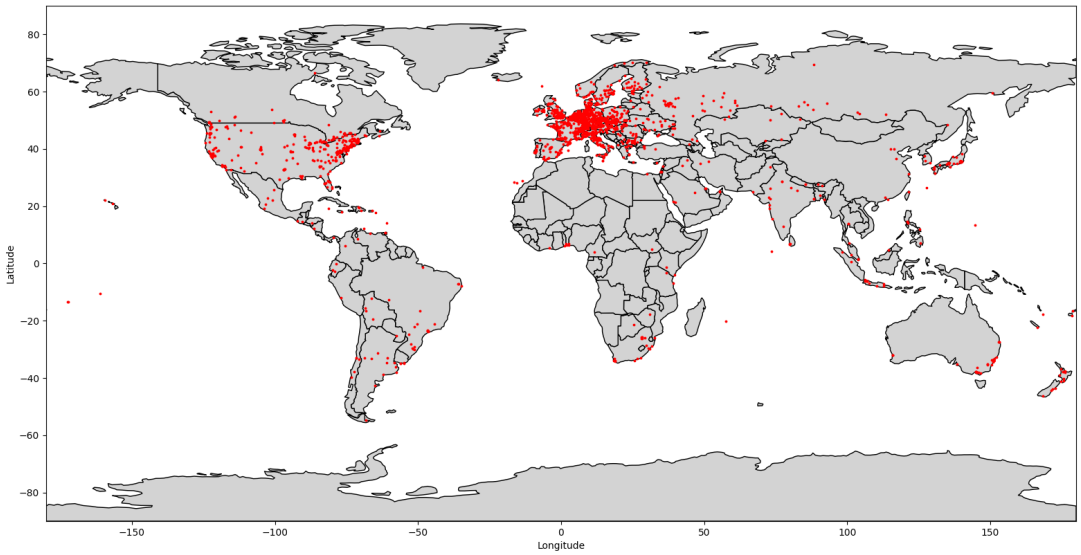

Figure 13: Map of probes used in the Rob's paper[18]

In 2018, as part of work on the Tor network, he wanted to model a global latency map of the world. To do this, he used Ripe Atlas, another Ripe NCC service consisting of a network of probes located around the world and allowing latency measurements to be taken in particular. He therefore published his results in the form of a file with latency measurements for a set of pairs of probes. There are 2035 probes in his publication. See Fig. 14

We therefore take advantage of the publication of its results. We use an IP address geolocation service (MaxMind in this case and its geoIP API[11]) to locate the location of the probes.

For our guard nodes, they are now clustered by AS or by IP prefix. So we have to approximate their location. In the case of an IP prefix, it's quite simple since the geolocation services do not allow to have a very high accuracy and locate all the IPs of the same block in the same place. In the case of the AS it is different, we can have IP address blocks announced by this AS which are located very far from each other. We will therefore look at the blocks of IPs announced by the AS and take the location that comes up most often among the blocks of IPs.

For ASes, the same process is carried out by looking at the location that comes up most often within the IP address blocks.

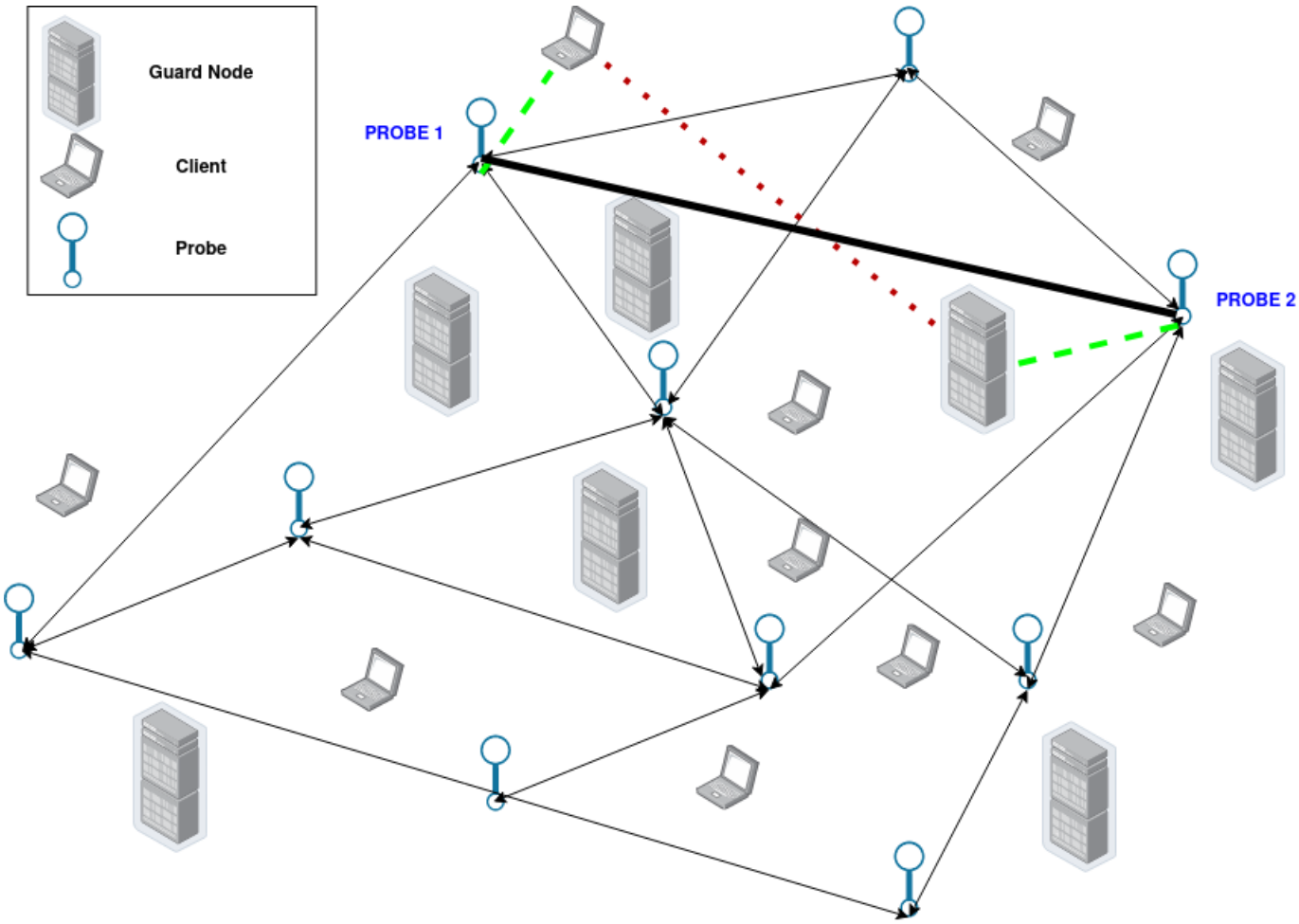

Figure 14: Approximation of latency using Rob's data

We would now like to approximate the latencies for the (client, guard) pairs by trying to find the link in Rob Jansen's data that minimizes the distance d(client,probe1)+d(guard,probe2). The problem is that we cannot content ourselves with calculating the minimum distance probe for the client and the minimum distance probe for the guard because nothing assures us that the link (probe1,probe2) exists in the database. Indeed, when Rob Jansen carried out these measurements, he did not succeed in obtaining measurements between all the probes in order to model all the possible links (reachability issues). We have 2035 probes, so we are supposed to have $2035 * \frac{2034}{2} \simeq 2{,}000{,}000$ but we only have about 1,200,000 links in database. Then, we can try to use an exhaustive search method but it takes so long time (about 10 hours from my experience) and we have to reuse the same process quite often because the topology of network evolves continuously.

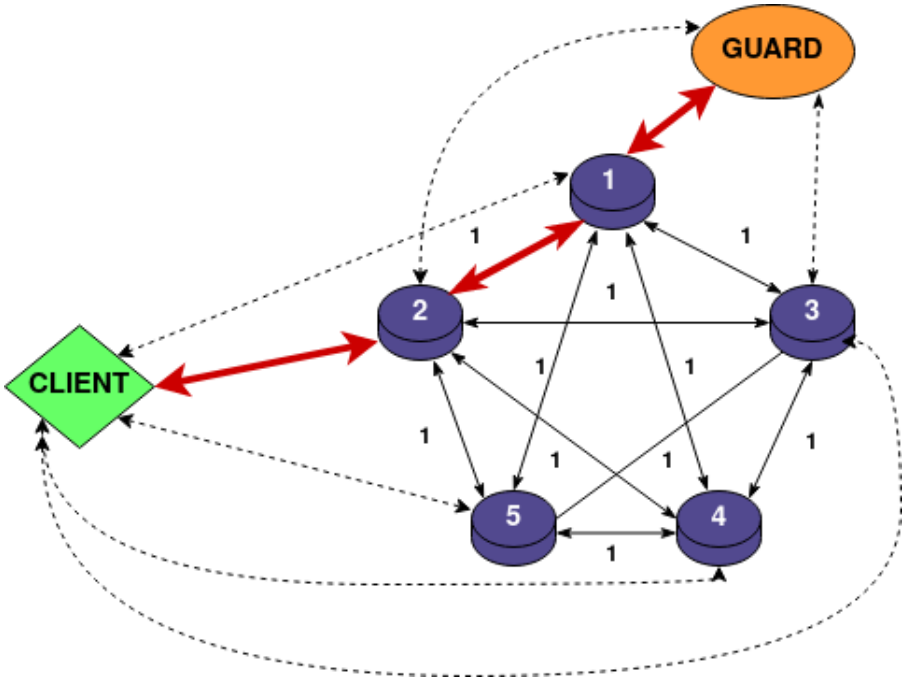

Figure 15: Models network as a graph and run Dijkstra algorithm

We therefore use another approach by modelling our network (probes-clients-guards) by a graph (see Fig. 15) and attempting to run a Dijkstra algorithm. The idea is as follows: compute the distances d(client, probe) and d(guard,probe) for all possible combinations (using IP locations from MaxMind database) and assign these distances as weights in the graph. Then set all the inter-probe distances to the same value. Dijkstra should then find the shortest path and the inter-probe links will not influence the result since they all have the same weight. We could thus extract the minimum latency by checking which segment is chosen between all probes in the smallest path. Then, we could find the corresponding latency to this segment in Rob's data. But there is still a problem.

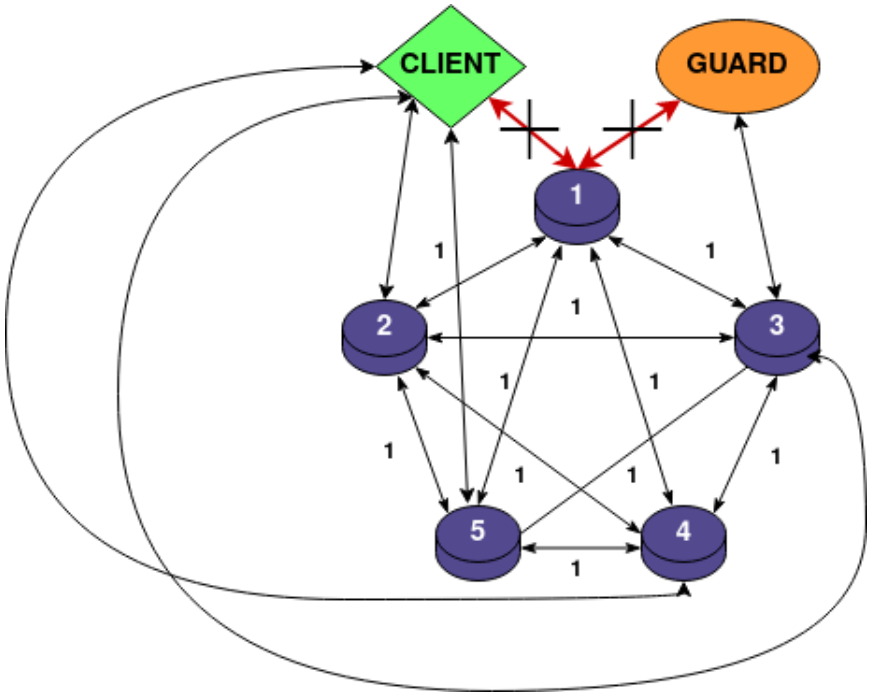

Figure 16: Problem 1 : Too short path

Indeed, the following two cases can occur: a client and a guard both have the same probe as the minimum distance probe (see Fig. 16). In this case, Dijkstra gives us a path of distance 3 and we need a path of distance 4 to be able to extract the corresponding latency. A second case (see Fig. 17) where the path found transit through more than one guard or client is possible and then we have a too long path and cannot extract the correct latency too.

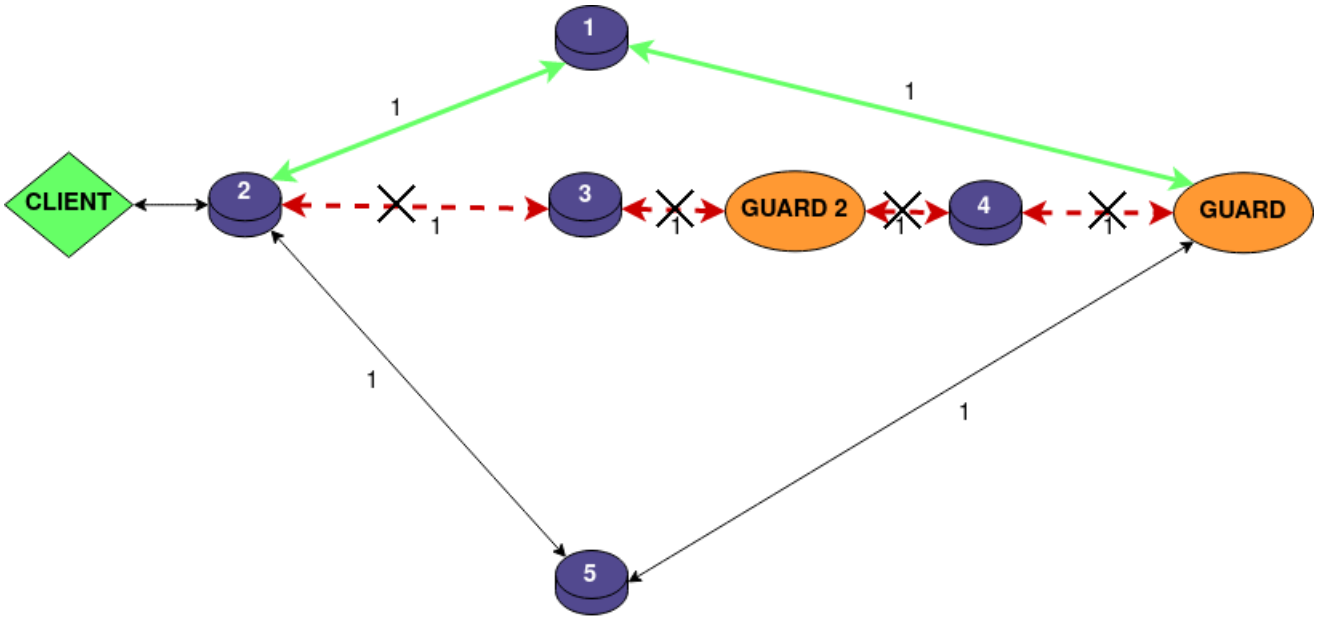

Figure 17: Problem 2 : Too long path

The solution is found by using a little trick : we transform our graph into a directed graph (see Fig. 18). Client-probe links only go to probes and probe-guard links only go to guards. In addition, the probe nodes are duplicated in order to have only "probe $--> $ duplicate probe" directional links, always leaving all the weights of the inter-probe links at the same value. In this case, we will always have a path of length 4 and we can always extract the latency that best approximates the link (client, guard).

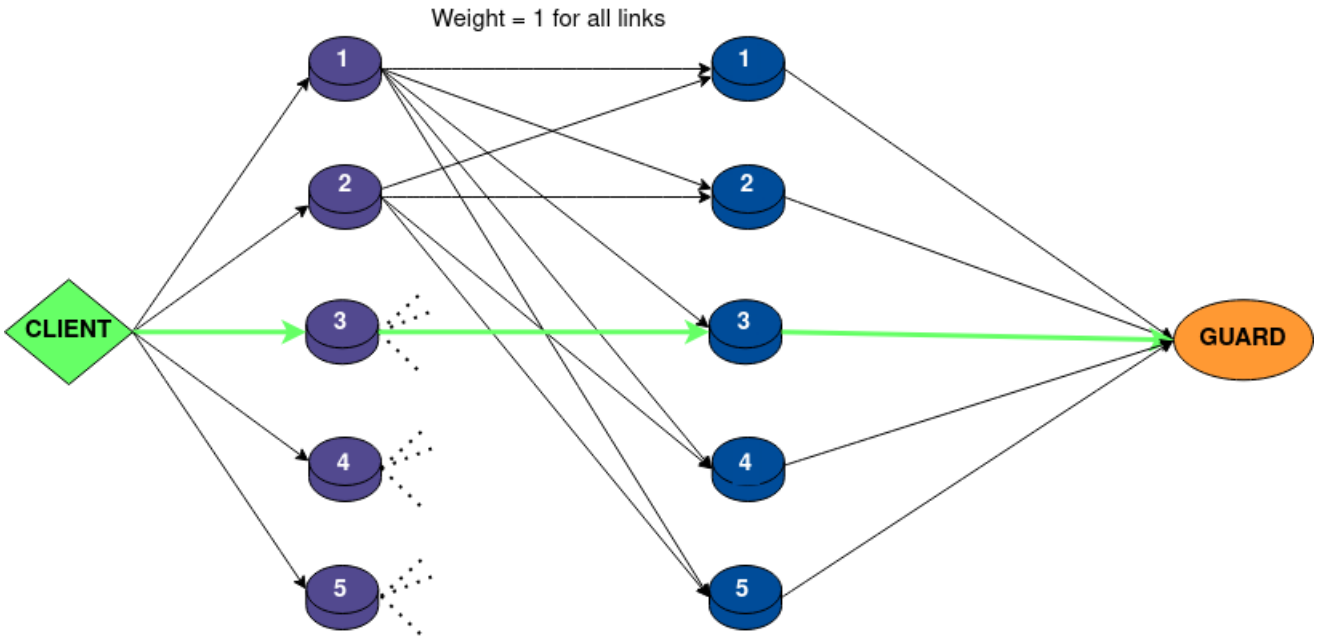

Figure 18: Solution : Directional graph

Thanks to this little trick, we avoid a solution that took more than 10 hours to run and we use another one that takes just under 3 min. It is important because this code must be run regularly.

4.4.2 Guard-Middle and Middle-Exit latencies

To calculate the latencies between nodes it is a little simpler. We can simply retrieve the list of all the relays in the network using the network state files that we created with the metrics at the beginning of the methodology. It is possible to create personalized circuits through Tor by imposing the relays through which it must pass (subject to respecting the conditions set by the network: exit policy, not the same relay twice, /16 different subnet). During the creation of a circuit, a cell is sent to the first relay to ask to build a circuit, a cryptographic key calculation is done, the cell returns to start point then leave again to the second relay and so until reaching its destination. In top of that, Tor send event messages at each of these steps. We will therefore be able to listen to the events and know exactly when a cell has made a round trip. This will allow us to calculate the RTT for each of the links (guard-middle and middle-exit).

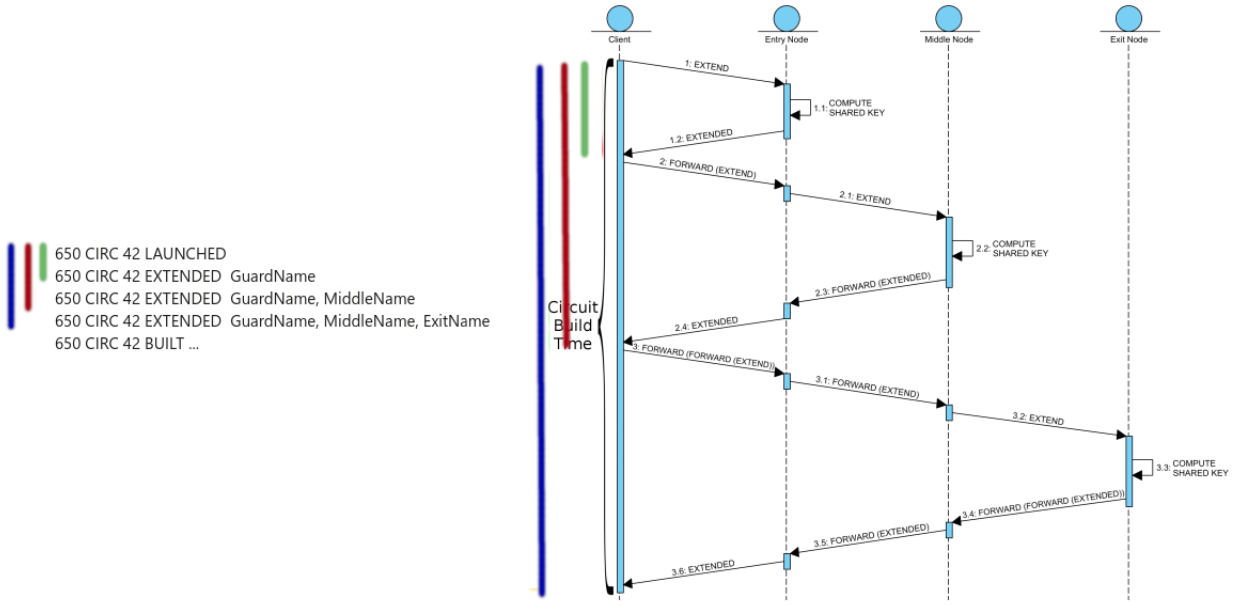

Figure 19: Circuit creation through Tor with event triggered during the process

If we look more closely at Fig. 19 it immediately appears that we can approximate the RTT for the ""guard-middle" links by the red link minus the green link. Similarly, for "middle-exit" links, the round-trip-time can be approximated by the blue link minus the red link. We therefore have an estimate of the latency ($\frac{RTT}{2}$ even if the RTT can be used directly).

We just have to be careful not to send too many requests at once to the same relay. We therefore proceed as follows :

- We create all possible combinations of links (guard-middle) and (middle-exit).
- We mix all the possible combinations to avoid sending requests via similar paths in a row
- We parallelize the circuit creation requests by using a 4-producers-1consumer pattern. Each producers send requests and producer process fetched data and create latency database.
- Each producer send at most 100 circuit creation requests per second

Due to the fact that the network may be disturbed at certain times, it is advisable to take the same measurement several times. In Fig. 20, we took 10 measurements for two links and we see that the data are quite far from each other. Therefore, we consider that the median of these values seems to be a good estimator for the latency value that we will use in the rest of our algorithm.

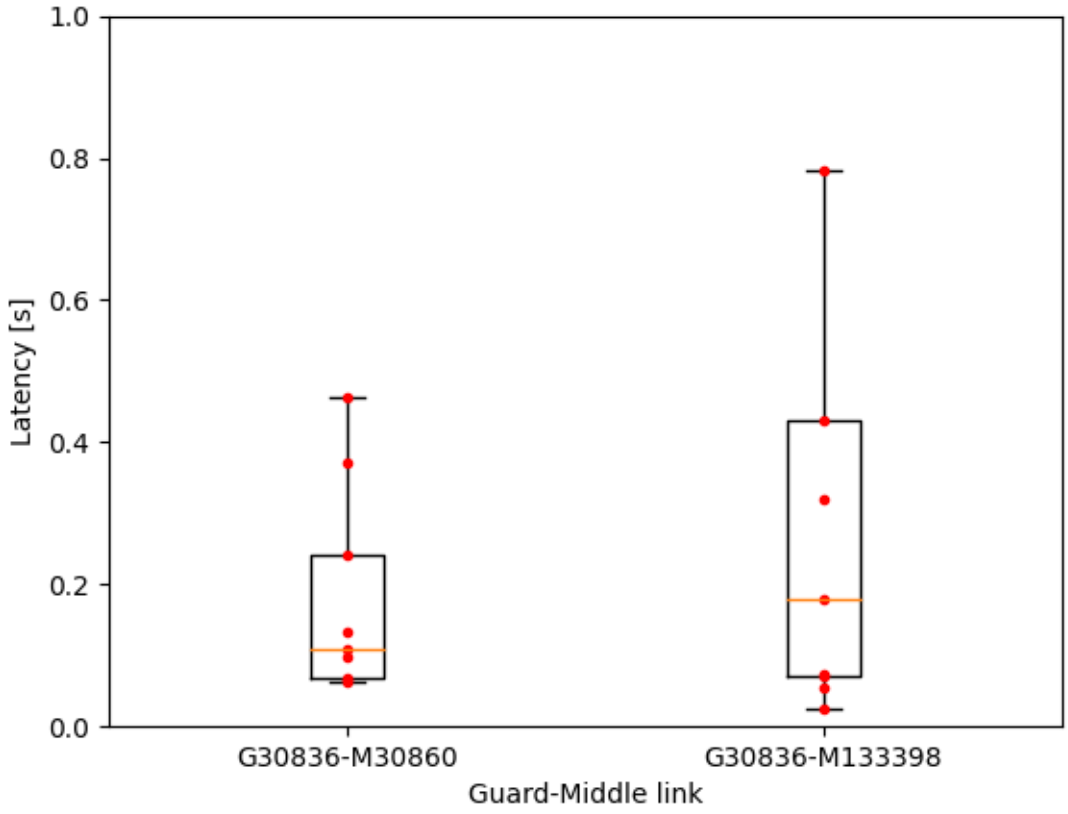

Figure 20: Latencies measurements for two Guard-Middle links

4.5 Vanilla penalties

Vanilla penalties are just expected penalties in position p ∈ {g,m,e} if we use previously computed penalties in combination to Vanilla weights from ABWSR algorithm. It can be expressed as $(\forall\ l \in \mathcal{L}_{\pi_i})$:

$$V_l^{\pi_i} = \sum_{l' \in \mathcal{L}_{\pi_i}} \lambda_{ll'}^{v} \sum_{r \in \mathcal{R}} \frac{v_r^p p_{l'r}^{\pi_i}}{\sum_{s \in \mathcal{R}} v_s^p}$$

with :

$$\lambda_{ll'} = \sum_{l'' \in \mathcal{L}_{\pi_{i-1}}} \lambda_{ll''} \sum_{r \in \mathcal{R}:\Lambda(l'',r)=l'} \frac{v_r^{\pi_i}}{\sum_s v_s^{\pi_i}}$$

Then, we just need Vanilla weights in addition of previous computed penalties. In order to get Vanilla weights, we get Consensus weights in *Network State File* :

- W_{gd} be the weight for choosing a Guard+Exit for the guard position
- W_{md} be the weight for choosing a Guard+Exit for the middle position
- W_{ed} be the weight for choosing a Guard+Exit for the exit position
- W_{me} be the weight for choosing an Exit for the middle position
- W_{mg} be the weight for choosing a Guard for the middle position
- W_{gg} be the weight for choosing a Guard for the guard position
- W_{ee} be the weight for choosing an Exit for the exit position

We also get flags and bandwidth of each relay in *Network State File*. Now we just multiply the appropriate consensus weight for each position (according to flags of each relay) by the bandwidth of the relay.

5 Results and interpretations

5.1 Databases

5.1.1 Client-guard databases

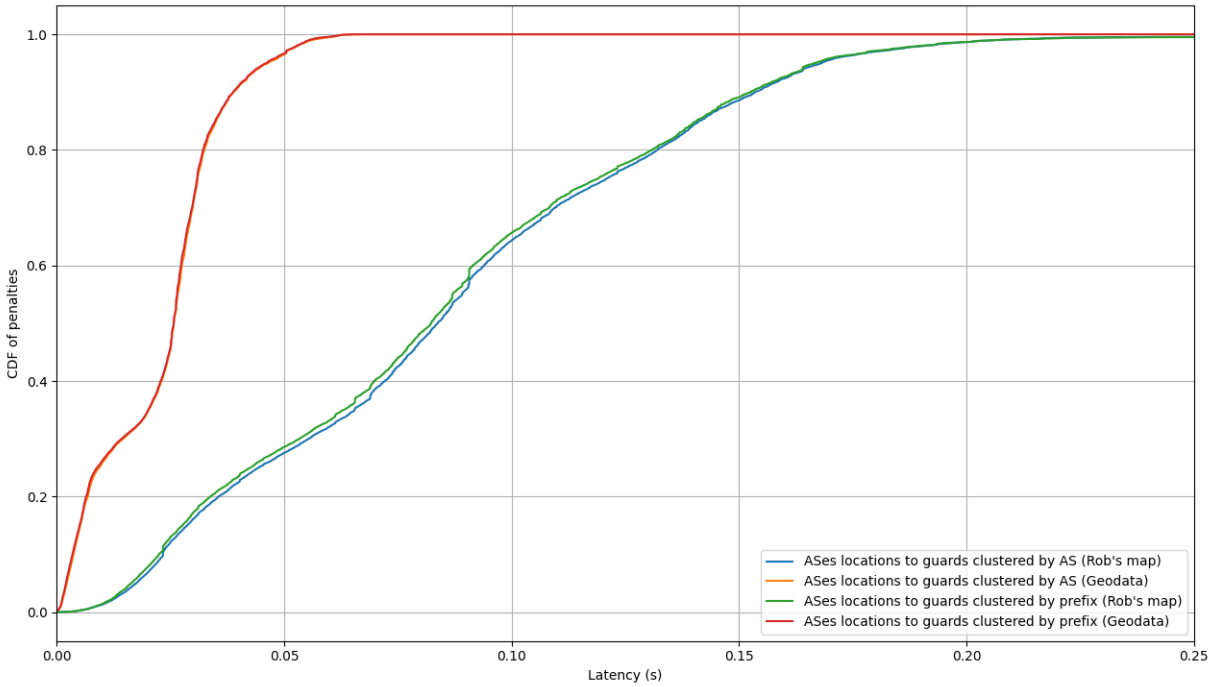

Figure 21: AS Clusters to Guard Clusters latency database CDF

We can see on Fig. 21 that the two CDF of latencies obtained using Geodata estimation (Maxmind) for latencies are almost identical. It is even more visible on the zoom in Fig. 22. We also observe latencies obtained using Geodata are smaller than those obtained using Rob's map. It seems logical as Geodata consider a perfect transmission medium without any loss then lantency are very weak. In fact, path latency on the internet is sum of a propagation delay, a queuing delay and a transmission delay. Geodata estimates get rid of that. However Rob's map estimates took into account those losses because measurements were done with real probes. It is why we have bigger latencies for Rob's curve.

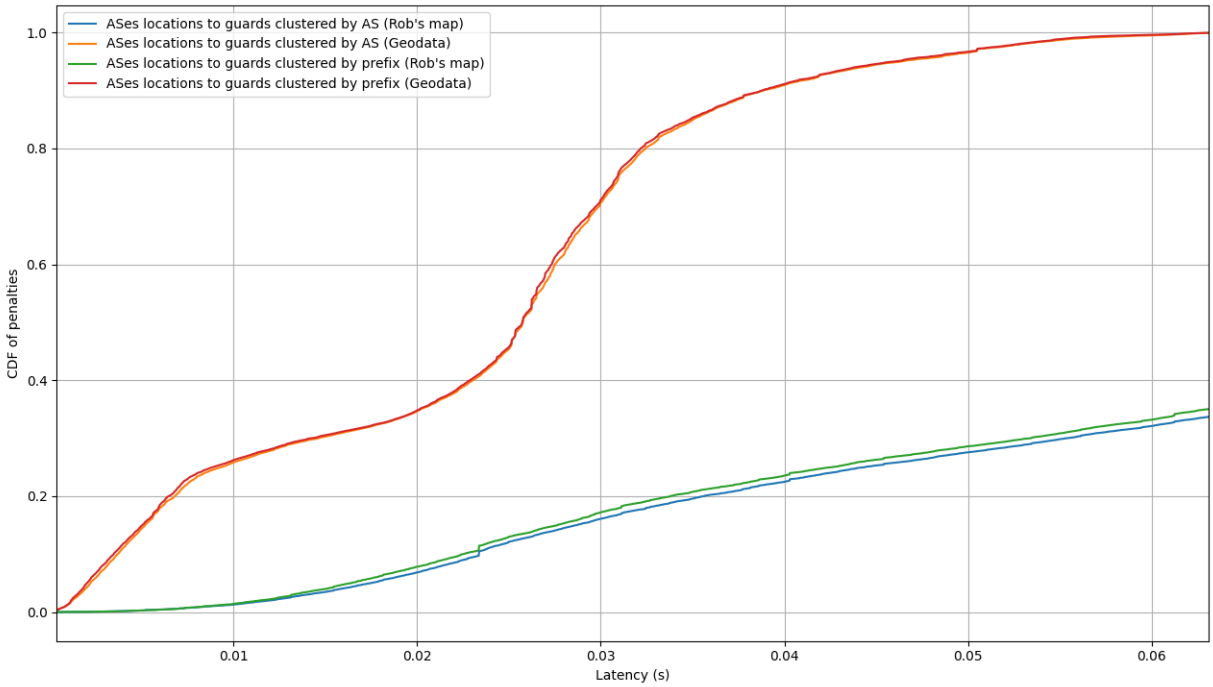

Figure 22: AS Clusters to Guard Clusters latency database CDF zoomed

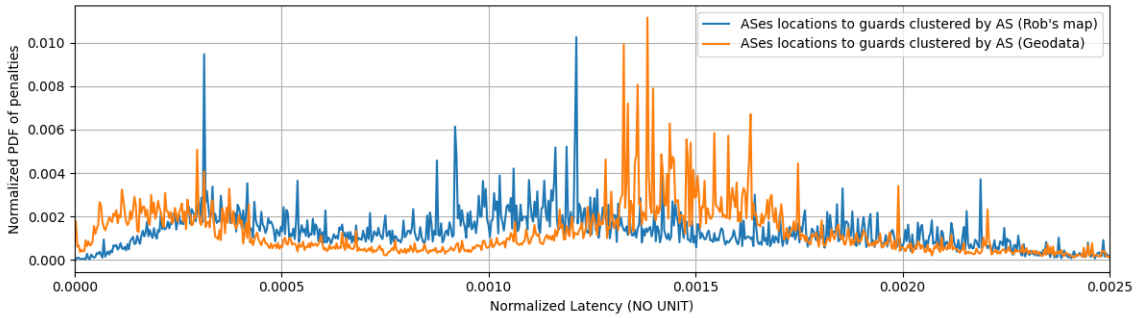

Figure 23: AS Clusters to Guard Clusters latency database PDF normalized

On Fig. 23, we normalize the PDF of the two curves representing latencies using AS clustering. We do that in order to be able to compare the shape of curves. As we see, the shape is not the same and then these two latencies won't have the same impact as penalties in CLAPS. Rob's estimates will probably give better results as measurements are more realistic.

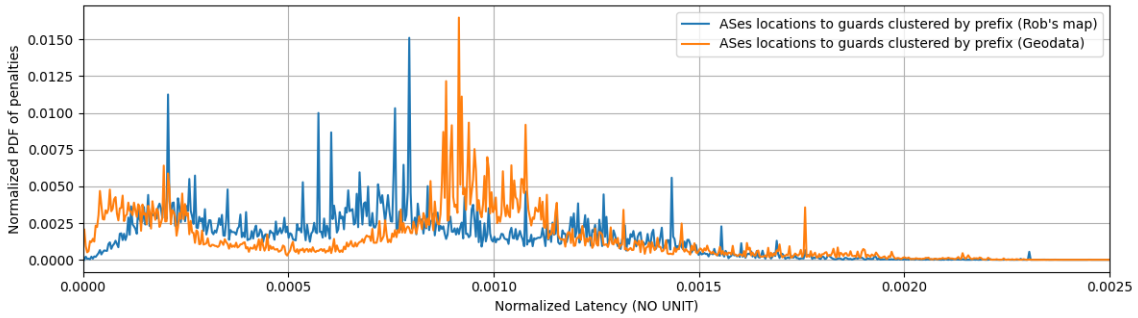

Figure 24: AS Clusters to Guard Clusters latency database PDF NORMALIZED

Here again on Fig. 24, we see that the two curves doesn't have the same shape and then we can conclude the same : latencies using geodata will probably be less meaningful in CLAPS than latencies using Rob's map.

5.1.2 Middle-Exit databases

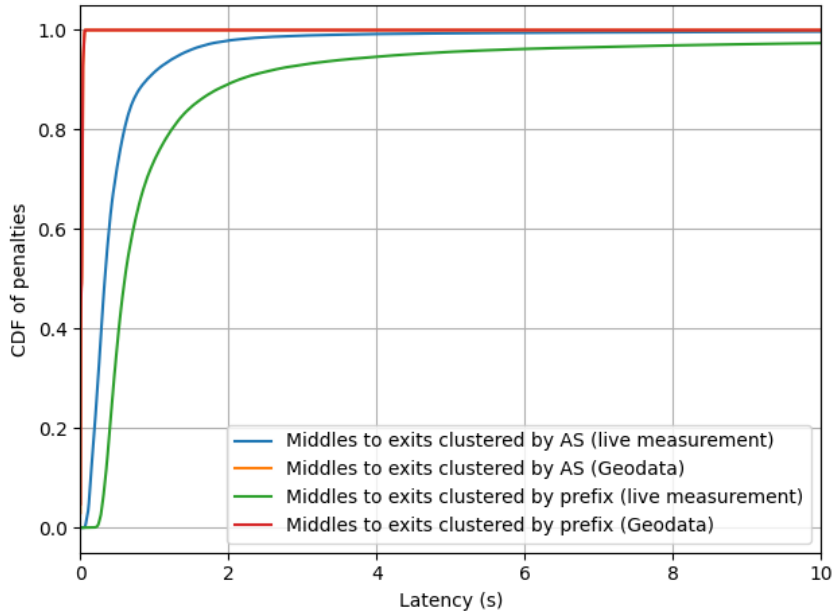

Figure 25: Middle Clusters to Exit Clusters latency database CDF

On Fig. 25, we still have weak latencies when using Geodata. Explanation is the same as before. But we find something interesting : using live latency measures, we find that latencies when clustering by AS are smaller than when clustering by prefixes. It may seem surprising at first glance, but we can interpret it : as there are less AS clusters than prefix clusters, latencies when clustering by AS less represent the realistic distribution of latencies in the network. Indeed an AS can contain a lot of IP prefixes and we choose only one as representative per AS. Therefore we don't cover a large spectrum of possible latencies.

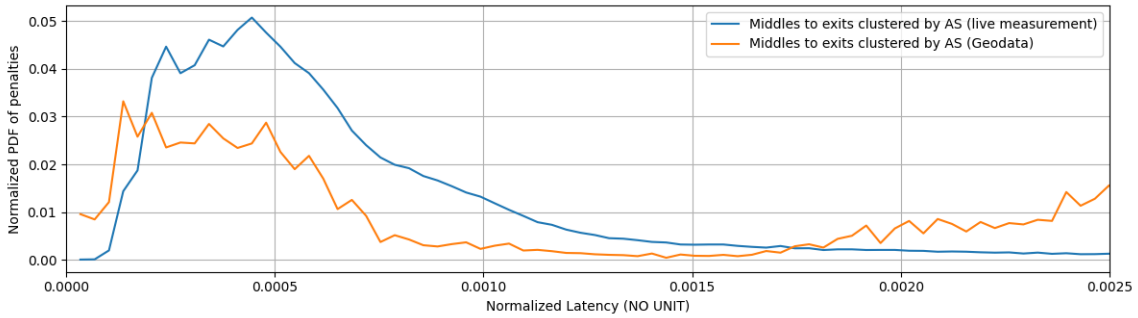

Figure 26: Middle Clusters to Exit Clusters latency database PDF NORMALIZED

On Fig. 26 and 27, we can do the same comments than in the previous section for normalized PDF. We see that the two curves doesn't have the same shape each time and then we can conclude the same : latencies using geodata will probably be less meaningful in CLAPS than latencies using live measures.

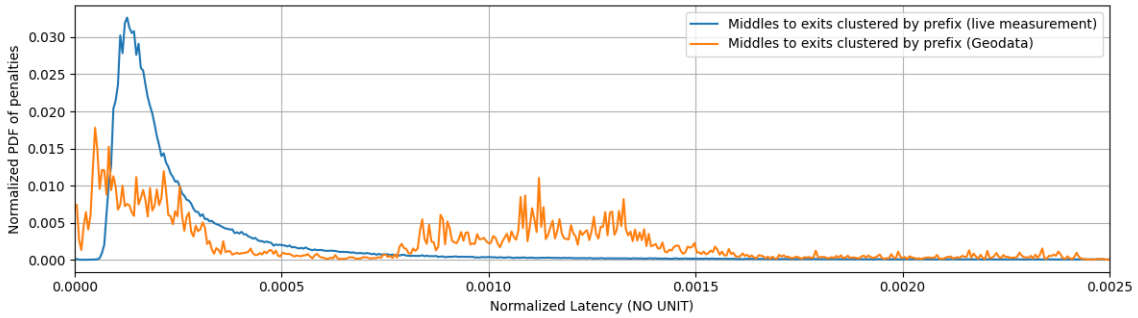

Figure 27: Middle Clusters to Exit Clusters latency database PDF NORMALIZED

5.1.3 Guard-Middle databases

The similar analyse than the previous section may be done for Middle-Exit database.

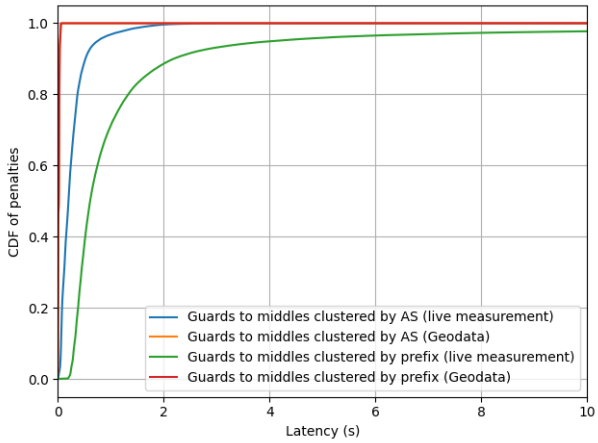

Figure 28: Middle Clusters to Guard Clusters latency database CDF

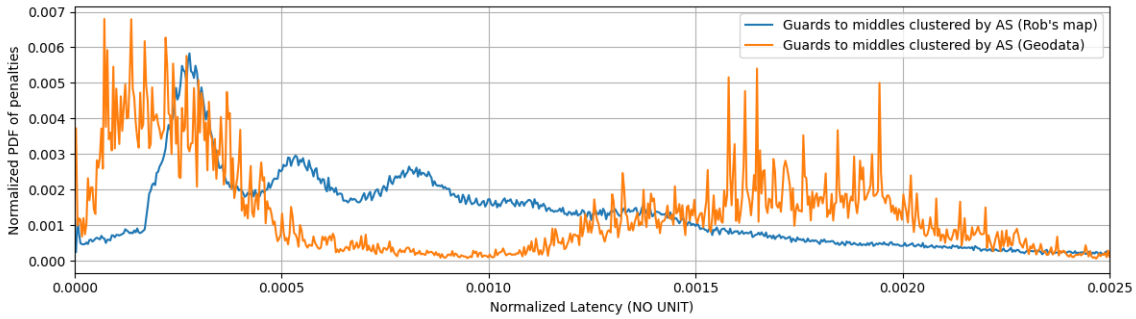

Figure 29: Middle Clusters to Guard Clusters latency database PDF NORMALIZED

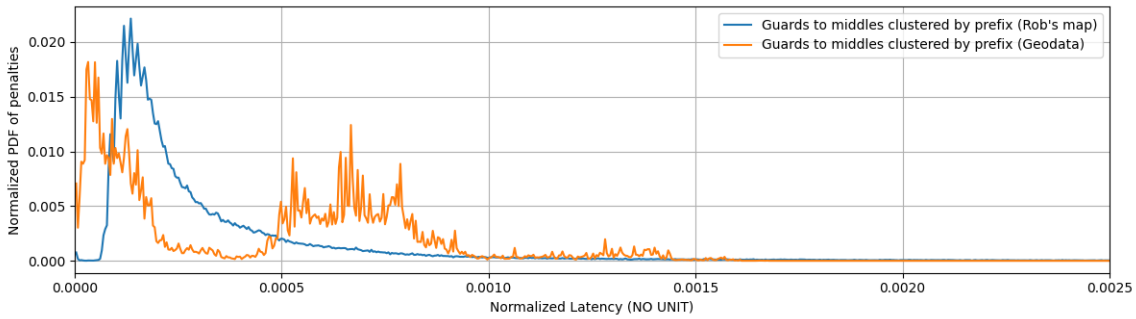

Figure 30: Middle Clusters to Guard Clusters latency database PDF NORMALIZED

5.2 CLAPS test results

For the following tests, I used the new weights I get from the second linear program to perform small live tests on the network. For the sake of realism, I deployed 9 virtual machines (VMs) in order to simulate ten clients around the world (thus, including me) and not only one client that corresponds to my IP address. This allows me to collect a wider variety of data and use a larger number of CLAPS weights across different clients so. Indeed, we will not use the same weights from one client to another since the 10 clients reside in different ases.
I still used the Microsoft Azure Service to deploy my 9 VMs. I decided to put those 9 customers around the world in order to be guaranteed that two of those clients don't lie in the same AS and thus reduce probability that two customers take the same path when I make requests. All of the deployed VMs had the following configuration (Standard B2s engine) :

- 2 vCPUs
- 4 GiB RAM
- 8 GiB SSD storage
- 4000 IOPS

These are not powerful machine configurations since we are only making requests to web sites. Here is the breakdown of my 10 clients:

	IP Address	***Approximate Location (according to MaxMind DB)***
My Home	/	Charleroi
North of South Africa	102.37.139.41	Johannesburg
East of Australia	20.211.41.153	Sidney
Brazil	20.197.225.125	Sao Paulo
North of Europe	20.67.246.160	Dublin
East of USA	20.41.114.92	Séoul
West of USA	102.37.144.64	San Diego
Japan	20.78.32.44	Tokyo
India	20.244.41.38	Pune
Switzerland	20.208.138.65	Zurich

Table 2: Clients test distribution : Microsoft Azure VMs + MyHome

After making all my requests with different set of weights, I get the following curves :

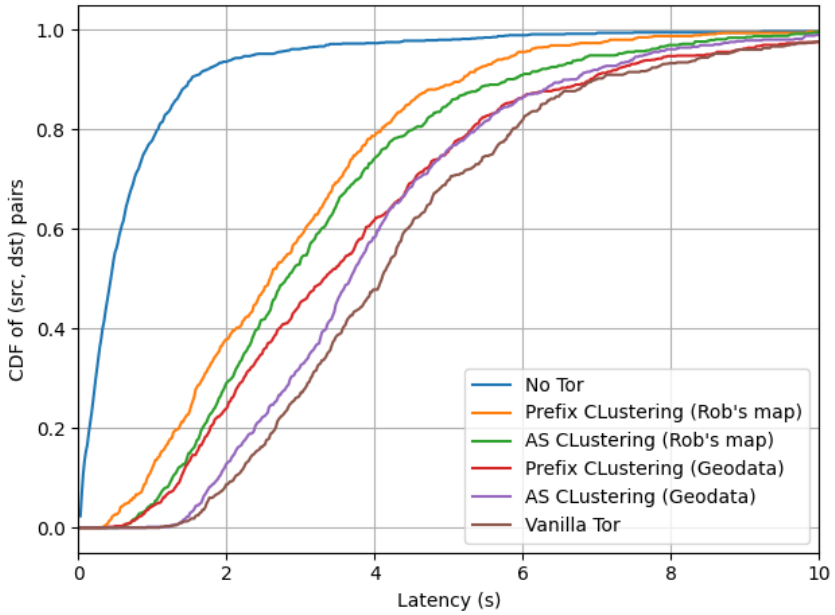

Figure 31: Latencies using weights from CLAPS

A first analysis that can be made is that clustering by prefix gives better results than clustering by AS. This is valid for the curves obtained whether estimated latencies or latencies from the live network have been used. It is quite logical indeed because the clusters by prefix contain fewer relays and are better located compared to the clusters by AS which contain more relays where one representative is taken as location whereas two relays within the same AS can actually be very far apart. Consequently, the paths chosen when using clustering by AS may turn out to be less precise than when using prefix clustering.

A second analysis that can be made is that we always obtain better results using latencies collected live on the network rather than using estimates from geographic data. This ties in with the previous section in which we specified that using latency estimates based on geography does not take into account possible network disturbances and delays. Consequently, the distribution of the estimated latencies is not really representative of the real distribution and then weights computed in CLAPS are less precise. Thus, we obtain better results with live data.

Here are median value for each curve :

- No Tor : 0.7673023493919551
- Latency measurements Prefixes :2.8379726453877194
- Latency measurements AS : 3.2368925580647
- Geodata Prefixes : 3.7658564580228817
- Geodata AS : 4.0203863781166085
- Tor : 4.431149151887786

We find using these medians that using latency live measurements with prefix clustering give us a 35% improvement of median latency compared to Vanilla Tor. We have 27% when we use AS clustering.
If we use instead latencies estimates, we find improvements of 15% and 9% according to prefix and AS clustering. IT shows us once again that using live date result in more precise computation in CLAPS and then we have more efficient weights and a better latency improvement.

6 Future works and improvements

Try another type of clustering

In the context of thiw work, we used two types of clustering in our methodology :

- Clustering by AS
- Clustering by Prefix

On one hand, we have seen in the previous results that we obtain better performances from prefix clustering than AS clustering. On the other hand, prefix clustering is much more time consuming to create (Guard Clusters, Middle Clusters) and (Middle Clusters, Exit Clusters) databases. As we have seen, latencies gathered in this part of the methodology are likely to be impacted by perturbations on the network (relay temporarily inaccessible, overload of a node, packet loss,...). This is why we have specified that we should make about ten queries for each link in the database and then take the median which seems to be a good estimator instead of the average which would be impacted by possible outliers.

Here are presented approximate timing to run one measurement for each kind of pair [database type/clustering type] (considering 507 guard, 979 middle and 222 exit clusters for AS clustering and 1180 guard, 2707 middle and 409 exit clusters for prefix clustering) :

Timing to run one iteration of database measure		
Clustering Type / Database Type	*AS*	*PREFIX*
Guard-Middle	1927 [s] $\simeq$ 32 [min]	8210 [s] $\simeq$ 137 [min]
Middle-Exit	702 [s] $\simeq$ 12 [min]	2739 [s] $\simeq$ 46 [min]

Table 3: Timing to run one iteration of database measurement

We easily see from these data that the run of ten of these measures quickly becomes restrictive in the case of a clustering by prefix. It is quite bad because this type of clustering is more efficient than the second one. So why not try a trade-off between the two types ? Imagine that we use a clustering by geographical area, as on Fig. 32 where we can see that we have delimited geographical cells. This would allow a clustering that would compensate for the defect of clustering by AS, i.e. we would have here blocks of IPs grouped in a defined geographical area. We would no longer find ourselves in the case of an AS clustering where two relays (included in two blocks of IPs) within an AS can be very far from each other.

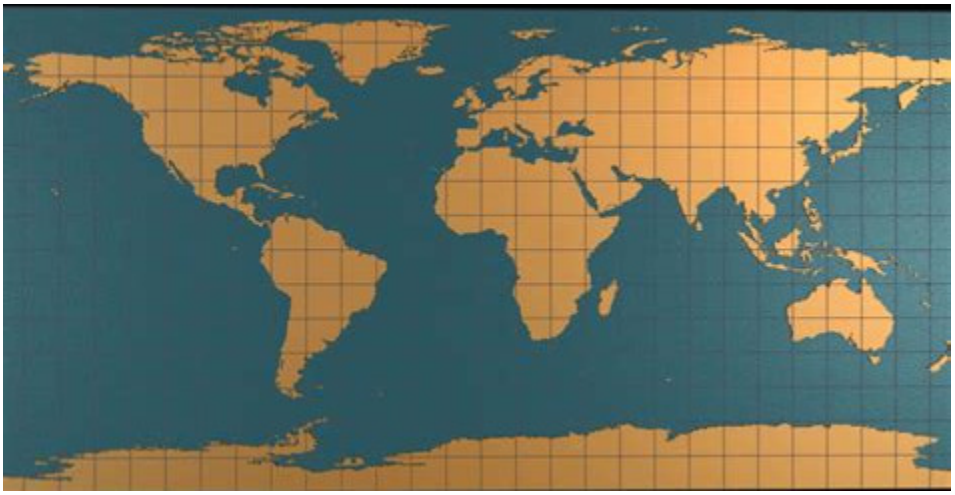

Figure 32: Geographic clustering by defined cell size

This kind of clustering is only limited by our imagination about the shape (square, triangle,...) or the size of the cells. We could probably find a good compromise in order to reduce the latency measurement time compared to prefix clustering but which guarantees a better efficiency in the algorithm than AS clustering.

Optimizing critical parts of the algorithm

Besides what we just mentioned in the previous section, it is important to optimize the runtime of all the pieces of our methodology to apply CLAPS. Indeed, if we look at Fig. 33, we can see that the distribution of relays changes quite strongly over 24 hours.

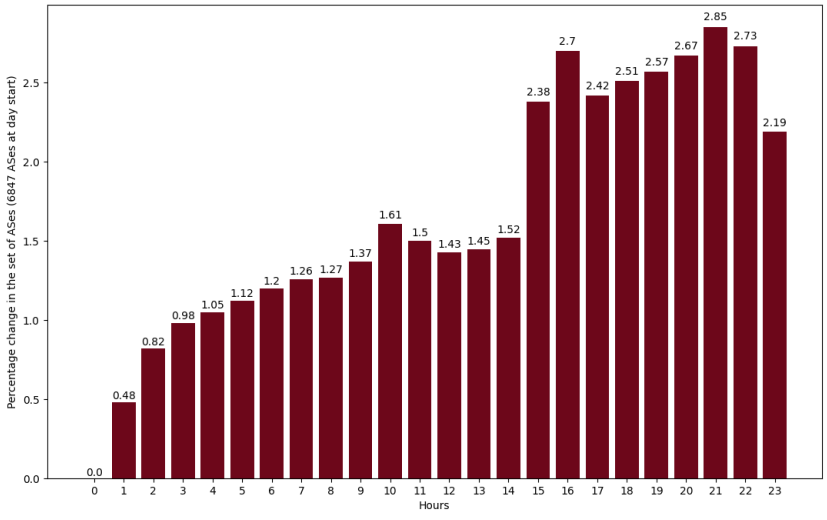

Figure 33

At the end of the day, about 2% of the relays are not in the relay set anymore. On Fig. 34, we can even see that at the end of the month, about 17% of nodes are not in the set anymore.

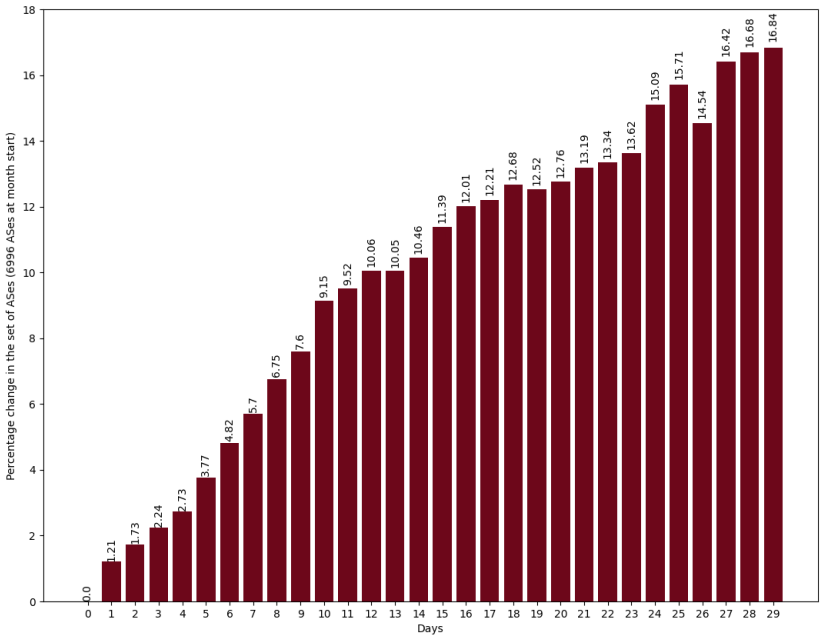

Figure 34

This can be due to a variety of reasons (compromised node removed from the set, node stopped by the volunteer operator, temporary node for experimentation, hibernation,...). The Tor network changes continuously, so it is necessary that a consensus file is published every hour to keep the list of relays in the network up to date. It is therefore useful to optimize the code runtime as much as possible in order to be able to restart the whole algorithm regularly. Among the things that can still be optimized :

- Code with purpose to fetch data like consensus file, relay descriptors, AS/prefix information. These pieces of code are still quite slow because it requires to connect to a server containing all data we need. But sometimes we fetch too much data, maybe it is possible to select more precisely according to our needs and then reduce runtime.
- We actually use Clp solver in order to solve our LPs. It is known as a performant solver in term of free product. Maybe considering a commercial solver like CPLEX solver[15] may help to solve faster and more efficiently our linear programs.

Large-scale live test or simulation tools

In our tests, we use a set of VMs as a set of clients. However, it is still very small. In order to have a more representative subset of clients worldwide to evaluate our methodology, it might be interesting to deploy even more nodes (100-200?). For instance, we can use nodes from PlanetLab[22] to construct a set of customers.

Another idea is to use simulation tools like Shadow ([19], [17]). It is an experimentation tool simplifying research, development, testing and evaluation of real network applications. It could allow us to test our algorithm inside an internally simulated network. Moreover, we could access/simulate some data which are quite difficult to obtain during live tests (packet loss, overloading some relays, ...)

Security evaluation

As we only focus on the improvement of global latency during this work, it may be interesting to make an evaluation of the actual security when using our algorithm. Although we leave the relay placement factor θ to 1 and we use the same criteria as Vanilla Tor to select nodes for circuit build during all of our tests, we know the duality between latency and anonymity. If we improve latency, then anonymity (and thus, security) is impacted. By using previously mentionned Shadow Tor Network, we are able to simulate some attacks in the network and then evaluate security.

7 Conclusion

To conclude, we were able to improve the overall latency in the Tor network. To achieve this, we began with ideas from one of the pioneering algorithms to improve latency. LASTor recommendeded using the latency estimate in the weight computation for relay selection. Since our CLAPS algorithm requires the use of a function that associates a penalty to each path (higher is worse), we decided to use latency as a penalty. In addition to using a theoretical value as used in LASTor, we also used real latency data collected from the live network.

This allowed us to compare the two approaches. We saw that using real values leads to better performances than using estimates that do not take into account some real parameters (network disturbance, transmission delay, propagation in a non-ideal environment,...). It is therefore the CLAPS algorithm combined with the use of latencies collected on the network which yields better results.

Now that we have improved the latency of the network, it is not over yet. In the future, some people may try to improve latency further, but they may also want to look at security and anonymity within the Tor network. The field of research in this field is still vast and many things are still to be tested.

From a personal perspective, this has not always been an easy job. The years of research behind Tor and all of its underlying terminology have not always been easy to understand. But I am proud to have acquired new knowledge, not pretending to be an expert. I now know a bit more about Tor than I did prior to this work, where Tor had been just a web browser for me.

I was also able to develop my programming skills and realize that rigor is required in projects of this size. Optimization is a key element, which we do not necessarily realize during studies where we do not necessarily optimize because we do not deal with data of the size of those used in this algorithm. Thus, I emerged from this experience with a lot of knowledge and a lot of personal experience !

References

[1] *Ahrefs - Outils et ressources SEO pour augmenter votre trafic de recherche*. 2022. URL: https://ahrefs.com/fr.

[2] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. *LASTor: A Low-Latency AS-Aware Tor Client*. 2012. DOI: 10.1109/SP.2012.35.

[3] *Alexa Top Sites*. 2022. URL: https://docs.aws.amazon.com/AlexaTopSites/latest/index.html.

[4] Hari Ram Balakrishnan. *Wide-area Internet Routing*. Massachusetts Institute of Technology - Department of Electrical Engineering and Computer Science. 2009.

[5] *CollecTor*. 2022. URL: https://metrics.torproject.org/collector.html.

[6] George Danezis, Roger Dingledine, and Nick Mathewson. *Mixminion: Design of a Type III Anonymous Remailer Protocol*. Oct. 2003.

[7] Roger Dingledine and Nick Mathewson. *Tor protocol specifications*. 2022. URL: https://github.com/torproject/torspec.

[8] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The Second-Generation Onion Router*. San Diego, CA, Aug. 2004. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[9] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. *Analyzing China's Blocking of Unpublished Tor Bridges*. 2018.

[10] Michael Freedman et al. *Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer*. June 2002. DOI: 10.1007/3-540-45748-8_12.

[11] *GeoIP and GeoLite API Requests*. 2022. URL: https://dev.maxmind.com/geoip/docs/web-services/requests?lang=en.

[12] Andy Greenberg. *Mapping How Tor's Anonymity Network Spread Around the World*. Sept. 2015. URL: https://www.wired.com/2015/09/mapping-tors-anonymity-network-spread-around-world.

[13] C. Gulcu and G. Tsudik. *Mixing E-mail with Babel*. 1996. DOI: 10.1109/NDSS.1996.492350.

[14] *IANA - Autonomous System (AS) Numbers*. 2022. URL: https://www.iana.org/assignments/as-numbers/as-numbers.xhtml.

[15] *ILOG CPLEX Optimization Studio - IBM ILOG CPLEX Optimizer*. 2022. URL: https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer.

[16] *Inequality Constraints, Complementary slackness condition, Maximisation and Minimisation, Kuhn-Tucker method: summary*. 2005. URL: http://www1.maths.leeds.ac.uk/~cajones/math2640/notes4.pdf.

[17] Rob Jansen and Nicholas Hopper. *Shadow: Running Tor in a Box for Accurate and Efficient Experimentation*. 2011.

[18] Rob Jansen, Matthew Traudt, and Nicholas Hopper. *Privacy-Preserving Dynamic Learning of Tor Network Traffic*. See also https://tmodel-ccs2018.github.io. 2018.

[19] Rob Jansen et al. *Methodically modeling the Tor network*. Aug. 2012.

[20] David Kurniadi. *The Difference Between Using Proxy Server and VPN*. Nov. 2015. DOI: 10.24167/sisforma.v2i1.406.

[21] *Onionperf*. 2022. URL: https://gitlab.torproject.org/tpo/network-health/metrics/onionperf.

[22] *PlanetLabEurope*. 2022. URL: https://www.planet-lab.eu.

[23] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. *Anonymous connections and onion routing*. 1998. DOI: 10.1109/49.668972.

[24] Michael K. Reiter and Aviel D. Rubin. *Crowds: Anonymity for Web Transactions*. New York, NY, USA, 1998. DOI: 10.1145/290163.290168. URL: https://doi.org/10.1145/290163.290168.

[25] Marc Rennhard. *MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection*. Jan. 2002.

[26] *RIPEstat*. URL: https://stat.ripe.net/about.

[27] Florentin Rochet et al. *CLAPS: Client-Location-Aware Path Selection in Tor*. 2020.

[28] *Semrush - Online Marketing Can Be Easy*. 2022. URL: https://www.semrush.com.

[29] Yixin Sun et al. *Counter-RAPTOR: Safeguarding Tor Against Active Routing Attacks*. 2017. DOI: 10.48550/ARXIV.1704.00843. URL: https://arxiv.org/abs/1704.00843.

[30] Paul Syverson et al. *Towards an Analysis of Onion Routing Security*. Ed. by Hannes Federrath. Berlin, Heidelberg, July 2001. DOI: 10.1007/3-540-44702-4_6. URL: https://doi.org/10.1007/3-540-44702-4_6.

[31] *Torperf - Tor performance evaluation tools*. 2022. URL: https://gitweb.torproject.org/torperf.git.

[32] TorPS. *TorPS*. 2022. URL: https://github.com/torps/torps.

[33] *Users – Tor Metrics*. 2022. URL: https://metrics.torproject.org/userstats-relay-country.html.

[34] Gerry Wan et al. *Guard Placement Attacks on Path Selection Algorithms for Tor*. Oct. 2019. DOI: 10.2478/popets-2019-0069.

[35] *What is a bridge? Tor Project Support*. 2022. URL: https://support.torproject.org/censorship/censorship-7.

A Use case exemple of CLAPS

Let's consider the following little network :

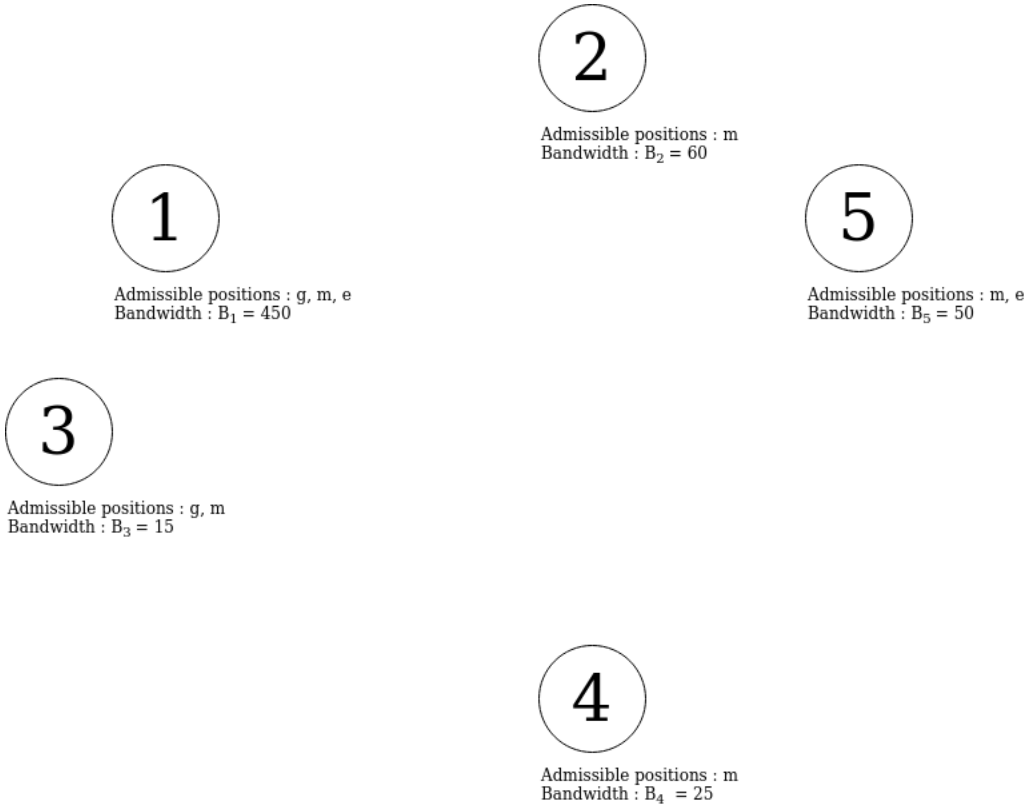

Figure 35: Example of Tor's network with 5 relays

This network is composed of five relays. Each relay has its own bandwidth and admissible positions as we can see on Fig. 35.

A.1 LP1

First, we must ensure that the network is load-balanced. For that purpose, we use the initial linear program called LP1 which allow us to compute the minimum bandwidth across all positions.

A.1.1 Expression of conditions

Let :

- $\mathcal{P} = \{\text{g,m,e}\}$ be the set of circuit positions
- $\mathcal{R} = \{\text{r1,r2,r3,r4,r5}\}$ the set of all relays and $\mathcal{R}_p$ the subset of relays that can be used in position $\text{p} \in \mathcal{P}$
- B_r the bandwidth of relay $\text{r} \in \mathcal{R}$

LP1 outputs the minimum bandwidth β that can be provided in all $\text{p} \in \mathcal{P}$ simultaneously (β takes the output value of b). It can be formulated as :

$$
\begin{aligned}
&\textbf{Maximize} && b && (1)\\
&\textbf{Subject to} &&&&\\
&\forall r \in \mathcal{R} : && \sum_{p \in \mathcal{P}} w_r^p \leq B_r && (2)\\
&\forall p \in \mathcal{P} : && \sum_{r \in \mathcal{R}} w_r^p \geq b && (3)\\
&\forall p \in \mathcal{P}, r \in \mathcal{R} : && w_r^p \geq 0 && (4)\\
&\forall p \in \mathcal{P}, r \in \mathcal{R} \backslash \mathcal{R}_p : && w_r^p = 0 && (5)
\end{aligned}
$$

Figure 36: Linear program LP1

Equation (5) set weights to zero when inadmissible in the position. In our case, we have :

$$
\begin{aligned}
w_{r2}^g =& w_{r2}^e = 0\\
& w_{r3}^e = 0\\
w_{r4}^g =& w_{r4}^e = 0\\
w_{r5}^g =& \quad = 0
\end{aligned}
\tag{6}
$$

Equation (2) limit each relay's bandwidth to its own bandwidth. In our case, we have :

$$
\begin{aligned}
r = r_1 : \quad & w_{r1}^g + w_{r1}^m + w_{r1}^e \leq B_1\\
r = r_2 : \quad & w_{r2}^g + w_{r2}^m + w_{r2}^e \leq B_2\\
r = r_3 : \quad & w_{r3}^g + w_{r3}^m + w_{r3}^e \leq B_3\\
r = r_4 : \quad & w_{r4}^g + w_{r4}^m + w_{r4}^e \leq B_4\\
r = r_5 : \quad & w_{r5}^g + w_{r5}^m + w_{r5}^e \leq B_5
\end{aligned}
\tag{7}
$$

Equations of (7) are simplified using results of (6) :

$$\begin{aligned}
r = r_1 : &\quad w^g_{r1} + w^m_{r1} + w^e_{r1} \leq B_1 \\
r = r_2 : &\quad w^g_2 + w^e_2 \leq B_2 \\
r = r_3 : &\quad w^g_3 + w^m_3 \leq B_3 \\
r = r_4 : &\quad w^m_4 \leq B_4 \\
r = r_5 : &\quad w^m_{r5} + w^e_5 \leq B_5
\end{aligned} \tag{8}$$

Equation (3) allocate at least b weight to each position. In our case, we have :

$$\begin{aligned}
p = g \ : &\quad w^g_{r1} + w^g_{r2} + w^g_{r3} + w^g_{r4} + w^g_{r5} \geq b \\
p = m : &\quad w^m_{r1} + w^m_{r2} + w^m_{r3} + w^m_{r4} + w^m_{r5} \geq b \\
p = e \ : &\quad w^e_{r1} + w^e_{r2} + w^e_{r3} + w^e_{r4} + w^e_{r5} \geq b
\end{aligned} \tag{9}$$

Using results of (6) once again, equations of (9) are simplified :

$$\begin{aligned}
p = g \ : &\quad w^g_{r1} + w^g_{r3} \geq b \\
p = m : &\quad w^m_{r1} + w^m_{r2} + w^m_{r3} + w^m_{r4} + w^m_{r5} \geq b \\
p = e \ : &\quad w^e_{r1} + w^e_{r5} \geq b
\end{aligned} \tag{10}$$

Finally, equation (4) requires weights to be non-negative :

$$\begin{array}{lllll}
w^g_{r1} \geq 0 & w^g_{r2} \geq 0 & w^g_{r3} \geq 0 & w^g_{r4} \geq 0 & w^g_{r5} \geq 0 \\
w^m_{r1} \geq 0 & w^m_{r2} \geq 0 & w^m_{r3} \geq 0 & w^m_{r4} \geq 0 & w^m_{r5} \geq 0 \\
w^e_{r1} \geq 0 & w^e_{r2} \geq 0 & w^e_{r3} \geq 0 & w^e_{r4} \geq 0 & w^e_{r5} \geq 0
\end{array} \tag{11}$$

Using results of (6) once again, equations of (11) are simplified :

$$\begin{array}{lllll}
w^g_{r1} \geq 0 & & w^g_{r3} \geq 0 & & \\
w^m_{r1} \geq 0 & w^m_{r2} \geq 0 & w^m_{r3} \geq 0 & w^m_{r4} \geq 0 & w^m_{r5} \geq 0 \\
w^e_{r1} \geq 0 & & & & w^e_{r5} \geq 0
\end{array} \tag{12}$$

A.1.2 Solving the maximisation problem

In this section we try to solve the maximisation problem using theory of *Appendix B*.

In our case we define $f(b, w^g_{r1}, w^g_{r3}, w^m_{r1}, w^m_{r2}, w^m_{r3}, w^m_{r4}, w^m_{r5}, w^e_{r1}, w^e_{r5}) = b$ subject to constraints :

$$\begin{array}{ll}
\lambda_1(w^g_{r1} + w^g_{r3} - b) = 0 & \lambda_5(w^m_{r2} - B_2) = 0 \\
\lambda_2(w^m_{r1} + w^m_{r2} + w^m_{r3} + w^m_{r4} + w^m_{r5} - b) = 0 & \lambda_6(w^g_{r3} + w^m_3 - B_3) = 0 \\
\lambda_3(w^e_{r1} + w^e_{r5} - b) = 0 & \lambda_7(w^m_{r4} - B_4) = 0 \\
\lambda_4(w^g_{r1} + w^m_{r1} + w^e_{r1} - B_1) = 0 & \lambda_8(w^m_{r5} + w^e_{r5} - B_5) = 0
\end{array} \tag{13}$$

$$
\begin{array}{lll}
 & & -w^g_{r1} \leq 0 \\
 & & -w^m_{r1} \leq 0 \\
w^g_{r1} + w^m_{r1} + w^e_{r1} - B_1 \leq 0 & & -w^e_{r1} \leq 0 \\
w^m_{r2} - B_2 \leq 0 & b - w^g_{r1} - w^g_{r3} \leq 0 & -w^m_{r2} \leq 0 \\
w^g_{r3} + w^m_3 - B_3 \leq 0 & b - w^m_{r1} - w^m_{r2} - w^m_{r3} - w^m_{r4} - w^m_{r5} \leq 0 & -w^g_{r3} \leq 0 \\
w^m_{r4} - B_4 \leq 0 & b - w^e_{r1} - w^e_{r5} \leq 0 & -w^m_{r3} \leq 0 \\
w^m_{r5} + w^e_{r5} - B_5 \leq 0 & & -w^m_{r4} \leq 0 \\
 & & -w^m_{r5} \leq 0 \\
 & & -w^e_{r5} \leq 0
\end{array}
\tag{14}
$$

N.B. Previous conditions are just conditions from (8), (10) and (12) rewrited under the form "*condition* ≤ 0"

Then, we define the Lagrangian :

$$
\begin{aligned}
L(b, w^g_{r1}, w^g_{r3}, w^m_{r1}, w^m_{r2}, w^m_{r3}, w^m_{r4}, w^m_{r5}, w^e_{r1}, w^e_{r5}) = b &- \lambda_1(b - w^g_{r1} - w^g_{r3}) - \lambda_2(b - w^m_{r1} - w^m_{r2} - w^m_{r3} - w^m_{r4} - w^m_{r5}) \\
&- \lambda_3(b - w^e_{r1} + w^e_{r5}) - \lambda_4(w^g_{r1} + w^m_{r1} + w^e_{r1} - B_1) \\
&- \lambda_5(w^m_{r2} - B_2) - \lambda_6(w^g_{r3} + w^m_3 - B_3) \\
&- \lambda_7(w^m_{r4} - B_4) - \lambda_8(w^m_{r5} + w^e_{r5} - B_5) \\
&- \lambda_9(-w^g_{r1}) - \lambda_{10}(-w^m_{r1}) - \lambda_{11}(-w^e_{r1}) - \lambda_{12}(-w^m_{r2}) \\
&- \lambda_{13}(-w^g_{r3}) - \lambda_{14}(-w^m_{r3}) - \lambda_{15}(-w^m_{r4}) - \lambda_{16}(-w^m_{r5}) - \lambda_{17}(-w^e_{r5})
\end{aligned}
$$

and solve the n first order equality conditions :

$$
\begin{array}{lll}
\dfrac{\partial L}{\partial b} = 1 - \lambda_1 - \lambda_2 - \lambda_3 = 0 & & \\
\dfrac{\partial L}{\partial w^g_{r1}} = \lambda_1 - \lambda_4 + \lambda_9 = 0 & \dfrac{\partial L}{\partial w^e_{r2}} = 0 = 0 & \dfrac{\partial L}{\partial w^m_{r4}} = \lambda_2 - \lambda_7 + \lambda_{15} = 0 \\
\dfrac{\partial L}{\partial w^m_{r1}} = \lambda_2 - \lambda_4 + \lambda_{10} = 0 & \dfrac{\partial L}{\partial w^g_{r3}} = \lambda_1 - \lambda_6 + \lambda_{13} = 0 & \dfrac{\partial L}{\partial w^e_{r4}} = 0 = 0 \\
\dfrac{\partial L}{\partial w^e_{r1}} = \lambda_3 - \lambda_4 + \lambda_{11} = 0 & \dfrac{\partial L}{\partial w^m_{r3}} = \lambda_2 - \lambda_6 + \lambda_{14} = 0 & \dfrac{\partial L}{\partial w^g_{r5}} = 0 = 0 \\
\dfrac{\partial L}{\partial w^g_{r2}} = 0 = 0 & \dfrac{\partial L}{\partial w^e_{r3}} = 0 = 0 & \dfrac{\partial L}{\partial w^m_5} = \lambda_2 - \lambda_8 + \lambda_{16} = 0 \\
\dfrac{\partial L}{\partial w^m_{r2}} = \lambda_2 - \lambda_5 + \lambda_{12} = 0 & \dfrac{\partial L}{\partial w^g_{r4}} = 0 = 0 & \dfrac{\partial L}{\partial w^e_5} = \lambda_3 - \lambda_8 + \lambda_{17} = 0
\end{array}
\tag{15}
$$

together with the following complementary slackness conditions :

$$
\begin{array}{lll}
\lambda_1(w^g_{r1} + w^g_{r3} - b) = 0 & \lambda_7(w^m_{r4} - B_4) = 0 & \lambda_{13}(-w^g_{r3}) = 0 \\
\lambda_2(w^m_{r1} + w^m_{r2} + w^m_{r3} + w^m_{r4} + w^m_{r5} - b) = 0 & \lambda_8(w^m_{r5} + w^e_{r5} - B_5) = 0 & \lambda_{14}(-w^m_{r3}) = 0 \\
\lambda_3(w^e_{r1} + w^e_{r5} - b) = 0 & \lambda_9(-w^g_{r1}) = 0 & \lambda_{15}(-w^m_{r4}) = 0 \\
\lambda_4(w^g_{r1} + w^m_{r1} + w^e_{r1} - B_1) = 0 & \lambda_{10}(-w^m_{r1}) = 0 & \lambda_{16}(-w^m_{r5}) = 0 \\
\lambda_5(w^m_{r2} - B_2) = 0 & \lambda_{11}(-w^e_{r1}) = 0 & \lambda_{17}(-w^e_{r5}) = 0 \\
\lambda_6(w^g_{r3} + w^m_3 - B_3) = 0 & \lambda_{12}(-w^m_{r2}) = 0 &
\end{array} \tag{16}
$$

$\Rightarrow$ We have 27 equations for 27 unknowns : $(b, w^g_{r1}, w^g_{r3}, w^m_{r1}, w^m_{r2}, w^m_{r3}, w^m_{r4}, w^m_{r5}, w^e_{r1}, w^e_{r5}, \{\lambda_i, \forall i = 1...17\})$. Then this system is in theory now resolvable. Final calculations are left to the reader

A.1.3 Solving the maximisation problem without weight-positivity constraints

Another approach is to solve the optimisation problem just considering constraints from (8) and (10) and not (12). If solutions are found, we then discard solutions disrespecting constraints of (12) (i.e. solutions with negative weights).

So we define the following Lagrangian :

$$
\begin{aligned}
L(b, w^g_{r1}, w^g_{r3}, w^m_{r1}, w^m_{r2}, w^m_{r3}, w^m_{r4}, w^m_{r5}, w^e_{r1}, w^e_{r5}) = b &- \lambda_1(b - w^g_{r1} - w^g_{r3}) - \lambda_2(b - w^m_{r1} - w^m_{r2} - w^m_{r3} - w^m_{r4} - w^m_{r5}) \\
&- \lambda_3(b - w^e_{r1} - w^e_{r5}) - \lambda_4(w^g_{r1} + w^m_{r1} + w^e_{r1} - B_1) \\
&- \lambda_5(w^m_{r2} - B_2) - \lambda_6(w^g_{r3} + w^m_3 - B_3) \\
&- \lambda_7(w^m_{r4} - B_4) - \lambda_8(w^m_{r5} + w^e_{r5} - B_5)
\end{aligned}
$$

and solve the n first order equality conditions :

$$
\begin{array}{lll}
\dfrac{\partial L}{\partial b} = 1 - \lambda_1 - \lambda_2 - \lambda_3 = 0 & & \\
\dfrac{\partial L}{\partial w^g_{r1}} = \lambda_1 - \lambda_4 = 0 & \dfrac{\partial L}{\partial w^e_{r2}} = 0 = 0 & \dfrac{\partial L}{\partial w^m_{r4}} = \lambda_2 - \lambda_7 = 0 \\
\dfrac{\partial L}{\partial w^m_{r1}} = \lambda_2 - \lambda_4 = 0 & \dfrac{\partial L}{\partial w^g_{r3}} = \lambda_1 - \lambda_6 = 0 & \dfrac{\partial L}{\partial w^e_{r4}} = 0 = 0 \\
\dfrac{\partial L}{\partial w^e_{r1}} = \lambda_3 - \lambda_4 = 0 & \dfrac{\partial L}{\partial w^m_{r3}} = \lambda_2 - \lambda_6 = 0 & \dfrac{\partial L}{\partial w^g_{r5}} = 0 = 0 \\
\dfrac{\partial L}{\partial w^g_{r2}} = 0 = 0 & \dfrac{\partial L}{\partial w^e_{r3}} = 0 = 0 & \dfrac{\partial L}{\partial w^m_5} = \lambda_2 - \lambda_8 = 0 \\
\dfrac{\partial L}{\partial w^m_{r2}} = \lambda_2 - \lambda_5 = 0 & \dfrac{\partial L}{\partial w^g_{r4}} = 0 = 0 & \dfrac{\partial L}{\partial w^e_5} = \lambda_3 - \lambda_8 = 0
\end{array} \tag{17}
$$

together with the following complementary slackness conditions :

$\Rightarrow$ Now, we have 18 equations for 18 unknowns :

$(b, w^g_{r1}, w^g_{r3}, w^m_{r1}, w^m_{r2}, w^m_{r3}, w^m_{r4}, w^m_{r5}, w^e_{r1}, w^e_{r5}, \{\lambda_i, \forall i = 1...8\})$.
This system is a little bit easier than precedent because we have less variables. We juste have to resolve it and verify that it satisfies the conditions $\lambda_i \geq 0$ for i = 1,...8, the positivity-weight constraints (12) and inequalities from (8) and (10). Final calculations are left to the reader.

A.1.4 Using linear equations instead of maximisation problem (from dirspector.txt[7] section 3.8.3)

We are not constrained to resolve the previous optimisation problem to find the minimum positional bandwidth. Indeed, we can use linear equations as in Vanilla Tor[7] :

Let :

- **G** be the total bandwidth for Guard-flagged nodes.
- **M** be the total bandwidth for non-flagged nodes.
- **E** be the total bandwidth for Exit-flagged nodes.
- **D** be the total bandwidth for Guard+Exit-flagged nodes.
- **T** = G + M + E + D

Let :

- W_{gd} be the weight for choosing a Guard+Exit for the guard position
- W_{md} be the weight for choosing a Guard+Exit for the middle position
- W_{ed} be the weight for choosing a Guard+Exit for the exit position
- W_{me} be the weight for choosing an Exit for the middle position
- W_{mg} be the weight for choosing a Guard for the middle position
- W_{gg} be the weight for choosing a Guard for the guard position
- W_{ee} be the weight for choosing an Exit for the exit position

We can balance the network using the following equations :

$$\begin{aligned} W_{gg} * G + W_{dd} * D &= M + W_{md} * D + W_{me} * E + W_{mg} * G \\ W_{gg} * G + W_{dd} * D &= W_{ee} * E + W_{ed} * D \\ W_{ed} * D + W_{md} * D + W_{gd} * D &= D \\ W_{mg} * G + W_{gg} * G &= G \\ W_{me} * E + W_{ee} * E &= E \end{aligned}$$

The two first equations imply that guard bandwidth = middle bandwidth = exit bandwidth. The three last equations verify that weights are consistent. We must have two more equations to be able to solve the previous system. Those two constraints depend from the state of network load.

CASE 1 : $E \geq \frac{T}{3}$ **and** $G \geq \frac{T}{3}$

In this case, nor exit nor guard is scarce. The two additional constraints are :

$$W_{mg} = W_{md}$$
$$W_{ed} = \frac{1}{3}$$

So we find the following solution :

$$W_{gd} = \frac{1}{3}$$
$$W_{ed} = \frac{1}{3}$$
$$W_{md} = \frac{1}{3}$$
$$W_{ee} = \frac{E + G + M}{3E}$$
$$W_{me} = 1 - W_{ee}$$
$$W_{mg} = \frac{2G - E - M}{3G}$$
$$W_{gg} = 1 - W_{mg}$$

CASE 2 : $E < \frac{T}{3}$ **and** $G < \frac{T}{3}$

Let :

- R denote the more scarce class between G and E
- S denote the less scarce class

SUBCASE A : $R + D < S$

In this case, we devote all D bandwidth to S

So we find the following solution :

$$W_{gg} = W_{ee} = 1$$
$$W_{mg} = W_{me} = W_{md} = 0$$

If $E < G$:

$$W_{ed} = 1$$
$$W_{gd} = 0$$

else :

$$W_{ed} = 0$$
$$W_{gd} = 1$$

SUBCASE B : $R + D \geq S$

In this subcase, if $M \leq \frac{T}{3}$, we have enough bandwidth to try to achieve a balancing condition.

We maximize bandwidth in guard position while still allowing exits to be used as middle nodes by adding constraints :

$$W_{gg} = 1$$
$$W_{md} = W_{gd}$$

So we find the following solution :

$$W_{gd} = \frac{1 - W_{ed}}{2}$$
$$W_{ed} = \frac{D - 2E + 4G - 2M}{3D}$$
$$W_{md} = \frac{1 - W_{ed}}{2}$$
$$W_{ee} = \frac{E - G + M}{E}$$
$$W_{me} = \frac{G - M}{E}$$
$$W_{mg} = 0$$
$$W_{gg} = 1$$

If we find values out of range (negative, or above 1), we add following constraints (since both those positions are scarce) :

$$W_{gg} = 1$$
$$W_{ee} = 1$$

So we find the following solution :

$$
\begin{aligned}
W_{gd} &= 1 - W_{ed} - W_{md} \\
W_{ed} &= \frac{D - 2E + G + M}{3D} \\
W_{md} &= \frac{D - 2M + G + E}{3D} \\
W_{ee} &= 1 \\
W_{me} &= 0 \\
W_{mg} &= 0 \\
W_{gg} &= 1
\end{aligned}
$$

If $M > \frac{M}{3}$, then the W_{md} weight above will become negative. We set it to zero and we have in this case :

$$
\begin{aligned}
W_{gd} &= 1 - W_{ed} \\
W_{ed} &= \frac{D - 2E + G + M}{3D} \\
W_{md} &= 0 \\
W_{ee} &= 1 \\
W_{me} &= 0 \\
W_{mg} &= 0 \\
W_{gg} &= 1
\end{aligned}
$$

CASE 3 : $E < \frac{T}{3}$ **or** $G < \frac{T}{3}$

Let :

- S denote the less scarce class between G and E

SUBCASE A : $S + D < \frac{T}{3}$

If S=G, we have (if E is more scarce than M, keep its bandwidth in place) :

$$
\begin{aligned}
W_{gg} = W_{gd} &= 1 \\
W_{md} = W_{ed} = W_{mg} &= 0 \\
\text{If}(E < M) : W_{me} &= 0 \\
\text{else} : W_{me} &= \frac{G - M}{2G} \\
W_{ee} &= 1 - W_{me}
\end{aligned}
$$

If S=E, we have (if G is more scarce than M, keep its bandwidth in place) :

$$\begin{aligned} W_{ee} &= W_{ed} = 1 \\ W_{md} &= W_{gd} = W_{me} = 0 \\ \text{If}(G < M) : W_{mg} &= 0 \\ \text{else} : W_{mg} &= \frac{G - M}{2G} \\ W_{gg} &= 1 - W_{mg} \end{aligned}$$

SUBCASE B : $S + D \geq \frac{T}{3}$

If $S = G$, we add the following constraints to maximize bandwidth in the guard position while still allowing exits to be used as middle nodes :

$$\begin{aligned} W_{gg} &= 1 \\ W_{md} &= W_{ed} \end{aligned}$$

So we find the following solution :

$$\begin{aligned} W_{gg} &= 1 \\ W_{gd} &= \frac{D - 2G + E + M}{3D} \\ W_{mg} &= 0 \\ W_{ee} &= \frac{E + M}{2E} \\ W_{me} &= 1 - W_{ee} \\ W_{md} &= \frac{1 - W_{gd}}{2} \\ W_{ed} &= \frac{1 - W_{gd}}{2} \end{aligned}$$

If $S = E$, we add the following constraints to maximize bandwidth in the exit position :

$$\begin{aligned} W_{ee} &= 1 \\ W_{md} &= W_{gd} \end{aligned}$$

So we find the following solution :

$$
\begin{aligned}
W_{ee} &= 1 \\
W_{ed} &= \frac{D - 2E + G + M}{3D} \\
W_{me} &= 0 \\
W_{gg} &= \frac{G + M}{2G} \\
W_{mg} &= 1 - W_{gg} \\
W_{md} &= \frac{1 - W_{ed}}{2} \\
W_{gd} &= \frac{1 - W_{ed}}{2}
\end{aligned}
$$

A.1.5 Example using linear equations

If we consider the following network :

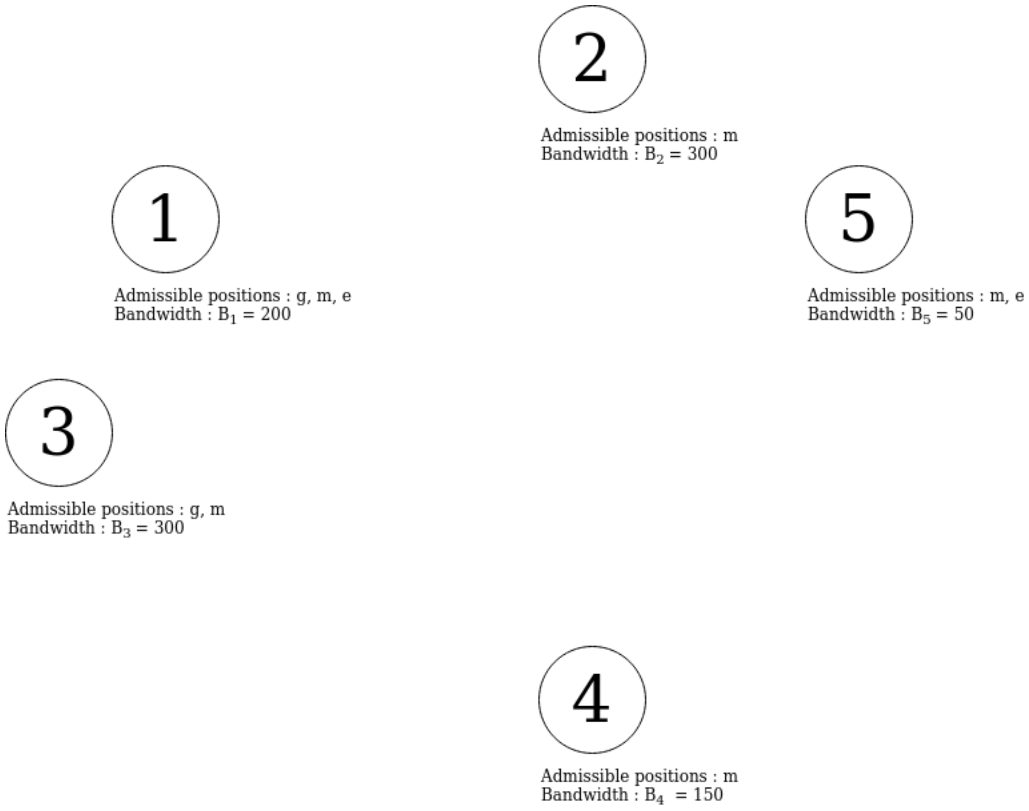

Figure 37: Example of Tor's network with 5 relays

We have that :

- $G = B_3 = 300$
- $M = B_2 + B_4 = 450$

- $E = B_5 = 50$
- $D = B_1 = 200$
- $T = G + M + E + D = 1000$

We are in the CASE 2 ($E = 50 \leq \frac{T}{3} = \frac{1000}{3}$ and $G = 300 \leq \frac{T}{3} = \frac{1000}{3}$). We can define :

- R = E = 50 (because $E = 50 < G = 300$)
- S = G = 300
- R + D = 50 + 200 = 250

So we are more precisely in CASE 2 SUBCASE B (because $R + D = 250 \leq S = 300$), then the solution to balance is :

$$W_{gg} = W_{ee} = 1$$
$$W_{mg} = W_{me} = W_{md} = 0$$

and because $E = 50 < G = 300$:

$$W_{ed} = 1$$
$$W_{gd} = 0$$

The solution shows that we use all our guard-flagged bandwidth for guard position, all our exit-flagged bandwidth for exit position and all our guard+exit-flagged bandwidth for exit position. It makes sense due to the very poor bandwidth available in B_5

Now we have :

- $G = W_{gg} * G + W_{gd} * D = 300$
- $M = M + W_{mg} * G + W_{me} * E + W_{md} * D = 450$
- $E = W_{ee} * E + W_{ed} * D = 250$

$\Rightarrow$ The minimal positional bandwidth is given by E = 250. Then we have $\beta = 250$ (what we must find with the LP1 problem). It is indeed easier with linear equations.

A.2 LP2

The linear program LP2 is solved to determine the weights in position π_i. It outputs a weight $w_{lr}^{\pi_i}$ for each $l \in \mathcal{L}_{\pi_\rangle}$ and $r \in \mathcal{R}$, which is the weight used by a client in location l for relay r. A instance of LP2 is solved for each circuit position in $\mathcal{P}$ in the order they appear in π. Here is this LP2 :

$$\textbf{Minimize} \qquad \sum_{l \in \mathcal{L}_{\pi_i}} \sum_{r \in \mathcal{R}_{\pi_i}} D_l^{\pi_i} P_{lr}^{\pi_i} w_{lr}^{\pi_i} \tag{18}$$

Subject to

$$\forall j \geq i, l \in \mathcal{L}_{\pi_i} : \qquad \sum_{r \in \mathcal{R}} w_{lr}^{\pi_j} = \beta \tag{19}$$

$$\forall j \geq i, l \in \mathcal{L}_{\pi_i}, r \in \mathcal{R} : \qquad \frac{w_{lr}^{\pi_j}}{\beta} \leq \frac{\theta_{\pi_j} v_r^{\pi_j}}{\sum\limits_{s \in \mathcal{R}} v_s^{\pi_j}} \tag{20}$$

$$\forall l \in \mathcal{L}_{\pi_1} : \qquad \sum_{l' \in \mathcal{L}_{\pi_i}} \lambda_{ll'} \sum_{r \in \mathcal{R}} \frac{w_{l'r}^{\pi_i} P_{l'r}^{\pi_i}}{\beta} \leq V_l^{\pi_i} \tag{21}$$

$$\forall r \in \mathcal{R} : \qquad \sum_{j<i} \omega_r^{\pi_j} + \sum_{j \geq i} \sum_{l \in \mathcal{L}_{\pi_i}} \delta_l^{\pi_i} w_{lr}^{\pi_j} \leq B_r \tag{22}$$

$$\forall j \geq i, l \in \mathcal{L}_{\pi_i}, r \in \mathcal{R} : \qquad w_{lr}^{\pi_j} \geq 0 \tag{23}$$

$$\forall j \geq i, r \in \mathcal{R} \backslash \mathcal{R}_{\pi_j} : \qquad w_r^p = 0 \tag{24}$$

Figure 38: Linear program LP2

We also define :

- $D_l \geq 0$ the client density in location $l \in \mathcal{L}$ with $\sum_l D_l = 1$ (Here, $D_{AS1} = \frac{2}{3}, D_{AS2} = \frac{1}{3}$)
- $\theta_p \geq 1$ the maximum factor by which any relay's selection probability in position p can increase over Vanilla tor.
- $P_{lr}^{\pi_i}$ the penalty for choosing relay r in position π_i from $l \in \mathcal{L}_{\pi_i}$.
- $\Lambda(l, r) : \mathcal{L}_{\pi_i} \times \mathcal{R} \to \mathcal{L}_{\pi_{i+1}}$ the map from the current positional location l to the next one after choosing r for the ith position.
- $\lambda_{ll'} = \sum\limits_{l'' \in \mathcal{L}_{\pi_{i-1}}} \lambda_{ll''} \sum\limits_{r \in \mathcal{R} : \Lambda(l'', r) = l'} \frac{w_{l''r}^{\pi_i}}{\beta}$ the probability that a client in $l \in \mathcal{L}_{\pi_j}$ chooses relays for the jth to (i-1)st positions that yield positional location l' $\in \mathcal{L}_{\pi_i}$ (for i=j, $\lambda_{ll'} = 1$ if $l = l'$ and $\lambda_{ll'} = 0$ otherwise).
- v_r^p the Vanilla weight for relay r in position p
- $V_l^{\pi_i}$ the expected penalty in π_i from positional location $l \in \mathcal{L}_{\pi_j}, j \leq i$, when using the vanilla weights to choose the jth to ith positions in π order.

- $\delta_l^{\pi_i} = \sum_{l' \in \mathcal{L}_{\pi_{i-1}}} \sum_{r \in \mathcal{R}: \Lambda(l', r) = l} \frac{\lambda_{l'}^{\pi_{i-1}} w_{l'r}^{\pi_{i-1}}}{\beta}$ the client density of positional location $l \in \mathcal{L}_{\pi_i}$ (given weights $w_{lr}^{\pi_{i-1}}$ for the (i-1)st position obtained from LP2.
- $\omega_r^{\pi_i} = \sum_{l \in \mathcal{L}_{\pi_i}} \delta_l^{\pi_i} w_{lr}^{\pi_i}$ the sum of the weights of relay r in position π_i weighted by the positional location densities.

Let's consider the previous network with three clients located in two ASes :

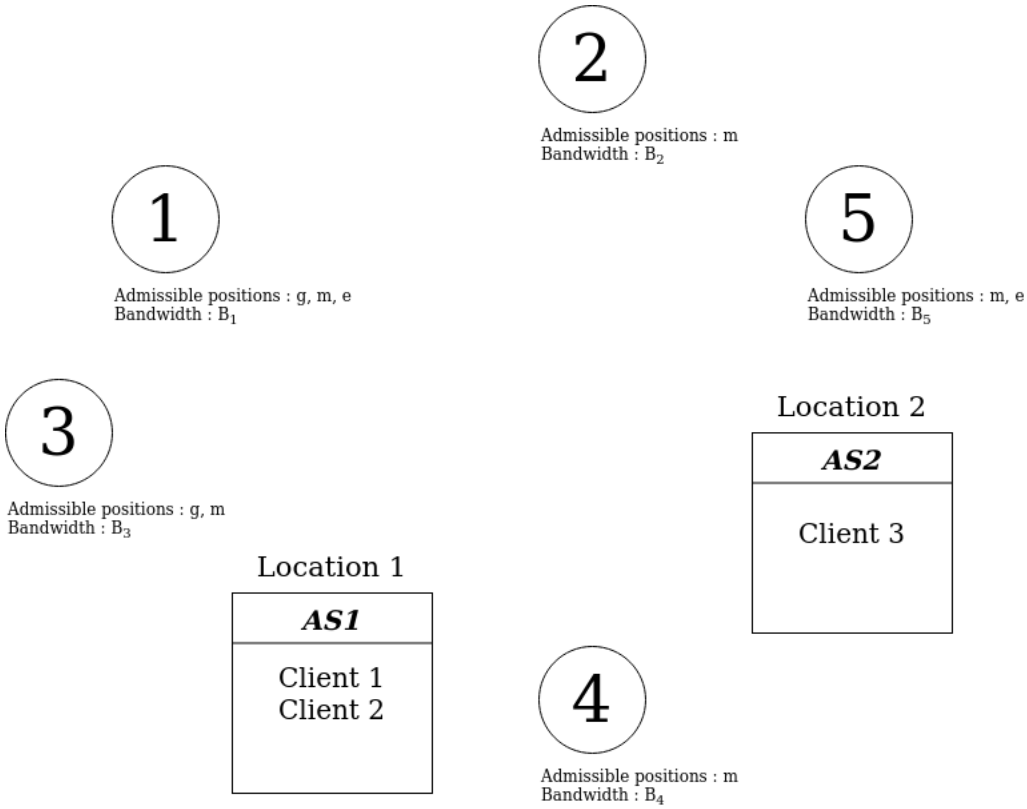

Figure 39: Example of Tor's network with 5 relays and 2 locations (Ases) containing 3 clients

A.2.1 Expression of conditions

Let $\pi = \{g, m, e\}$ and solve the first instance of LP2 for $\pi_1 = \text{g}$ (we must solve the three instance in the same order positions appear in π). To formulate weights, let $\mathcal{L}_{\pi_i}$ be the set of positional locations for the ith position (can bet different element for each combination of previous client location and (i-1)relays sequence. In our network (see Fig. 4), we have :

- $\mathcal{L}_{\pi_1} = \mathcal{L} = \{AS1, AS2\}$
- $\mathcal{L}_{\pi_2} = \{(AS1, r_1), (AS1, r_3), (AS2, r_1), (AS2, r_3)\}$
- $\mathcal{L}_{\pi_3} = \{(AS1, r_1, r_2), (AS1, r_1, r_3), (AS1, r_1, r_4), (AS1, r_1, r_5), (AS1, r_3, r_1), (AS1, r_3, r_2), (AS1, r_3, r_4), (AS2, r_1, r_2), (AS2, r_1, r_3), (AS2, r_1, r_4), (AS2, r_1, r_5), (AS2, r_3, r_1), (AS2, r_3, r_2), (AS2, r_3, r_4), (A$

Equation (24) sets weights to zero when inadmissible in the position. We find :

$$\begin{aligned} w^g_{r2} =& w^e_{r2} = 0 \\ & w^e_{r3} = 0 \\ w^g_{r4} =& w^e_{r4} = 0 \\ w^g_{r5} & = 0 \end{aligned} \tag{25}$$

and then for π_1 we have :

$$\begin{aligned} w^g_{AS1,r2} = w^g_{AS2,r2} =& w^e_{AS1,r2} = w^e_{AS2,r2} = 0 \\ & w^e_{AS1,r3} = w^e_{AS1,r3} = 0 \\ w^g_{AS1,r4} = w^g_{AS2,r4} =& w^e_{AS1,r4} = w^e_{AS2r4} = 0 \\ w^g_{AS1,r5} = w^g_{AS2,r5} & = 0 \end{aligned} \tag{26}$$

Equation (23) requires weights to be non-negative. We'll check out this condition after the optimisation by choosing the appropriate solution (as for LP1 previously). We have (using (26)) :

$$\begin{array}{lllll} w^g_{r1} \geq 0 & & w^g_{r3} \geq 0 & & \\ w^m_{r1} \geq 0 & w^m_{r2} \geq 0 & w^m_{r3} \geq 0 & w^m_{r4} \geq 0 & w^m_{r5} \geq 0 \\ w^e_{r1} \geq 0 & & & & w^e_{r5} \geq 0 \end{array} \tag{27}$$

and then for π_1 we have :

$$\begin{array}{lllllllll} w^g_{AS1,r1} \geq 0 & w^g_{AS2,r1} \geq 0 & w^m_{AS1,r2} \geq 0 & w^g_{AS1,r3} \geq 0 & w^g_{AS2,r3} \geq 0 & w^m_{AS1,r4} \geq 0 & & & \\ w^m_{AS1,r1} \geq 0 & w^m_{AS2,r1} \geq 0 & w^m_{AS2,r2} \geq 0 & w^m_{AS1,r3} \geq 0 & w^m_{AS2,r3} \geq 0 & w^m_{AS2,r4} \geq 0 & w^m_{AS1,r5} \geq 0 & w^m_{AS2,r5} & \\ w^e_{AS1,r1} \geq 0 & w^e_{AS2,r1} \geq 0 & & & & & w^e_{AS1,r5} \geq 0 & w^e_{AS2,r5} & \end{array} \tag{28}$$

Equation (19) sets weight in each position to value β obtained from LP1. In our case, we have :

$$\begin{aligned}
l \in \mathcal{L}_{\pi_1} \text{and i} = 1: \quad & w^g_{(AS1),r_1} + w^g_{(AS1),r_2} + w^g_{(AS1),r_3} + w^g_{(AS1),r_4} + w^g_{(AS1),r_5} + \\
& w^g_{(AS2),r_1} + w^g_{(AS2),r_2} + w^g_{(AS2),r_3} + w^g_{(AS1),r_4} + w^g_{(AS2),r_5} = \beta \\
l \in \mathcal{L}_{\pi_1} \text{and i} = 2: \quad & w^m_{(AS1),r_1} + w^m_{(AS1),r_2} + w^m_{(AS1),r_3} + w^m_{(AS1),r_4} + w^m_{(AS1),r_5} + \\
& w^m_{(AS2),r_1} + w^m_{(AS2),r_2} + w^m_{(AS2),r_3} + w^m_{(AS1),r_4} + w^m_{(AS2),r_5} = \beta \\
l \in \mathcal{L}_{\pi_1} \text{and i} = 3: \quad & w^e_{(AS1),r_1} + w^e_{(AS1),r_2} + w^e_{(AS1),r_3} + w^e_{(AS1),r_4} + w^e_{(AS1),r_5} + \\
& w^e_{(AS2),r_1} + w^e_{(AS2),r_2} + w^e_{(AS2),r_3} + w^e_{(AS1),r_4} + w^e_{(AS2),r_5} = \beta
\end{aligned} \tag{29}$$

Using results of (26), equations of (29) are simplified :

$$\begin{aligned}
l \in \mathcal{L}_{\pi_1} \text{and i} = 1: \quad & w^g_{(AS1),r_1} + w^g_{(AS1),r_3} + w^g_{(AS2),r_1} + w^g_{(AS1),r_3} = \beta \\
l \in \mathcal{L}_{\pi_1} \text{and i} = 2: \quad & w^m_{(AS1),r_1} + w^m_{(AS1),r_2} + w^m_{(AS1),r_3} + w^m_{(AS1),r_4} + w^m_{(AS1),r_5} + \\
& w^m_{(AS2),r_1} + w^m_{(AS2),r_2} + w^m_{(AS2),r_3} + w^m_{(AS1),r_4} + w^m_{(AS2),r_5} = \beta \\
l \in \mathcal{L}_{\pi_1} \text{and i} = 3: \quad & w^e_{(AS1),r_1} + w^e_{(AS1),r_5} + w^e_{(AS2),r_1} + w^e_{(AS1),r_5} = \beta
\end{aligned}$$

Equation (20) limits relay-placement advantage to θ_{π_i}. In our case, we have :

$$\begin{aligned}
& w^g_{(AS1),r_1} \le \beta \frac{\theta_g v^g_{r_1}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS1),r_1} \le \beta \frac{\theta_m v^m_{r_1}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS1),r_1} \le \beta \frac{\theta_m v^m_{r_1}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS1),r_2} \le \beta \frac{\theta_g v^g_{r_2}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS1),r_2} \le \beta \frac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS1),r_2} \le \beta \frac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS1),r_3} \le \beta \frac{\theta_g v^g_{r_3}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS1),r_3} \le \beta \frac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS1),r_3} \le \beta \frac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS1),r_4} \le \beta \frac{\theta_g v^g_{r_4}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS1),r_4} \le \beta \frac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS1),r_4} \le \beta \frac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS1),r_5} \le \beta \frac{\theta_g v^g_{r_5}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS1),r_5} \le \beta \frac{\theta_m v^m_{r_5}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS1),r_5} \le \beta \frac{\theta_m v^m_{r_5}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS2),r_1} \le \beta \frac{\theta_g v^g_{r_1}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS2),r_1} \le \beta \frac{\theta_m v^m_{r_1}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS2),r_1} \le \beta \frac{\theta_m v^m_{r_1}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS2),r_2} \le \beta \frac{\theta_g v^g_{r_2}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS2),r_2} \le \beta \frac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS2),r_2} \le \beta \frac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS2),r_3} \le \beta \frac{\theta_g v^g_{r_3}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS2),r_3} \le \beta \frac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS2),r_3} \le \beta \frac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS2),r_4} \le \beta \frac{\theta_g v^g_{r_4}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS2),r_4} \le \beta \frac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS2),r_4} \le \beta \frac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} \\
& w^g_{(AS2),r_5} \le \beta \frac{\theta_g v^g_{r_5}}{\sum_{s\in\mathcal{R}} v^g_s} \qquad w^m_{(AS2),r_5} \le \beta \frac{\theta_m v^m_{r_5}}{\sum_{s\in\mathcal{R}} v^m_s} \qquad w^m_{(AS2),r_5} \le \beta \frac{\theta_m v^m_{r_5}}{\sum_{s\in\mathcal{R}} v^m_s}
\end{aligned} \tag{30}$$

Using results of (26), equations of (30) are simplified :

$$
\begin{array}{lll}
 & w^m_{(AS1),r_2} \le \beta \dfrac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} & \\
w^g_{(AS1),r_1} \le \beta \dfrac{\theta_g v^g_{r_1}}{\sum_{s\in\mathcal{R}} v^g_s} & w^m_{(AS1),r_3} \le \beta \dfrac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} & w^e_{(AS1),r_2} \le \beta \dfrac{\theta_e v^e_{r_1}}{\sum_{s\in\mathcal{R}} v^e_s} \\
w^g_{(AS1),r_2} \le \beta \dfrac{\theta_g v^g_{r_2}}{\sum_{s\in\mathcal{R}} v^g_s} & w^m_{(AS1),r_4} \le \beta \dfrac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} & w^e_{(AS1),r_5} \le \beta \dfrac{\theta_e v^e_{r_5}}{\sum_{s\in\mathcal{R}} v^e_s} \\
w^g_{(AS2),r_1} \le \beta \dfrac{\theta_g v^g_{r_1}}{\sum_{s\in\mathcal{R}} v^g_s} & w^m_{(AS2),r_2} \le \beta \dfrac{\theta_m v^m_{r_2}}{\sum_{s\in\mathcal{R}} v^m_s} & w^e_{(AS2),r_2} \le \beta \dfrac{\theta_e v^e_{r_1}}{\sum_{s\in\mathcal{R}} v^e_s} \\
w^g_{(AS2),r_2} \le \beta \dfrac{\theta_g v^g_{r_2}}{\sum_{s\in\mathcal{R}} v^g_s} & w^m_{(AS2),r_3} \le \beta \dfrac{\theta_m v^m_{r_3}}{\sum_{s\in\mathcal{R}} v^m_s} & w^e_{(AS2),r_5} \le \beta \dfrac{\theta_e v^e_{r_5}}{\sum_{s\in\mathcal{R}} v^e_s} \\
 & w^m_{(AS2),r_4} \le \beta \dfrac{\theta_m v^m_{r_4}}{\sum_{s\in\mathcal{R}} v^m_s} &
\end{array}
\tag{31}
$$

Equation (21) limit per-location average penalty to that of Vanilla Tor :

$$
\begin{aligned}
l = AS1 &: \frac{w^g_{AS1,r_1} P^g_{AS1,r_1} + w^g_{AS1,r_2} P^g_{AS1,r_2} + w^g_{AS1,r_3} P^g_{AS1,r_3} + w^g_{AS1,r_4} P^g_{AS1,r_4} + w^g_{AS1,r_5} P^g_{AS1,r_5}}{\beta} < V^g_{AS1} \\
l = AS2 &: \frac{w^g_{AS2,r_1} P^g_{AS2,r_1} + w^g_{AS2,r_2} P^g_{AS2,r_2} + w^g_{AS2,r_3} P^g_{AS2,r_3} + w^g_{AS2,r_4} P^g_{AS2,r_4} + w^g_{AS2,r_5} P^g_{AS2,r_5}}{\beta} < V^g_{AS2}
\end{aligned}
\tag{32}
$$

Using results of (26), equations of (32) are simplified :

$$
\begin{aligned}
l = AS1 &: \frac{w^g_{AS1,r_1} P^g_{AS1,r_1} + w^g_{AS1,r_3} P^g_{AS1,r_3}}{\beta} < V^g_{AS1} \\
l = AS2 &: \frac{w^g_{AS2,r_1} P^g_{AS2,r_1} + w^g_{AS2,r_3} P^g_{AS2,r_3}}{\beta} < V^g_{AS2}
\end{aligned}
\tag{33}
$$

Equation (22) limits relay's total client weight to its bandwidth :

$$
\begin{aligned}
&\delta^g_{AS1} w^g_{AS1,r1} + \delta^g_{AS2} w^g_{AS2,r1} + \delta^m_{AS1} w^m_{AS1,r1} + \delta^m_{AS2} w^m_{AS2,r1} + \delta^e_{AS1} w^e_{AS1,r1} + \delta^e_{AS2} w^e_{AS2,r1} \le B_{r1} \\
&\delta^g_{AS1} w^g_{AS1,r2} + \delta^g_{AS2} w^g_{AS2,r2} + \delta^m_{AS1} w^m_{AS1,r2} + \delta^m_{AS2} w^m_{AS2,r2} + \delta^e_{AS1} w^e_{AS1,r2} + \delta^e_{AS2} w^e_{AS2,r2} \le B_{r2} \\
&\delta^g_{AS1} w^g_{AS1,r3} + \delta^g_{AS2} w^g_{AS2,r3} + \delta^m_{AS1} w^m_{AS1,r3} + \delta^m_{AS2} w^m_{AS2,r3} + \delta^e_{AS1} w^e_{AS1,r3} + \delta^e_{AS2} w^e_{AS2,r3} \le B_{r3} \\
&\delta^g_{AS1} w^g_{AS1,r4} + \delta^g_{AS2} w^g_{AS2,r4} + \delta^m_{AS1} w^m_{AS1,r4} + \delta^m_{AS2} w^m_{AS2,r4} + \delta^e_{AS1} w^e_{AS1,r4} + \delta^e_{AS2} w^e_{AS2,r4} \le B_{r4} \\
&\delta^g_{AS1} w^g_{AS1,r5} + \delta^g_{AS2} w^g_{AS2,r5} + \delta^m_{AS1} w^m_{AS1,r5} + \delta^m_{AS2} w^m_{AS2,r5} + \delta^e_{AS1} w^e_{AS1,r5} + \delta^e_{AS2} w^e_{AS2,r5} \le B_{r5}
\end{aligned}
\tag{34}
$$

Using results of (26), equations of (34) are simplified :

$$
\begin{aligned}
&\delta^g_{AS1} w^g_{AS1,r1} + \delta^g_{AS2} w^g_{AS2,r1} + \leq B_{r1} \\
&\delta^m_{AS1} w^m_{AS1,r2} + \delta^e_{AS1} w^e_{AS1,r2} + \delta^m_{AS2} w^m_{AS2,r2} + \delta^e_{AS2} w^e_{AS2,r2} \leq B_{r2} \\
&\delta^g_{AS1} w^g_{AS1,r3} + \delta^m_{AS1} w^m_{AS1,r3} + \delta^g_{AS2} w^g_{AS2,r3} + \delta^m_{AS2} w^m_{AS2,r3} \leq B_{r3} \\
&\delta^m_{AS1} w^m_{AS1,r4} + \delta^m_{AS2} w^m_{AS2,r4} \leq B_{r4} \\
&\delta^e_{AS1} w^e_{AS1,r5} + \delta^e_{AS2} w^e_{AS2,r5} \leq B_{r5}
\end{aligned}
\tag{35}
$$

A.2.2 Solving the minimisation problem

Now we can solve the optimisation problem as in section 1.1.2 (see *Appendix B*). The only difference is that we have a minimisation problem, then we must rewrite our constraint for our Lagrangian in the form "condition ≥ 0" instead of "condition ≤ 0".

The procedure is exactly the same for position $\pi_2 = m$ and $\pi_3 = e$.

B Theory

Optimisation with inequality constraints

From [16], we remember the procedure to resolve an optimisation problem.

To maximise f($\mathbf{x}$) with $\mathbf{x} = (x_1, ..., x_n)$ subject to :

$$g_1(\mathbf{x}) \leq b_1, \ \ ..., \ \ g_n(\mathbf{x}) \leq b_n, \quad i = 1, ..., k$$

we define the Lagrangian

$$L = f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - ... - \lambda_n g_n(\mathbf{x})$$

and solve the n first order equality conditions

$$\frac{\partial L}{\partial x_1} = 0, \ \ ..., \ \ \frac{\partial L}{\partial x_n} = 0$$

together with the k complementary slackness conditions

$$\lambda_1(g_1(\mathbf{x}) - b_1) = 0, \ \ ..., \ \ \lambda_n(g_n(\mathbf{x}) - b_n) = 0$$

These are $n + k$ equations for the $n + k$ unknowns $(x_1, \cdots, x_n, \lambda_1, \cdots, \lambda_k)$. However, only solutions which satisfy the following 2k inequalities are admissible :

$$\lambda_1 \geq 0, \ \ ..., \ \ \lambda_k \geq 0 \quad \text{and} \quad g_1(\mathbf{x}) \leq b_1, \ \ ..., \ \ g_k(\mathbf{x}) \leq b_k$$

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**