**Distributed File Synchronization System**

**Project Description:**

The Distributed File Synchronization System (DFS) is a network application designed to allow users to synchronize files across multiple devices in real-time. This system will enable users to access their files from any connected device, ensuring that the most up-to-date version of each file is always available.

The DFS will consist of a central server and multiple client applications. The central server will be responsible for managing user accounts, storing file metadata, and coordinating synchronization between clients. Client applications will run on various devices (desktops, laptops, mobile devices) and handle local file operations, detect changes, and communicate with the server to maintain synchronization.

Key features of the DFS include:

a) User Authentication: Secure login system to protect user data and ensure only authorized access to files.

b) File Versioning: The system will maintain a history of file changes, allowing users to revert to previous versions if needed.

c) Conflict Resolution: When conflicting changes occur on different devices, the system will implement a conflict resolution mechanism to merge changes or alert users to manually resolve conflicts.

d) Bandwidth Optimization: The system will use delta synchronization to transfer only the changes in files rather than entire files, reducing bandwidth usage.

e) Encryption: All data transfers between clients and the server will be encrypted to ensure privacy and security.

f) Offline Support: Clients will be able to work offline and synchronize changes when a connection is reestablished.

g) Selective Synchronization: Users can choose which folders or files to synchronize across devices.

h) File Sharing: Users can share files or folders with other users, with customizable permissions.

i) Cross-platform Support: Client applications will be developed for major operating systems to ensure wide compatibility.

The DFS will utilize various network protocols and concepts, including:

* TCP for reliable data transfer and connection management
* UDP for quick status updates and heartbeats
* HTTP/HTTPS for RESTful API communication between clients and server
* WebSockets for real-time notifications
* SSL/TLS for secure communication
* Load balancing for scalability
* Caching mechanisms to improve performance

**Design Requirements of Transport Services:**

|  |  |
| --- | --- |
| **Service** | **Requirement** |
| Reliable Data | - Guaranteed delivery of file chunks and metadata |
| Transfer | - In-order delivery of file chunks  - Error detection and correction mechanisms |
| Timing | - Real-time synchronization (within seconds of file changes)  - Low latency for small file transfers (<500ms)  - Timeout and retry mechanisms for failed transfers |
| Throughput | - Support for large file transfers (up to 1GB)  - Efficient use of available bandwidth (adaptive based on network conditions)  - Concurrent transfers to/from multiple clients |
| Security | - End-to-end encryption for all data transfers  - Secure user authentication and authorization  - Protection against man-in-the-middle attacks  - Resistance to DDoS attacks |

**Architecture Diagram:**