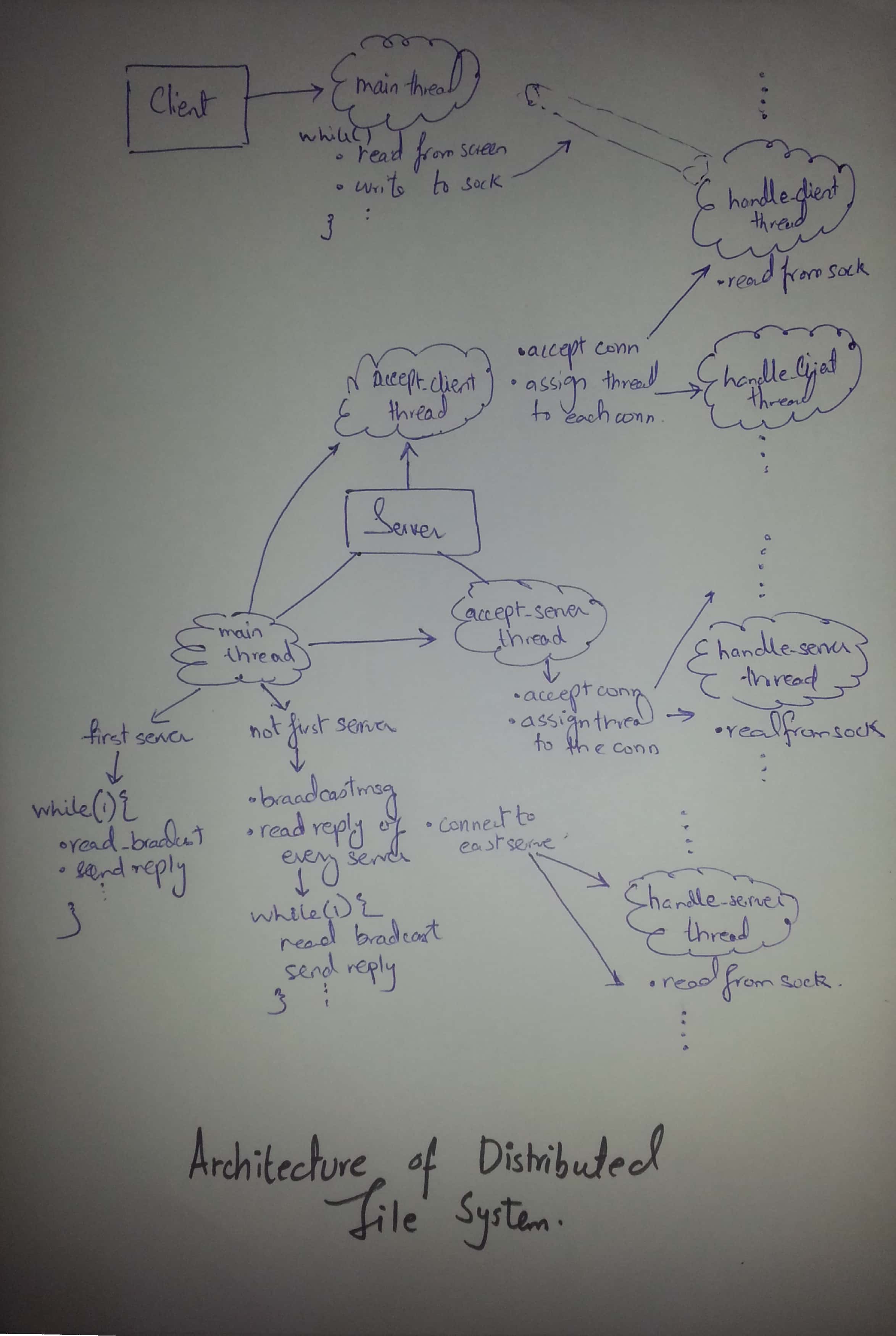


Architecture of Distributed file system

Muhammad Wasi Naseer-11578



Architecture Diagram:

# Server:

## Starting Process:

* When server starts,
  + It creates broadcast\_sock for broadcasting request and listening to the request
  + It creates udp\_sock for sending ack and hostname to specific incoming broadcast request
  + It creates tcp\_client\_sock for accepting client connections
  + It creates tcp\_server\_sock for accepting server connections
  + it creates two threads accept\_server\_thread and accept\_client\_thread
  + If the directory structure file is already there on the root folder, it will just load the file
  + If the file is not there because it is starting first time, it will create an empty directory structure file
* If this is the first server to run,
  + It will start the two threads.
  + It won’t broadcast to other servers because it is the first server.
  + Hence, it will only listen to broadcast requests from servers and reply the ack to the servers. When the server, which requested connection, sent the connect(), this server will reply by accepting the connection.
* If this is not the first server to run,
  + It will broadcast requests to all for getting connected.
  + The other servers will reply by sending ack and hostnames
  + It will register their ips and send connect() to all of them
  + Once connect() is accepted, it will be connected to the distributed file system. Hence eligible for accepting client requests and server requests
  + Every server after accepting connection will send the directory structure file to this newly connected server.
    - This server will compare the version of its own dir\_struct file and the one which it gets most updated. If this server has the most updated dir\_struct file, it will send the its file to other servers. If it has the less version of dir\_struct file, it will get the most updated dir\_struct file and
      * It will compare its own dir\_struct files’ version to the most updated dir\_struct files’ version and get the one which is most updated. It will also compare for the deletion.
  + Administrator will have to specify how many servers are currently running in the system so that it will wait for only those acknowledgements.
  + Now the main thread will listen to the broadcast requests from servers and reply the ack to the servers. When the server, which requested connection, sent the connect(), this server will reply by accepting the connection.
* Accept\_client\_thread will accept the connections from client
* Accept\_server\_thread will accept the connections from server.

## Intermediate Interaction:

* When the dir command is requested, it will just send the files in directory structure
* When the touch command is requested,
  + It will create the file on its own server
  + It will create the file to any other random server if a server is available for replication. Otherwise, no replication is possible. Hence, it will create the file on only one server.
* When the download command is requested,
  + It will check if it has the requested file. If it has the file, it will just return the file to the client.
  + If it has not the file, it will check the hostname that has the file and find its corresponding ip and send the ip to the client if the server is up and running. If the server is not up and running, it will give an error.
  + The client will send the connect() to the ip and request the same file. The server now send the requested file.
  + Since client is not multithreaded, it will not be able to do any task unless the file has been uploaded by saving and closing it. Once the file is uploaded, the version of this file is updated first on this server and then it will send the updated file to other servers to update the version of this file virtually in dir\_struct and the file physically if they have.
* When the delete command is requested,
  + It will just update its dir\_struct by deleting the entry of that file. It will check if it has the requested file. If it has the file, it will delete the file physically. And, it will also send the delete request to other servers to do the same.
* When the server is getting shutdown, it will send the connected clients either the ip of any other server so that the client will connect to the server or if no other server is up and running, it will send No Server available error to the client.`

# Client:

## Starting Process:

* The client will send the connect to the set of servers that it has been configured. If no server is available, corresponding message will be shown.
* If the connect is successful, it will read commands from user’s screen

## Intermediate Interaction:

* The client is not multithreaded. It will read a command from screen and wait for the result from server.
* If the shutdown command is requested, client will just exit the process. And the server will read ‘’ nothing from client’s socket, it will remove the entry of the client.
* If the download command is requested, it just sent to the server because server will handle this automatically. The reply of the command can be of three types:
  + IP type: it means that server has sent the ip of the server which has the file. So, it will send the connect to the ip and request the file.
  + FL type: It means that server has sent the whole file, it will read the file and store it to the client temporarily and open it on the client. When client save and close the file, it will be automatically uploaded to the server.
  + ER type: it means that either server does not have that file or any other error occurred.
* When the delete command is requested, it just sent to the server
* When the client reads No server Available, it will just terminate
* When the client reads IP type and ip value, it will send connect to the ip and request the same command which it has requested previously.

# Assumptions and Limitations:

* Upload download model is used.
* Virtual directory structure is made in which dir\_struct is being dumped to the server physically each time when an update is made.
* When the client is updating file and the server goes down at the same time. The client will not be able to save the file to the server
* When only one server is available, no replication is possible.
* Administrator must know how many servers are running at a time.
* Administrator must know if he/she is starting first server.
* Every server is identified by its hostname. If hostname is changed, the file cannot be recovered.
* N+1 replication has been done.
* A message protocol has been introduced in which every time a message is sent, it appends its length with the message.
* Every server has root directory
* Sub directories are not implemented.
* Session semantics has been used when client close the file and it will be uploaded, then changes will be reflected to others.
* It will only run on Linux OS