ShardNede: A Peer-to-Peer File Sharing System

Project Report

Group Members:
22k5146 M. Talha Yousif (Leader)
22k5031 Rehan Khan
22k5073 Sohaib Qazi

Contents

Executive Summary

ShardNet is an innovative decentralized peer-to-peer file sharing system designed to provide users with greater control over their data while eliminating reliance on centralized servers. The system leverages advances in peer-to-peer networking technology to create a scalable, secure, and cost-effective solution for modern file-sharing needs.

The project includes a user-friendly web interface built with Next.js for uploading, searching, and downloading files, coupled with a robust Python-based client that handles the peer-to-peer file distribution. By distributing files across multiple peers in encrypted chunks, ShardNet ensures data privacy and security while maintaining high availability.

This project report details the motivation, architecture, implementation plan, and technical specifications for ShardNet, demonstrating its feasibility and potential impact on decentralized file sharing.

Chapter 1

Motivation

The motivation for this project is to create a decentralized file-sharing system that eliminates reliance on centralized servers, giving users greater control over their data. The increasing demand for privacy and decentralization, coupled with advancements in peer-to-peer networking, makes this project an ideal solution for modern file-sharing needs. We aim to solve the inefficiencies of centralized cloud storage while providing a secure and scalable alternative.

In today's digital landscape, users increasingly seek alternatives to centralized storage solutions due to concerns about data privacy, security breaches, and service disruptions. Traditional cloud storage systems typically store data on centralized servers, creating single points of failure and raising concerns about unauthorized access. By developing ShardNet, we aim to address these issues through a decentralized approach that distributes data across multiple peers, enhancing both security and availability.

Chapter 2

Overview

2.1 Significance of the Project

The importance of ShardNet lies in its decentralized architecture, which offers several key advantages over traditional centralized file storage systems:

- **Enhanced Privacy and Security:** By distributing encrypted file chunks across multiple peers rather than storing complete files on centralized servers, ShardNet significantly reduces the risk of unauthorized access or data breaches. Users maintain full control over their data, with no need to trust third-party storage providers.
- **Improved Scalability:** As more users join the network, the system's capacity and performance naturally scale up. This organic growth pattern eliminates the need for expensive infrastructure upgrades that centralized systems require to handle increased loads.
- **Cost-Effectiveness:** The elimination of centralized servers substantially reduces both development and operational costs. Users contribute storage space in exchange for access to the network, creating a collaborative economy of storage.
- **Resilience Against Outages:** With data distributed across multiple nodes, the system remains operational even if individual peers go offline, providing greater reliability compared to centralized services that may experience complete outages.

If successful, ShardNet could significantly impact how individuals and organizations approach file storage and sharing, particularly for privacy-conscious users seeking alternatives to conventional cloud services. The project also has potential applications in areas requiring resilient data storage, such as disaster recovery, collaborative work environments, and regions with limited internet infrastructure.

2.2 Description of the Project

ShardNet is a comprehensive decentralized file-sharing platform that leverages peer-to-peer (P2P) networking for efficient and secure file distribution. The system architecture consists of two primary components:

2.2.1 Frontend Component

The frontend is developed using Next.js, providing a responsive and intuitive web interface that allows users to:

- Register and manage their account
- Upload files to the network

- Search for files using metadata and keywords
- Download files from other peers
- Monitor upload and download progress in real-time
- Manage their shared and stored files

2.2.2 Backend Client Component

The client backend is implemented in Python and handles the core P2P functionality:

- Automatic peer registration with the tracker
- File chunking and encryption for secure storage
- Chunk distribution across multiple peers
- Chunk retrieval and file reconstruction during downloads
- Local metadata management using SQLite
- API exposure via FastAPI for frontend communication

When a user uploads a file through the web interface, the client splits it into multiple encrypted chunks and distributes them across available peers in the network. File metadata is stored in a distributed index that enables efficient searching. During downloads, the client locates and retrieves the necessary chunks from various peers, reassembles them, and delivers the complete file to the user.

2.3 Background of the Project

The concept of decentralized file sharing has evolved significantly since the early days of Napster and Gnutella. Modern implementations like BitTorrent have demonstrated the efficiency of P2P file distribution, while systems like IPFS (InterPlanetary File System) have expanded on these concepts to create content-addressable storage networks.

ShardNet builds upon these foundations while addressing several limitations in existing systems:

- **Usability:** Many current P2P systems require technical knowledge, limiting their adoption. ShardNet provides an intuitive web interface accessible to average users.
- **Privacy:** Unlike some P2P networks that focus primarily on distribution efficiency, ShardNet emphasizes privacy through chunk-level encryption and distribution.
- **Search Functionality:** ShardNet implements a distributed search index that allows users to quickly locate files based on metadata and keywords.

2.3.1 Relevant Literature and Technologies

- **BitTorrent Protocol:** Our chunk distribution methodology draws inspiration from BitTorrent's efficient peer-to-peer file sharing approach [1].
- **IPFS (InterPlanetary File System):** The content-addressing model used in IPFS influences our approach to file identification and retrieval [2].
- **Distributed Hash Tables (DHTs):** We utilize concepts from DHT implementations like Kademlia for peer discovery and metadata storage.
- **Academic Resources:** "Designing Distributed Systems" by M. Balakrishnan provides theoretical foundations for our architecture [3].

2.4 Project Category

ShardNet falls under the **Product-based** project category. It is designed as a fully functional decentralized application intended for practical use by individuals and small businesses requiring secure, distributed file-sharing capabilities. While the project incorporates research elements, particularly in optimizing chunk distribution algorithms and security mechanisms, its primary focus is on delivering a usable solution to real-world file-sharing challenges.

Chapter 3

Features, Scope and Modules

3.1 Key Features

1. **Decentralized File Storage and Distribution**

   ShardNet eliminates central servers by distributing file chunks across participating peers. This architecture ensures no single entity controls the entire file, enhancing both privacy and redundancy. Files remain accessible even if some peers disconnect from the network.

2. **Secure File Chunking and Encryption**

   Files are automatically split into smaller chunks before distribution, with each chunk individually encrypted. This approach ensures that even if a malicious actor gains access to some chunks, they cannot reconstruct the complete file without the appropriate decryption keys.

3. **Distributed Tracker System**

   Unlike traditional P2P systems that rely on central trackers, ShardNet implements a distributed tracker system. Peers maintain a partial network map and collaboratively track chunk locations, eliminating single points of failure in network coordination.

4. **Intelligent Chunk Distribution Algorithm**

   The system implements a smart distribution algorithm that considers factors such as peer reliability, geographic distribution, and network conditions when placing chunks. This approach optimizes both retrieval speed and file availability.

5. **Comprehensive Search Functionality**

   Users can search for files across the network using metadata, keywords, and file characteristics. The distributed search index ensures efficient queries without centralized servers.

6. **Real-Time Progress Monitoring**

   The interface provides detailed visual feedback on file upload and download progress, including transfer speeds, estimated completion times, and chunk availability status.

7. **Automatic Network Optimization**

   ShardNet continuously analyzes network conditions and peer behavior to optimize chunk distribution and retrieval paths, ensuring maximum performance with minimal user intervention.

8. **Cross-Platform Compatibility**

   The system works seamlessly across different operating systems and device types, allowing users to access their files from any platform with an internet connection.

3.2 System Modules

ShardNet's architecture consists of several interconnected modules, each responsible for specific functionality:

3.2.1 Frontend Modules

1. **User Authentication Module**
   - Handles user registration and login
   - Manages user profiles and preferences
   - Implements secure session management
2. **File Management Interface**
   - Provides intuitive file upload mechanisms
   - Displays user's shared and downloaded files
   - Allows file organization and metadata editing
3. **Search Interface**
   - Implements advanced search functionality
   - Displays search results with relevant metadata
   - Provides filtering and sorting options
4. **Transfer Monitor**
   - Shows real-time progress of uploads and downloads
   - Displays network statistics and performance metrics
   - Allows transfer prioritization and management

3.2.2 Backend Client Modules

1. **Peer Manager**
   - Handles peer discovery and registration
   - Maintains connections with other peers
   - Monitors peer availability and performance
2. **File Processor**
   - Splits files into chunks for distribution
   - Handles encryption and decryption

- Manages chunk verification and integrity

3. **Chunk Distributor**
    - Implements the chunk distribution algorithm
    - Manages chunk replication for redundancy
    - Handles chunk retrieval during downloads
4. **Metadata Manager**
    - Maintains the local database of file metadata
    - Participates in the distributed search index
    - Manages file reconstruction information
5. **API Server (FastAPI)**
    - Exposes endpoints for frontend communication
    - Handles file upload and download requests
    - Provides search functionality and system status

Chapter 4

Project Feasibility

4.1 Technical Feasibility

ShardNet is technically feasible with existing technologies and established programming paradigms. The project leverages several mature technologies:

- **Python and FastAPI:** For developing the backend client and API services. Python's extensive libraries for cryptography, networking, and data handling significantly simplify development.
- **Next.js:** A well-established React framework for building the frontend interface, offering server-side rendering and an optimized development experience.
- **WebSockets:** For real-time communication between peers and the frontend interface.
- **SQLite:** For local metadata storage, providing a lightweight database solution without external dependencies.
- **Cryptographic Libraries:** Such as PyCryptodome for implementing secure file encryption and integrity verification.

4.1.1 Technical Risks and Mitigation

- **Network Address Translation (NAT) Traversal:** Connecting peers behind different NATs can be challenging. We will implement techniques such as UDP hole punching and STUN/TURN servers to facilitate connections.
- **Peer Reliability:** Peers may disconnect unexpectedly. The system will implement redundant chunk storage and intelligent chunk selection to ensure file availability.
- **Security Vulnerabilities:** To address potential security issues, we will implement strong encryption, secure authentication methods, and regular security audits throughout development.
- **Scalability Challenges:** As the network grows, maintaining efficient search and retrieval could become challenging. We will implement distributed indexing and caching mechanisms to maintain performance.

4.2 Economic Feasibility

ShardNet offers compelling economic advantages over traditional centralized file-sharing systems:

4.2.1 Development Costs

- **Infrastructure:** Minimal server infrastructure is required since the system operates primarily on user devices. Only lightweight coordination servers are needed.
- **Software Development:** Development utilizes open-source technologies, eliminating licensing costs.
- **Testing and Deployment:** Testing can be conducted using virtualized peers on development machines, requiring no additional hardware.

4.2.2 Operational Costs

- **Maintenance:** The decentralized nature significantly reduces ongoing maintenance costs compared to centralized alternatives.
- **Scaling:** Scaling costs are distributed among users who contribute resources to the network, eliminating the need for expensive server upgrades.
- **Support:** Documentation and community forums can address most user issues, reducing the need for extensive support staff.

4.2.3 Benefits

- **For Users:** Free or low-cost file storage and sharing without compromising privacy.
- **For Developers:** Reduced infrastructure costs and simplified scaling.
- **For Organizations:** Enhanced data security and resilience with lower operational expenses.

4.3 Schedule Feasibility

The proposed 8-week timeline is realistic for delivering a functional ShardNet system based on the following considerations:

- **Modular Development:** The system architecture allows for parallel development of different components, optimizing team productivity.
- **Leveraging Existing Libraries:** Using established libraries for cryptography, networking, and UI components reduces development time.
- **Agile Approach:** The development plan incorporates iterative testing and refinement, allowing early identification and resolution of issues.
- **Clear Milestone Structure:** The project plan divides development into manageable phases with specific deliverables, helping to track progress and maintain momentum.

4.3.1 Potential Schedule Risks

- **Integration Challenges:** Frontend-backend integration may require additional time for troubleshooting. We've allocated extra time in Week 6 specifically for this purpose.
- **Unanticipated Technical Hurdles:** Particularly in P2P connectivity and distributed search. The team has built contingency time into the schedule to address these if they arise.
- **Testing Complexity:** Testing distributed systems presents unique challenges. We've allocated sufficient time in Weeks 7-8 for thorough system testing.

Chapter 5

Hardware and Software Requirements

5.1 Development Requirements

5.1.1 Hardware Requirements

- Computers with minimum 8GB RAM (16GB recommended)
- Processors: Intel Core i5/AMD Ryzen 5 or better
- Storage: 20GB+ free space for development environment
- Stable internet connection (10+ Mbps)

5.1.2 Software Requirements

- Operating System: Windows 10/11, macOS, or Linux
- Python 3.8+ with pip package manager
- Node.js 16.x+ and npm/yarn
- Git version control system
- Visual Studio Code or similar IDE
- Docker (for testing multiple peer instances)

5.1.3 Libraries and Frameworks

- FastAPI for backend API development
- PyCryptodome for encryption functionality
- SQLite for local database
- WebSockets for real-time communication
- Next.js for frontend development
- React for UI components
- Tailwind CSS for styling

5.2 Deployment Requirements

5.2.1 For Users

- Computer with minimum 4GB RAM
- Modern CPU (dual-core or better)
- Storage space (variable depending on shared file sizes)
- Stable internet connection
- Modern web browser (Chrome, Firefox, Edge, Safari)
- Python 3.8+ runtime (for the client component)

5.2.2 For Developers

- Git for version control
- Python 3.8+ with pip
- Node.js and npm
- Docker (optional, for containerized deployment)

Chapter 6

System Architecture

6.1 High-Level Architecture

The ShardNet system follows a hybrid decentralized architecture that combines peer-to-peer networking with a lightweight coordination mechanism. The system can be conceptually divided into the following components:

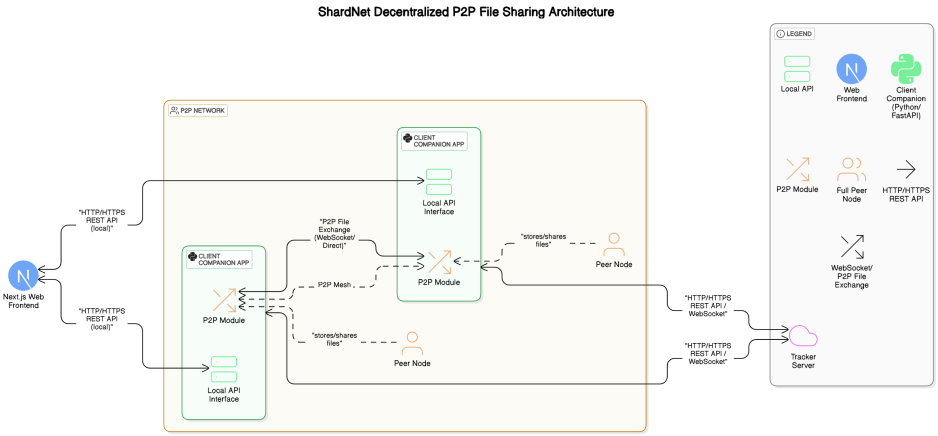

Figure 6.1: High-Level Architecture of ShardNet

6.2 Component Interaction

6.2.1 File Upload Flow

When a user uploads a file, the following sequence occurs:

1. The user selects a file through the Next.js frontend
2. The frontend sends the file to the local client via the FastAPI interface
3. The client processes the file:
   - Splits the file into chunks
   - Encrypts each chunk with unique keys
   - Calculates checksums for integrity verification
4. The client distributes chunks to available peers based on the distribution algorithm
5. Metadata about the file and chunk locations is stored in the distributed index
6. The frontend displays success confirmation and sharing options

6.2.2 File Download Flow

When a user downloads a file, the following sequence occurs:

1. The user searches for and selects a file through the frontend
2. The frontend requests the file from the local client
3. The client queries the distributed index for chunk locations
4. Chunks are requested and retrieved in parallel from multiple peers
5. As chunks arrive, they are:
    - Verified for integrity using checksums
    - Decrypted using the appropriate keys
    - Assembled in the correct order
6. The complete file is made available to the user
7. The client optionally retains chunks to share with other peers

6.3 Security Model

ShardNet implements multiple layers of security:

- **Chunk-Level Encryption:** Each file chunk is independently encrypted
- **Distributed Storage:** No single peer holds the complete file
- **Multi-Factor Authentication:** For user accounts
- **Secure Communication:** All peer-to-peer communication is encrypted
- **Integrity Verification:** Checksums ensure chunks haven't been tampered with

6.4 Diagrammatic Representation of the Overall System

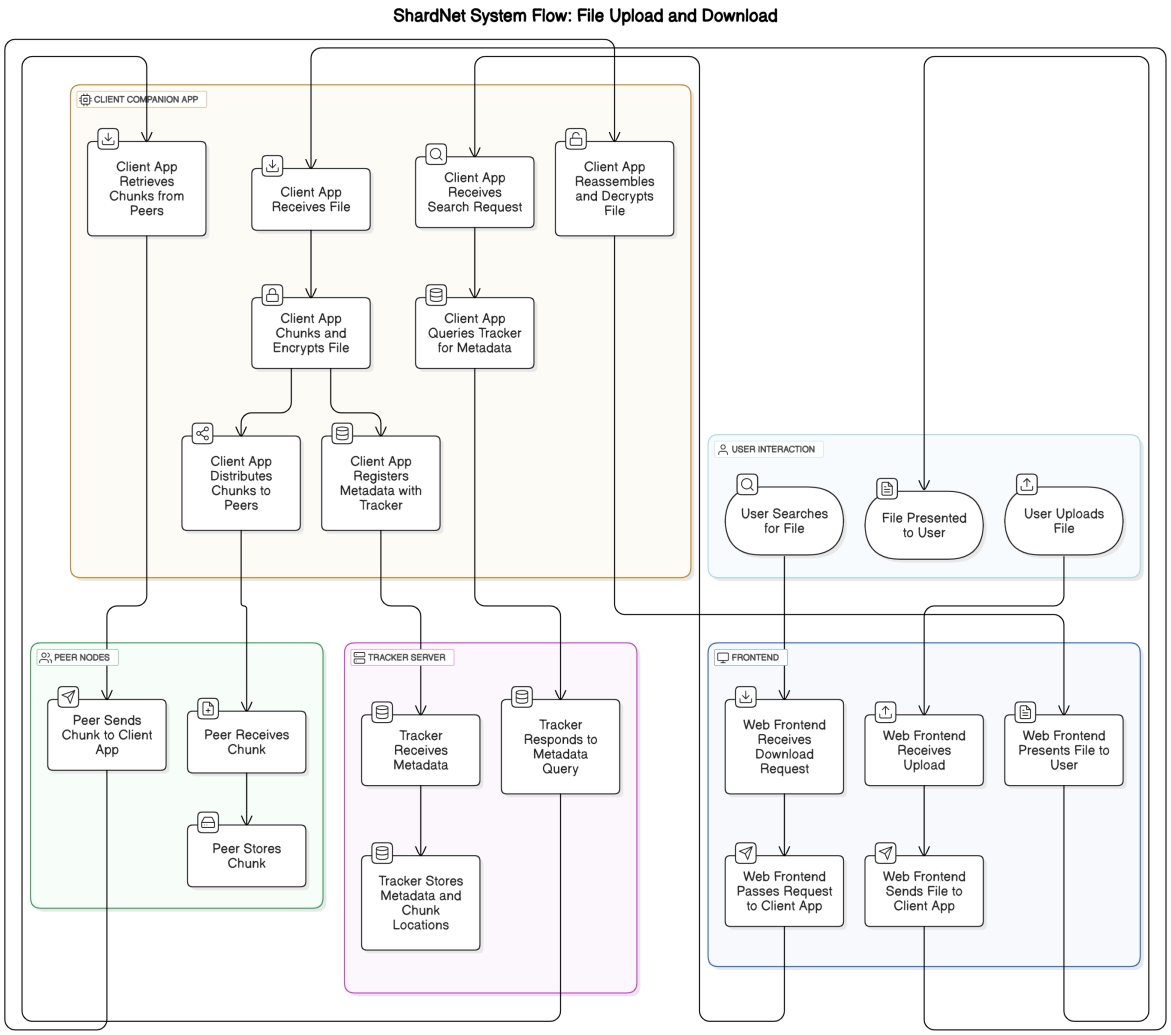

Figure 6.2: Comprehensive System Diagram of ShardNet

Chapter 7

Future Enhancements

While the current scope of ShardNet provides a robust decentralized file sharing solution, several potential enhancements could be implemented in future iterations:

1. **Mobile Application Development:** Creating native mobile applications for iOS and Android to expand accessibility.
2. **Incentive Mechanism:** Implementing a token-based incentive system to reward users who contribute storage and bandwidth resources.
3. **Versioning and Collaboration:** Adding file versioning capabilities and real-time collaborative editing features.
4. **Advanced Analytics:** Providing users with insights about their storage usage, file popularity, and network performance.
5. **Integration with Existing Systems:** Developing plugins or APIs to integrate ShardNet with popular productivity tools and content management systems.
6. **Smart Contracts:** Implementing blockchain-based smart contracts to enhance security and enable advanced sharing permissions.

These enhancements would build upon the foundation established in the current project, further expanding ShardNet's capabilities and user appeal.

Chapter 8

Conclusion

ShardNet represents a significant step forward in decentralized file sharing technology, offering users a secure, efficient, and privacy-focused alternative to traditional cloud storage systems. By distributing files across multiple peers and implementing robust security measures at the chunk level, the system addresses many of the limitations of both centralized services and existing P2P networks.

The project's feasibility has been thoroughly assessed from technical, economic, and scheduling perspectives, with clear plans established for implementation and risk mitigation. The modular architecture ensures scalability and adaptability for future enhancements.

Upon completion, ShardNet will demonstrate the viability of user-controlled decentralized storage as a practical solution for everyday file sharing needs, potentially influencing how data storage and distribution are approached in various applications.

Bibliography

[1] BitTorrent, "BitTorrent Protocol," Retrieved from `https://www.bittorrent.com`

[2] IPFS, "InterPlanetary File System," Retrieved from `https://ipfs.io`

[3] M. Balakrishnan, "Designing Distributed Systems," O'Reilly Media, 2018.

[4] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," Peer-to-Peer Systems, Springer, 2002, pp. 53-65.

[5] Vercel, "Next.js Documentation," Retrieved from `https://nextjs.org/docs`

[6] FastAPI, "FastAPI Documentation," Retrieved from `https://fastapi.tiangolo.com`