Storage Simulation

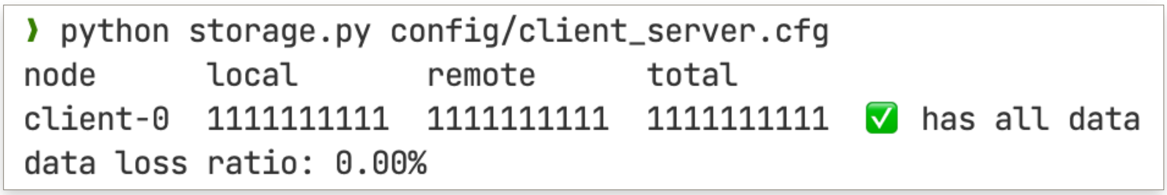
Erasure Coding

Giorgio Rengucci S4483986
Milo Galli S4793560
12 December 2022

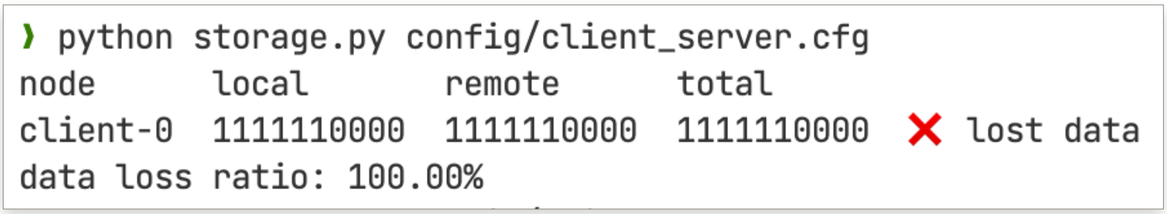
Simulation’s results analysis

Once we finished implementing the code, we tested it with the default configurations provided.

`client_server.cfg`



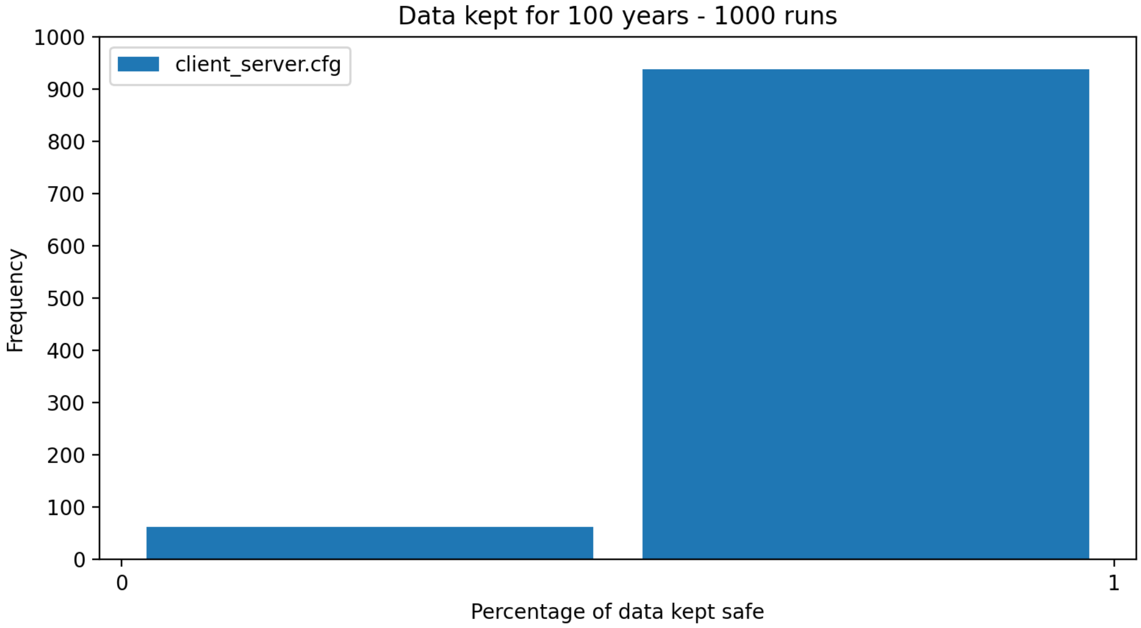
```
❯ python storage.py config/client_server.cfg
node       local       remote      total
client-0   1111111111  1111111111  1111111111  ✅ has all data
data loss ratio: 0.00%
```



```
❯ python storage.py config/client_server.cfg
node       local       remote      total
client-0   1111110000  1111110000  1111110000  ❌ lost data
data loss ratio: 100.00%
```

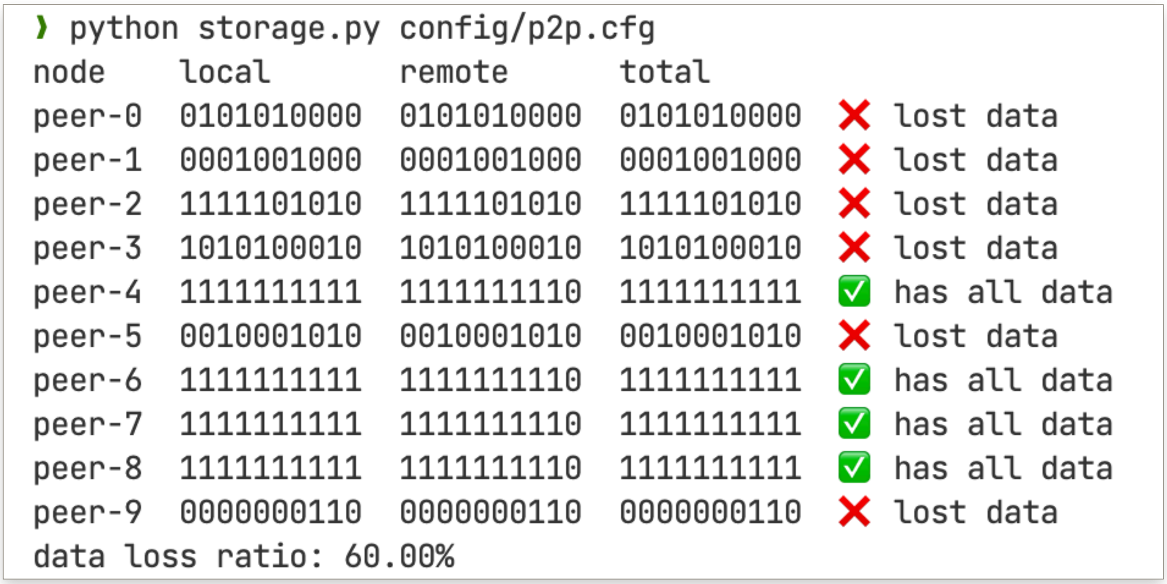
As we can see, given this configuration, there are two possible outcomes. Either all data is kept safe for 100 years or all data gets lost.

By running the simulation 1000 times we find that on the majority of runs (~940 out of 1000) the data is maintained in the system while on a smaller fraction the data gets lost because we don’t have enough blocks to restore the others.

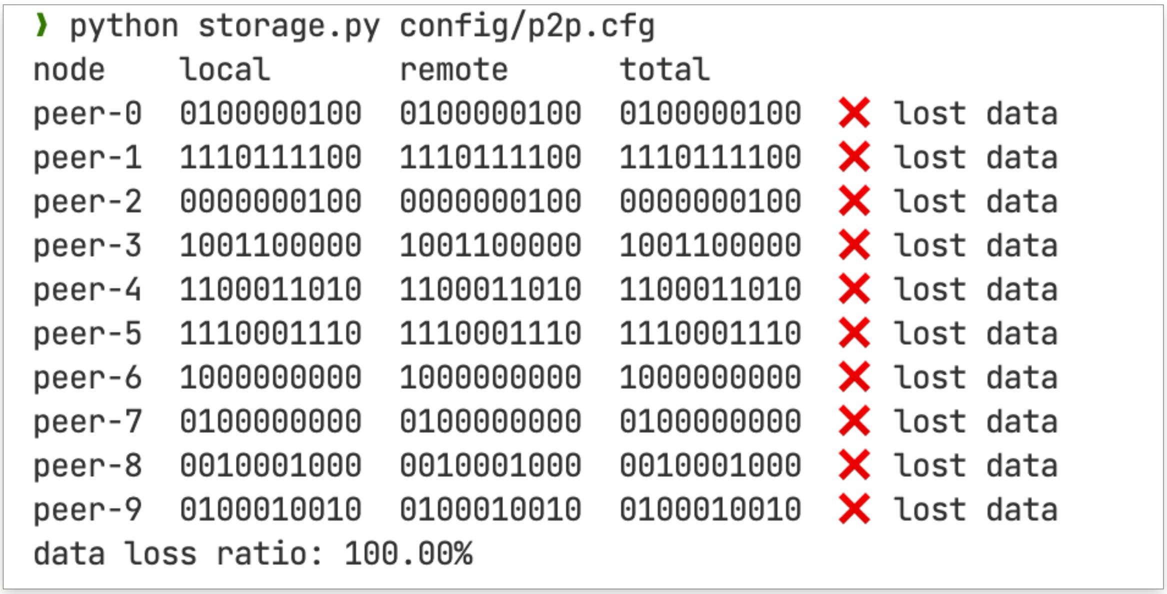


Looking back at the theory we ca see that to restore N blocks we need at least K blocks that in the current configuration are respectively 10 and 8.
The failing outcome, where we can’t restore the data, is possibly due to the slow upload and download speeds and the short average uptime, that lead to a higher probability of data loss.

p2p.cfg



```
❯ python storage.py config/p2p.cfg
node    local       remote      total
peer-0  0101010000  0101010000  0101010000  ❌ lost data
peer-1  0001001000  0001001000  0001001000  ❌ lost data
peer-2  1111101010  1111101010  1111101010  ❌ lost data
peer-3  1010100010  1010100010  1010100010  ❌ lost data
peer-4  1111111111  1111111110  1111111111  ✅ has all data
peer-5  0010001010  0010001010  0010001010  ❌ lost data
peer-6  1111111111  1111111110  1111111111  ✅ has all data
peer-7  1111111111  1111111110  1111111111  ✅ has all data
peer-8  1111111111  1111111110  1111111111  ✅ has all data
peer-9  0000000110  0000000110  0000000110  ❌ lost data
data loss ratio: 60.00%
```



```
❯ python storage.py config/p2p.cfg
node    local       remote      total
peer-0  0100000100  0100000100  0100000100  ❌ lost data
peer-1  1110111100  1110111100  1110111100  ❌ lost data
peer-2  0000000100  0000000100  0000000100  ❌ lost data
peer-3  1001100000  1001100000  1001100000  ❌ lost data
peer-4  1100011010  1100011010  1100011010  ❌ lost data
peer-5  1110001110  1110001110  1110001110  ❌ lost data
peer-6  1000000000  1000000000  1000000000  ❌ lost data
peer-7  0100000000  0100000000  0100000000  ❌ lost data
peer-8  0010001000  0010001000  0010001000  ❌ lost data
peer-9  0100010010  0100010010  0100010010  ❌ lost data
data loss ratio: 100.00%
```

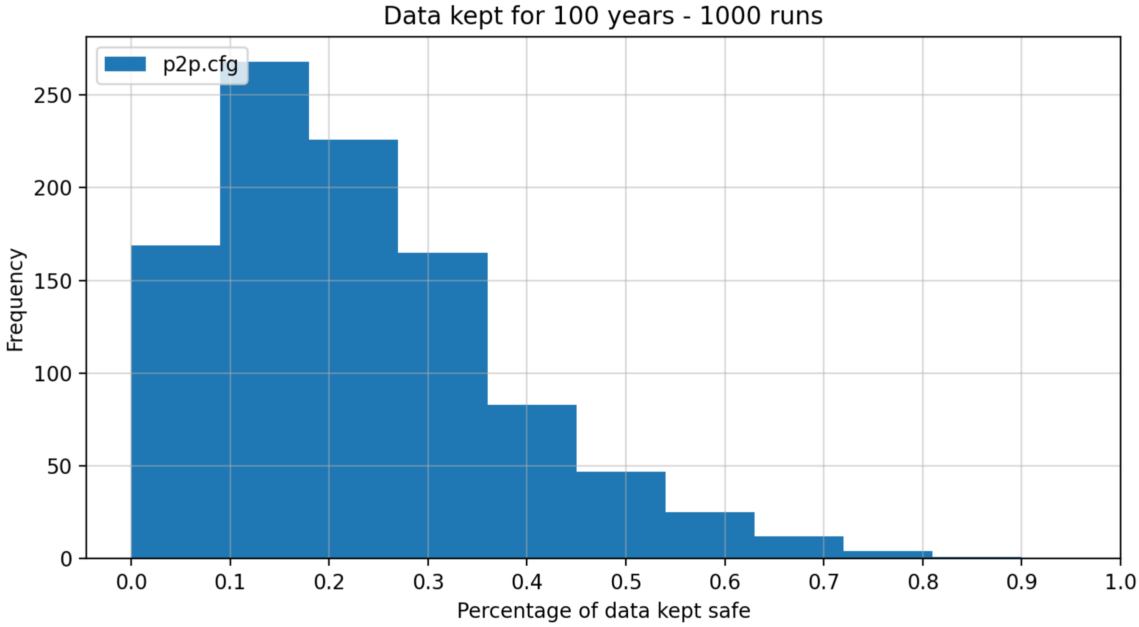
Using the p2p configuration we see that the data is never kept safe for 100 years.

The majority of times only a few peers are able to keep all their data while the others don't have enough blocks to restore the original information.

We suspect that short uptime of the machines and long recovery times lead to this results.

A great slice of the simulation's total running time is crowded with offline machines that are recovering and if the others that are up and running fail the data is lost forever.

Running the simulation 1000 times with the original client server configuration and plotting the results it's clear that the system is never able to keep the data safe for all peers of the network and the majority of times only one to three peers are able to keep the data.

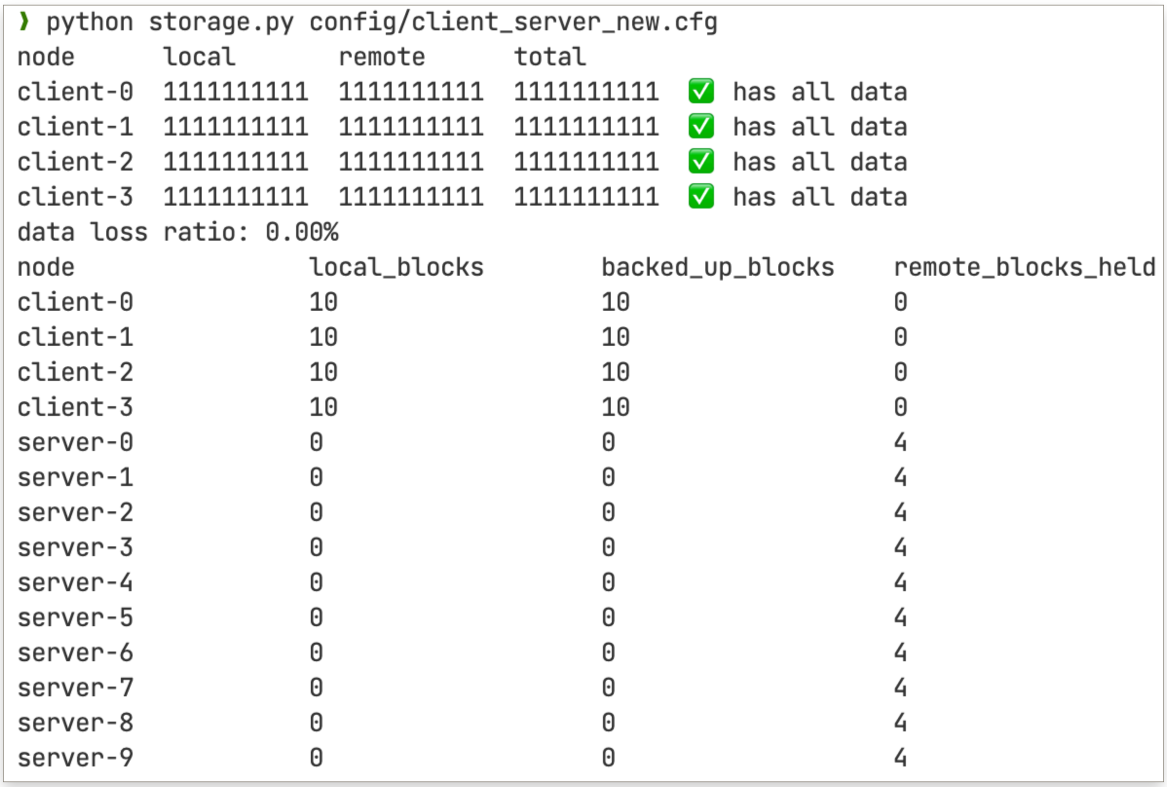


Extensions

Multiple clients in client_server simulation

The original code doesn’t support simulation of multiple clients on multiple distributed servers since every peer of the network can freely interact with the others.

We fixed that by adding new checks on schedule_next_upload and schedule_next_download, specifically each time an upload or download is scheduled, it checks whether the node is a client or a server and behaves differently in each case.



```
❯ python storage.py config/client_server_new.cfg
node      local       remote      total
client-0  1111111111  1111111111  1111111111  ✅ has all data
client-1  1111111111  1111111111  1111111111  ✅ has all data
client-2  1111111111  1111111111  1111111111  ✅ has all data
client-3  1111111111  1111111111  1111111111  ✅ has all data
data loss ratio: 0.00%
node                local_blocks        backed_up_blocks    remote_blocks_held
client-0            10                  10                  0
client-1            10                  10                  0
client-2            10                  10                  0
client-3            10                  10                  0
server-0            0                   0                   4
server-1            0                   0                   4
server-2            0                   0                   4
server-3            0                   0                   4
server-4            0                   0                   4
server-5            0                   0                   4
server-6            0                   0                   4
server-7            0                   0                   4
server-8            0                   0                   4
server-9            0                   0                   4
```

Even though we have multiple clients all the remote blocks are sent to the servers so no client is backing up data on other clients.

Modifications to p2p.cfg

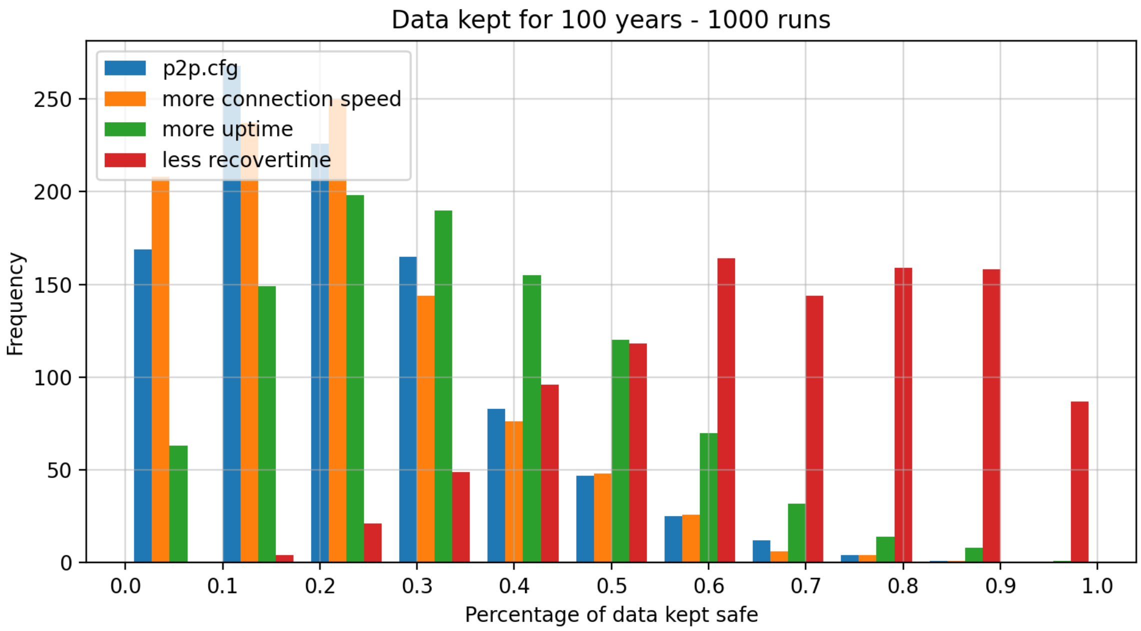
The default configuration for p2p provided isn’t able to keep data safe for 100 years as said before.

Considering this, we ran the simulation with different configurations to find which parameter better improves data retention.

The modified parameters are:

- Faster connection speeds: 1000Mbps up and down
- More peer uptimes: 16 hours up, 8 hours down
- Less peer recover times: 1 day instead of 3 days

We ran each modified configuration 1000 times and plotted the results

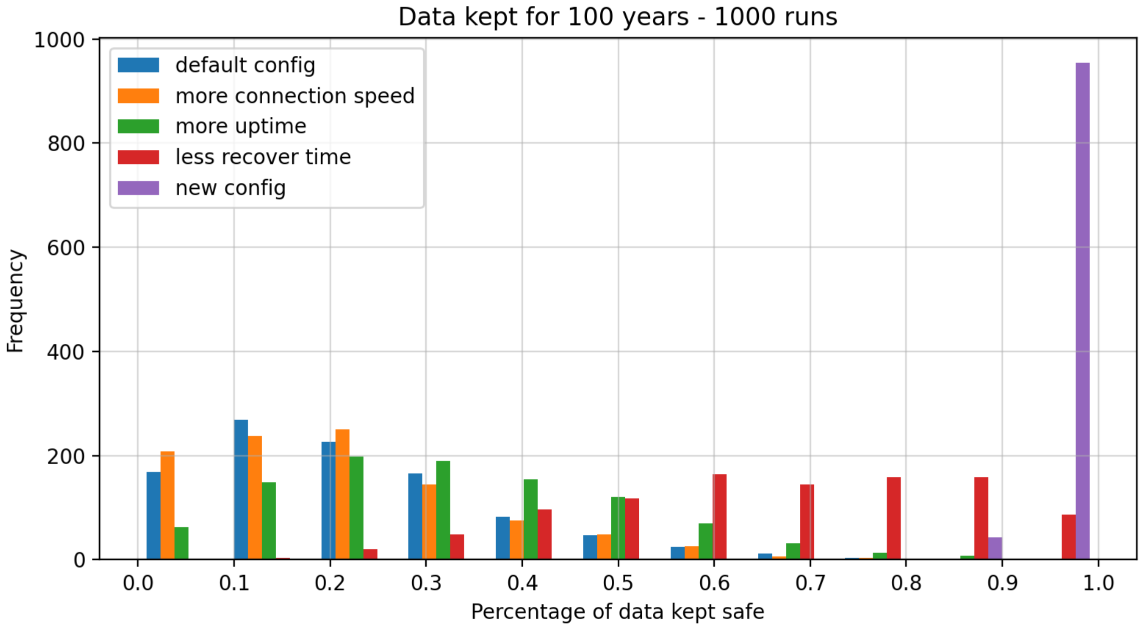


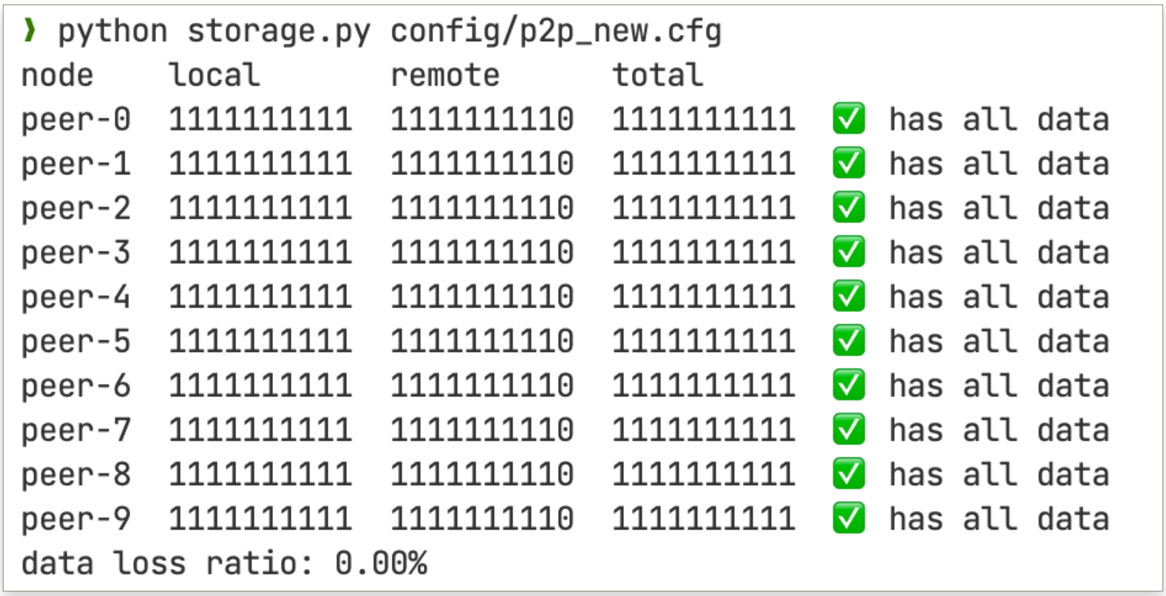
From the histogram is clear that each modified parameter slightly improves the results but recover time of peers is by far the parameter that most influences data retention in the system.

From these findings we changed the original configuration as a combination of the before mentioned parameters, more specifically

- Upload speed is increased from 2MiB to 1000MiB
- Download speed is increased from 10MiB to 1000MiB
- Average uptime from 8 hours is increased to 16 hours
- Average downtime is reduced from 16 hours to 8 hours
- Average recover time is increased from 3 days to 1 hour

We again run the simulation 1000 times and plotted the result in a histogram





```
❯ python storage.py config/p2p_new.cfg
node    local       remote      total
peer-0  1111111111  1111111110  1111111111  ✅ has all data
peer-1  1111111111  1111111110  1111111111  ✅ has all data
peer-2  1111111111  1111111110  1111111111  ✅ has all data
peer-3  1111111111  1111111110  1111111111  ✅ has all data
peer-4  1111111111  1111111110  1111111111  ✅ has all data
peer-5  1111111111  1111111110  1111111111  ✅ has all data
peer-6  1111111111  1111111110  1111111111  ✅ has all data
peer-7  1111111111  1111111110  1111111111  ✅ has all data
peer-8  1111111111  1111111110  1111111111  ✅ has all data
peer-9  1111111111  1111111110  1111111111  ✅ has all data
data loss ratio: 0.00%
```

Given these changes the simulation is able to keep data for 100 years on every peer of the network more than 95% of the times.