SRS Document
UDP Peer To Peer Data Transfer

Yash Shivhare 2021101105
Rayaan Khan 2021101120

Overview

This project's objective is to integrate the pre existing code for UDP Peer to Peer transfer to the DFS server and adding more functionalities like Authentication, increasing overall efficiency and building the frontend that will be intractable with the sender and receiver.

Project's main purpose is to establish a secure and efficient peer-to-peer data transfer system between two entities, namely the Data Foundation and the Partner, using NodeJS servers. The data transfer mechanism utilizes UDP for the actual data transmission and TCP for maintaining data transfer quality. As part of the project, specific Data Transfer Profiles are created for each agent, containing essential information such as IP addresses, host names, and port details.

The initiation of data transfer occurs through the configuration of a transfer request, specifying parameters like the number of threads, IP addresses involved, and the scheduled transfer time. Furthermore, the project encompasses the development of a web-based monitoring system for overseeing the progress of data transfer sessions.

Additionally, the project entails integrating the Data Foundation with both the file system and MinIO, a versatile object storage solution. We ultimately would like to create a better and a faster way for sending data between two peers.

System Requirements

1. **Functional requirements**

   Peer-to-peer data transfer is a method employed for moving data between two computers without the involvement of an intermediary server or central coordinator. Ensuring the security of such data transfers is imperative, particularly when dealing with sensitive or confidential information.

In this document, we will delve into the blueprint and structure of a robust, high-speed, secure peer-to-peer data transfer system. This system will utilize UDP for efficient data transfer and TCP for maintaining transfer quality. The architecture will be constructed using NodeJS servers and will facilitate data exchanges between two entities: the Data Foundation Agent and the Partner Agent.

While some progress has already been made earlier on the project, our primary focus now revolves around the following key tasks:

1. **Integration:** Our foremost objective is to seamlessly integrate the existing project with the actual servers and execute it in a real-world environment. During this process, we will rigorously identify and address any potential bugs or errors that may arise.Also after the integration ensuring multiple device support as well as ensuring cross browser compatibility.

2. **Authentication:** Given that the UDP protocol lacks a built-in "Handshake" mechanism to confirm package reception by the receiver, we must enhance the current solution. Our goal is to implement this functionality, bolstering the assurance level of data transfer.

3. **Performance Interface:** To enhance user interaction and provide essential insights into the data transfer process, we will develop a user-friendly frontend interface. This interface will enable users to monitor transfer speed, estimated time of completion, and the percentage of data transferred at any given point in time.

4. **Efficiency**: Evaluating the current efficiency of the existing product is vital. We will conduct comprehensive tests to assess its performance and identify areas for improvement. Our aim is to make significant enhancements to boost the overall efficiency of the system.Majorly the existing product works on HTTP protocol,which is relatively slow and hence the **Main aim is to convert it into UDP based protocol.**

   In summary, our key objectives encompass integrating the project, enhancing data transfer security, providing a user-friendly interface for monitoring, and optimising the system's efficiency for a more robust and reliable peer-to-peer data transfer solution.

Use cases for this platform include:

1. **Educational Use:** This platform has the potential to be employed within educational institutions to facilitate the swift transfer of substantial data quantities. This could encompass the exchange of materials like video lectures and research papers among educators and students or among various departments within the same institution.

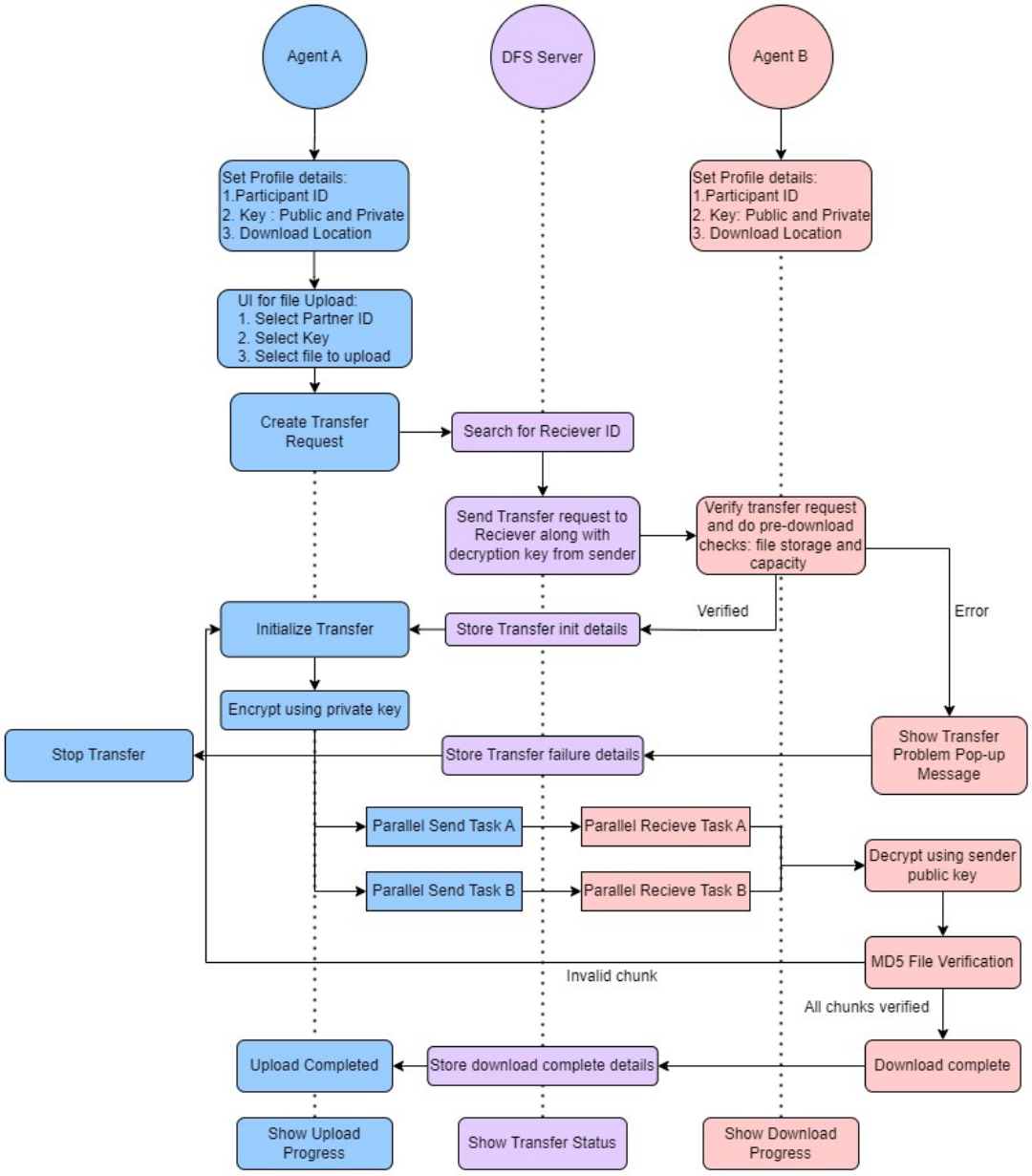
2. **Movie Industry**: Our product can be used to transfer big movies and videos across different areas in a seamless way which can be used to distribute the big movies quicker to be displayed in theatres.

3. **Research Application:** Researchers have the opportunity to utilize this platform for the efficient transfer of extensive datasets, such as scientific data or simulations. It can serve as a means of data exchange between different research institutions or among researchers collaborating on a shared project.

4. **Industrial Utility:** Companies can benefit from this platform for the secure and high-speed transfer of large volumes of data, including sensitive information like financial data or customer records. This can be useful for data sharing among different company divisions or between separate companies.

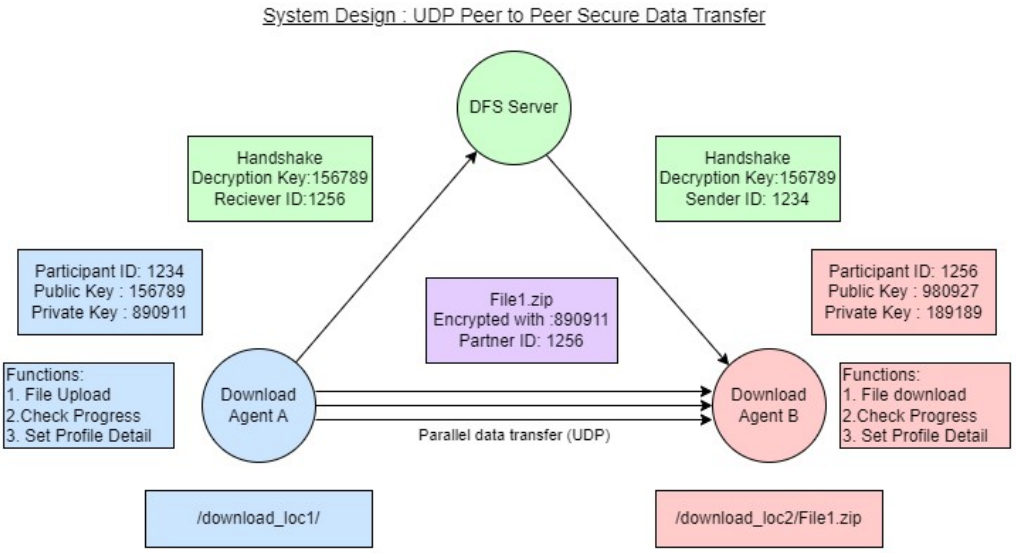
5. **Community Enhancement:** The platform also has the potential to foster community growth by enabling the seamless transfer of sizeable data sets, such as event recordings or community resources. This can facilitate data exchange between various communities or among members within a specific community.

Workflow of the Product

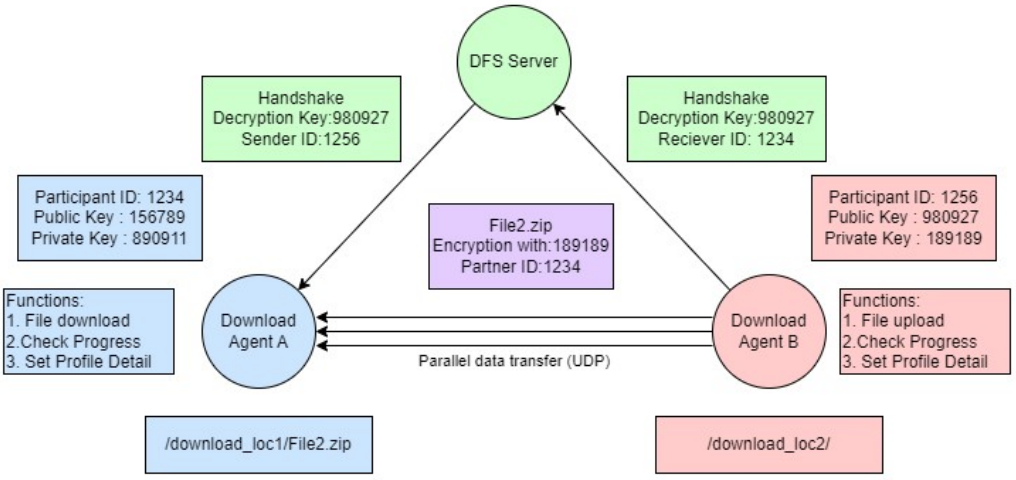


Workflow of the System

System Design



Transferring file1.zip from Agent A to Agent B



Transferring file2.zip from Agent B to Agent A

Steps of how the system would work:

1. Initiating Transfer:

a. Agent A logs into the website, fetching its unique Public and Private keys.
b. Navigates to the "Point-to-Point" transfer page and clicks "Start to Transfer File."
c. Agent A enters the recipient's ID, initiating a Handshake with Agent B via the DFS Server.

2. Recipient Information Retrieval:

a. The server searches for the recipient's details (Public and Private Key) and provides Agent A with the recipient's ID and Key.
b. Agent A prepares the transfer request, including the decryption key (A's public key) from the database.
c. Agent A sends the request to the recipient.

3. Request Verification:

a. Agent B receives and verifies the transfer request.
b. Upon successful verification, Agent B sends a confirmation to the server.

4. Confirmation Handling:

a. The server receives Agent B's confirmation and notifies Agent A, signaling that the transfer can begin.

5. Parallel Data Transfer:

a. Agent A breaks down the file into segments and creates concurrent threads to send each segment encrypted with A's private key.
b. Agent B, the recipient, concurrently receives and decrypts each segment using A's public key.
c. Integrity checks using MD5 are conducted to verify each segment. If corruption is detected, a Negative Acknowledgement prompts retransmission.

6. Real-time Progress Monitoring:

a. The server monitors the upload and download progress of the transfer, offering real-time updates from both Agent A and Agent B.

7. Completion and History:

a. Once the transfer concludes, the recipient can access the received file in their local directory.
b. The server updates the transfer history, accessible by both parties to review the status of prior transfers.

Non Functional Requirements

1. Performance:

a. Throughput: The system should support high data transfer rates to ensure efficient communication between agents.
b. Response Time: User interactions with the performance interface should have minimal latency.
c. Scalability: The system should be capable of handling a growing number of concurrent data transfers without significant degradation in performance.

2. Reliability:

a. Availability: The system should be available for data transfer operations 24/7, with minimal downtime for maintenance or upgrades.
b. Fault Tolerance: It should be resilient to failures, with the ability to recover from errors gracefully.
c. Data Integrity: Ensure data is transferred accurately and without corruption.

3. Security:

a. Authentication: The authentication mechanism should be robust to prevent unauthorised access to the system.
b. Data Encryption: Sensitive data should be encrypted during transfer to protect against eavesdropping or data breaches.
c. Access Control: Implement role-based access control to restrict unauthorized users' actions within the system.

4. Usability:

a. User-Friendly Interface: The performance interface should be intuitive and easy to use, even for non-technical users.
b. Accessibility: Ensure that the interface is accessible to users with disabilities.

5. Scalability:

a. Load Handling: The system should be able to handle a variable load, scaling resources as necessary to accommodate increased traffic.

6. Compatibility:

a. Cross-Platform Compatibility: Ensure that the system works seamlessly on different operating systems and devices.
b. Browser Compatibility: If the interface is web-based, it should be compatible with popular web browsers.

7. Logging and Auditing:

a. Audit Trail: Maintain an audit trail of system activities, including user actions and data transfers, for security and compliance purposes.

b. Error Logging: Record and log errors and exceptions for debugging and troubleshooting.

8. Documentation:

a. User Documentation: Provide comprehensive user documentation, including guides and tutorials for both agents and administrators.
b. Technical Documentation: Document the system architecture, APIs, and any custom protocols used.

9. Compliance:

a. Regulatory Compliance: Ensure that the system complies with relevant data protection and privacy regulations, such as GDPR or HIPAA, if applicable.

10. Resource Utilisation:

a. Resource Efficiency: Optimise resource usage, such as CPU and memory, to ensure efficient operation of the system.

11. Monitoring and Reporting:

a. Performance Monitoring: Implement monitoring tools to track system performance and alert administrators to any anomalies.
b. Reporting: Generate reports on data transfer statistics, errors, and system usage.

12. Backup and Recovery:

a. Data Backup: Regularly backup transferred data to prevent data loss in case of system failures.
b. Disaster Recovery: Have a plan in place for recovering the system in the event of a catastrophic failure.

These non-functional requirements help ensure that the peer-to-peer data transfer system not only functions correctly but also meets the performance, security, and usability expectations of its users while adhering to industry standards and regulations.

Deliverables

Our final product would consist of modified code integrating it with DFS server and having a UI interface showing the progress report of the download or upload status and deploying UDP for data transfer between two peers.

Authentication feature will also be added for enhancing the security and increasing the reliability.