# Systems Privacy Project 2: The Safer Path

| | |
|---|---|
| Professor | Kevin Gallagher |
| Week | 9 |
| Due | 4 December, 2025 at 23:59 |
| Version | 1.5 |

## Project Description

A lot of Tor's security depends on its path selection. If multiple nodes in a circuit are owned by an adversary, the probability of deanonymization goes up significantly. This, combined with continued and growing surveillance by nation states on their people (for example, the NSA spying leaked by Edward Snowden) paints a concerning picture: if an entire circuit is in a country, that country can attempt to use legal force to deanonymize a client. However, the current implementation of the Tor path selection algorithm only avoids /16 subnets from appearing in the same circuit, rather than country-level adversaries. While a start, this may not be enough for security.

On the other hand, too many filters or restrictions on the path selection algorithm may cause the lack of possible circuits, or undue load on smaller nodes, potentially harming usability of the network. Therefore, instead of creating a filter-based solution, it may be better to improve the Tor path selection algorithm by probabilistically avoiding nodes in the same country, instead of using a hard filter.

In this project you will be implementing a new Tor path selection algorithm that uses a new weighting mechanism to probabilistically avoid nodes in the same country. It relies on several new parameters, which will be described in the algorithm section below. Though you can do this project in any language you wish, the skeleton code for the project will be provided in Java, as usual.

This project contains four steps: building a simple consensus parser, implementing the base Tor path selection algorithm, implementing the new Tor path selection algorithm, and simulating circuit creation to determine the efficacy of this new solution. Each of these parts is described in detail below.

## Implementing a simple consensus parser

In order to implement a new path selection algorithm in Tor, we first must be able to determine what nodes are in the network. To do this, we will need to parse the Tor consensus documents, reading the information about each node in the document and saving it as a Node object (provided in the template code) within our code.

Remember, the structure of a relay within the Tor consensus is as follows:

```
r <NodeNickname> <Fingerprint> <Digest> <PublicationTime> <IP Address> <ORPort> <DIRPort>
a <IPv6 address>:<port> (this line is optional)
s <Flags> (1 or more)
v <VersionNumber>
pr <SupportedProtocols> (1 or mode, not needed for our project)
w Bandwidth=<Bandwidth>
p <ExitPolicy>
```

As you can see from the lines above and our Node class in the provided Java code discussed in Section , we will not be using some of this information, and therefore will not need to parse it. You will also notice that the country of a relay is not listed in the consensus document. You will need to use some form of Geo-location in order to determine what country the node is in.

By the time you finish this section, you should have a method that will take a consensus document, parse the nodes out of it, geo-locate each node, and then store the nodes in an array in the program. Note that you are

not required to check the validity of any passed consensus document. You only need to parse the relays from the document.

### Skeleton Code

In order to aid in this endeavor, I have provided some skeleton code on Clip in a zip file called Project2.zip. It contains a currently unimplemented main class (Project2.java) as well as two classes in the model folder: Node and Circuit. The Node class contains the following data, as well as their associated setters and getters.

- Nickname
- Fingerprint
- Time Published
- IP Address
- OR Port
- DIR Port
- Flags (String Array)
- Version
- Bandwidth
- Country
- Exit Policy

You may modify this class as you see fit. The Circuit class is currently a lot simpler. It contains the following data, as well as their associated setters and getters.

- ID
- Nodes (Node Array)
- Minimum Bandwidth

You may modify this class as you see fit. Finally, there is one entirely unimplemented class called ConsensusParser in the utils/ConsensusParser.java file. This will need to be implemented as described above. This class contains a single String for the consensus filename, as well as getters and setters, and one method called parseConsensus, which will read the filename and return a list of type Node. Currently it returns null, and must be implemented by you as described above.

Your parser only needs to parse information about Nodes from the consensus. **All lines in the consensus that have nothing to do with nodes can be ignored!**

You can find the current consensus document at one of the links below:

`http://217.196.147.77/tor/status-vote/current/consensus`
`http://171.25.193.9:443/tor/status-vote/current/consensus`
`http://216.218.219.41/tor/status-vote/current/consensus`
`http://45.66.35.11/tor/status-vote/current/consensus`

I tested all of these on Eduroam and they all worked for me.

## Implementing Current Path Selection

After implementing a ConsenusParser, you will be ready to implement Tor's current path selection mechanism. I recommend reviewing the slides for information about this, if needed. The algorithm is described in Algorithm 1.

As you can see, this will imply the implementation of various functions. You may implement these functions wherever seems best; this could be in the Circuit class, in main, or in another class altogether. However you choose to do it, the output should always be an array of three nodes.

## Implementing Geography-Aware Path Selection

While Tor's existing path selection mechanism is decent at avoiding congestion within the network and ensuring that resources are relatively well used, it falls very short of protecting from circuits that exist entirely within one nation. For this reason, we are going to adapt the node selection algorithm to consider nation states. However, if we were to filter out all nodes belonging to a nation state, we may end up very little node diversity in our circuits. For this reason, we will need to take a more subtle approach.

Specifically, we will be modifying the Tor Path Selection algorithm to apply a higher weight to nodes that do not share a country with the nodes already chosen to our circuit, and lower weight to those nodes that do share a country with a node already in our circuit. To do this we will have two security parameters, $\alpha$ and $\beta$, which will

**Algorithm 1** Simplified Version of Tor's Current Path Selection

```
1: procedure SelectPath()
2:     Exit ← SelectExit()
3:     Guard ← SelectGuard(Exit)
4:     Middle ← SelectMiddle(Guard, Exit)
5:
6:     return (Guard, Middle, Exit)
7: end procedure
8: function SelectExit()
9:     Filter relays to those with: Fast flag, & suitable exit policy
10:    Calculate the total bandwidth weight W_t
11:    Weight candidates by bandwidth W_i / W_t
12:
13:    return a relay sampled by weight
14: end function
15: function SelectGuard()
16:    Filter relays to those with: Guard, flag
17:    Filter relays to remove those with the same family and 16 subnet as the exit
18:    Prioritize relays from persistent SAMPLED_GUARDS and CONFIRMED_GUARDS sets
19:    Calculate the total bandwidth weight W_t
20:    Weight candidates by bandwidth W_i / W_t
21:
22:    return a relay sampled by weight
23: end function
24: function SelectMiddle(Guard, Exit)
25:    Filter relays to those with: Fast flag
26:    Filter relays to remove those with the same family and 16 subnet as the exit and guard
27:    Calculate the total bandwidth weight W_t
28:    Weight candidates by bandwidth W_i / W_t
29:
30:    return a relay sampled by weight
31: end function
```

be between 0 and 1. $\alpha$ will be used when selecting a guard, and due to the possibility of end-to-end correlation attacks should be a relatively large parameter. $\beta$ will be used when selecting a middle node, and due to there being less danger inherent in selecting a middle node, can be much smaller.

Our solution will not eliminate weighting by bandwidth. Bandwidth weighting is too useful for performance, and therefore should be kept. However, it will be used to adjust the weights in path selection so that the weights are not based purely on bandwidth, but also on geographical factors. The complete algorithm can be found in Algorithm 2.

In this algorithm we use two security parameters, $\alpha$ and $\beta$ to weigh different parts of path selection differently. For guard selection, if a guard does not share a country with the circuit's exit, the node's weight gets multiplied by $1 + \alpha$, giving it a higher chance to be selected which scales linearly with $\alpha$. For middle node selection, nodes get weighted based on the number of nodes already in the circuit with which they share a country. If a middle node shares a country with two nodes in the circuit, it is multiplied by $1 + \beta$. If it shares a country with one node in the circuit, it is $1 + (2 \times \beta)$. If it share a country with no node in the circuit, it is $1 + (3 \times \beta)$. This will weight node diversity higher, potentially making stronger circuits.

**Please be sure to implement this in a separate function so we can run both algorithms to compare them!**

## Evaluating Our Path Selection

Of course, after we implement our algorithm we must measure its performance. In order to have the best picture of how well it performs, we need to compare the vanilla Tor circuit selection algorithm with our new and improved version. We will use a few metrics to determine this. First, we need a metric to evaluate the security of our solution. Intuitively, our solution can be seen as stronger if the set of nodes that were chosen in our experiments is larger than the set of the control (Tor's current weighted algorithm). Measuring this is simple: we keep a set of the nodes chosen, and compare the cardinality of these sets. In addition to this global metric, we can also keep a similar count for all of the nodes in each position: one set for the guards, one set for the middle nodes, and one set for the exit nodes.

While this metric is interesting, it's also a bit naive. If a node by coincidence is chosen once it alters the size of the set, even if its impact on node diversity is minimal. However, we can measure the *Shannon Entropy* of the country selection both globally and in each set (Guard, Middle, Exit). To calculate the *Shannon Entropy* of a set $X$, we can calculate the following:

$H(X) = -1 \times \sum_{x_i \in X} [p(x_i) \times log_2(p(x_i))]$

In this formula, $p(x_i)$ is the probability of a node being selected, assuming that a node $x_i$ is selected $n_i$ times during the execution of your experiments for a given set (Global, Guard, Middle, or Exit), and the total number of nodes selected globally was $n_t = 3 \times m$ where $m$ is the number of circuits you built during your experiments. Then, the probability $p(x_i)$ for the global set is $\frac{x_i}{n_t}$ while, for each node position set (Guard, Middle, or Exit), it is equal to $fracx_im$. Higher entropy values means more security.

Finally, we can measure our success through the bandwidth of a circuit. Note that for each circuit, the bandwidth of the circuit is limited to the lowest bandwidth among all of the nodes in that circuit. To compare the two, we can graph the bandwidths of both experiments, both our modified algorithm and the control. This will allow us to visually see the difference our algorithm makes in terms of circuit bandwidth.

All of the metrics mentioned above should be reported in the Technical Report. In an ideal world we would also measure latency of our solution. However, this would complicate the project too much, and therefore I will not require it.

## Objectives

By the end of this project, you should have experience with:

- Implementing node selection for Tor paths
- Modifying node selection algorithms for better security

**Algorithm 2** New Version of Tor's Geographically Aware Path Selection

```
1: procedure SelectPath()
2:     Exit ← SelectExit()
3:     Guard ← SelectGuard(Exit, α)
4:     Middle ← SelectMiddle(Guard, Exit, β)
5:
6:     return (Guard, Middle, Exit)
7: end procedure
8: function SelectExit()
9:     Filter relays to those with: Fast flag, & suitable exit policy
10:    W_i is set from the consensus document.
11:    Calculate the total bandwidth weight W_t
12:    Weight candidates by bandwidth W_i / W_t
13:
14:    return a relay sampled by weight
15: end function
16: function SelectGuard()
17:    Filter relays to those with: Guard, flag
18:    Filter relays to remove those with the same family and 16 subnet as the exit
19:    Prioritize relays from persistent SAMPLED_GUARDS and CONFIRMED_GUARDS sets
20:    for Node N_i in the guard sets do
21:        if N_i.country is not in the circuit then
22:            W_i = W_i × (1 + α)
23:        else
24:            W_i is set from the consensus document.
25:        end if
26:    end for
27:    Calculate the total bandwidth weight W_t
28:    Weight candidates by bandwidth W_i / W_t
29:
30:    return a relay sampled by weight
31: end function
32: function SelectMiddle(Guard, Exit)
33:    Filter relays to those with: Fast flag
34:    Filter relays to remove those with the same family and 16 subnet as the exit and guard
35:    for Node N_i in the consensus do
36:        W_i is set from the consensus document.
37:        if N_i.country ≠ Guard.country ≠ Exit.country then
38:            c = 3
39:        else if N_i.country ≠ Guard.country or N_i.country ≠ Exit.country then
40:            c = 2
41:        else
42:            c = 1
43:        end ifW_i = W_i × (1 + (β × c))
44:    end for
45:    Calculate the total bandwidth weight W_t
46:    Weight candidates by bandwidth W_i / W_t
47:
48:    return a relay sampled by weight
49: end function
```

- Running comparative experiments to determine which solution is better.
- Writing a technical report.

### The Technical Report

As mentioned in the previous section, your group must turn in a technical report with your solution. This technical report should be no longer than 3 pages with 12 point Ariel fond. Brevity is a skill, so please prioritize what you want to tell me in the report. Compilation instructions or other information I need to run your code does not count towards this page limit. Citations also do not count towards this page limit. The report should be submitted as a PDF, and should only contain your student numbers, not names. **In order to ensure blind grading, I do not want to see your names on the report.**

Reports can be in English or in Portuguese. The choice of language will not impact your grade.

## Submission

To submit, please send all of your code in a zip, 7z, or tar file to my email address, k.gallagher@fct.unl.pt, with the following subject:

"[PdS 25/26] Project 2 Submission - XXXXX and YYYYY"

where XXXXX and YYYYY are your student numbers. **Again, to ensure blind grading I'd like there to not be any student names in the code or in the email subject. I will be using a script to extract all files and information from the emails while keeping myself blind to the names of the students who sent them.**

## Grading Rubric

To be added soon.

## Frequently Asked Questions

Here is a list of questions I expect will be frequent about the project, along with their respective answers.

1. **Is this exercise graded?**

   Yes. It is a graded project.

2. **Do we do the projects alone or in pairs?**

   I recommend doing the project in pairs.

3. **Where do I turn in the project?**

   See the section immediately above this one.

4. **Can you describe how this code works?**

   I am happy to clarify these doubts during a class or during office hours, or by email. However, this document is already long, and I would basically only be regurgitating information that can be found online.

5. **I really like/want to learn language x. Can I do my lab and project in that language?**

   Conditionally on approval.

6. **I found this bug/problem/issue with the report/code.**

   Skill issue on my end. Please let me know and I'll do my best to update the document and/or code to fix the issue as fast as possible. I'll also update the version number on top and add your name to the acknowledgements section below.

## Acknowledgements

Credit for the base of the LaTeX template goes to Gilles Callebaut. If you find any issue in this document, let me know and I'll add your name here.