SAPIENZA
UNIVERSITÀ DI ROMA

Design and Implementation of a Replicated File Storage System Using Conflict-free Replicated Data Types

Department of Computer, Control and Management Engineering “Antonio Ruberti”
Master’s Degree in Engineering in Computer Science

Candidate: Giorgio Francescone (*Matr. 1974461*)
Thesis Advisor: Prof. Andrea Vitaletti

January 28, 2026

PROBLEM STATEMENT

Introduction

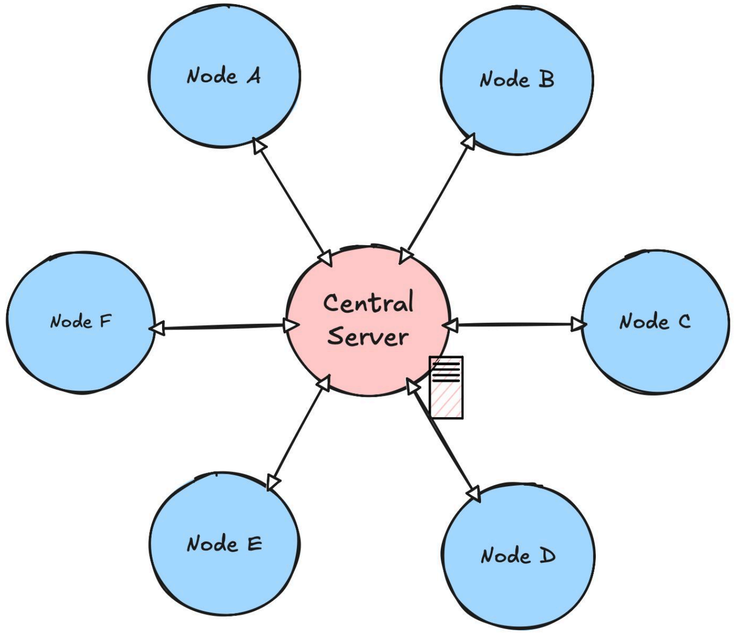
The shift towards distributed systems

- Modern systems distribute the workload onto **multiple machines** (as opposed to handling everything on a single, powerful machine)
- **Key benefits:** *scalability*, *availability*, *efficiency*

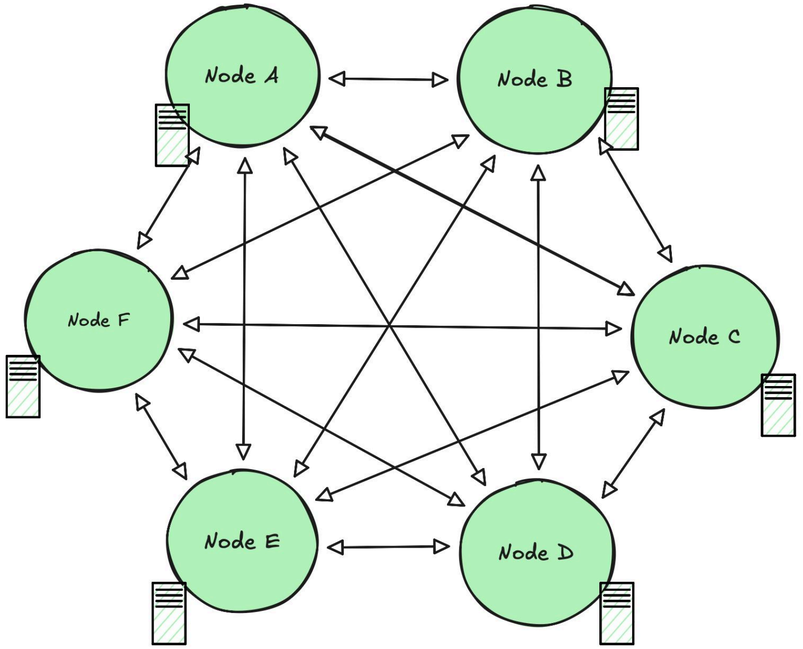
The challenges of collaborative applications

- Real-time, concurrent editing of shared data by multiple users (e.g., *Google Docs*)
  - How is everything kept **consistent**?
  - How to seamlessly handle *conflicts*?

Approaches to Replication

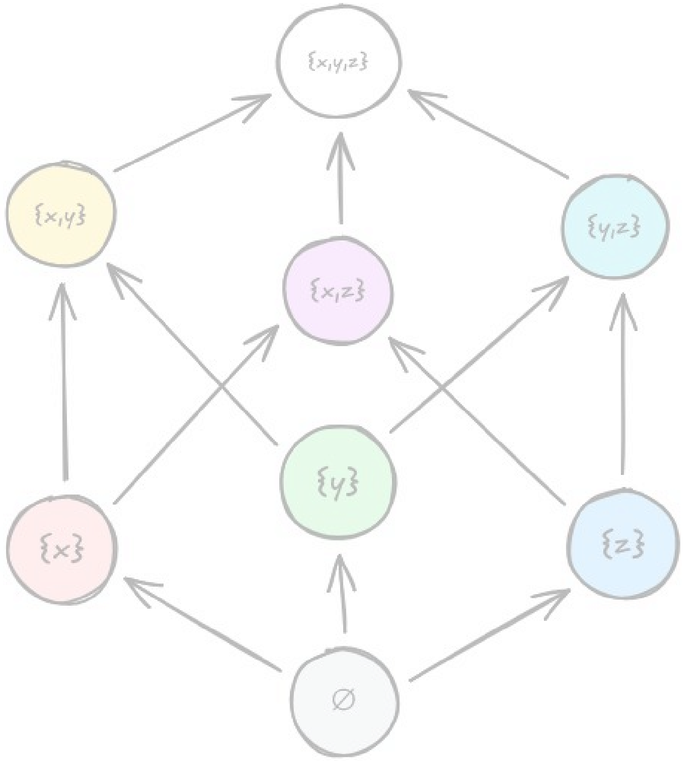


Centralized replication: a *central server* coordinates operations for the entire network and resolves conflicts



Decentralized replication: Each node individually handles (i.e. *merges*) operations received from other nodes — no need for coordination

Foundational Concepts on CRDTs



1

Strong Consistency

All nodes in the system see the same data at the same time

2

Eventual Consistency (EC)

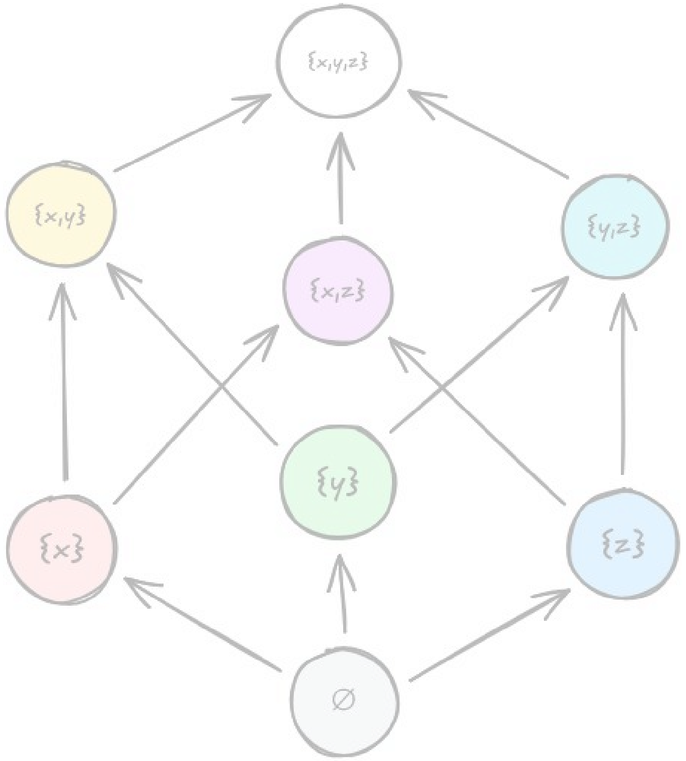
Once updates are no longer being issued, all nodes in the system *eventually* converge to the same version of data

3

Strong Eventual Consistency (SEC)

Extension of EC which ensures *Strong Convergence* (nodes that receive the same updates converge to the same state)

Foundational Concepts on CRDTs



1

Strong Consistency

All nodes in the system see the same data at the same time

2

Eventual Consistency (EC)

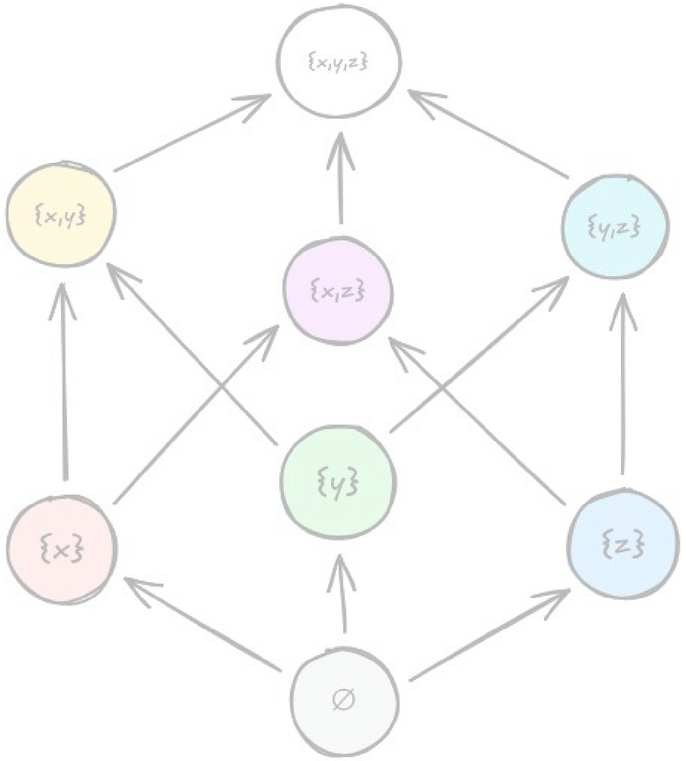
Once updates are no longer being issued, all nodes in the system *eventually* converge to the same version of data

3

Strong Eventual Consistency (SEC)

Extension of EC which ensures *Strong Convergence* (nodes that receive the same updates converge to the same state)

Foundational Concepts on CRDTs



1

Strong Consistency

All nodes in the system see the same data at the same time

2

Eventual Consistency (EC)

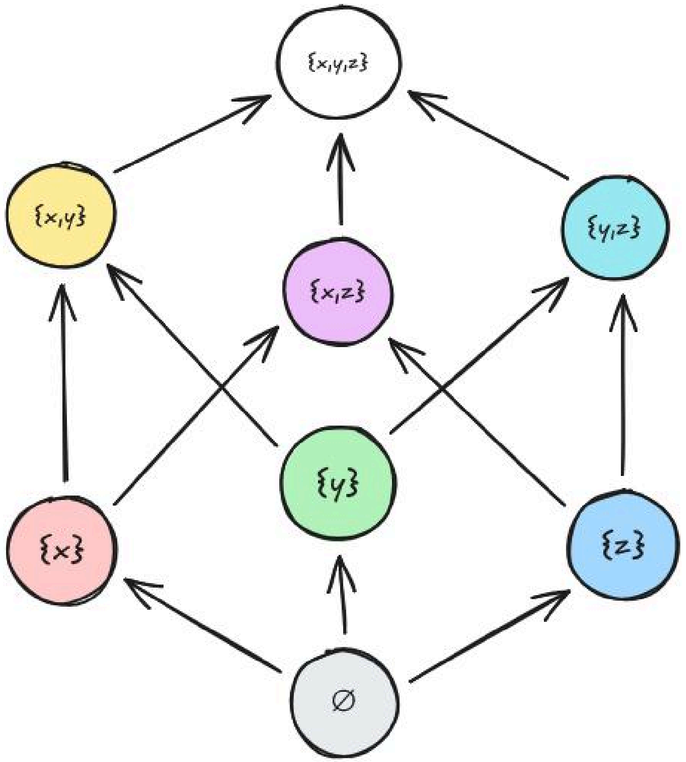
Once updates are no longer being issued, all nodes in the system *eventually* converge to the same version of data

3

Strong Eventual Consistency (SEC)

Extension of EC which ensures *Strong Convergence* (nodes that receive the same updates converge to the same state)

Foundational Concepts on CRDTs

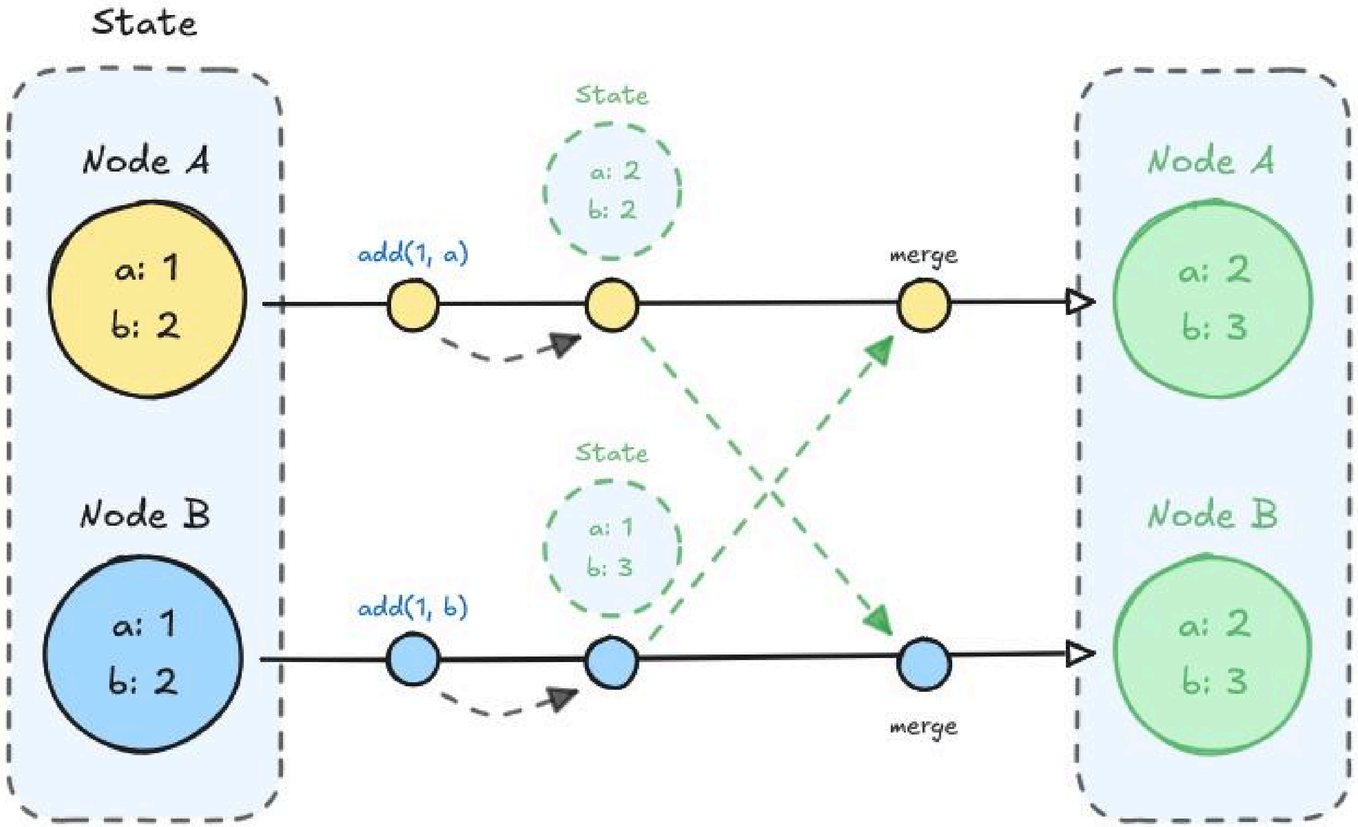


Conflict-free Replicated Data Types (CRDTs) are structures that enable *Strong Eventual Consistency* in a principled manner, allowing replicas in distributed systems to be updated independently and concurrently [1].

Replicas that have received the same state of updates will *eventually converge* to the same state, without requiring coordination or consensus protocols.

[1] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, “Conflict-free replicated data types”, In *Proceedings of the 13th International Conference on Stabilization, Safety, and Security of Distributed Systems*, 2011, pp. 386–400. doi: 10.1007/978-3-642-24550-3_29.

Types of CRDT



State-based CRDTs

Broadcast the entire state to other nodes.

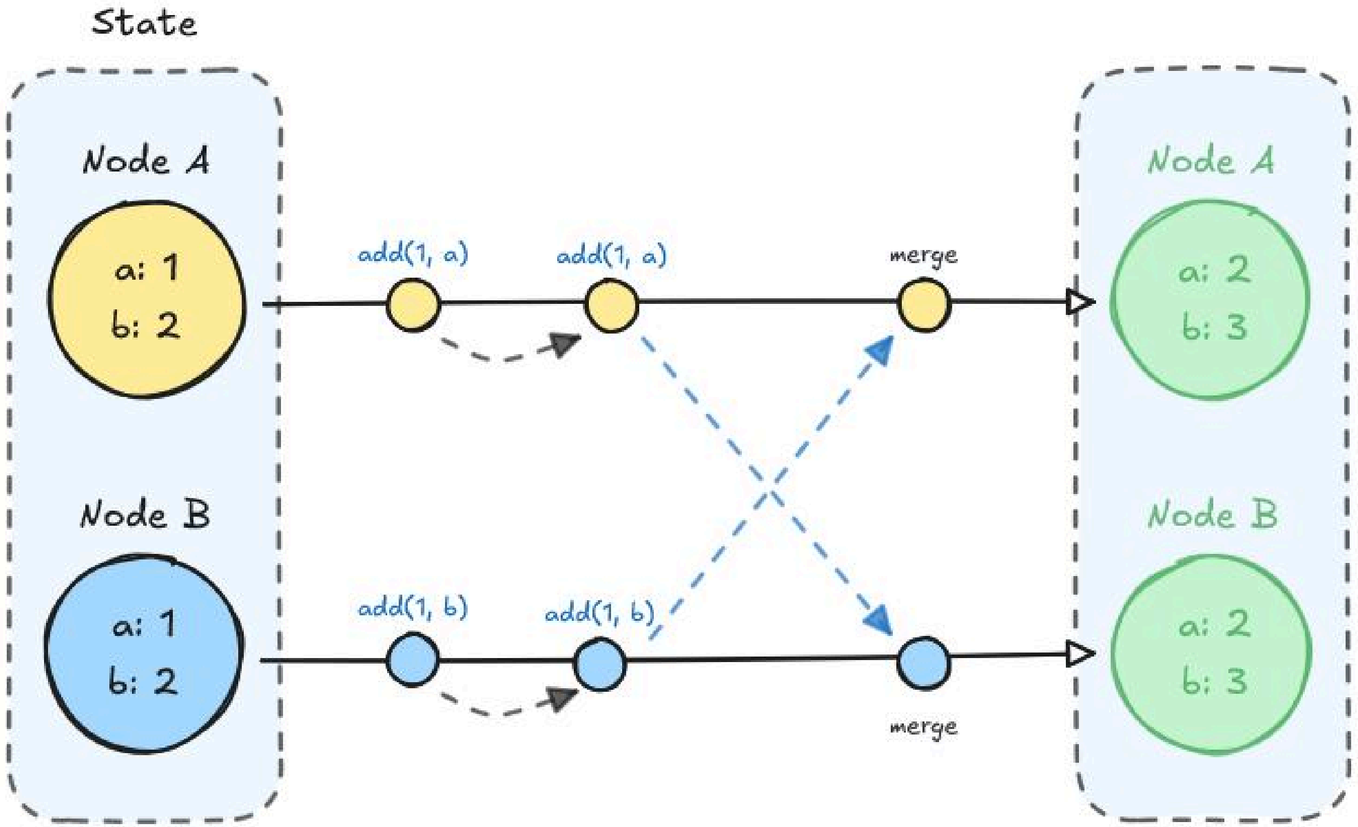
Operation-based CRDTs

Broadcast individual operations to other nodes.

δ state-based CRDTs

Transmit a synthetic state containing only the incremental changes.

Types of CRDT



State-based CRDTs

Broadcast the entire state to other nodes.

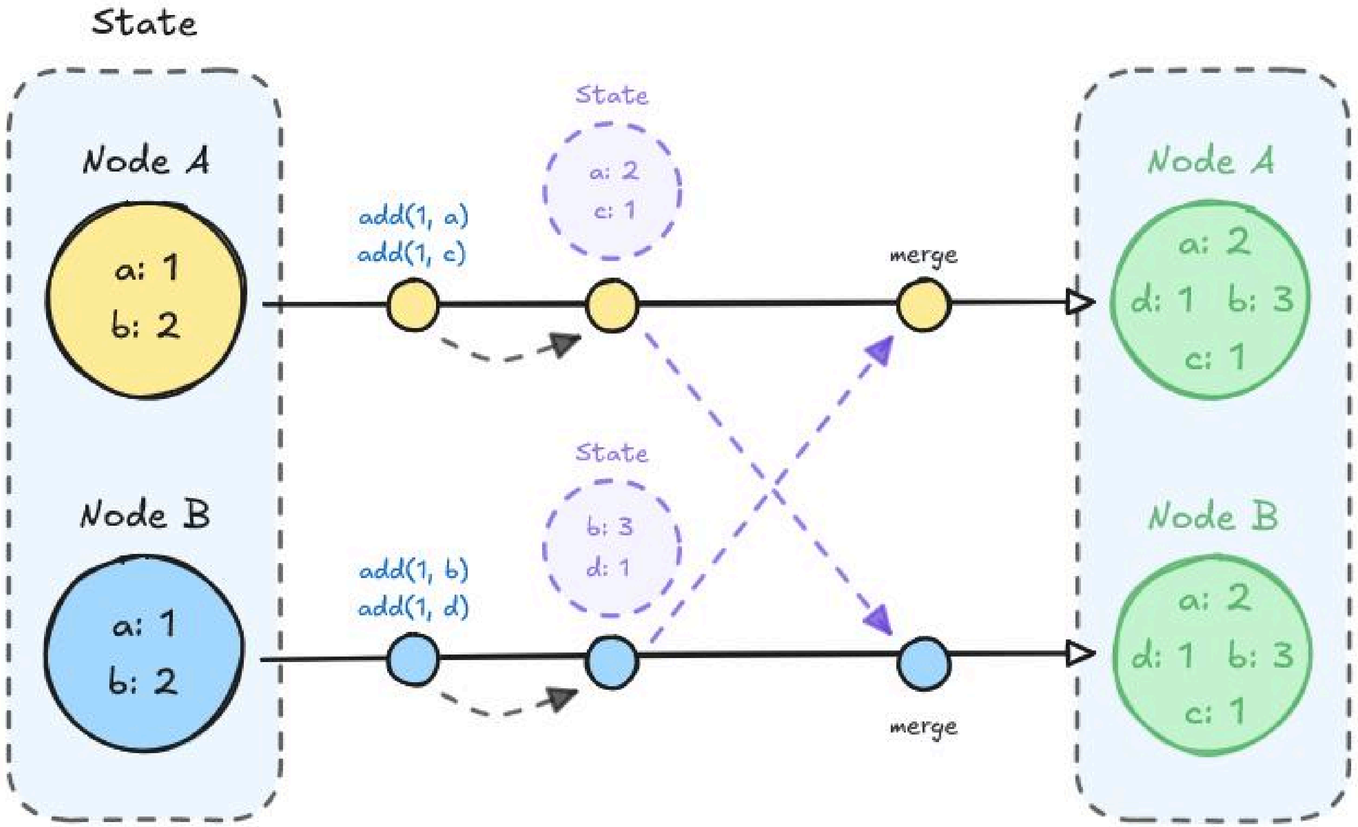
Operation-based CRDTs

Broadcast individual operations to other nodes.

δ state-based CRDTs

Transmit a synthetic state containing only the incremental changes.

Types of CRDT



State-based CRDTs

Broadcast the entire state to other nodes.

Operation-based CRDTs

Broadcast individual operations to other nodes.

δ state-based CRDTs

Transmit a synthetic state containing only the incremental changes.

Advanced CRDT Frameworks

Yjs is a high-performance CRDT framework for building collaborative applications that sync automatically[3].

- It exposes its internal CRDT model as **shared data types** that can be manipulated concurrently.
- It allows for modularity by supporting **several communication protocols** (e.g., *WebRTC*, *WebSockets*, *XMPP*).

[3] P. Nicolaescu, K. Jahns, M. Derntl, and R. Klamma. “Yjs: A Framework for Near Real-Time P2P Shared Editing on Arbitrary Data Types”. In *15th International Conference on Web Engineering*. ICWE 2015. Springer LNCS volume 9114, June 2015, pp. 675–678. doi: 10.1007/978-3-319-19890-3_55.

Building a Replicated File Storage System using CRDTs

The Proof-of-Concept

A **decentralized file storage** where users can *share files* and certify ownership through *digital signatures*.

Key features

- **File insertion** and **removal**
- On-demand **digital signature validation**
- **Peer-to-peer syncrhonization**
- **File retention policy** (to prevent storage overhead)

SYSTEM ARCHITECTURE

Building a Replicated File Storage System using CRDTs

crdtsign

CRDT Storage

Map-like data type that holds the raw files and their metadata (including their digital signature)

Backend

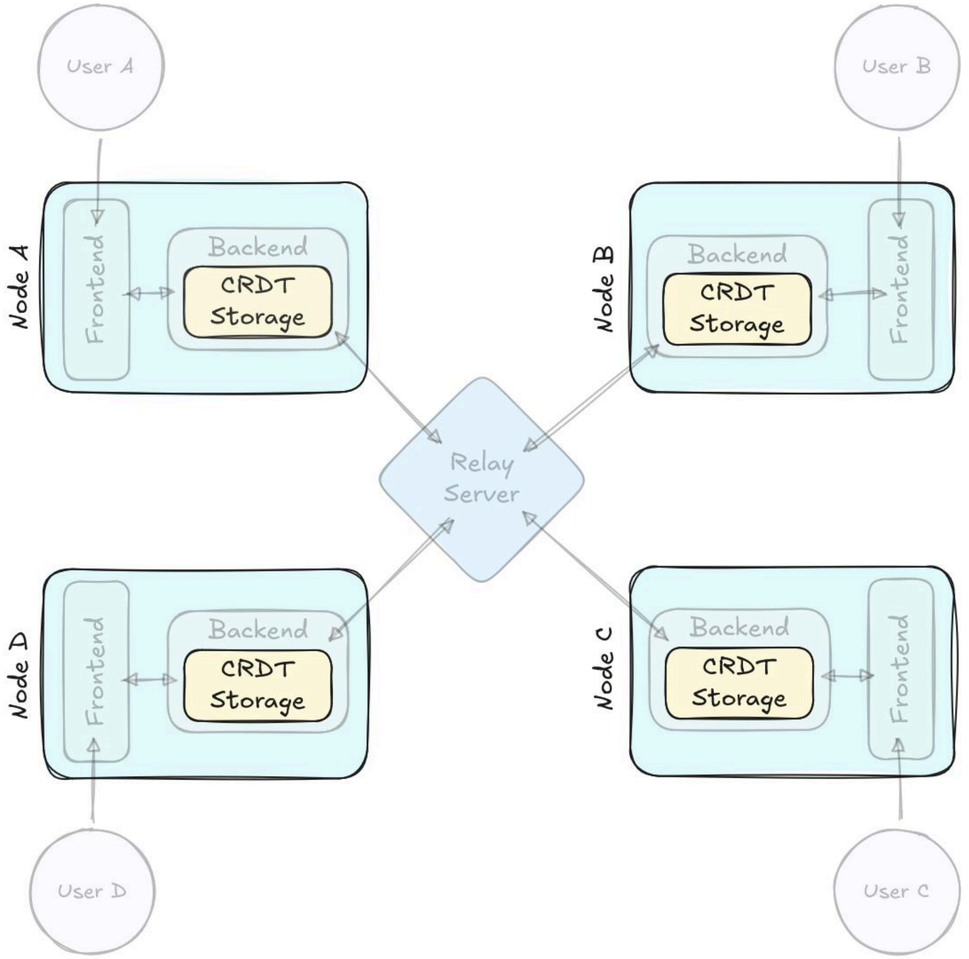
Manages the *CRDT Storage* and the *File Signature* module

Relay Server

Simulates a P2P environment by conveying updates from one node to the others

Frontend

Provided to the *User* as a way to interact with the *Backend*



SYSTEM ARCHITECTURE

Building a Replicated File Storage System using CRDTs

crdtsign

CRDT Storage

Map-like data type that holds the raw files and their metadata (including their digital signature)

Backend

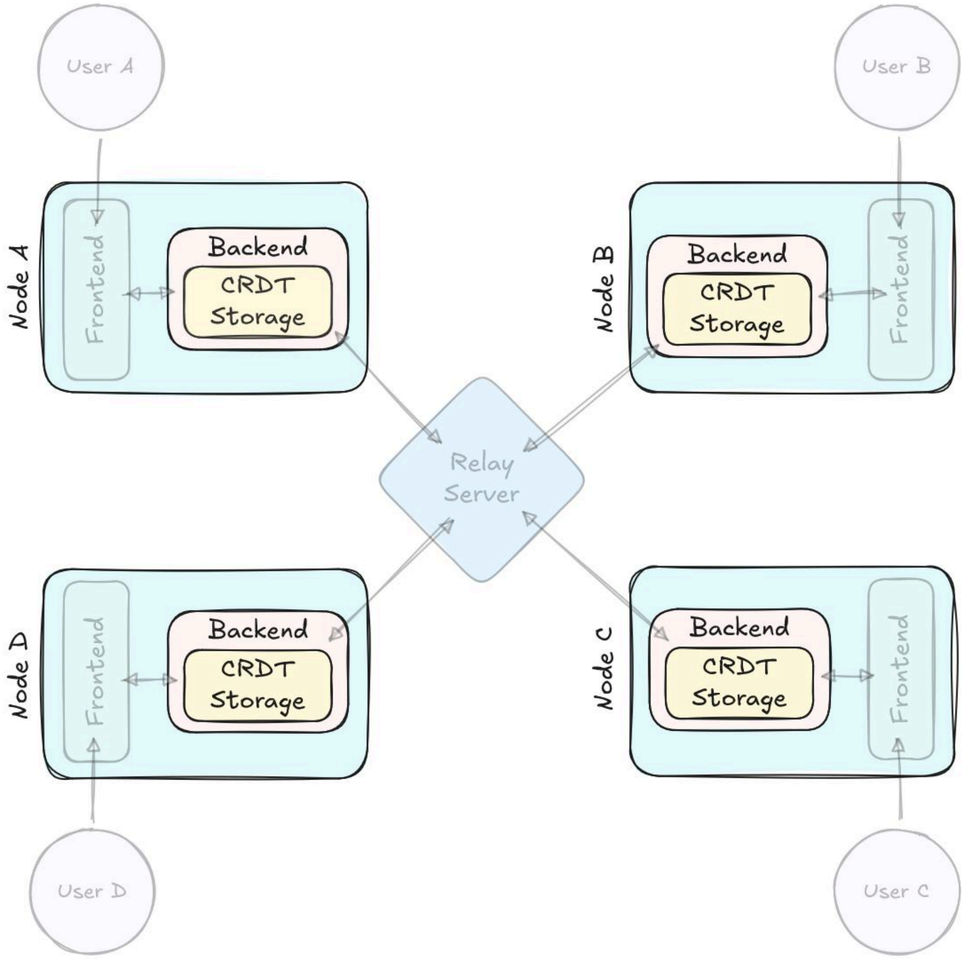
Manages the *CRDT Storage* and the *File Signature* module

Relay Server

Simulates a P2P environment by conveying updates from one node to the others

Frontend

Provided to the *User* as a way to interact with the *Backend*



SYSTEM ARCHITECTURE

Building a Replicated File Storage System using CRDTs

crdtsign

CRDT Storage

Map-like data type that holds the raw files and their metadata (including their digital signature)

Backend

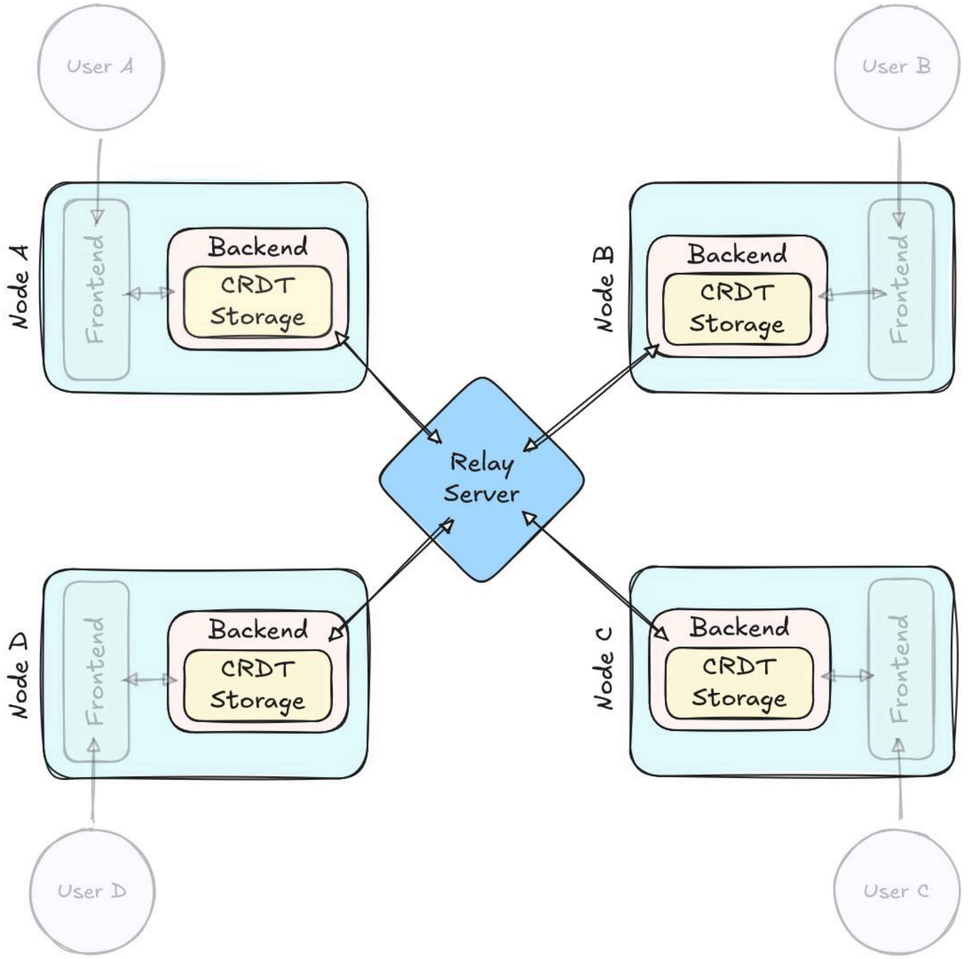
Manages the *CRDT Storage* and the *File Signature* module

Relay Server

Simulates a P2P environment by conveying updates from one node to the others

Frontend

Provided to the *User* as a way to interact with the *Backend*



SYSTEM ARCHITECTURE

Building a Replicated File Storage System using CRDTs

crdtsign

CRDT Storage

Map-like data type that holds the raw files and their metadata (including their digital signature)

Backend

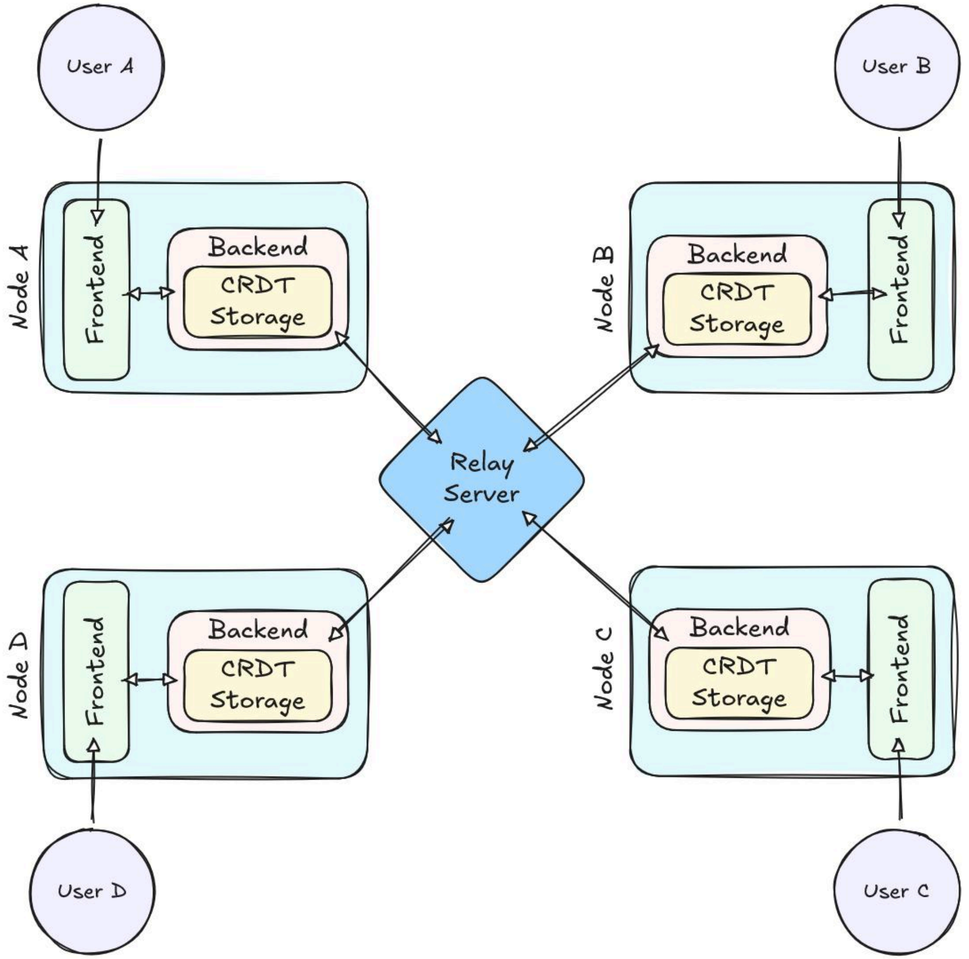
Manages the *CRDT Storage* and the *File Signature* module

Relay Server

Simulates a P2P environment by conveying updates from one node to the others

Frontend

Provided to the *User* as a way to interact with the *Backend*

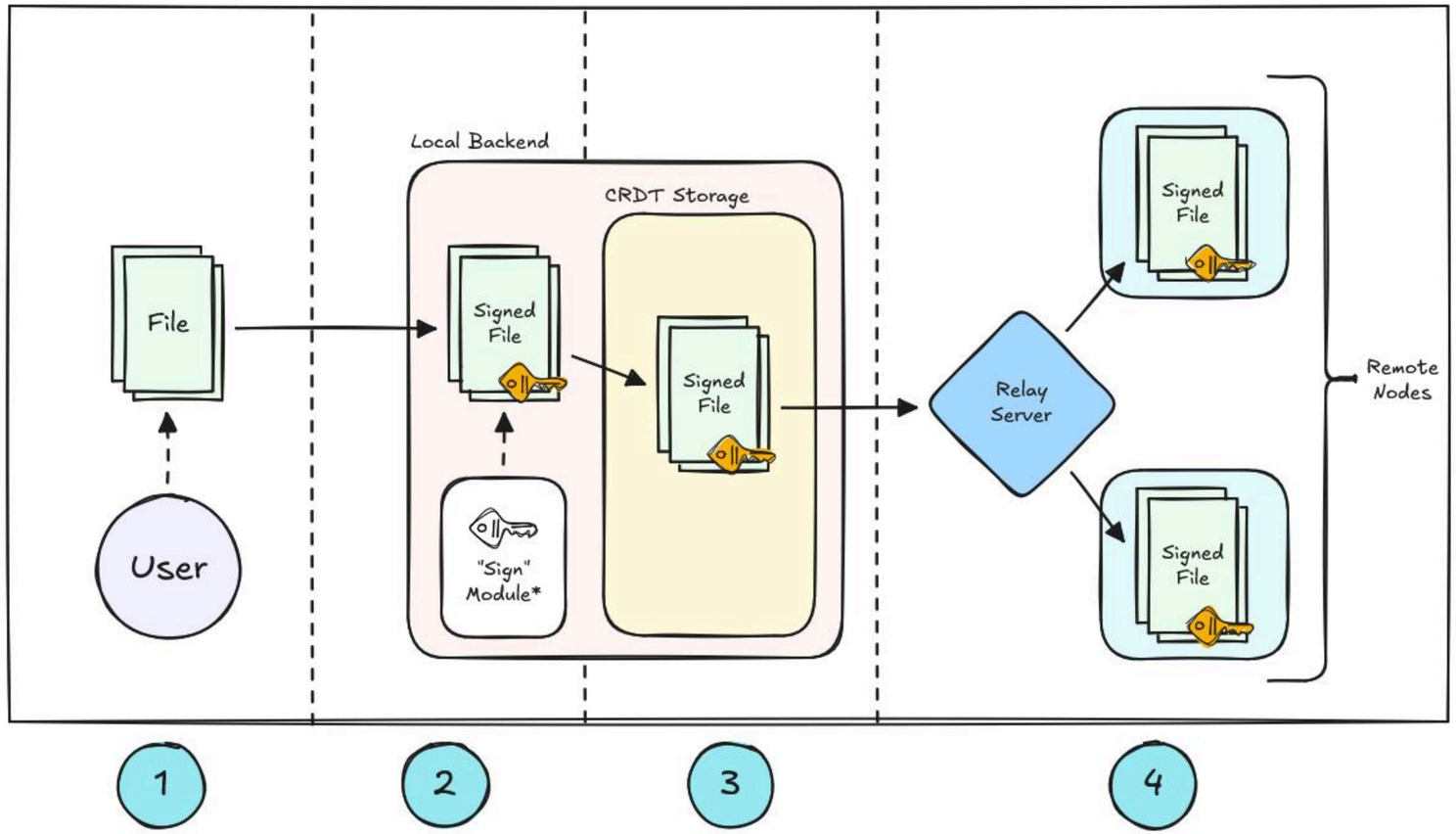


EXAMPLE - FILE INSERTION

Building a Replicated File Storage System using CRDTs

crdtsign

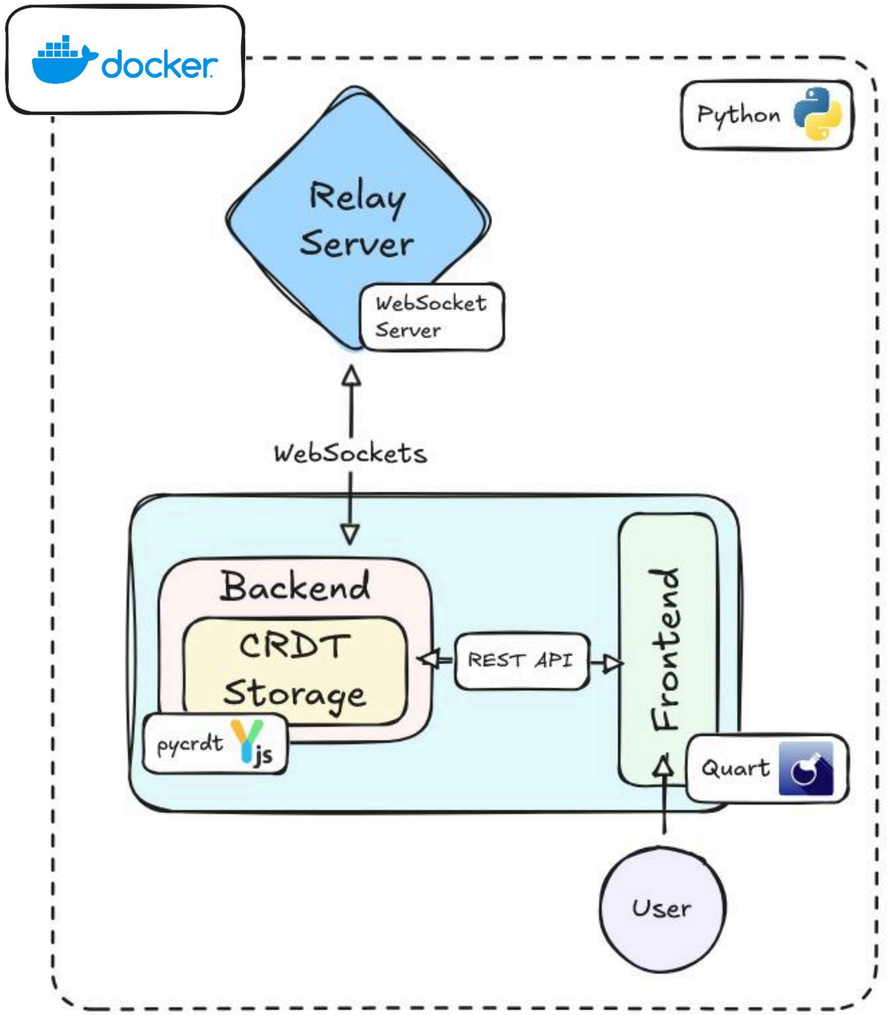
1. The user **uploads** a file
2. The Backend **signs** the file
3. The Backend **saves** the file to CRDT storage
4. The Backend **sends** the file to other nodes (via the Relay Server)



* The *"Sign" Module* uses the *Ed25519 signature scheme* to generate a valid and immutable signature associated to the user's pre-computed keypair.

Building a Replicated File Storage System using CRDTs

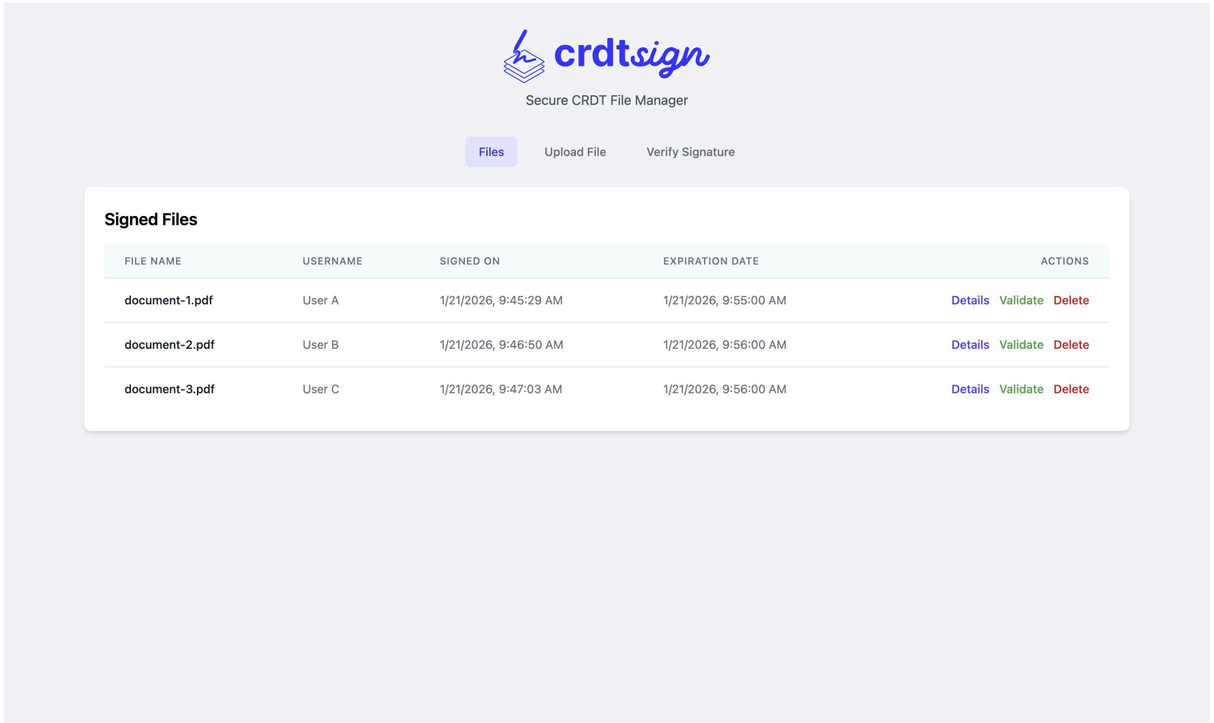
- Written in **Python**
- **Backend** (library): *pycrdt* library (Yjs for Python)
- **Frontend**: *REST API* accessed client-side through a *Web application* (written in *HTML* and *JavaScript*, deployed with *Quart*)
- Communication with **Relay Server**: *WebSockets* protocol
- **Deployment**: Docker (each component has its own container)

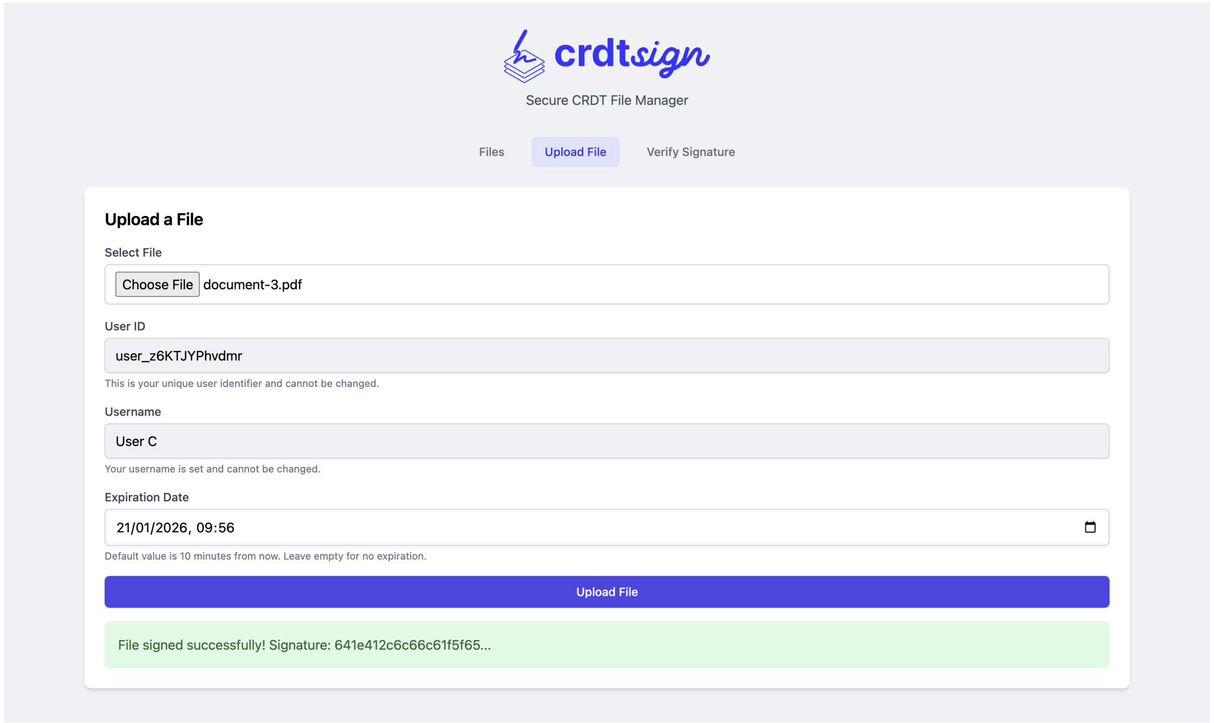


crdtsign

WEB APPLICATION

Building a Replicated File Storage System using CRDTs

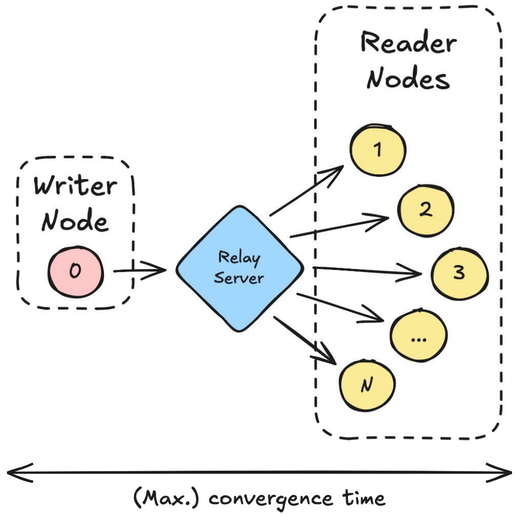




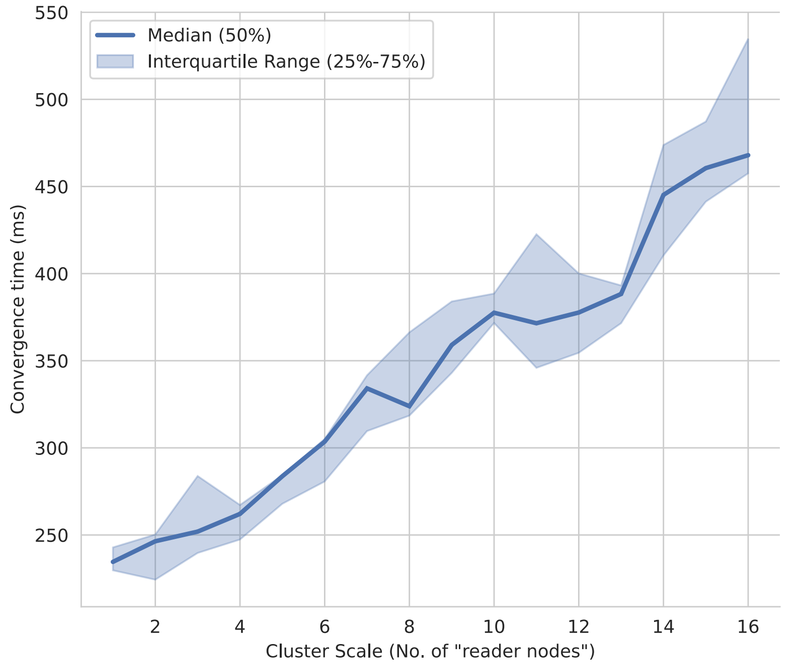
Building a Replicated File Storage System using CRDTs

crdtsign

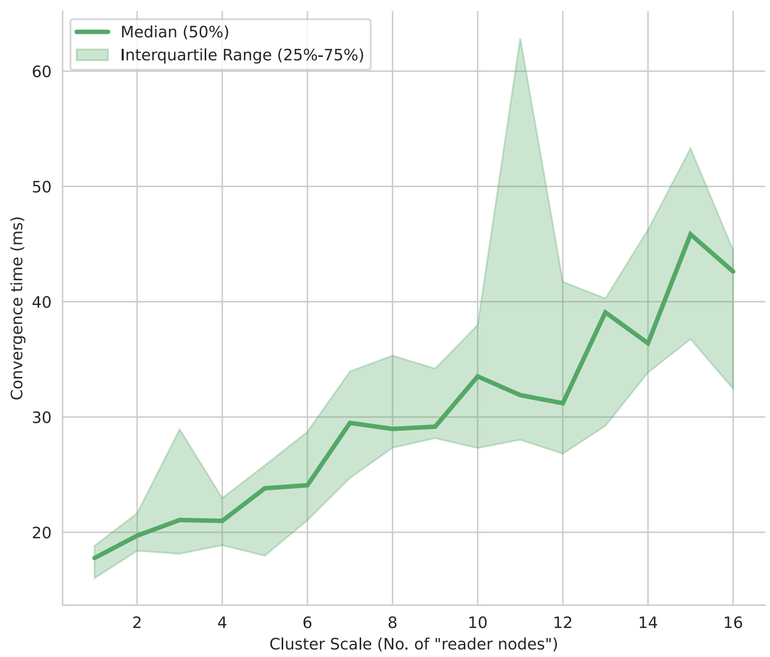
Experimental Setup



"Add" operations



"Remove" operations



Overall, the system maintains *<1 second* (1000ms) *response times* across all tested scales, meeting user expectations for responsiveness.

Conclusion

Contributions of the Thesis

- **Replicated File Storage**
  - Basic functionalitites (*file insertion* and *removal*)
  - Blockchain-like traceability

Limitations of the proof-of-concept

- File embedded within the CRDT map (suboptimal)
- Test performance limited by experiment hardware

Commercial CRDT-based products

- *Redis*
- Distributed databases (*Riak*, *AntidoteDB*)

Possibility for future improvements

- Implement “true” P2P protocols (e.g., *Dat*)
- Extended file storage functionalities (e.g., folders, file renaming, ...)
- Large-scale experimental setup (over multiple geographic locations)

Thank you for your attention!