Gateway Optimal Orchestration and Deployment

Analysis of current Pocket Network Portal deployment and possible optimal deployment strategies.

Report by:

Ramiro Rodríguez Colmeiro (ramiro@poktscan.com)

Nicolas Aguirre (nicolas@poktscan.com)

Michael ORourke (michael@poktscan.com)

POKTscan Data Science Team

2023/02/08

Disclaimer

This work was founded by Pocket Scan Technologies LLC, a company operating on the Pocket Network.

Contact

Pocket Scan Technologies LLC

1621 Central Ave

Cheyenne, WYOMING (WY) 82001

Contact: Michael ORourke (michael@poktscan.com)

Changelog

v1.0	2023-02-02	First version.

Table of Contents

1 Abstract **4**

2 Introduction **4**

3 Data Source and Structure **5**

3.1 Node-Runner Data 5
3.2 App Users Data 6
3.3 Graph Representation 6

4 Mathematical Modeling **8**

4.1 Heuristic Approach 9
4.2 Linear Programming Approach 11

5 Results **13**

5.1 Optimal Solution 13
5.2 Convergence to Optimal Latency 14
5.3 Minimum Cost Scenario 14

6 Conclusions **16**

Reference List **17**

1 Abstract

This report analyses the current deployment locations of the Pocket Network Portal (PNP) interfaces in the Amazon Web Services ecosystem. Currently the PNPs are deployed in multiple locations, this represents a high cost for both the Pocket Network Inc. and the servicer node operators. In this document we demonstrate that the current deployment strategy is sub-optimal and a better Quality of Service (QoS) can be achieved by reducing the number of portals. More specifically, we show that the average QoS of the network can be reduced from 155 ms to 120 ms by keeping only three PNPs: *ap-southeast-1*, *eu-central-1* and *us-east-1*.

2 Introduction

As of time of this writing (2023/02/08) The Pocket Network is composed of 14 active Pocket Network Portals (PNP). The PNPs are the only available entrance to the Pocket Network and they are controlled by the PNI. Applications that wish to connect to the network must first pass through one of these portals to reach any of the staked nodes, as it is shown in fig. 1.

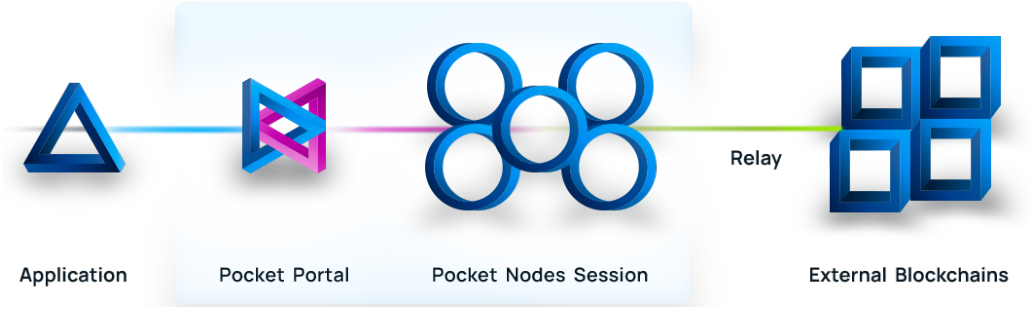

Figure 1: Basic diagram of the Pocket Network relays path. Image source: Pocket Network docs.

Over the last year the servicer nodes operators (or node-runners) deployed their sericer nodes to obtain the lowest latency to the PNPs, as a lower latency results in higher rewards. However, deploying nodes is costly and to maximize their income the node-runners deployed their infrastructure only in regions with high traffic. As the node-runner competence in the Pocket Network growth, the optimization of the servicers nodes became critical, which lead to the creation of modified clients such as the Geo-Mesh [5]. With Geo-Mesh open to the community, the relocation of nodes became much cheaper, and also much faster. Given the stochastic behavior of the Pocket Network traffic volume [3] (which is even more complex when it is analyzed in a per-portal basis), the node-runners began a never ending race for the optimal deployment of servicer nodes. This race add little value to the ecosystem and requires lots of resources from the node-runners to stay on top of the game. The main reasons behind this are the following:

1. Economic pressure on the PNI : PNI has to maintain more PNPs, which means more resources are used for maintaining and infrastructure.

2. More PNPs does not mean more decentralization since all of them are controlled by the PNI.
3. Economic pressure on the node-runners : Node-runners have to deploy to more locations (more infrastructure cots) and spend resources tracking the optimal location (human resources costs).
4. Network cost is higher and difficult to estimate. In the current stage of the Pocket Network and market conditions is important to keep track of the real cost of the network which is fundamental for economic models.

In the following section we will show how to model the PNP deployment problem, find an optimal solution and propose a better strategy to reduce the costs for all the players of the Pocket Network ecosystem. In a few words, the problem is tackled as a linear optimization problem based on a graph structure of the Amazon Web Services (AWS) inner ping times and the node-runners latency to the different PNPs located on specific AWS Availability Zones (AZs).

3 Data Source and Structure

The proposed model requires two kinds of data, first the servicers data, which corresponds to the node-runners latency and then the clients data, which is the latency of the app users to the PNPs. All the gathered data is then modeled as a graph and prepared for further processing.

3.1 Node-Runner Data

The node-runner data was obtained from POKTscan internal databases. The majority of the data is collected from the Cherry Picker (CP) tracker process [1], which is written to POKTscan internal databases and served to the public through several sections of the site.

For each day in a period of approx. two months, from 2022-11-08 to 2023-01-03 [2], the CP data was summarized, obtaining for each day and each PNP deployment location the following features:

- Median Success Latency (MSL) : The weighted average of the MSLs reported by the CP for each session.
- Weighted Success Latency (WSL) : The weighted average of the WSLs reported by the CP for each session.
- Percentile 90 Latency (p90) : The weighted average of the p90 latency reported by the CP for each session.
- Success Rate (SR) : The weighted average of the SR reported by the CP for each ses-

[1] Citation needed. The structure of this data can be seen here.
[2] The month of January was removed since the metrics of one of the main PNPs, *eu-central-1*, ceased to be available due to problems external to POKTscan.

sion.

- Total Success : The total number of successful relays observed in the given day and gateway.

From these, the MSL was used as target feature. In fig. 2 the evolution of the latency in the observed period is shown. From this data, the average MSL and the total number of relays in the observed period is summarized in table 1. Currently the network has an $AvgMSL = 155.66$ ms.

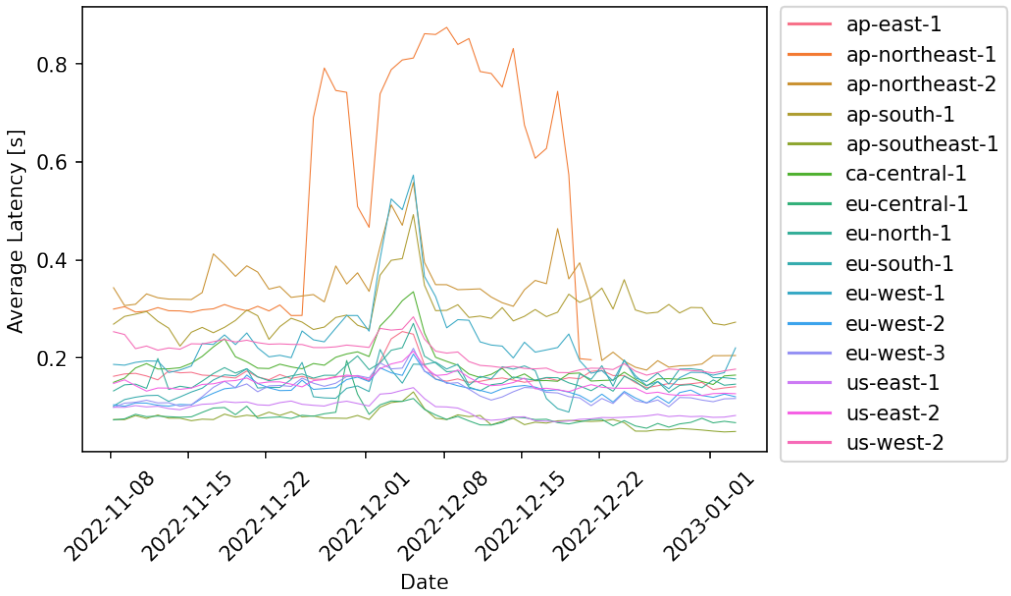

Figure 2: Evolution of the MSL in time for each of the 15 available PNPs from 2022-11-08 to 2023-01-03.

3.2 App Users Data

The app users data is not available, since the Pocket Network does not track users. This means that there is no way to obtain the latency of the the apps to each of the PNPs, for this reason the optimization cannot be optimized for app users directly. What is available to the public is the latency between the different PNPs, this means that it is possible to optimize the deployment of the PNP to take advantage of the current AWS infrastructure, redirecting app users internally once they reach a portal. In this case the required information is publicly available in Cloud Ping. This website present the ping times between all of the AWS regions, which can be easily converted into a fully connected graph of the AWS infrastructure.

3.3 Graph Representation

We choose to represent the problem as a transport problem embedded in a graph. The graph is composed of two elements, nodes and edges. The nodes, in our case, are the

Table 1: Average MSL and total successful relays for each of the PNPs over two months.

PNP Location	MSL [ms]	Relays
ap-east-1	163	3075 M
ap-northeast-1	534	4246 M
ap-northeast-2	322	3225 M
ap-south-1	295	407 M
ap-southeast-1	75	11016 M
ca-central-1	185	521 M
eu-central-1	81	13848 M
eu-north-1	161	2015 M
eu-south-1	153	943 M
eu-west-1	234	162 M
eu-west-2	133	1327 M
eu-west-3	133	1558 M
us-east-1	94	5814 M
us-east-2	148	2763 M
us-west-2	207	1586 M

source of the relays (apps locations) and also the PNPs possible locations (AWS available locations). The node-runners are represented by a single node labeled as *servicers*. The edges represent the connectivity between all the possible PNP locations/Apps and between the node-runners and these locations. Each edge is characterized by the response latency observed between a pair of nodes. The edges reaching the servisers node also represent the PNPs, since each connection between them and the Apps can only exist if a PNP is deployed. For example, a node representing the traffic in *eu-central-1* and the *servicers* node will have a connection if a PNP is available there, since there is one deployed in that location and the latency is low (81 ms, see table 1), this connection will be strong. In other regions, such as *ap-south-1*, there is also a PNP deployed, hence there is a connection between the region and the *servicers*, however this connection will be a weak connection since in average the latency to the servicers is higher (295 ms, see table 1). Finally, there is no connection between *sa-east-1* and the *servicers*, since there is no PNP deployed there. Using this concept, the problem graph was constructed and it is shown in fig. 3.

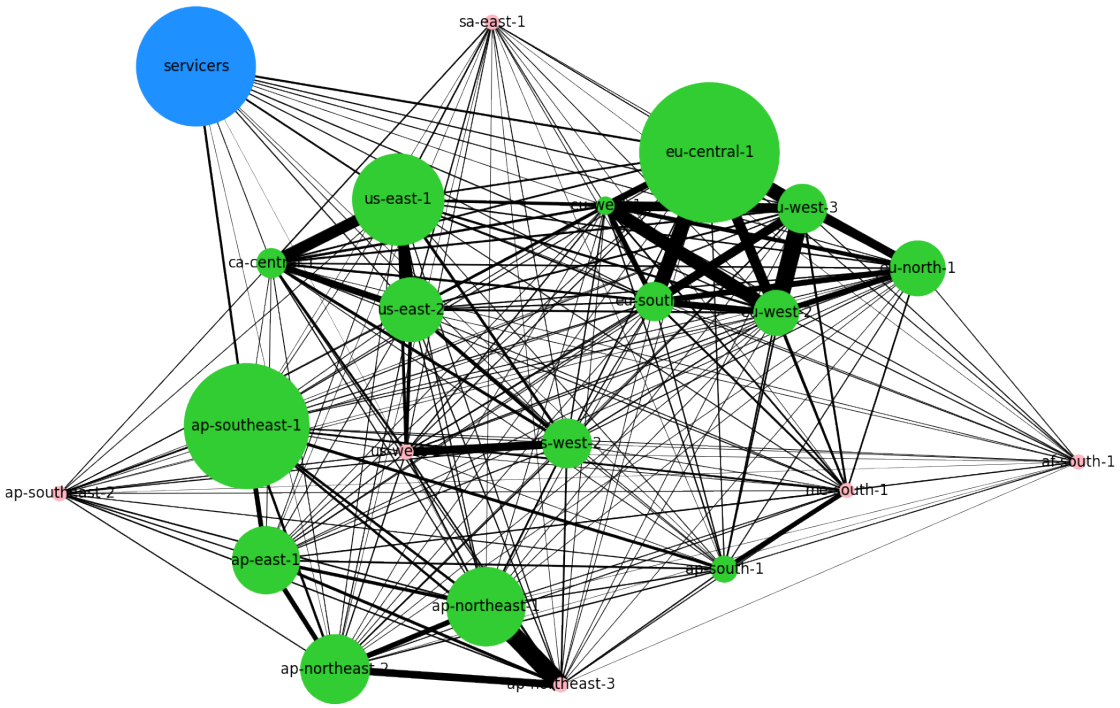

Figure 3: Graph representation of the PNP location problem. Nodes in green represent the AWS locations with traffic, nodes in pink represent AWS location with no traffic (no PNPs there, used for reference) and the blue node represent the servicers. The nodes sizes, excepting the servicers node, are proportional to their relay volume. The edges width represent the latency of the connection, thicker edges represent lower latency.

4 Mathematical Modeling

The problem to solve can be expressed as finding which are the connections that can be eliminated between the serviser node and the rest of network, while not degrading the average latency of the network too much. In simple words, we need to answer:

- How many lines (edges) connected to the blue node of fig. 3 can we cut without making the system too slow?
- Which of those lines should we keep?
- How should the traffic be routed to reduce latency and hops? (using a hop time $p = 50$ ms)

These questions can be answered by many approaches, two of them are presented here. The first one is an heuristic (or brute-force) approach and the second is an optimal search. While the second one is a more formal and precise formulation of the problem, it is more complex to follow, for this reason (and for double checking), the empirical approach is also presented, which is much simpler to understand.

The notation used in this section is shown in table 2:

Sets	
V	Set of graph nodes (AWS locations), $V = \{1, ..., n-1\}$
V'	Set of graph nodes + Servicer, $V' = \{V \cup n\}$
A	Set of arcs or edges, $A = \{(j,k) \mid j,k \in V', j \neq k\}$
Parameters	
n	Number of graph nodes + Servicer
$\mathcal{L}$	Average latency of the network.
$\mathcal{A}$	Set of active arcs/edges connected to the Servicer (PNPs deployed).
M	Big value (used for linear search)
D	Total node demand (relays) $\sum_{i \in V'} d_i$
d_i	node demand (relays) $i \in V'; d_n = 0$
c_{jk}	Latency between nodes j and k
q	Max number of active arcs/edges to node Servicer
p	Hop penalization
Decision variables (used for linear search)	
x^i_{jk}	1 if the relays/demand originated in i pass trough node j to node k, 0 otherwise
y_j	if the server n is reached from j, 0 otherwise

Table 2: Parameters and variables of the model.

4.1 Heuristic Approach

One of the simplest ways of finding a solution is to test each of the possible scenarios and calculate the lowest response time of each of them. This is also known as a *brute force* approach, since no assumption is used to simplify the search space. The algorithm is simple to follow, the first step is to create a list of all the possible PNPs deployments that are requested to test. This list will have the following format:

- Case 1: [*us-east-1*]
- Case 2: [*us-east-2*]
- ...
- Case M: [*us-east-1*, *eu-cental-1*]
- Case M+1: [*us-east-2*, *eu-cental-1*]
- ...
- Case N-1: [*us-east-1*, *eu-cental-1*, *eu-west-1*, *eu-north-1*, *us-west-2*]
- Case N: [*us-east-2*, *eu-cental-1*, *eu-west-1*, *eu-north-1*, *us-west-2*]

It can be seen that this list is filled with lists of PNPs to deploy, and is extensive. Due to limitations of the heuristic approach (time consuming), the search space was restricted to explore up to six deployed PNPs. The search space is generated then using a recursive solution (or a series of for loops).

Once the search list is created, each of its elements is tested in order. The procedure is described in algorithm 1. It is implemented using the NetworkX package [2], which implements the Dijkstra algorithm [1] for shortest path calculation, using a modified weighting mechanism that adds a hop time p each time the path reaches a node that is not the servicers node.

Algorithm 1 Heuristic method, PNP deploy exhaustive search.

procedure Heuristic Search(Search Space, Max PNPs)
    $\overline{\mathcal{L}} \leftarrow$ zeros
    $\overline{\mathcal{A}} \leftarrow []$
    $n \leftarrow 0$
    for Num PNPs $\in$ Max PNPs **do**
        SubSpace $\leftarrow$ Find all cases in $mboxSearchSpace$ with Num PNPs elements.
        $\overline{\mathcal{L}'} \leftarrow$ zeros
        $\mathcal{A}' \leftarrow []$
        $m \leftarrow 0$
        for case $\in$ SubSpace **do**
            $\lambda \leftarrow$ Get shortest path from each node to the servicers.
            $\mathcal{L}'[m] \leftarrow (D \times \lambda) + p$ Get total cost of moving relays.
            $\mathcal{A}'[m] \leftarrow$ case
            $m++$
        end for
        argBest $\leftarrow argMin(\overline{\mathcal{L}'})$
        $\mathcal{L}[n] \leftarrow \mathcal{L}'$[argBest]
        $\mathcal{A}[n] \leftarrow \mathcal{A}'$[argBest]
        $n++$
    end for
    Return $(\overline{\mathcal{L}}, \overline{\mathcal{A}})$.
end procedure

The result of implementing algorithm 1 is a list containing the minimum latency for each case and a list containing the deployed PNPs for each case. This algorithm is fast for low number of gateways but becomes more computationally demanding as Max PNPs grows, for this reason it was executed only up to Max PNPs $= 6$.

4.2 Linear Programming Approach

As it is presented here, this problem is a good candidate for linear optimization techniques. The problem can be expressed as an optimization problem with a series of linear relationships and constraints. As such, and given that the dimensionality of the problem is not too high, an optimal solution can be reached.

Objective function:

$$\min_{x} \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in V'; k \neq j} d_i c_{jk} x^i_{jk} + \sum_{i \in V} \sum_{j \in V} \sum_{k \in V; k \neq j} d_i p x^i_{jk},$$

subject to:

$$\sum_{k \in V'; k \neq i} x^i_{ik} - \sum_{k \in V'; k \neq i} x^i_{ki} = 1 \quad \forall i \in V \tag{1}$$

$$\sum_{k \in V} x^i_{nk} - \sum_{k \in V} x^i_{kn} = -1 \quad \forall i \in V \tag{2}$$

$$\sum_{k \in V'; k \neq j} x^i_{jk} = \sum_{k \in V'; k \neq j} x^i_{kj} \quad \forall i \in V; j \in V \tag{3}$$

$$\sum_{k \in V'} x^i_{jk} \leq 1 \quad \forall i \in V; j \in V' \tag{4}$$

$$\sum_{i \in V} x^i_{jn} \leq M y_j \quad \forall j \in V \tag{5}$$

$$\sum_{i \in V} x^i_{jn} \geq y_j \quad \forall j \in V \tag{6}$$

$$\sum_{j \in V} y_j \leq q \tag{7}$$

$$x^i_{jk}, y_j \in \{0, 1\} \quad i \in V; j \in V'; k \in V' \tag{8}$$

The presented constrains have the following meanings:

- Constraint 1: The sum of the arcs that leave node i minus the sum of the arcs that arrive at node i must be 1; This guarantees that from node i, being this the initial node of the path, only 1 arc comes out, for the route i.
- Constraint 2: The sum of the arcs leaving the node n minus the sum of the arcs arriving at the node n must be -1; this guarantees that from the servicers node only 1 arc arrives for route i, since it is the end of the path.
- Restrictions 3: For each route i the number of arcs that arrive at node j must be equal to the number of arcs that leave node j.
- Restrictions 4: Prevent a node j from being visited more than once in the same route i.
- Constraints 5 and 6 relate the variables x and y. If the sum of arcs leaving node j towards the servicers node is 0; then $y_j = 0$. Conversely, if the sum of the arcs leaving node j towards the servicers node is greater than or equal to 1, $y_j = 1$.

- Constraints 7: Guarantees that the number of arcs arriving at the servicers node is less than or equal to q.
- Constraints 8: Establishes the nature of the variables.

Once the problem is formulated it can be processed using the OR-Tools package [4], specifically using a *SCPI* backend. This way of presenting the problem allows for further scaling compared to the heuristic method presented in section 4.1, and enabling the computation of the optimal number of PNPs for the lowest possible latency. Moreover, this method has the capacity to add higher order restrictions, such as edge capacity (maximum traffic between locations), which the heuristic method cannot handle directly.

5 Results

In this section we summarize the results of this report and comment on some of the outcomes.

5.1 Optimal Solution

Using the linear programming method described in section 4.2 and no restriction in the number of PNPs to open, the network latency was minimized, resulting in 8 PNPs deployed and a network average latency of $Avg.MSL = 118.76$ ms. The resulting network is shown in fig. 4 and uses the following PNPs:

1. ap-southeast-1
2. eu-central-1
3. us-east-1
4. ap-east-1
5. us-east-2
6. eu-west-2
7. eu-west-3
8. us-west-2

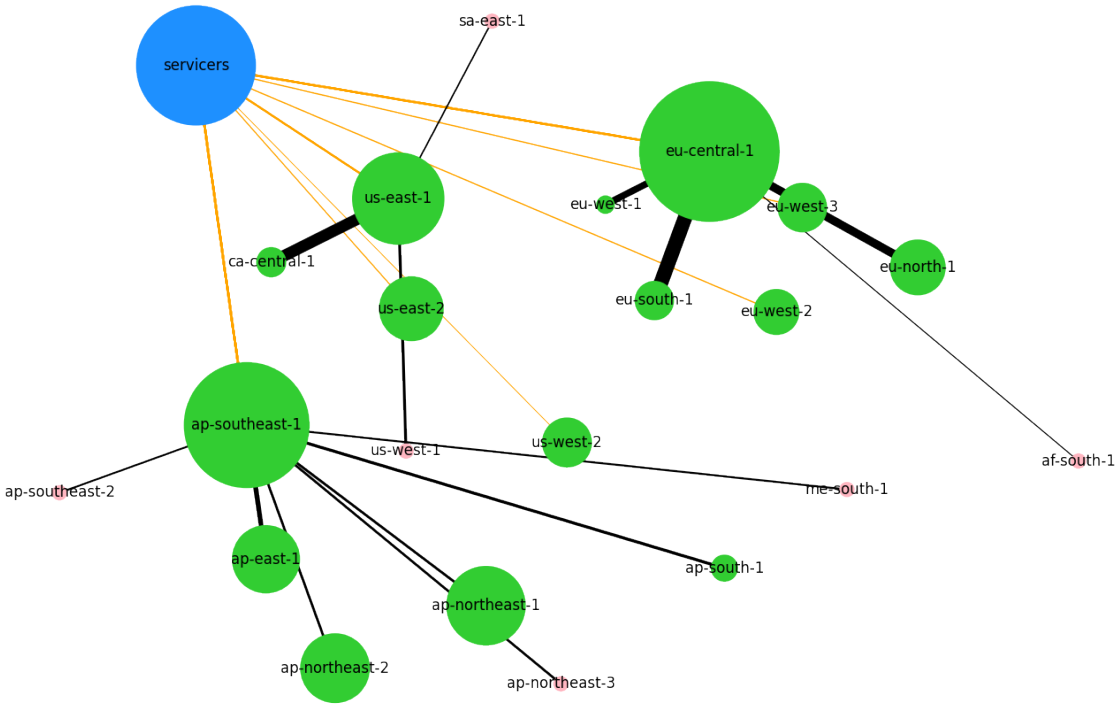

Figure 4: Graph representation of the optimal PNP locations.Edges in orange represent the active PNPs.

5.2 Convergence to Optimal Latency

An interesting point to observe is how fast the network latency reaches the optimal value of $Avg.MSL = 118.76$ ms. By means of this curve the number of PNPs required to reach a close to optimal latency can be obtained. To do this the heuristic method was used to obtain the network latency for the scenarios of opening 1 to 6 PNPs. The results are shown in fig. 5. It is clear from the figure that the network latency reaches values near the optimal with three or more deployed PNPs. Moreover, it can be seen that with only two PNPs and routing the traffic through AWS network, the network latency improves from 155.66 ms to 140.89 ms. These results were re-validated running the linear programming approach using lower q variable values and same results were obtained. In this way, contrasted that the proposed configuration is optimal.

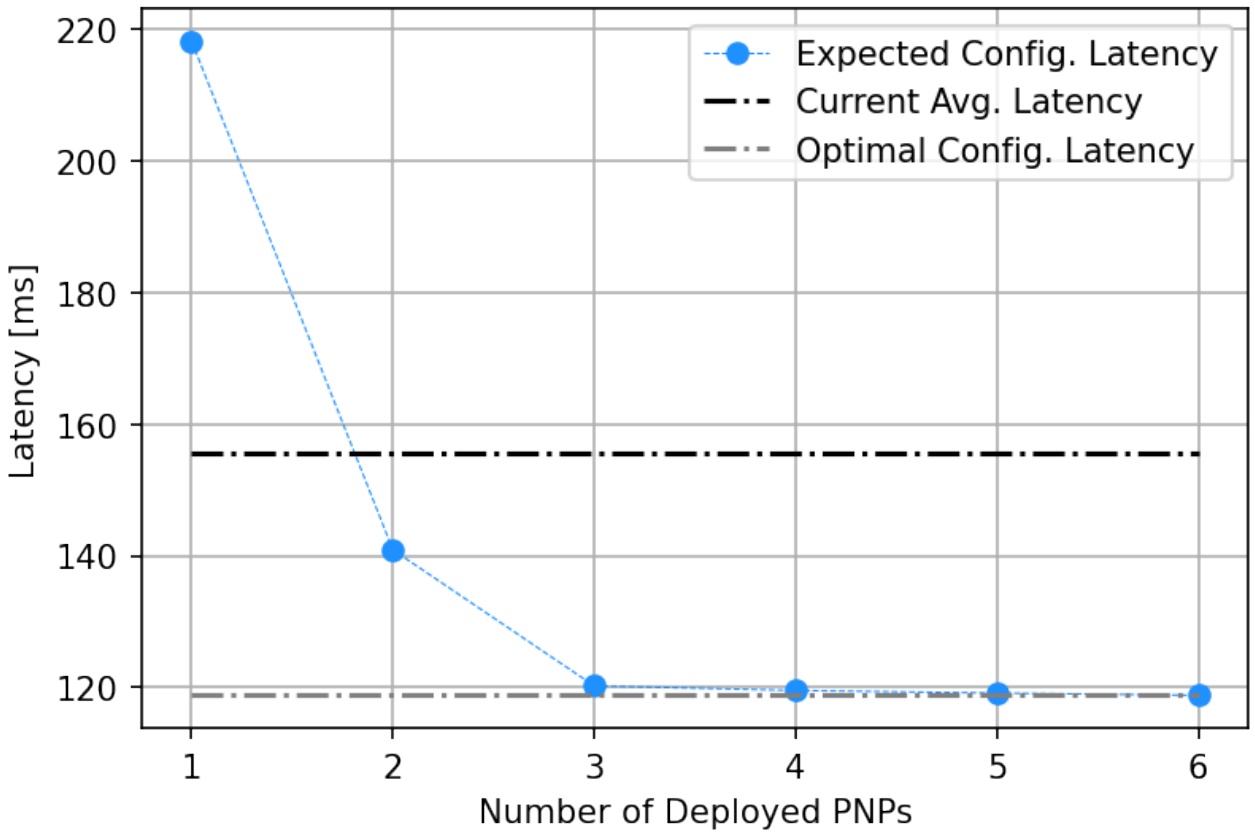

Figure 5: Network average Median Success Latency as a function of the number of active PNPs. The black dashed line represents the current MSL of the network of 155.66 ms and in grey the optimal network latency of 118.76 ms

5.3 Minimum Cost Scenario

If the objective is not only to reduce the network latency but also to reduce the costs of maintaining the PNPs (both hardware and human resources costs). Then the number of deployed can be reduced from the optimal solution presented in section 5.1. To select the best scenario, the table 3 can be used as a reference. It can be seen that with only three

active PNPs the difference to the optimal latency is only 1.22% and the network latency improves a 22.76% from current values.

Table 3: Network Avg. MSL and difference to current and optimal network latency for opening one to six PNPs.

Active PNPs	Avg. MSL [ms]	Diff. to Optimum [%]	Diff. to Current [%]
1	218.13	83.65	40.13
2	140.88	18.61	-9.49
3	120.22	1.22	-22.76
4	119.53	0.64	-23.20
5	119.14	0.31	-23.45
6	118.80	0.02	-23.67

The resulting network, if only three PNPs are selected, is shown in fig. 6 and uses the following PNPs:

1. ap-southeast-1
2. eu-central-1
3. us-east-1

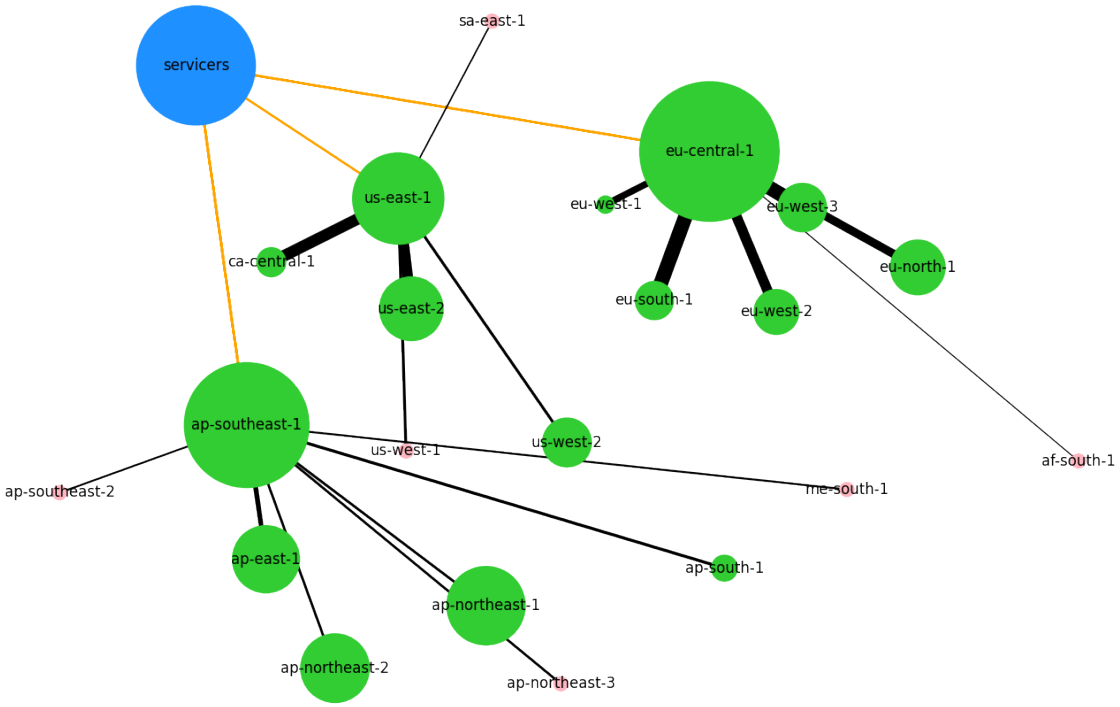

Figure 6: Graph representation of the optimal PNP locations.Edges in orange represent the active PNPs.

6 Conclusions

In this document it is shown that the current number of Pocket Network Portals (PNPs) is too high. By means of linear optimization we demonstrated that if the number of PNPs is reduced and the apps traffic is re routed to the remaining portals, not only the cost of maintaining PNPs can be reduced, but also the average latency of the network can be improved.

We advise the Pocket Network Inc. (PNI) to:

1. Reduce the number of active PNPs to three. The remaining portals should be placed in:
   - ap-southeast-1
   - eu-central-1
   - us-east-1
2. Re-route the traffic of all other AWS locations to the remaining PNPs.

By implementing these changes we expect to:

1. Improve the network latency by 22.76%, reaching an average latency of ~ 120.22 ms.
2. Reduce costs for the PNI.
3. Reduce costs for the node-runners (less hardware and overhead work of following traffic).

The impact should be positive in all aspects of the network, reducing the network overall cost and sell pressure on the POKT token. These changes do not affect the decentralization of the network, since all the PNPs belong to the PNI.

The code and data requiered to reproduce the presented results is being released to the public.

Reference List

[1] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematlk 1*. 1959, pp. 269–271.

[2] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[3] Ramiro Rodríguez Colmeiro Pablo Frigerio. *Analysis and Forecasting of the Main Pocket Network Chains Traffic Using Statistical Tools.* 2022. url: `https://github.com/pokt-scan/infracon-2022` (visited on 05/01/2022).

[4] Laurent Perron and Vincent Furnon. *OR-Tools*. Version v9.5. Google, Nov. 25, 2022. url: `https://developers.google.com/optimization/`.

[5] POKTscan. *Geo-Mesh*. 2022. url: `https://forum.pokt.network/t/poktscan-s-geo-mesh/3585` (visited on 10/11/2022).