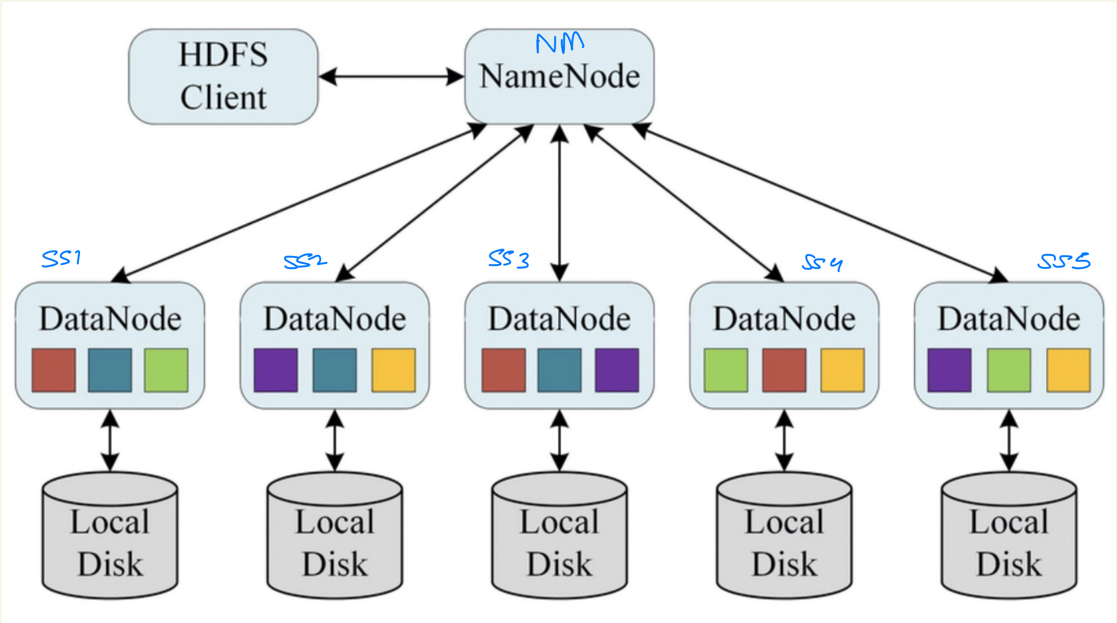


The overview of the Hadoop Distributed File System (HDFS).

NM

- Stores relevant data for all the SS
  - $\downarrow$ IP and 2 ports / SS

- Regulates the 3 commands
  - Create a empty file or directory
  - Delete a file or directory
  - Copy files or directories

- Send acknowledgements to clients when a request is made

- Handle multiple clients at once

→ Send ACK to client when request is made, if no ACK is recieved by client, then client shows a timeout message
(?? → when & why would this happen)

→ Only one client can write at a time, but multiple clients can read simultaneously

→ Send initial ACK and final ACK to handle multiple clients

- Error handling
- Trie / hashmap search
- LRU Caching (store accessible paths and the corresponding Socket information (IP + PORT))
- SS backup and recovery (+ async updation in the backups)
- Logging

CLIENTS

- READ dir1/dir2/dir3/yo.txt → NM finds the SS where the file is stored and returns the IP and port of the server where it found the file
- Stop communication when a STOP packet is sent by the SS

- Read, Write, info $\downarrow$ WITHOUT NM

Create, Delete & Copy $\downarrow$ THROUGH NM

Copy $\rightarrow$ Source and destination is given by the Client

Storage Server (SS)

- When new SS is started, signal the NM with it's IP and 2 ports
- Creating and deleting files and directories
- Manage port to destination

$\rightarrow$ These 3 are SS $\longleftrightarrow$ NM functionalities

- Read, write, file permissions $\longrightarrow$ SS $\longleftrightarrow$ Client functionality

DATA STRUCTURES

1. NM lookup table → Hash map

{ key : value }

key → acc path (string)

value → IP (string)

value → this can be a pair (for storing both sockets)

2. STATES

```
{
    ~arjun/IZZT/yo.txt : {
                          num_readers : integer = 0
                          writingON : true ,
                            ⋮
                         } ,
    ~hardik/IZZT/yo.txt : {
                          num_readers : integer = 2
                          writingON : false ,
                            ⋮
                         }
}
```

3 BACKUP ACC. PATHS

↳ struct same as acc. paths

4 ~~{ IP: [list of acc. path]~~

key: label

FILES

1. Client → ⊙ main.c (asks for name & port (if wanted))
    - ↳ UTILS → ⊙ server setup.c
        - ⊙ operations → ⊙ read.c
            - ⊙ write.c
            - ⋮

2. Name Server → ⊙ main.c
    - ⊙ UTILS
        - ↳ NS server init
        - ↳ Client
        - ↳ SS → ⊙ remove all paths of a server.c
        - ↳ NS operations (caching, searching, SS_info_table, Accessible_paths)
            - Accessible_paths → hash, trie
        - ↳ Backup_acc_paths
        - ↳ STATES { who is reading[]; write_on bool; etc.

if num_servers > 2 then start backing up.

③ SS

main.c

UTILS/

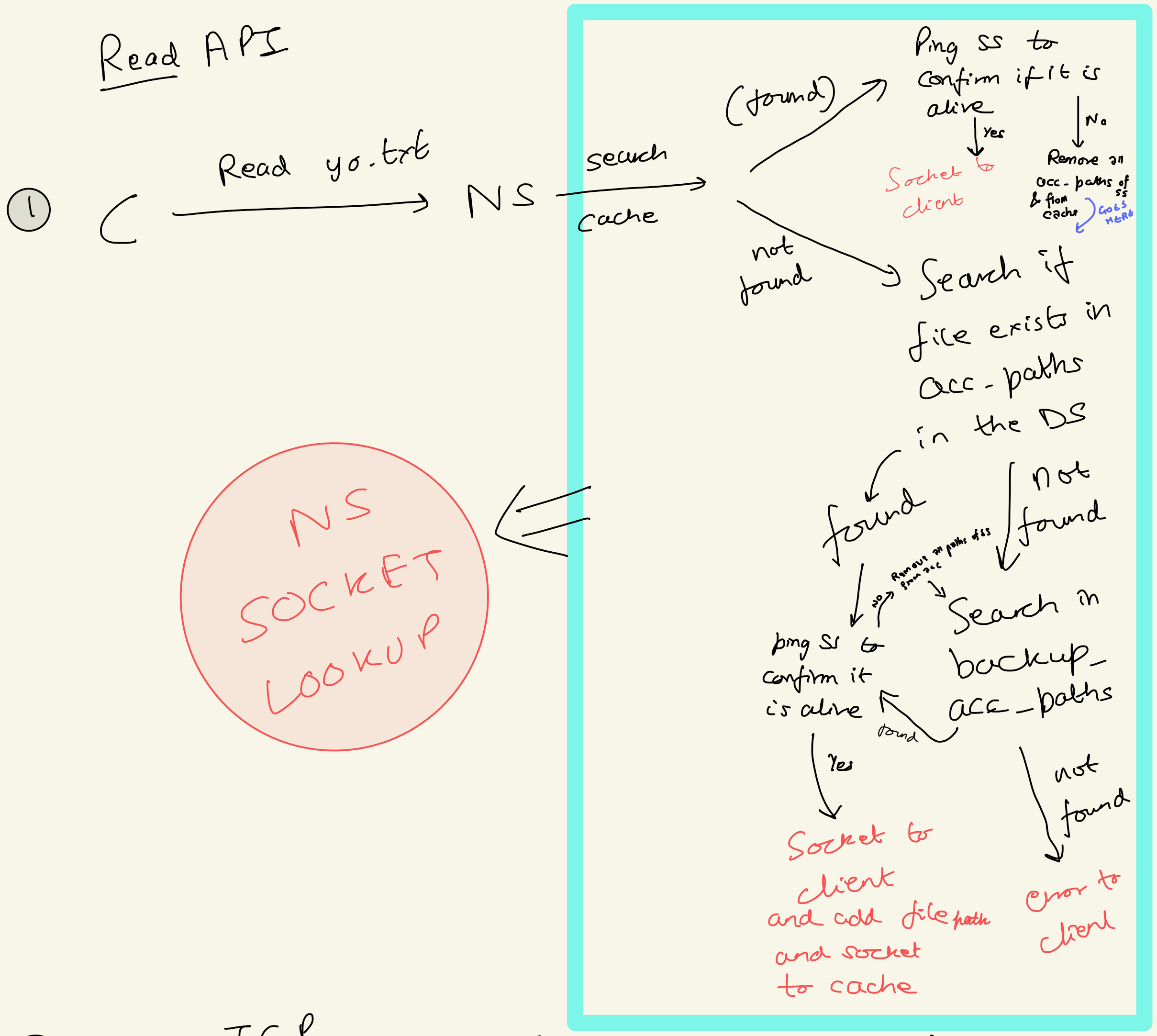
$$\begin{tikzcd}
 & \text{backup file} \\
\text{backup.c} \arrow[ur] \arrow[dr] & \\
 & \text{backup server}
\end{tikzcd}$$

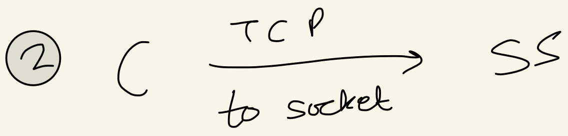
init.c

copy.c

APIs

Read API


1
C
Read yo.txt
NS
search cache
(found)
Ping SS to confirm if it is alive
Yes
Socket to client
No
Remove all occ-paths of SS & from cache
GOES HERE
not found
Search if file exists in acc-paths in the DS
NS SOCKET LOOKUP
found
ping SS to confirm it is alive
No
Remove all paths of SS from acc
Yes
Socket to client and add file path and socket to cache
not found
Search in backup_acc_paths
found
not found
error to client



1) Send init ACK to NM num_readers ++;
2) Send packets to C with STOP
3) Send final ACK to NM
↳ NM states update readers

Write API

①

$$\begin{tikzcd}[column sep=huge] C \arrow[r, "\text{Write yo.txt ``hi''}"] & NS \arrow[r, "\text{NS SOCKET}", "\text{LOOKUP}"'] & \text{}\end{tikzcd}$$

with "/0"

(without checking backups)

↓

socket found

NEW

1) write bool for yo.txt = 1 (NS STATES)

May have to handle simultaneous reading & writing

↓

Return socket to client

②

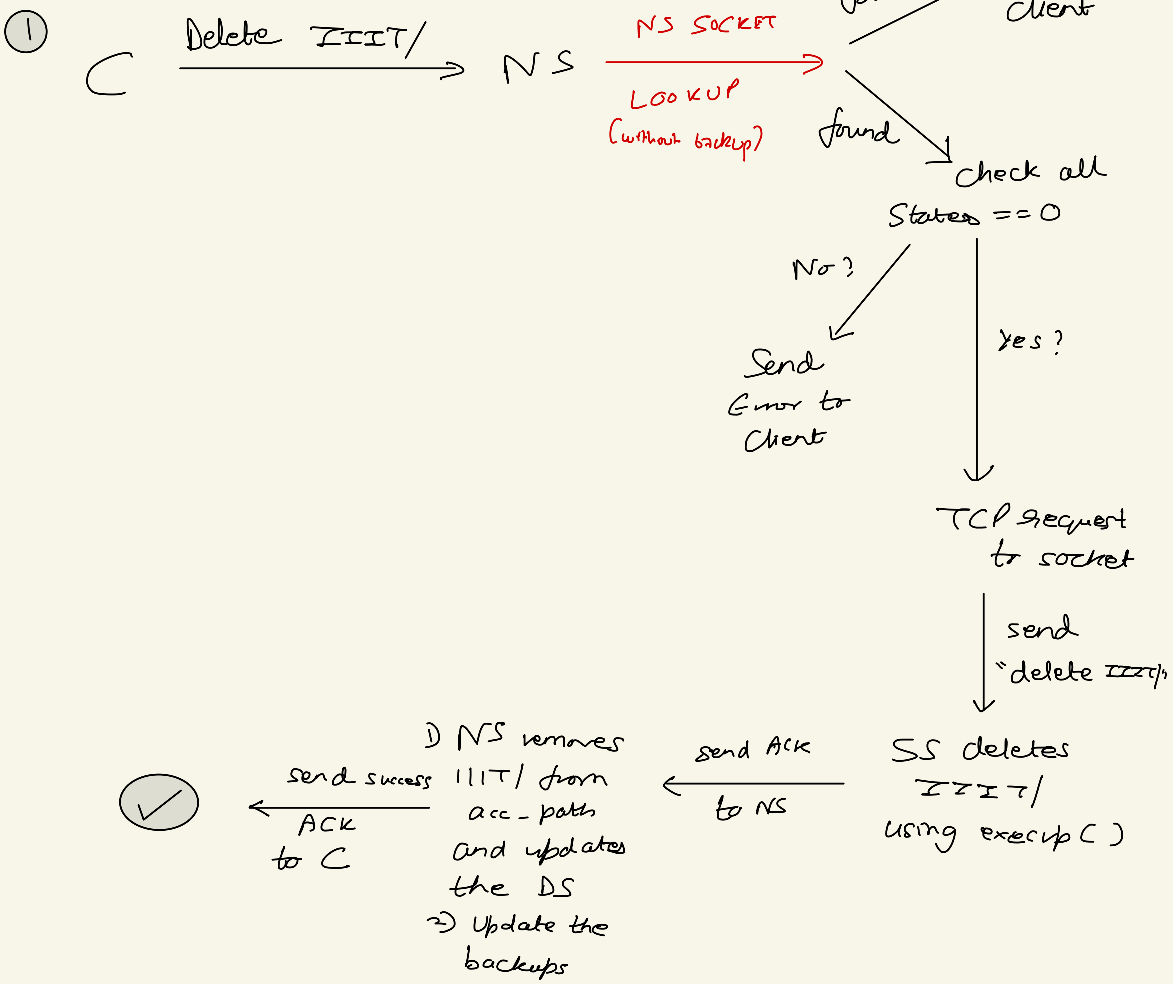
$$\begin{tikzcd}[column sep=huge] C \arrow[r, "\text{``hi''}", "\text{open TCP socket}"'] & SS \arrow[r] & \text{}\end{tikzcd}$$

1) Send init ACK to NM
2) Write in SS
3) Send final ACK to NM, client
   ↳ NM state update writers
4) Update the 2 backups

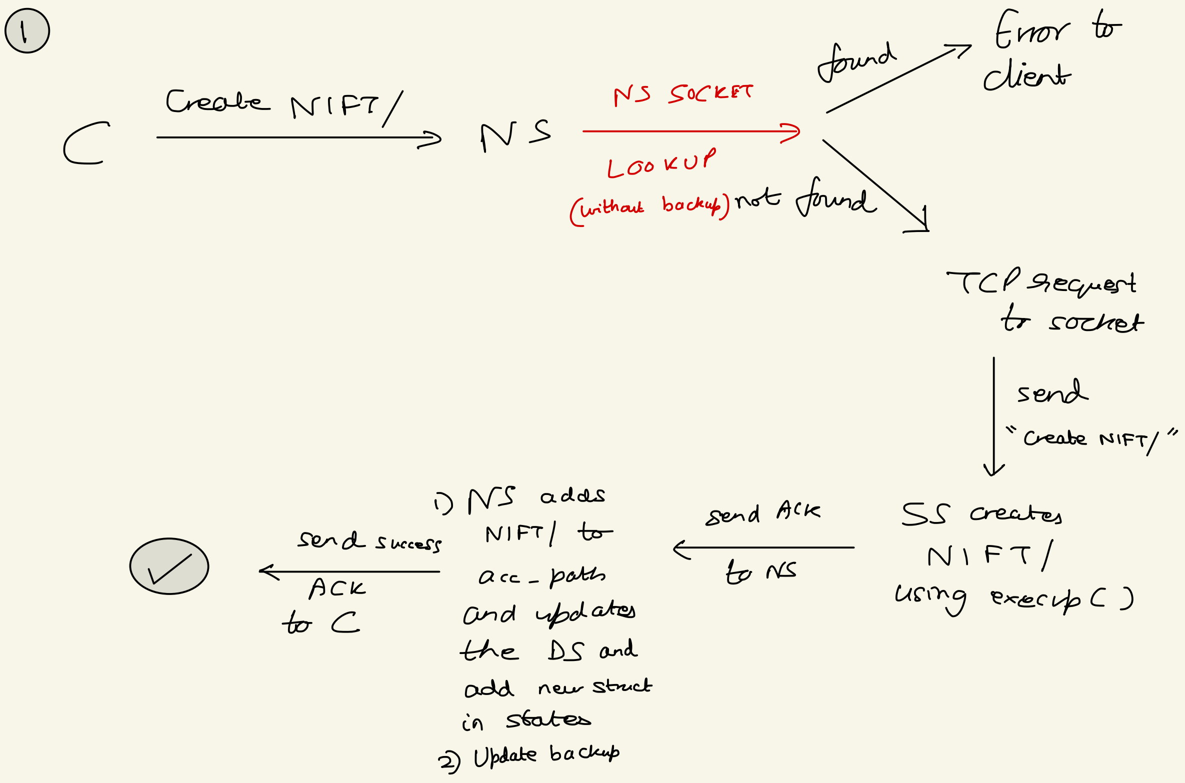
Info API

Ditto same as read API

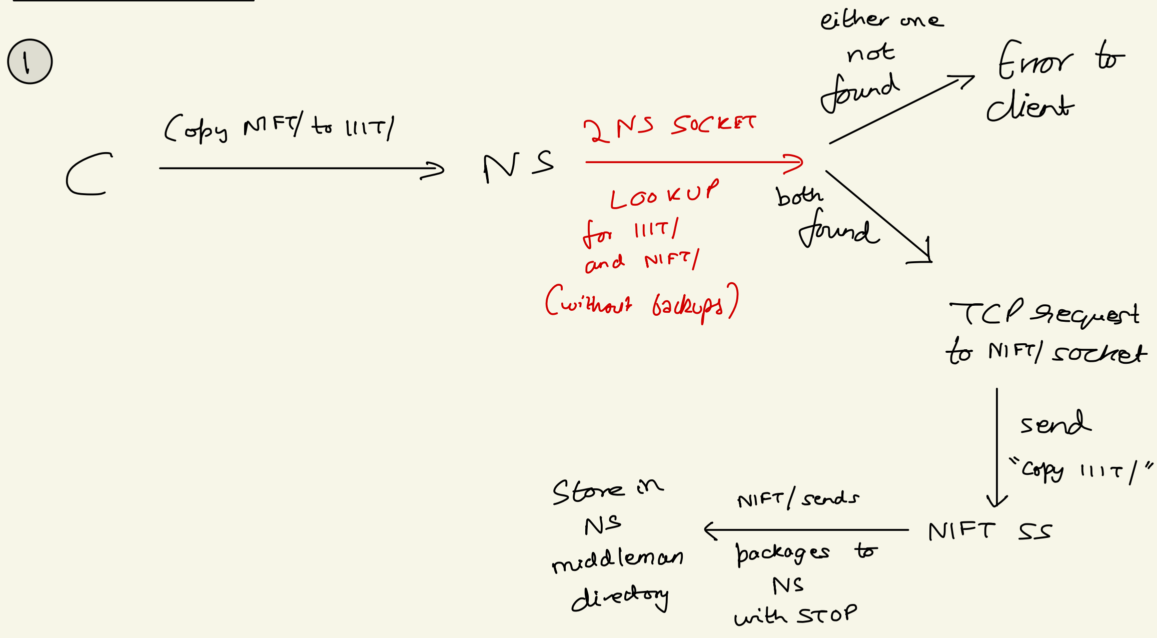
DELETE API

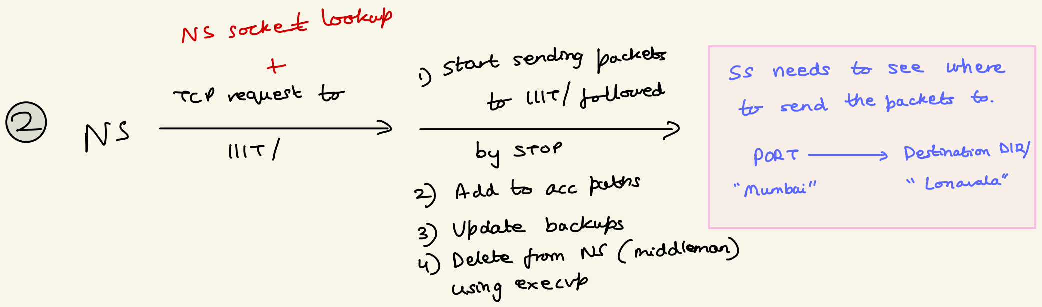

1
C
Delete IIIT/
NS
NS SOCKET
LOOKUP
(without backup)
not found
Error to client
found
check all States == 0
No ?
Send Error to Client
Yes ?
TCP request to socket
send "delete IIIT/"
SS deletes IIIT/ using execvp( )
send ACK
to NS
1) NS removes IIIT/ from acc_path and updates the DS
2) Update the backups
send success ACK to C
✓

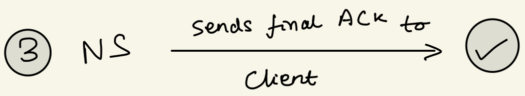
CREATE API


1
C
Create NIFT/
NS
NS SOCKET
LOOKUP
(without backup)
found
Error to client
not found
TCP request to socket
send "Create NIFT/"
SS creates NIFT/ using execvp( )
send ACK to NS
1) NS adds NIFT/ to acc_paths and updates the DS and add new struct in states
2) Update backup
send success ACK to C
✓

COPY API


1
C
Copy NIFT/ to IIIT/
NS
2 NS SOCKET
LOOKUP for IIIT/ and NIFT/ (without backups)
either one not found
Error to client
both found
TCP request to NIFT/ socket
send "copy IIIT/"
NIFT SS
NIFT/ sends packages to NS with STOP
Store in NS middleman directory


2
NS
NS socket lookup + TCP request to IIIT/
1) start sending packets to IIIT/ followed by STOP
2) Add to acc paths
3) Update backups
4) Delete from NS (middleman) using execvp
SS needs to see where to send the packets to.
PORT → Destination DIR/
"Mumbai" "Lonavala"


3
NS
sends final ACK to Client
✓