

UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI



REPORT

Subject: Network Programming
Implementation of Multi-Node File Storage System

Class: B2 - CS

Vũ Nhật Anh	22BA13036
Bùi Quang Huy	22BI13188
Lê Việt Trung	22BA13305
Trần Anh Quốc	22BA13266
Nguyễn Quốc Khánh	22BA13173
Nguyễn Quang Anh	22BA13023
Vũ Quang Thành	22BA13291

Lecturer: Huynh Vinh Nam

Table of Contents

1. Introduction	3
2. System Components	3
a. Index Server (index_server.js)	3
b. Storage Nodes (servernode.js)	3
c. Client (client.js)	3
3. Features Implemented	3
4. Usage	4
5. Demo	5
Demo that the file will be saved in 3 different nodes	6

1. Introduction

This project implements a Multi-Node File Storage System that enables file upload, download, listing, and deletion through a centralized index server. The system leverages multiple storage nodes for distributed file storage and replication, ensuring redundancy, integrity, and accessibility of files.

2. System Components

a. Index Server (`index_server.js`)

- Acts as the central coordinator.
- Maintains metadata of files and mapping to storage nodes.
- Handles client requests for upload, download, listing, and deletion.
- Manages communication with storage nodes using WebSocket.

b. Storage Nodes (`servernode.js`)

- Registers with the index server.
- Stores file data locally in a `data/` directory.
- Responds to file retrieval and deletion requests.
- Sends acknowledgments for storage operations.

c. Client (`client.js`)

- Provides a command-line interface.
- Connects to the index server via WebSocket.
- It supports commands: `upload`, `list`, `download`, and `delete`.
- Manages local file operations and communicates using JSON messages.

3. Features Implemented

- Index Server (`index-server/`): tracks metadata and node registry and coordinates replication.
- Storage Nodes (`storage-node/`): register themselves, store files under `data/`, handle store/retrieve/delete.
- Client (`client/`): CLI is used for uploading, listing, downloading, and deleting.
- **File Upload:**
 - Clients upload files to the index server.
 - Files are assigned a unique file ID.
 - The file is replicated to multiple storage nodes (default: 3).

- **File Download:**
 - Clients download files using file IDs.
 - Files are retrieved from one of the replicated nodes.
- **File Listing:**
 - Clients can list all available files with metadata.
- **File Deletion:**
 - Files can be deleted from all nodes, and the metadata removed from the index server.
- **File Replication:**
 - Ensures fault tolerance by storing files on multiple nodes.
- **Checksum Verification:**
 - Used to verify the integrity of stored and transferred files.
- **Command-Line Interface:**
 - User-friendly interaction for managing files.

Distribution Algorithm

- On upload, the index server picks the first $N=3$ registered nodes.
- Sends full file data over WebSocket to each.
- Waits for **STORE_ACK** from all before confirming to the client.

Node Selection

- Currently, it uses simple FIFO selection of the first N nodes.
- Can be extended with latency measurements or geo-location.

4. Usage

!! Create 3 separate terminals

1. For the **index server**:
`cd index-server && npm install && node index.js`
2. For each **storage node**:
`cd storage-node && npm install && node storageNode.js <NODE_ID>`
3. For **client** terminal:
`cd client && npm install && node client.js`
4. From client prompt:
`upload <file> (file uploaded in /storage-node/data)`
`list`
`download <id> (file downloaded in /client/downloads)`
`delete <id>`

5. Demo

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/index-server]
$ npm install && node index.js

up to date, audited 3 packages in 617ms

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Index server listening on ws://localhost:8000
```

– Start index server –

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node storageNode.js node-1

up to date, audited 2 packages in 580ms
found 0 vulnerabilities
```

– Start Storage Node –

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/client]
$ npm install && node client.js

up to date, audited 3 packages in 612ms

found 0 vulnerabilities
Connected to index server.
> file
Commands: upload <path>, list, download <id>, delete <id>
> list
Available files:
> upload /home/kali/Desktop/CTF/idad_data.bin
Uploading /home/kali/Desktop/CTF/idad_data.bin...
Upload complete. File ID: 29E88BC0
> list
Available files:
29E88BC0 - idat data.bin (1.7 kB) - 2025-05-19T13:47:32.324Z
> download 29E88BC0
Download complete: idat_data.bin
> delete 29E88BC0
Deleted: 29E88BC0
>
```

Upload file to the storage

List of command

List the available file And Download it

Delete file

– Run the client and demo –

Demo that the file will be saved in 3 different nodes

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node index.js
up to date, audited 2 packages in 124ms
up to date, audited 3 packages in 615ms
1 package is looking for funding
  run `npm fund` for details
found 0 vulnerabilities
Index server listening on ws://localhost:8000
found 0 vulnerabilities
```

– Start index server –

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node storageNode.js node-1
found 0 vulnerabilities
up to date, audited 2 packages in 542ms
found 0 vulnerabilities
```

– Start node 1 –

```
(kali㉿kali)-[~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node storageNode.js node-2
up to date, audited 2 packages in 584ms
found 0 vulnerabilities
up to date, audited 2 packages in 584ms
```

– Start node 2 –

```
(kali@kali) - [~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node storageNode.js node-3

up to date, audited 2 packages in 584ms

found 0 vulnerabilities
```

– Start node 3 –

```
(kali@kali) - [~/Desktop/NP/Network-Programming/index-server]
$ npm install && node index.js

up to date, audited 3 packages in 615ms

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Index server listening on ws://localhost:8000
Node registered: node-1
Node registered: node-2
Node registered: node-3
```

– After that, the server will register 3 nodes –

```
(kali@kali) - [~/Desktop/NP/Network-Programming/client]
$ npm install && node client.js

up to date, audited 3 packages in 567ms

found 0 vulnerabilities
Connected to index server.
> upload /home/kali/Desktop/a.js
Uploading /home/kali/Desktop/a.js...
Upload complete. File ID: EBD88C31
> download EBD88C31
Download complete: a.js
>
```

If there are 3 running storageNode
 -> Can download -> File is still available

– If 3 nodes are running –

```
(kali㉿kali) - [~/Desktop/NP/Network-Programming/storage-node]
$ npm install && node storageNode.js node-1
up to date, audited 2 packages in 542ms
found 0 vulnerabilities
^C
```

← Make 1 one node crash

↓

```
> download EBD88C31
Download complete: a.js
```

→ Still downloadable -> File is still available
-> File is also saved in other nodes

– If we make 1 node crash –

```
Download complete: a.js
> download EBD88C31 packages in 604
Error: No available replica nodes
>
```

– If all of the nodes are crashed –