Rapport for Handin Four

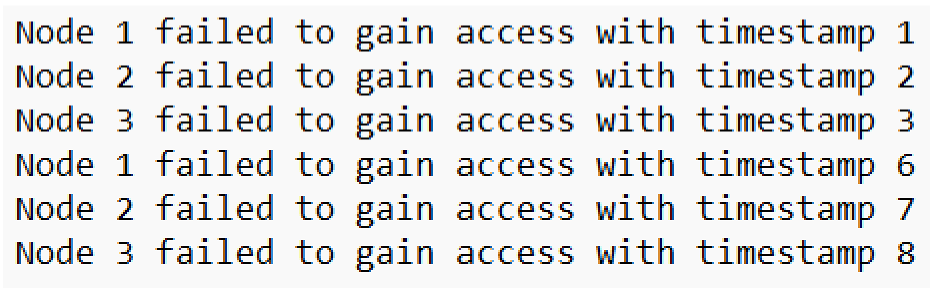
The following report describes how fault-tolerant communication between 3 grpc servers could work. It explores how they handle critical data together and how they ensure reliable synchronized data. On the following link you can inspect the code: https://github.com/Rpou/Distributed-system/tree/main/Consensus

The program operates by continually having 3 servers operating at the same time. They can communicate with each other directly without a central server.

Central to the program is a global variable, which acts as the critical data, that the nodes want to access. They gain access to this data by asking the other nodes for permission and if they gain total permission they can then emulate the data. If the node is not granted access it will then wait until it eventually gains access from all nodes. This ensures that they do not emulate the data at the same time and cause issues. Sometimes the nodes do not want access to the critical data. This is implemented by having a random number generator, and then one in three times the node do not want to have access to the critical data.

As previously said, the program works by trying to gain access by comparing their timestamp. If this request is denied, it will add 5 to its current timestamp. If they do gain access, they will not gain any points to their timestamp. This ensures that every node will eventually gain access, because if it continually fails to gain access, it will get a higher amount of points than everyone else - thus higher priority.

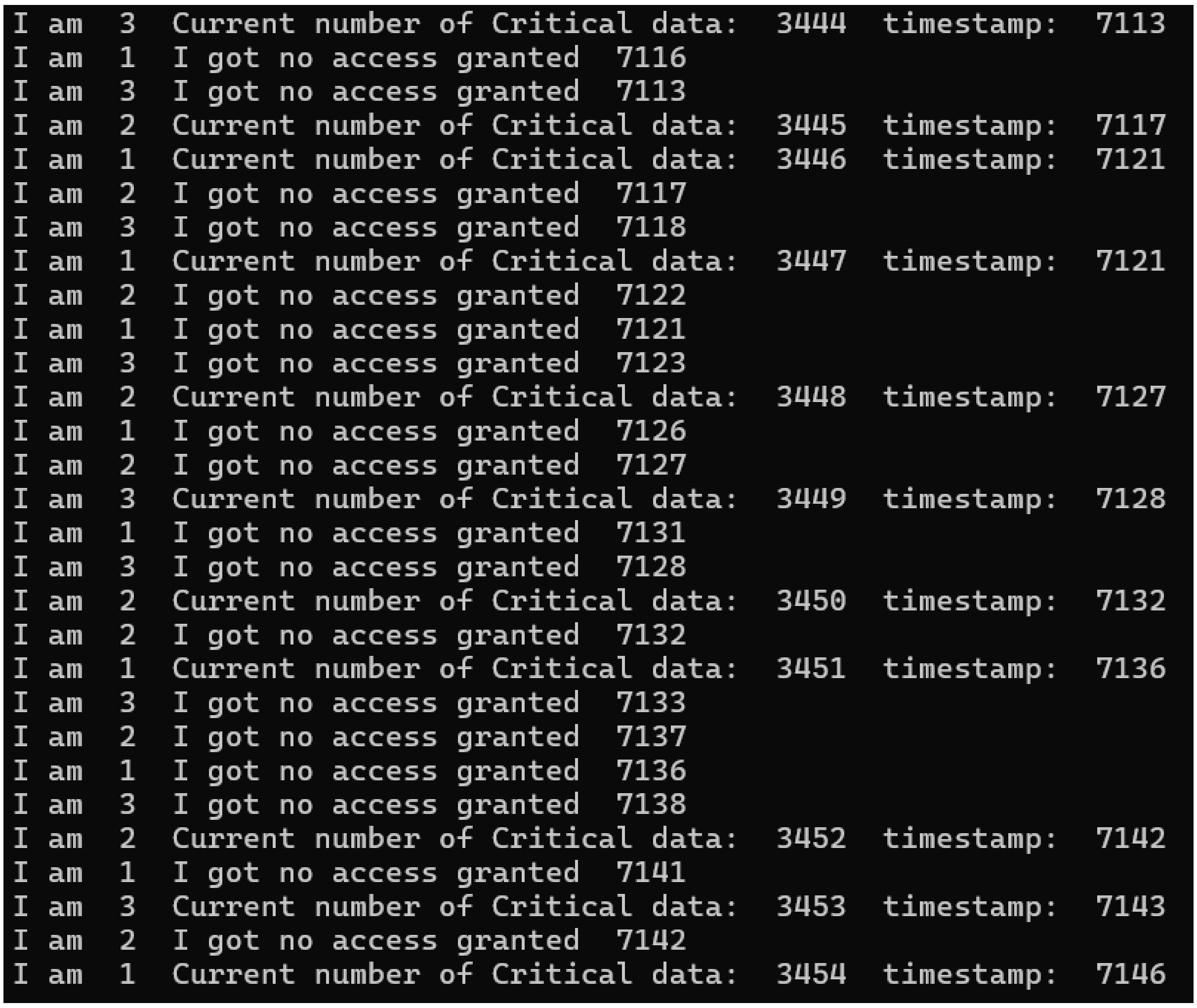
The program also has a sleep timer, before adding points to themselves, and after gaining access to the critical data. This ensures that the example in the figure below never happens. That example happens, if node 3 requests access AFTER node 1 and 2 has added 5 to their own timestamp. That means that 3 is no longer the highest number, although it should be.



```
Node 1 failed to gain access with timestamp 1
Node 2 failed to gain access with timestamp 2
Node 3 failed to gain access with timestamp 3
Node 1 failed to gain access with timestamp 6
Node 2 failed to gain access with timestamp 7
Node 3 failed to gain access with timestamp 8
```

In the beginning the program gives each node a timestamp, based on their ID (1,2,3). The program then adds 5 to the timestamp, if they do not gain access. The reason behind the number 5, is that they can never reach the same number, and the program therefore never gets confused about who should gain access. It removes scenarios where two nodes become synced and keep denying each other by adding priority to their number.

Appendix



```
I am  3  Current number of Critical data:  3444  timestamp:  7113
I am  1  I got no access granted  7116
I am  3  I got no access granted  7113
I am  2  Current number of Critical data:  3445  timestamp:  7117
I am  1  Current number of Critical data:  3446  timestamp:  7121
I am  2  I got no access granted  7117
I am  3  I got no access granted  7118
I am  1  Current number of Critical data:  3447  timestamp:  7121
I am  2  I got no access granted  7122
I am  1  I got no access granted  7121
I am  3  I got no access granted  7123
I am  2  Current number of Critical data:  3448  timestamp:  7127
I am  1  I got no access granted  7126
I am  2  I got no access granted  7127
I am  3  Current number of Critical data:  3449  timestamp:  7128
I am  1  I got no access granted  7131
I am  3  I got no access granted  7128
I am  2  Current number of Critical data:  3450  timestamp:  7132
I am  2  I got no access granted  7132
I am  1  Current number of Critical data:  3451  timestamp:  7136
I am  3  I got no access granted  7133
I am  2  I got no access granted  7137
I am  1  I got no access granted  7136
I am  3  I got no access granted  7138
I am  2  Current number of Critical data:  3452  timestamp:  7142
I am  1  I got no access granted  7141
I am  3  Current number of Critical data:  3453  timestamp:  7143
I am  2  I got no access granted  7142
I am  1  Current number of Critical data:  3454  timestamp:  7146
```