**Concurrent Programming Project**

**High Level Design**

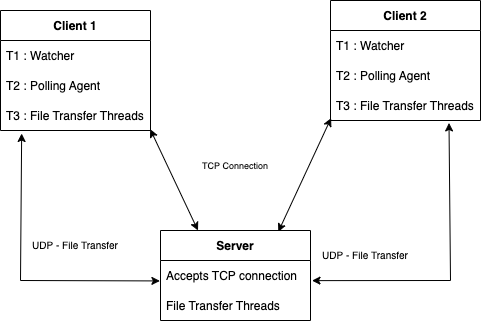
**2225-CSE-6324-001**

**Summary:**

To create a java based application to replicate cloud-based storage application by implementing concurrent programming by using network protocol (UDP) and implementing error handling and logging.

**Architecture**

Component Interaction:

****

**Project has 4 major components**

1. **Directory Watcher**
2. **File Synchronisation**
3. **Error Handling**
4. **Polling**
5. **Directory Watcher**

* The WatcherService library in java is used to check if the new file is created or modified or deleted.
* The issue that was faced here was that 2 events are triggered when a new file is created(i.e Create and modify).

1. **File Synchronisation: This has 2 core components: File Sync and Delta Sync**

* The file is transferred from client to server using UDP and vice versa.
* We have File Reader, File Writer, Packet Sender and Packet Receiver classes for doing this.
* This has been integrated in the file transfer utility class.
* For the Delta sync, we are dividing the file into chunks of 2MB and then comparing it. The data is then transferred from the modified block to the Server and vice versa.
* FileComparator class is used to identify change in file blocks, and post getting the index from where file is changed we are updating only those chunks via UDP to server.

1. **Error Handling:**

* Exceptions for I/O, Network / Connection failure, Package drops are logged by logger.
* The code is being terminated if any exceptions are thrown.
* For Package drop and sequence issues, we are using the packet index. The acknowledgement is sent and received on the basis of an index.
* Connectivity Exception: We are using a logger to check if there are any exceptions.
* The logger is used to log the data to the console.So that if any exception occurs, it is written to the console.
* The issue faced by watchers is handled by WatchFolder class.

1. **Polling:**

* We are using polling to send and receive the message. A message is sent to the server from the client to request a new/modified/delete file.
* The buffer is maintained on the server side.
* If one client is sending a file to the server, the details like clientid, filename and requesttype(modify/new/delete) are added to the buffers of the other client.
* Server sends a message to the client saying that it has updates when the client sends the poll requests during regular intervals..
* The client requests the file and then send-receive operation happens.

**Thread Management**

The base means for concurrency is the java.lang.Threads class. A Thread executes an object of type java.lang.Runnable.

Runnable is an interface which defines the run() method. This method is called by the Thread object and contains the work which should be done. Therefore the Runnable is the task to perform. The Thread is the worker who is doing this task.

File transfer is built using a client server model.

The client here is the sender of a file and server is the receiver.

Before the client sends the data to the server via UDP, it tells the port number that is used to send the data.

The client also gets the port number that is used by the server to receive the data.

The request is sent by the client to the server that has a UDP port number using TCP.

The response is sent by the server which contains the port number that is used to receive the data.

Now the file transfer will begin as the client and server now know the UDP port numbers used for transfer.

We consider four threads here.

Two on the client side and the other two on the server side.

One on the client and server side each for file reading and writing respectively.

One on the client and server side each for networking, sending and receiving data.

Here we benefit from concurrent programming as we have two different resources for file and network. Pipelining them can increase resource utilisation.

The diagram below shows execution of the threads.

At Client side we have

ReadFile - It constantly reads the data from the file and deposits it in the buffer until the entire data is read.

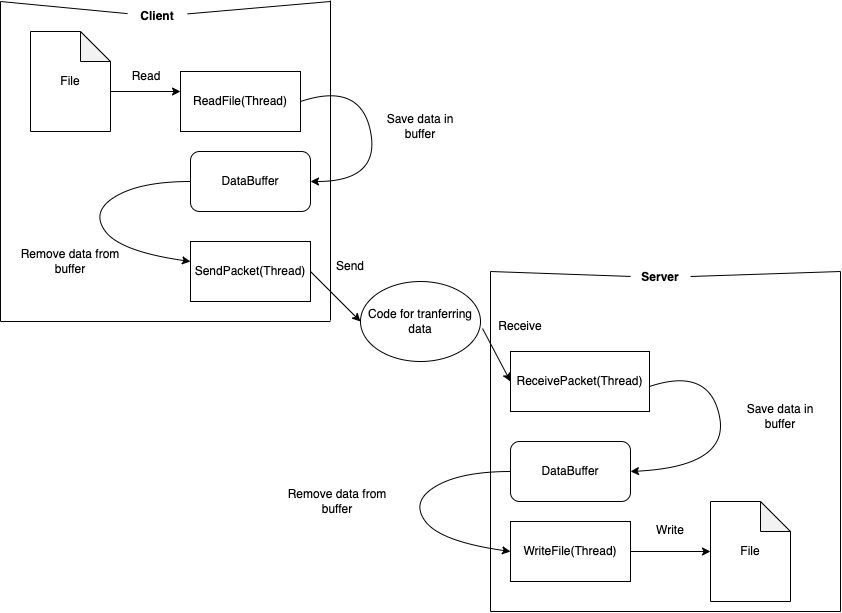
SendPacket - It withdraws packet from buffer and sends it to the server.

At Server side we have

ReceivePacket - It receives packets from the client and deposits them in a buffer.

WriteFile - It withdraws packet from buffer and writes it to a file.

The acknowledgement packets are sent to ensure the reliability. SendPacket and ReceivePacket use this to maintain reliability.



**Directory Watcher thread:**

Directory Watcher continuously executes on clients directory to monitor for changes when a file is updated, added or deleted.

Three events gets triggered in this directory: Events: Add/Modify/Delete

Consider below example for add event:

1. When a new file is added by Client 1 in the directory, change is detected, Fileransfer utility transfers the file over udp connection.
2. Along with the file to be transferred, the client also sends the message to the server eg “CLIENT1\nFADDED\n1.txt\n”.

Consider Event for file deletion:

1. When a client deletes a file, “CLIENT1\nFDELETED\n1.txt\n” this message is communicated to server over TCP connection and on server side, once it receives “DELETE” request type it deletes the file. Server then stores data in Buffer for client
2. Once the client 2 polls the server for changes, it receives data from the buffer by server , and the required file gets deleted .

**Polling Thread:**

A new thread has been implemented for polling the requests from and to client and server.

The client constantly polls the server for updates. The new file created by the client is notified to the server and then the server starts receiving the file once the client sends it. Same thing happens when a new file is created on client 2. For syncing the files between all the clients we maintain a buffer at the server side that has all the details about the client.The buffer will store updates for client two if a new file is created by client1. When client2 will poll the server, server will send out updates to client2 and then a new file will be requested by client2.The send-receive operation between client2 and server takes place.

