Assignment 2

Distributed Computing – Mohammad Hossein Mahrooghi – Universita di Genova 2026

Introduction

- In this assignment, we study a peer-to-peer backup system based on erasure coding, using a discrete-event simulator.
- The goal of the simulation is to evaluate the long-term reliability of data storage in a distributed peer-to-peer environment, where nodes frequently connect and disconnect, and may permanently fail, losing all locally stored data.
- Each node encodes its data into multiple blocks using erasure coding and stores the encoded blocks on other peers.
- A node can recover its entire data as long as at least k encoded blocks remain available somewhere in the network.
- By simulating very long time spans, we investigate whether data remains available under different system configurations, and to what extent data loss occurs. We evaluate different deployment scenarios and study how system parameters affect data durability.

Implementation details and execution parameters are reported in the Appendix.

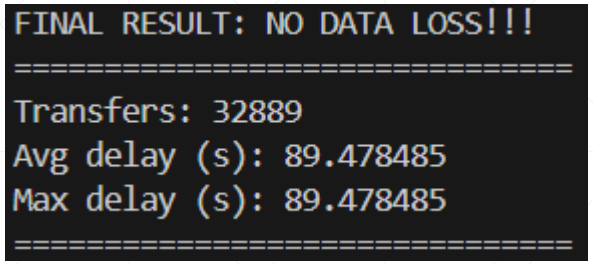
System Model and Simulation Overview

- The simulator models a peer-to-peer backup system composed of multiple nodes.
- Each node:
  - owns some data to back up
  - encodes its data into n blocks using erasure coding
  - can recover its data as long as at least k blocks are available in the system (k/n)
- Nodes frequently go online and offline.
- Nodes may also permanently fail, losing all locally stored data.
- After a failure, nodes attempt to recover their data by downloading blocks from other peers in the network.

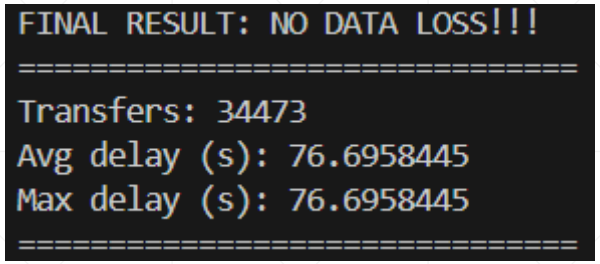
Simulation Setup and Configurations

- The simulator is used to model very long time periods (e.g., several decades or up to 100 years) in order to evaluate long-term data durability.
- Multiple simulation runs are performed for each configuration to obtain statistically meaningful results.
- Two main configurations are considered:
  - p2p.cfg: a fully peer-to-peer system where all nodes have similar roles
  - client_server.cfg: a client–server–like system where some nodes provide storage but do not own data

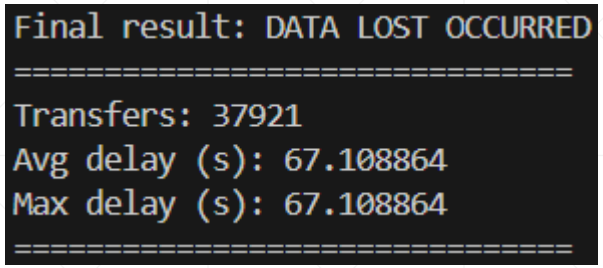
P2P Results – Effect of the Recovery Threshold k



Result for n = 10, k = 6



Result for n = 10, k = 7



Result for n = 10, k = 8

Peers = 20, average lifetime = 1 year, data size = 1 GiB, storage size = 10 GiB

Transfers: total number of block transfers performed by the system.

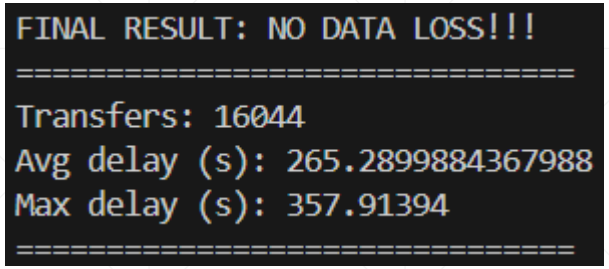
Avg delay: average time required to complete a block transfer.

Max delay: longest block transfer observed during the simulation.

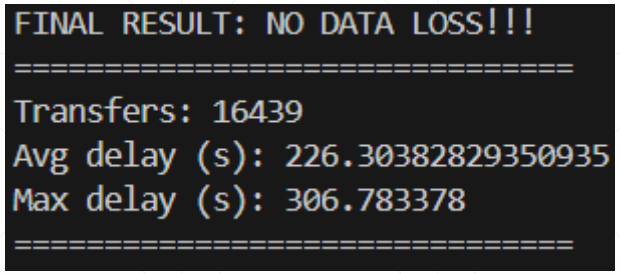
Interpretation of Experimental Results

- Increasing the recovery threshold k reduces redundancy, making data recovery progressively more difficult.
- As k increases, the system performs fewer successful recovery operations, and data loss becomes more likely.
- Higher values of k are associated with increased transfer delays: recovery operations take longer, and worst-case delays increase, indicating higher stress on the system before data loss occurs.
- As observed in the results, the average and maximum transfer delays are equal. This happens because all nodes have the same upload and download speeds in this simulation, which leads to the same delay for every transfer.
- Choosing k is a critical design decision in erasure-coded backup systems.

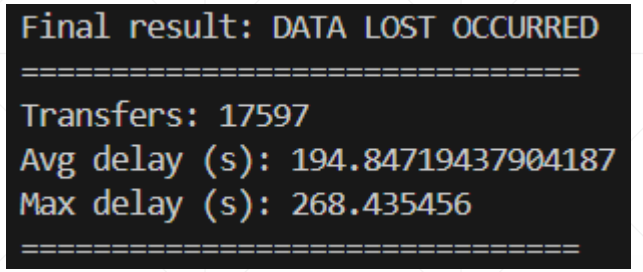
Client–Server Configuration Results



Result for n = 10, k = 6



Result for n = 10, k = 7



Result for n = 10, k = 8

servers = 10, nodes = 10 average lifetime = 1 year, data size = 1 GiB, nodes storage size = 2 GiB, servers storage size = 1 TiB

Interpretation (Client–Server Scenario)

- In this configuration, the system consists of two different node classes: clients (data owners) and servers (storage providers), with different upload/download speeds and stability characteristics.
- In the runs with NO DATA LOSS, data remains available thanks to the presence of more stable storage servers. However, recovery operations are often bottlenecked by server-side upload capacity and concurrent downloads, resulting in higher average and maximum transfer delays.
- In the run with DATA LOST, the system performs even more transfers, but recovery ultimately fails because not enough blocks remain available to reach k. This occurs when critical blocks are placed on a small number of storage providers or when stable nodes temporarily or permanently leave the system.
- These results show that data durability depends not only on redundancy (n, k), but also on block placement, node stability, and available upload/download bandwidth.

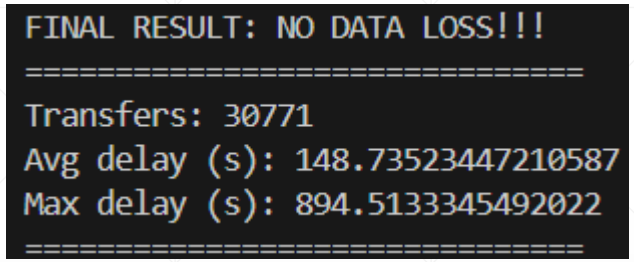
Simulator Extension (Upload and Download Noise)

- In the base simulator, upload and download speeds are constant.
- In real-world systems, network bandwidth fluctuates over time due to congestion, contention, and external factors.
- This extension introduces noise on upload and download speeds to model more realistic network behavior(Noise is applied dynamically during transfers).

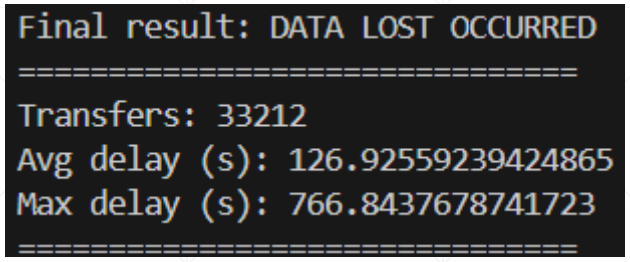
- **Implementation Overview**
- Upload and download speeds are perturbed by a random noise factor at transfer time.
- The noise affects only the transfer duration, while preserving the average bandwidth over time.

P2P Configuration Results

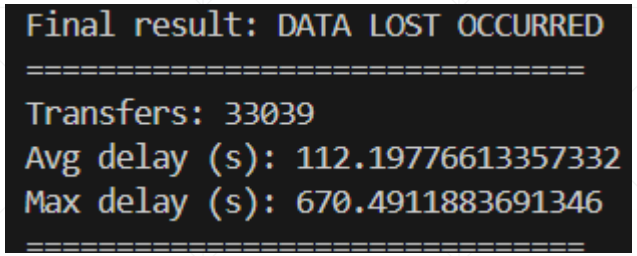
With Noised upload and download speed



Result for n = 10, k = 6



Result for n = 10, k = 7



Result for n = 10, k = 8

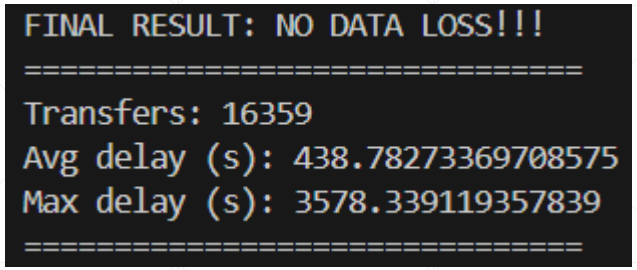
Peers = 20, average lifetime = 1 year, data size = 1 GiB, storage size = 10 GiB, upload noise = 0.9, download noise = 0.9

Results with Upload/Download Noise (p2p)

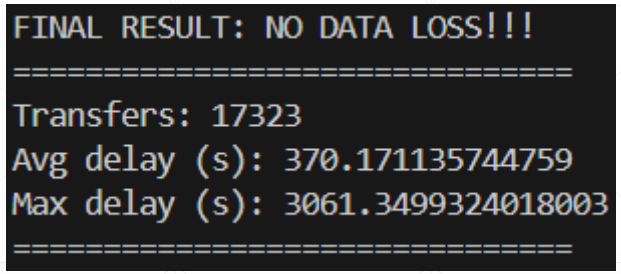
- With strong bandwidth noise (upload_noise = 0.9, download_noise = 0.9), the system exhibits high variability in transfer times.
- For k = 6, the system remains stable and no data loss occurs. The system survives long enough for very slow transfers to complete, resulting in high maximum transfer delays.
- For k = 7 and k = 8, redundancy is insufficient to tolerate bandwidth fluctuations. Recovery operations are delayed, failures overlap, and data loss occurs. In these cases, the system fails earlier, leading to lower observed maximum delays.

Client–Server Configuration Results

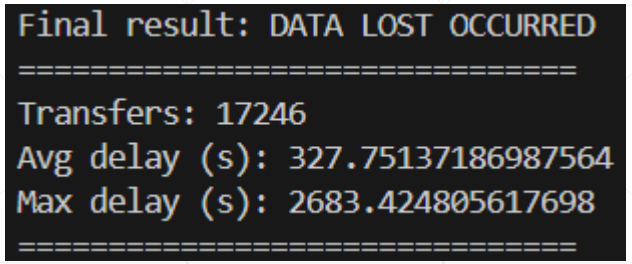
With Noised upload and download speed



Result for n = 10, k = 6



Result for n = 10, k = 7



Result for n = 10, k = 8

Peers = 20, average lifetime = 1 year, data size = 1 GiB, storage size = 10 GiB

Node: upload noise = 0.9, download noise = 0.9

Server: upload noise = 0.1, download noise = 0.1

Results with Upload/Download Noise (client-server)

- In this experiment, bandwidth noise is modeled asymmetrically. Client nodes use upload_noise = 0.9 and download_noise = 0.9,while server nodes use upload_noise = 0.1 and download_noise = 0.1.
- Compared to the pure peer-to-peer experiments, the client–server model shows higher robustness under the same redundancy level.
- In previous experiments, data loss occurred already for k = 7. In contrast, in the client–server configuration, the system remains durable for both k = 6 and k = 7, despite strong bandwidth noise on client nodes.
- This indicates that the presence of stable storage servers significantly improves recovery reliability, allowing the system to tolerate higher values of k before data loss occurs.
- When k = 8, redundancy becomes insufficient even in the client–server setup, and data loss is eventually observed.

Final Conclusions

- Data durability strongly depends on the erasure coding parameters (n, k). Higher values of k reduce redundancy and increase the risk of data loss.
- Compared to a pure peer-to-peer system, the client–server configuration is more robust, tolerating higher values of k before data loss occurs.
- Bandwidth variability plays a critical role: strong upload/download noise can cause data loss even when average bandwidth remains unchanged.
- Stable storage servers significantly improve recovery reliability, at the cost of higher recovery delays due to bandwidth bottlenecks.

Use of External Resources

- This assignment was completed individually.
- Large language models (ChatGPT) were extensively used as a support and learning tool, both to understand how the provided simulator code works and to assist in the implementation of the assignment, including writing and modifying parts of the code. ChatGPT was also used to help with Python syntax, code structuring, and implementation ideas, due to limited prior experience with Python.
- In addition, ChatGPT was used to support the writing of the report, helping with structure, clarity, and grammatical correctness.
- All design decisions, simulator modifications and experimental setups were independently carried out by the author.

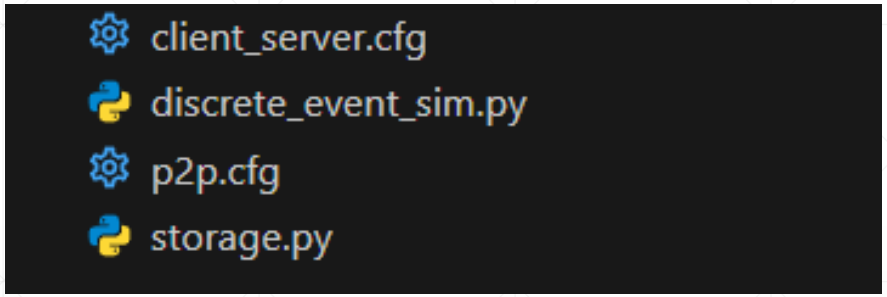
Project Repository

The source code of the simulator is available at:

https://github.com/Argon2002/DC_Assignment2.git

Appendix – Implementation Details

Files overview



A quick look of the functionality of each file:

- storage.py: main simulator implementing a peer-to-peer, client-server backup system based on erasure coding, including node behavior, failures, recovery, and data transfers.
- discrete_event_sim.py: core discrete-event simulation engine used to manage simulation time and schedule events.
- p2p.cfg: configuration file defining a peer-to-peer system where all nodes have **similar characteristics** and store data for each other.
- client_server.cfg: configuration file modeling a **client-server–like** system, where some nodes do not own data but provide storage to others.

Running the Simulator

- The simulator is executed from the command line and requires a configuration file describing the classes of nodes to simulate (e.g., p2p.cfg or client_server.cfg).
- Example command:

```
python storage.py p2p.cfg --max-t "100 years" --seed 1
```

- Parameters
  - cfg: the system configuration (p2p.cfg or client_server.cfg)
  - --max-t (or simulation time): e.g., 100 years, 10 minutes …
  - --seed: random seed (enables reproducible runs)

Thanks for your attention!

Mohammad Hossein Mahrooghi – S7499834

Università di Genova