UCLouvain

École polytechnique de Louvain

***SINF2MS* | LINFO2345**

Random Peer Sampling Project

Adrien Giot Guillaume Jadin

1 Random Peer Sampling Implementation

1.1 Objective

Implement in Erlang the Cyclon Random Peer Sampling protocol and analyze its metrics.

1.2 Settings considered for the experiments

The simplified version of the Cyclon Random Peer Sampling protocol was implemented.
Experiments were performed while varying the following parameters :

— 3 sizes of the system : 100, 500, and 1000.
— 3 view proportions : 0.10, 0.15 and 0.20 (of the number of nodes of the system)
— 3 subset proportions : 0.05, 0.075, and 0.10 (of the number of nodes of the system).

View proportions were varied while fixing the subset proportion to 0.05. Subset proportions were varied while fixing the view proportion to 0.15. System of different sizes are compared for a view proportion of 0.15, while considering diffrent subset proportions (0.05, 0.075, 0.10).

All results were averaged over five different runs.

Log files used to produce the graphs can be accessed through this link : `https://uclouvain-my.sharepoint.com/:f:/g/personal/adrien_giot_student_uclouvain_be/EmQdHwlQaQdMjdehkGGeMgkBuaNs_WwR1odb3DcksfYr3w?e=YehGmQ`

1.3 Discovery proportion

As stated in the assignment, the discovery proportion represents the proportion of nodes that have been at least one turn in the view of a given node.

To evaluate it, we measure at each turn the average of that proportion over all nodes. We do that for 150 turns.

Varying the subset proportion

The following results indicate that, independently of the system size, the number of turns to discover at least 75% of all nodes in the system is smaller if the subset proportion is smaller.

This is intuitive as a smaller subset limits the number of nodes that can be exchanged/discovered when exchanging subsets with another node.

The horizontal green line represents the 75% discovery threshold.

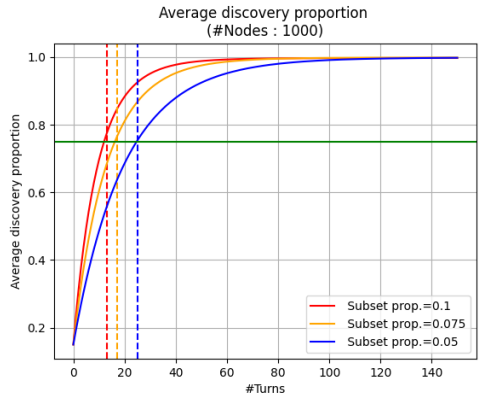

FIGURE 1 – Comparison for different subset proportions, 1000 nodes in the system ; Threshold reached at turn 13, 17, and 25

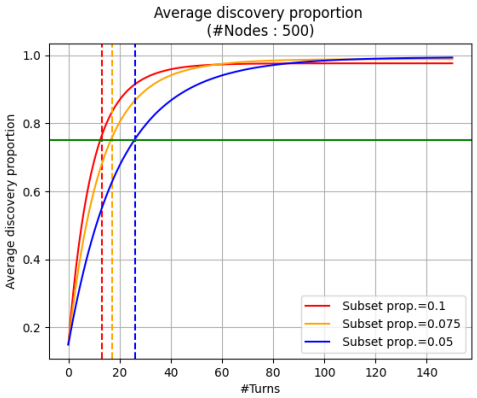

FIGURE 2 – Comparison for different subset proportions, 500 nodes in the system ; Threshold reached at turn 13, 17, and 26

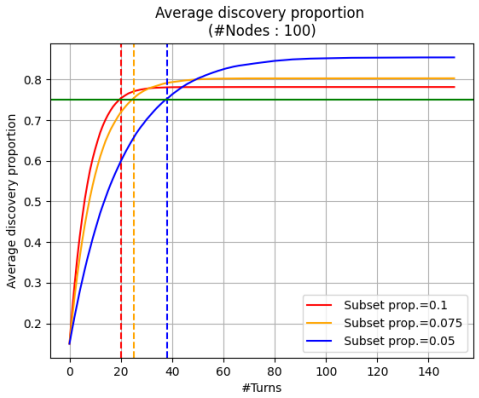

FIGURE 3 – Comparison for different subset proportions, 100 nodes in the system ; Threshold reached at turn 20, 25 and 38

For 100 nodes, the discovery proportion that is achieved at convergence is greater with a smaller subset (0.05). Smaller subsets might seems to delay convergence in this case. For 500 and 1000 nodes, results don't indicate such phenomenon and the discovery proportion seems to converge to the same value.

Varying the number of nodes of the system

Here, we compare results between different system sizes on different subset proportions.

We see that, whatever the subset proportion, the system containing 100 nodes takes more time to reach the 75% discovery threshold than the systems containing 500 and 1000 nodes (that take almost exactly the same time). Also, at convergence time, the discovery proportion is always smaller than that of the two bigger systems, that converge around the same discovery proportion.

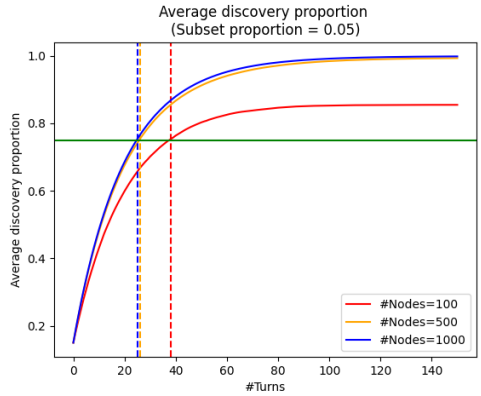

FIGURE 4 – Comparison for different sizes of the system, subset proportion = 0.05 ; Threshold reached at turn 25, 26, and 38

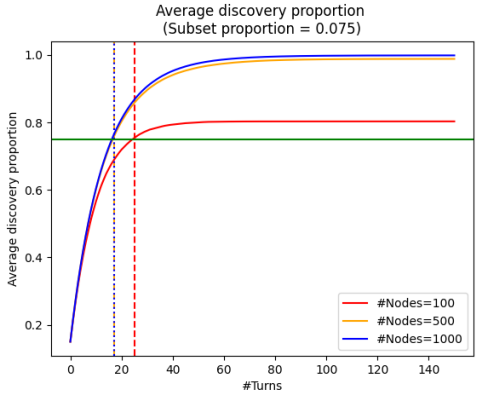

FIGURE 5 – Comparison for different sizes of the system, subset proportion = 0.075 ; Threshold reached at turn 17 (twice) and 25

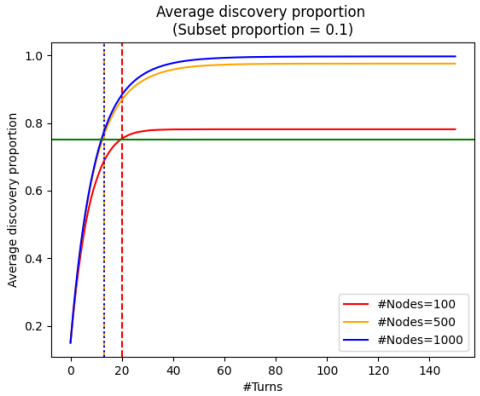

FIGURE 6 – Comparison for different sizes of the system, subset proportion = 0.1 ; Threshold reached at turn 13 (twice), and 20

Varying the size proportion

The discovery proportion threshold of 0.75 is reached slightly faster when the view proportion is greater. This is more clear for a system size of 100 nodes. In this case, the smallest view proportion does not even reach the 0.75 threshold. A smaller view size seems to speed up the convergence, at least for this system size.

One idea to explain this result is that a bigger view allows for better sampling when choosing subsets to share to other nodes (there are more choices at hand). A smaller view could favor received subsets that are redundant with the current view, which would limit the growth of the discovery proportion.

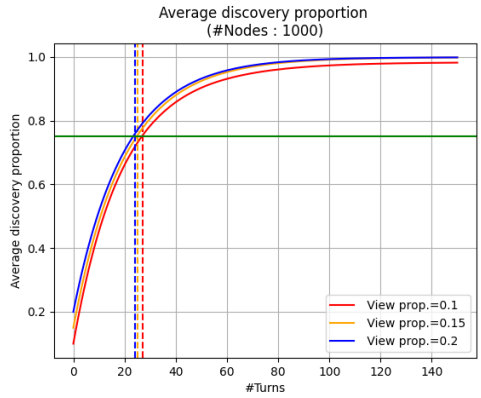

FIGURE 7 – Comparison for different view proportions, 1000 nodes in the system ; Threshold reached at turn 24, 25, and 27

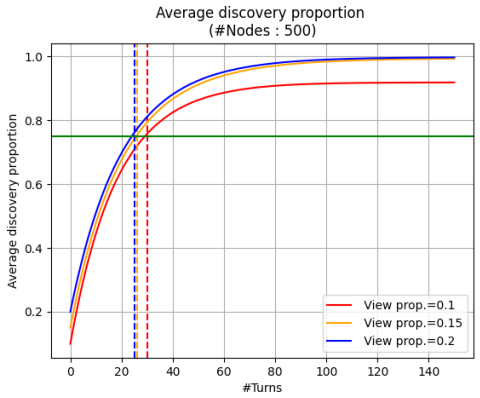

FIGURE 8 – Comparison for different view proportions, 500 nodes in the system ; ; Threshold reached at turn 25, 26, and 30

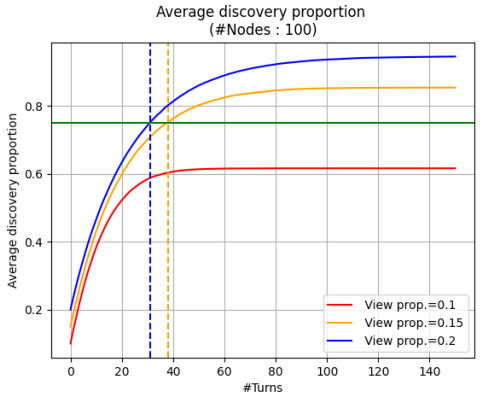

FIGURE 9 – Comparison for different view proportions, 100 nodes in the system ; ; Threshold reached at turn 31 and 38

1.4 Churn resilience - Resilience against the exiting node

As stated in the assignment, the churn resilience represents the ability of the RPS protocol to handle churn. Here, we evaluate the resilience against the exiting node. It is defined as the average number of turns to see an exiting node be removed from at least 75% of all node's views.

To evaluate it, we compute the proportion of the views that contain the exiting node after its exit. We compute that over the 100 turns that follow the exit of a node. We normalize that proportion by dividing by the view proportion (0.15, except when we vary the view itself).

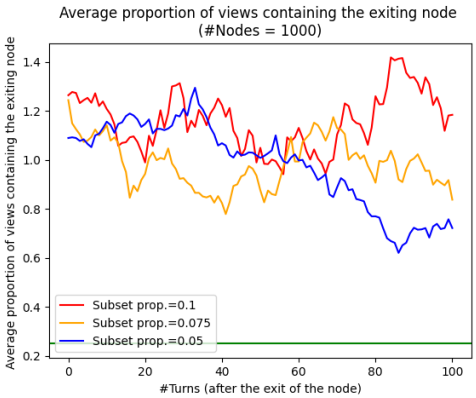

FIGURE 10 – Comparison for different subset proportions, 1000 nodes in the system

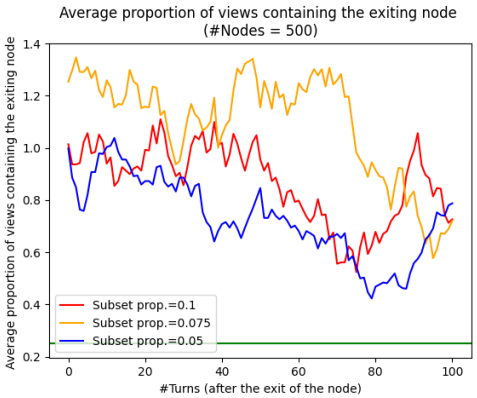

FIGURE 11 – Comparison for different subset proportions, 500 nodes in the system

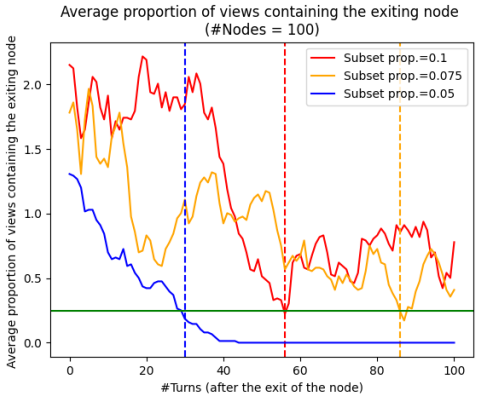

FIGURE 12 – Comparison for different subset proportions, 100 nodes in the system ; Threshold reached at turn 30, 56, and 86

Varying the subset proportion

In general, the results seem to show that the bigger the subset proportion, the longer the exiting node will stay in the views of the system. This is less clear/obvious than the impact of the subset size on the discovery proportion of the nodes.

One reason for that is the following : the proportion of views that contain the exiting node when it exits the network is variable (even when averaging over 5 runs). For example, for 500 nodes, experiments with subset proportions of 0.075 and 0.01 have different proportions at exit time (they are respectively equal to 1.25 and 1.0). That difference makes the comparison more difficult, as the exit node can be further propagated to other nodes with such a greater proportion at exit time.

The impact of the subset size can be explained : to remove an exiting node from a view, it must become the "oldest" node of the view, so that the exiting node does not respond and is considered to have left the network. When increasing the subset proportion, the risk of propagating the exiting node increases, which delays its exit of the views of the network. That risk is also increased on nodes that receive the exiting node.

Varying the the size of the system

When varying the number of nodes in the system, the results indicate that the more nodes there are in the system, the longer the exiting node will stay in the views of the nodes. This is clear as the 0.25 threshold (= exiting node left 75 % of the views) is only reached by systems of 100 nodes, whatever the subset size (when averaging over 5 runs).

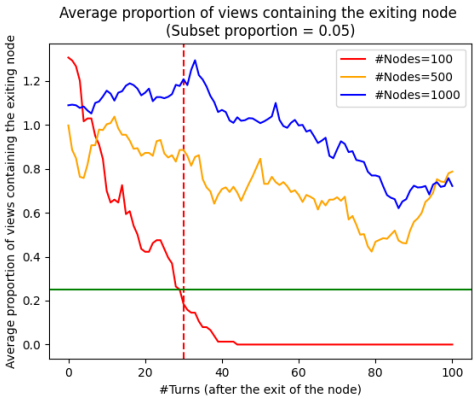

FIGURE 13 – Comparison for different sizes of the system, subset proportion = 0.05 ; Threshold reached at turn 30

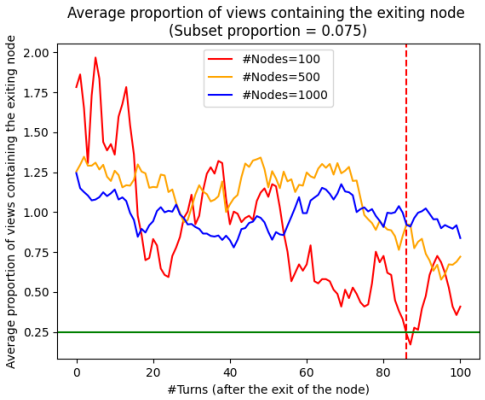

FIGURE 14 – Comparison for different sizes of the system, subset proportion = 0.075 ; Threshold reached at turn 86

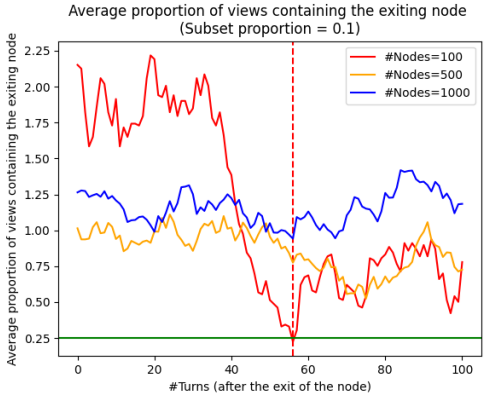

FIGURE 15 – Comparison for different sizes of the system, subset proportion = 0.1 ; Threshold reached at turn 56

To explain this result, we consider the following reasoning : a bigger system implies bigger views (in absolute numbers), so we can assume it takes more time for a given node to elect the exit node as the oldest and detect its exit. This increases the risk of propagation of the exiting node to other views, whose nodes face the same issue.

Varying the view proportion

Despite averaging the results over five runs, the various proportions at exit time complicate analysis, though it seems likely that bigger views deteriorate the resiliency against exiting nodes. We might have tried more extreme view proportions to check that.

1.5 Perfect Configuration

Greater subset/view proportions allow for quicker discovery, but it deteriorates resilience against exiting nodes, that's a trade-off.

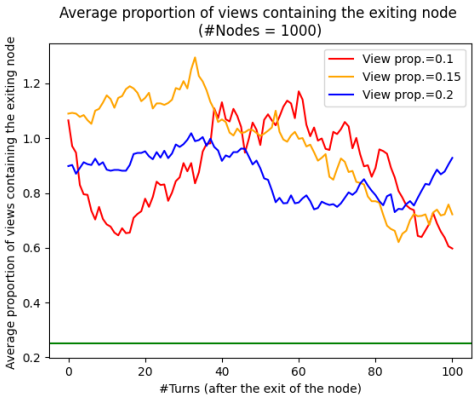

FIGURE 16 – Comparison for different view proportions, 1000 nodes in the system

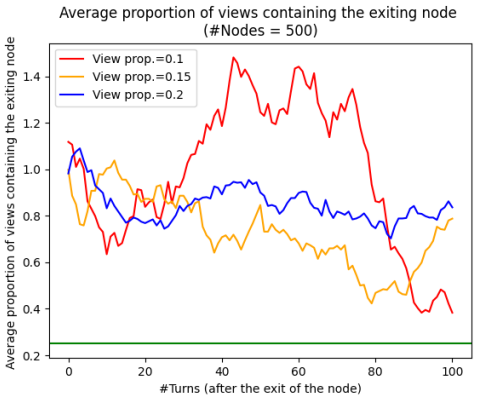

FIGURE 17 – Comparison for different view proportions, 500 nodes in the system

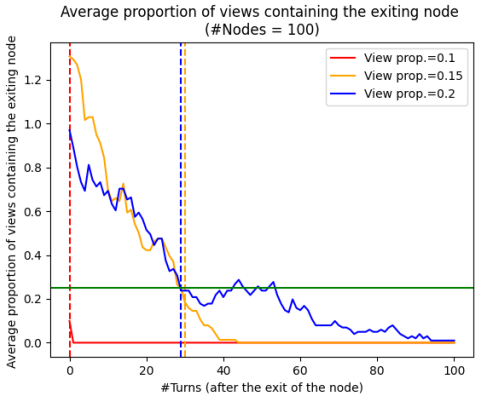

FIGURE 18 – Comparison for different view proportions, 100 nodes in the system ; Threshold reached at turn 0, 29, and 30

For systems of size 500 or 1000, it seems critical to not consider a too high subset size, as bigger systems make the resilience against exiting nodes harder.

For systems of size 100, a view proportion too small (0.1) led to not reach the discovery threshold of 0.75, so we should be careful to make that parameter sufficiently high for this system size.

2 The Byzantine Nodes Problem

In a system, byzantine nodes respond with subsets made of byzantine node in order to isolate parts of the system.

2.1 Objective

Implement a byzantine node in erlang. Print graph of discovering proportion and convergence time for the best configuration of variables found previously with 10% of Byzantines nodes. Check the proportion of byzantine nodes in views at convergence, for different numbers of byzantine nodes in the system.

2.2 Configuration

The configuration of variables that was kept is the following : a system of 100 nodes, view and subset proportions of 0.15 and 0.075.

We consider 100 nodes for a smaller processing time. View and subset proportions of respectively 0.15 and 0.075 allow for a fast convergence of the discovery proportion, which goes above the 0.75 threshold (in the absence of byzantine nodes). We chose a subset proportion of 0.075 over 0.05 because we don't consider exiting nodes here, and we prefer the fast convergence of that greater subset size.

2.3 Results

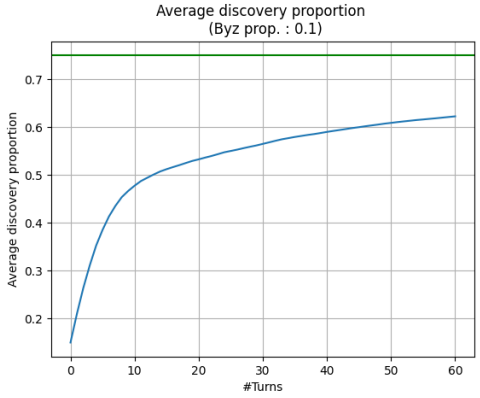

FIGURE 19 – Discovery proportion for a system of 100 nodes with a proportion of 0.1 byzantine nodes

We measured the convergence time to be around 60 turns. We then measure the average proportion of byzantine nodes in views of healthy nodes. We consider the value reached at convergence time.

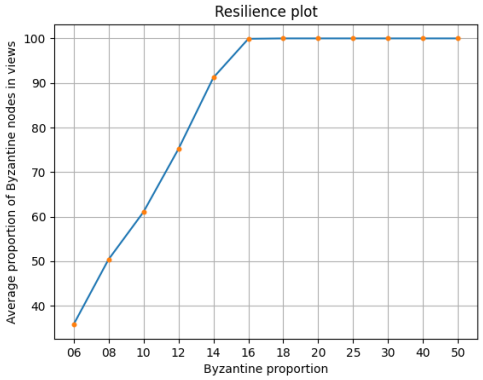

FIGURE 20 – Resilience plot for a system of 100 nodes, view and subset proportion of 0.15 and 0.075

This plots shows that the more byzantine nodes, the higher the proportion of byzantine nodes in view at convergence time. For the configuration that was kept, the proportion of byzantine nodes in views of "normal" nodes reaches 100 % at convergence time for a Byzantine proportion ≥ 18.

Note that the x-axis is not scaled (12-10=25-20).

All results were averaged over five different runs. Log files used to produce the graphs can be accessed through this link :

`https://uclouvain-my.sharepoint.com/:f:/g/personal/adrien_giot_student_uclouvain_be/EmQdHwlQaQdMjdehkGGeMgkBuaNs_WwR1odb3DcksfYr3w?e=YehGmQ`

3 Countermeasure to Byzantines Nodes

Multiple mechanisms to improve cyclon's security against byzantines nodes were considered. We didn't have time for to implement them.

3.1 Centralized SPSS-like solution

One idea found is based on the SPSS service mentioned in the paper `Brahms: Byzantine resilient random membership sampling`. It would consist to gather statistics (about subsets sent to each node) for identifying nodes that are over represented. Such identified nodes can then be removed from the system.

To implement such a service, one could use a centralized approach, with a trusted server that receives logs (subsets received) from nodes of the system, and then computes a list of suspected nodes. That list can then be queried by nodes of the system to prevent harm.

This requires trust to be established between nodes and the server (Certificate authority, use of private/public key, encryption, ...), to not create an additional surface for attackers.

3.2 Local detection

Nodes could also run a local byzantine detection procedure. Using an history of the subsets that were sent to it, it could detect byzantine nodes through statistical analysis.

This could be coupled by the centralized detection server mentioned before.

3.3 Voting scheme

Instead of sending all the subsets it receives, a node could run its local detection process to summarize those data within a vote. The vote would then be sent to the centralized server, that would elect suspected nodes from all the votes. This allows to balance the detection load between the nodes of the system, instead of leaving it all to one centralized detection server.