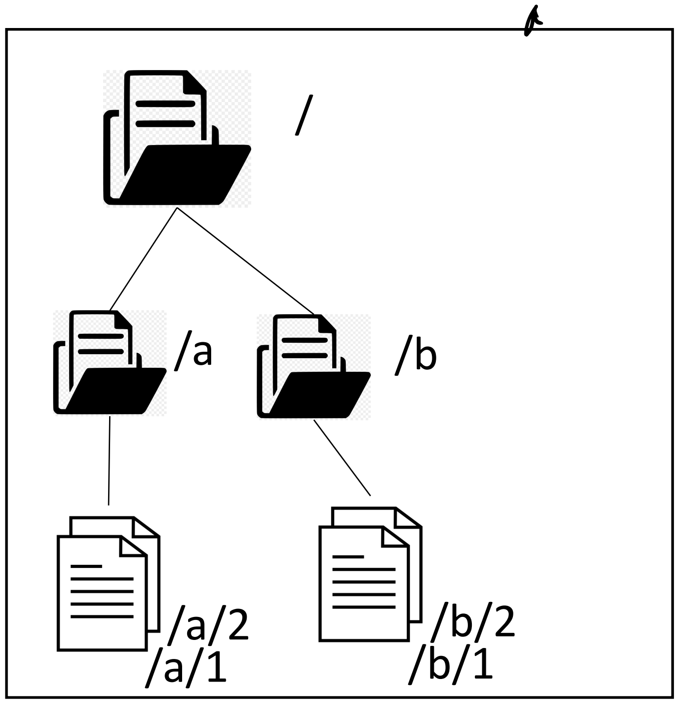
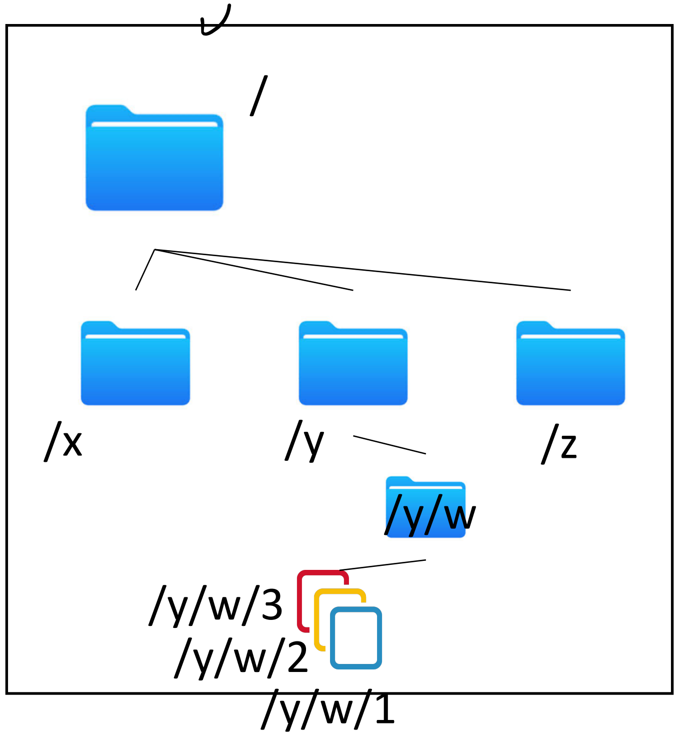
# Networked File System

NFS

# How to share files across machines?





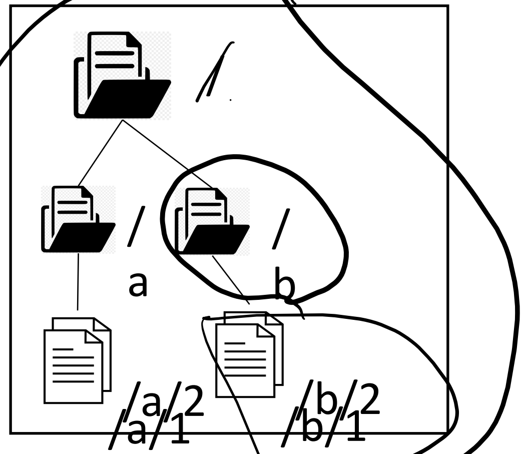
# Goals of NFS

- Small collaboration environment
- Transparent
  - Accessing remote files is just like accessing local files
- Performance
- Easy failure recovery

# NFS procedure from a user’s perspective

- mount <M2>:/y /b
- fd = open (“/b/w/1”, …);
- read (fd, …, …)
- write (fd, …, …)
- close (fd);

l f.s.



mount dev:/ /usb

mount /usb dev:/

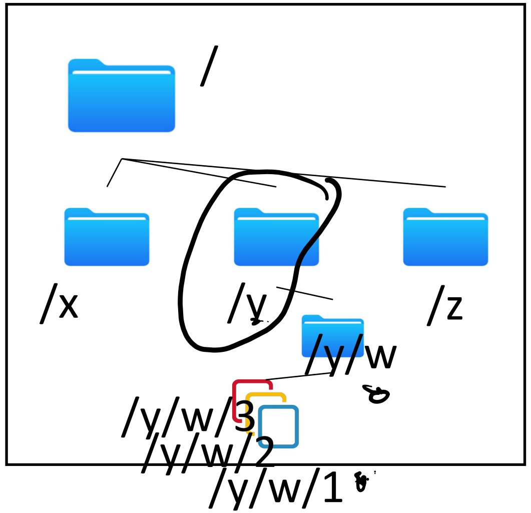
/a/2

/m

/b/1 X

/b/w/1 ✓

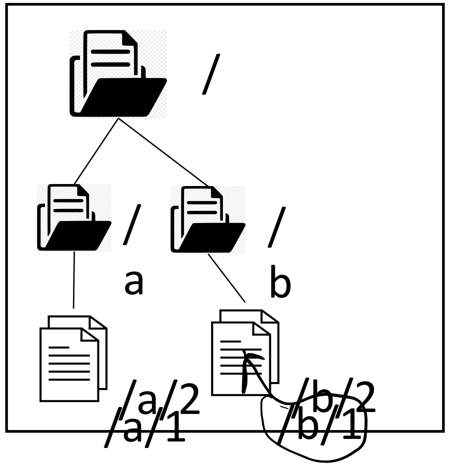
F: /usb x/z/y

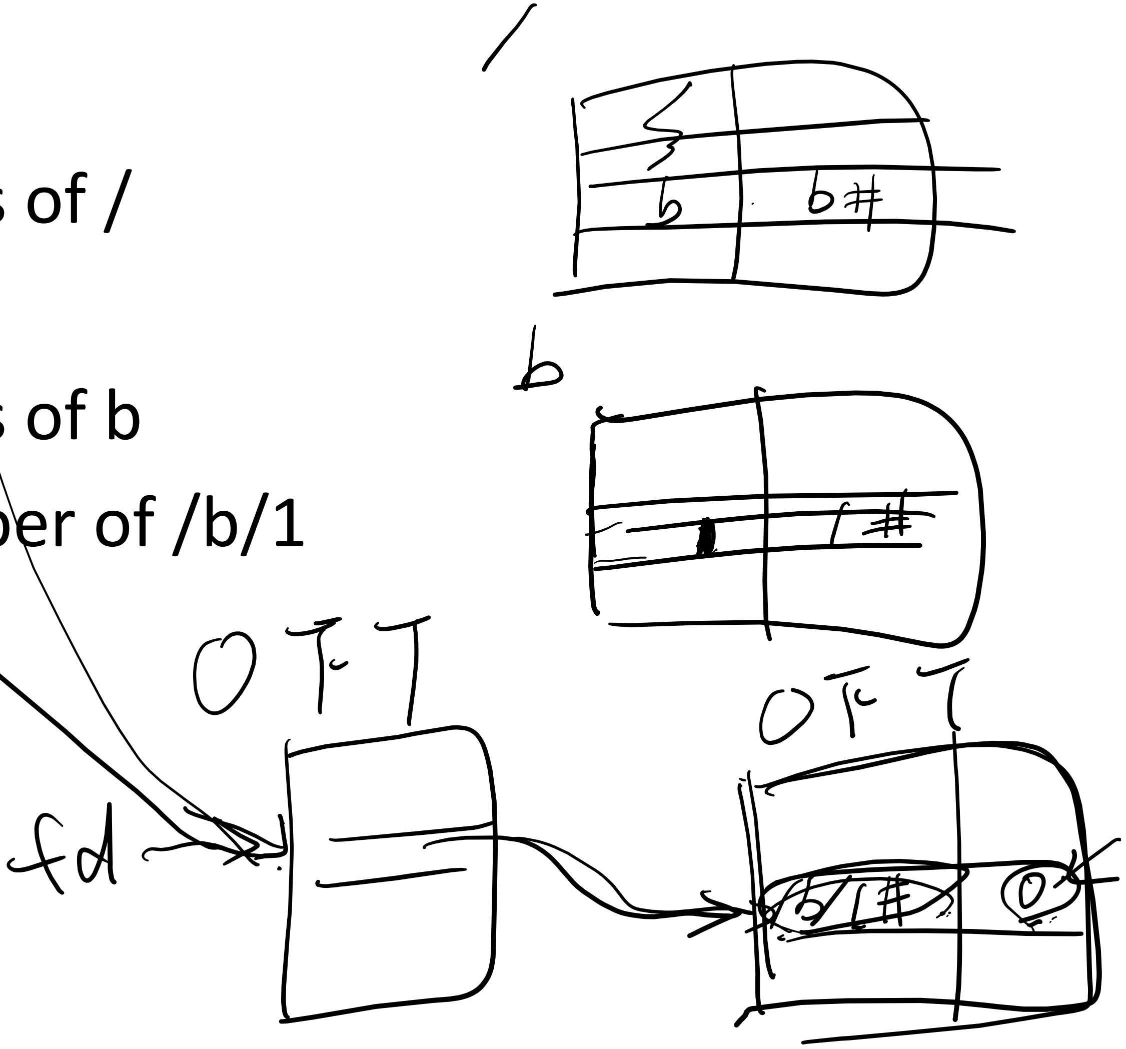


read (fd, ~ )

# Open “/b/1” on local machine

- Read i-node of /
- Read data blocks of /
- Read i-node of b
- Read data blocks of b
- Get i-node number of /b/1





mount

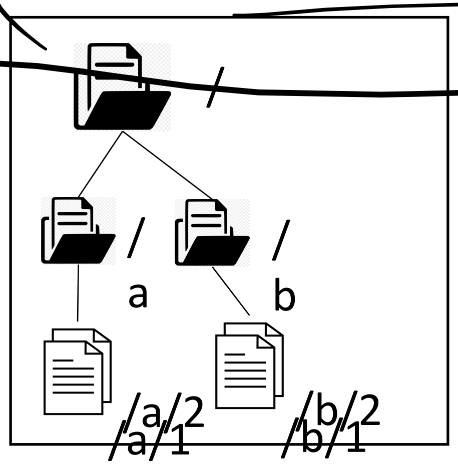
# Open “/b/1” on local machine

IN: 1

- Read i-node of “/”
- Read data block of “/”   (locate [b, i-node b])

IN: b#

- Read i-node of “/b”
- Read data block of “/b”   (locate [1, i-node of 1])
- Read i-node of “/b/1”; update open file table



How/Where does it change to get remote files?

# Open “/y/w/1” on a remote machine

- Mount

Mount table

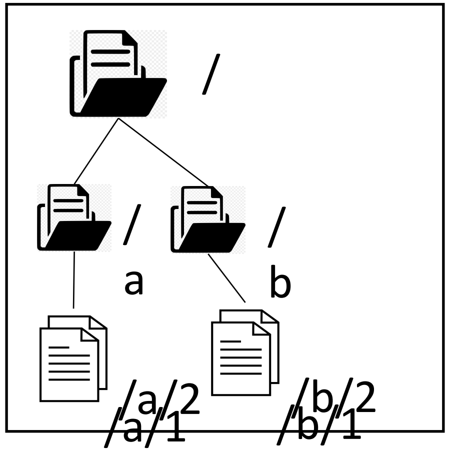
| <path> | Device |
|---|---|
| <path> | Remote machine address,<br>***file handle*** of remote mount point |

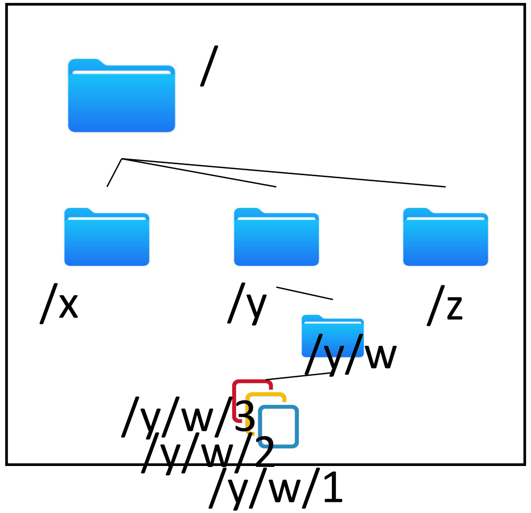
/b

M2

NFS

/y





ID

# Open "/y/w/1" on a remote machine

i-node access right

fs version#

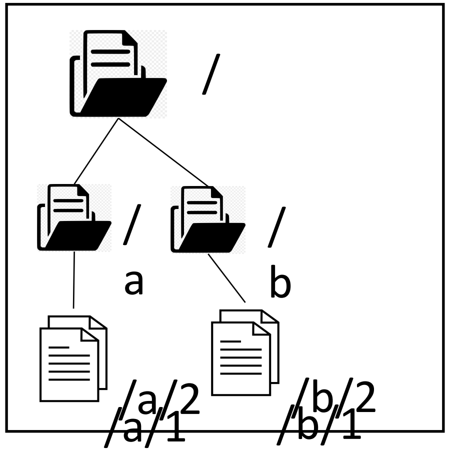
- Mount

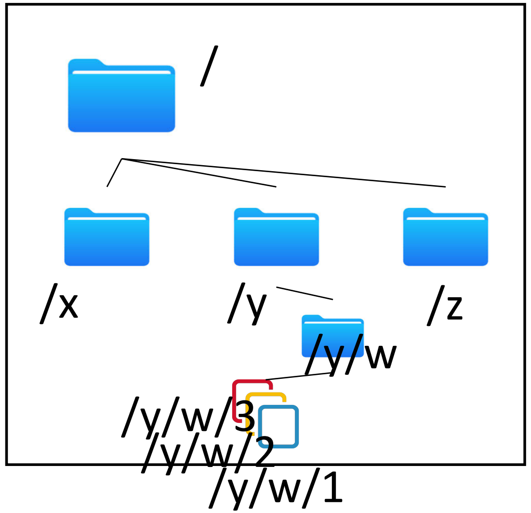
Mount table

| <path> | Device <M2> |
|---|---|
| <path><br>/b | Remote machine address,<br>***file handle*** of remote mount point |

fs : i-node : v#

How to identify a directory/file on a remote machine??



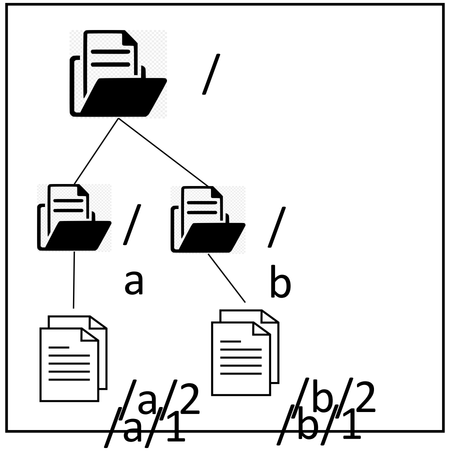


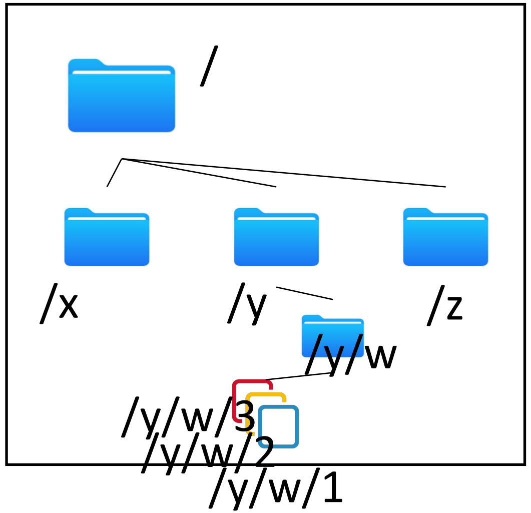
# Open "/y/w/1" on a remote machine

- Mount m2:/y /b

Mount table

| <path> | Device |
|---|---|
| <path> | Remote machine address,<br>***file handle*** of remote mount point |
| | |



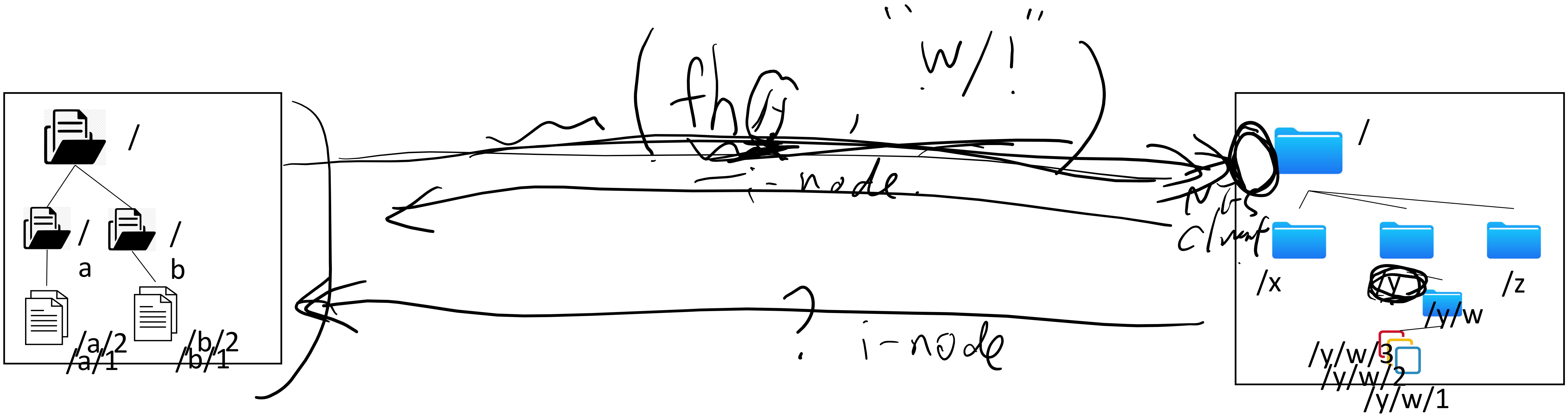


# Open “/y/w/1” on a remote machine

- Mount m2:/y /b
- open(“/b/w/1”, …)
  - Step 1: look up the mount table
  - Step 2: send ??? to machine M2

Mount table

| <path> | Device |
|---|---|
| /b | M2, file handle of /y |
| | |



read(fd, );

Ask about one file at a time!

# Open "/y/w/1" on a remote machine

- Mount m2:/y /b
- fd = open ("~~/b~~/w/1", …)
  - Step 1: look up the mount table
  - Step 2: send ??? to machine M2

<M2>

fh

OFT

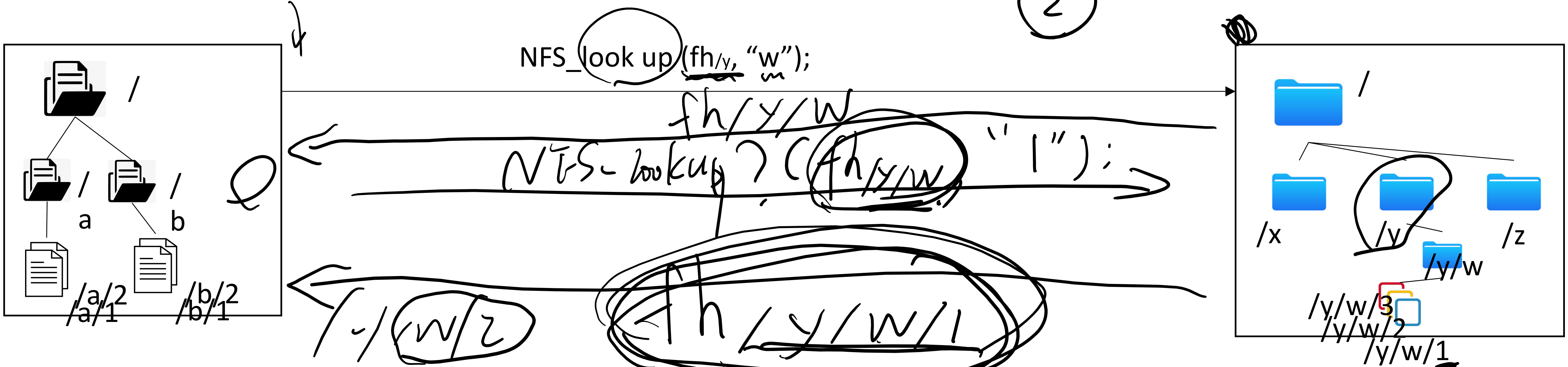
offset

① fh/y, w/1

②

Mount table

| <path> | Device |
|---|---|
| /b | M2, file handle of /y |
| | |



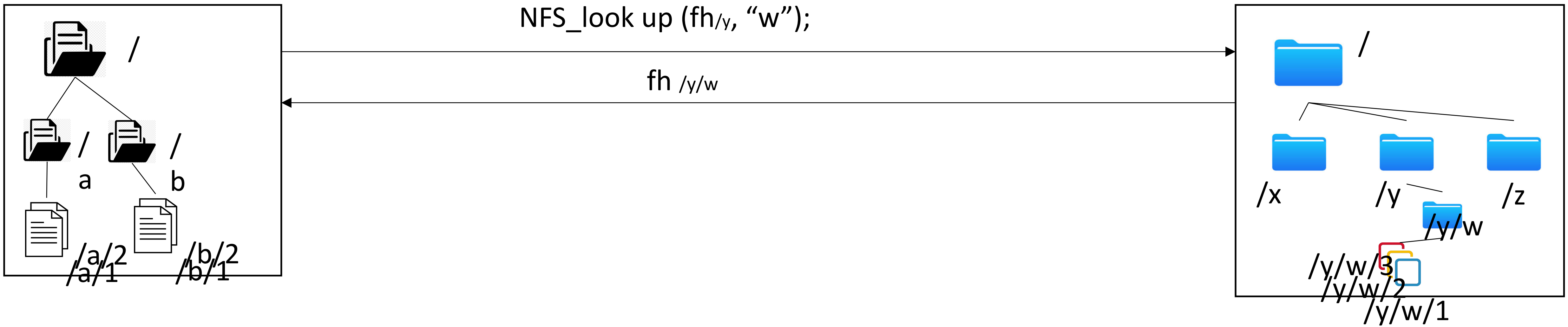
Ask about one file at a time!

# Open “/y/w/1” on a remote machine

- Mount m2:/y /b
- open (“/b/w/1”, …)
  - Step 1: look up the mount table
  - Step 2: send ??? to machine M2

Mount table

| <path> | Device |
|---|---|
| /b | M2, file handle of /y |
| | |



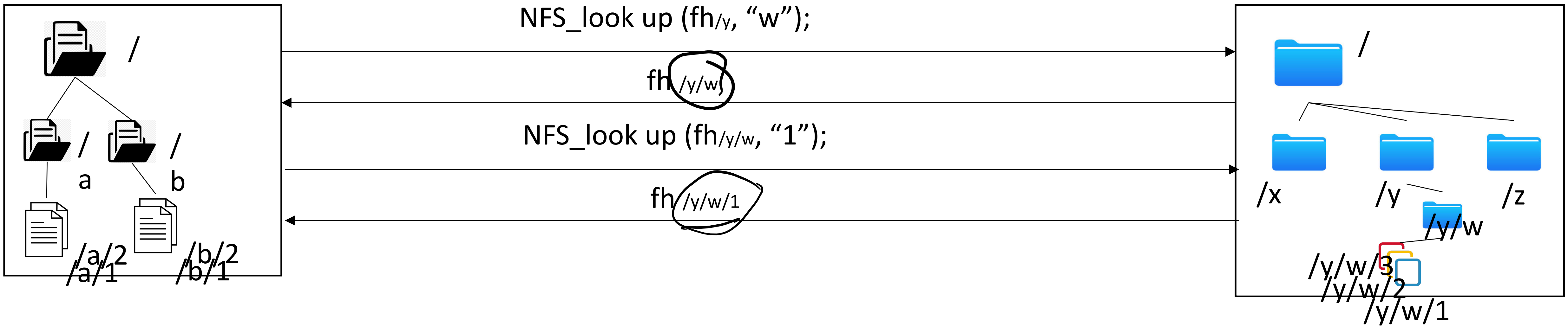
Ask about one file at a time!

# Open “/y/w/1” on a remote machine

- Mount  m2:/y   /b
- open (“/b/w/1”, …)
    - Step 1: look up the mount table
    - Step 2: send ??? to machine M2

Mount table

| <path> | Device |
|---|---|
| /b | M2, file handle of /y |
| | |



NFS_look up ($fh_{/y}$, “w”);

$fh_{/y/w}$

NFS_look up ($fh_{/y/w}$, “1”);

$fh_{/y/w/1}$

# Open "/y/w/1" on a remote machine

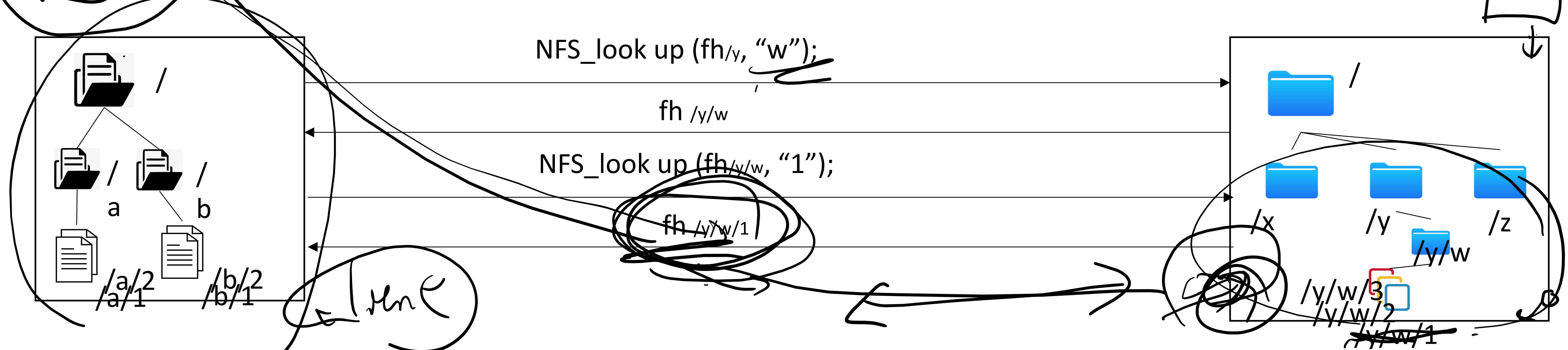
- Mount  m2:/y   /b
- fd = open ("/b/w/1", …)
  - Step 1: look up the mount table
  - Step 2: send ??? to machine M2
  - Step 3: store <m2>:$fh_{/y/w/1}$ to open file table on M1

OFT

Mount table

| <path> | Device |
|---|---|
| /b | M2, file handle of /y |
| | |

Is it stored on M2?

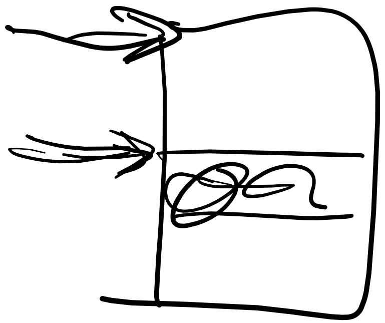


# NFS server is stateless

- Not keeping information about files opened by another machine
  - ➔ Easy for failure recovery

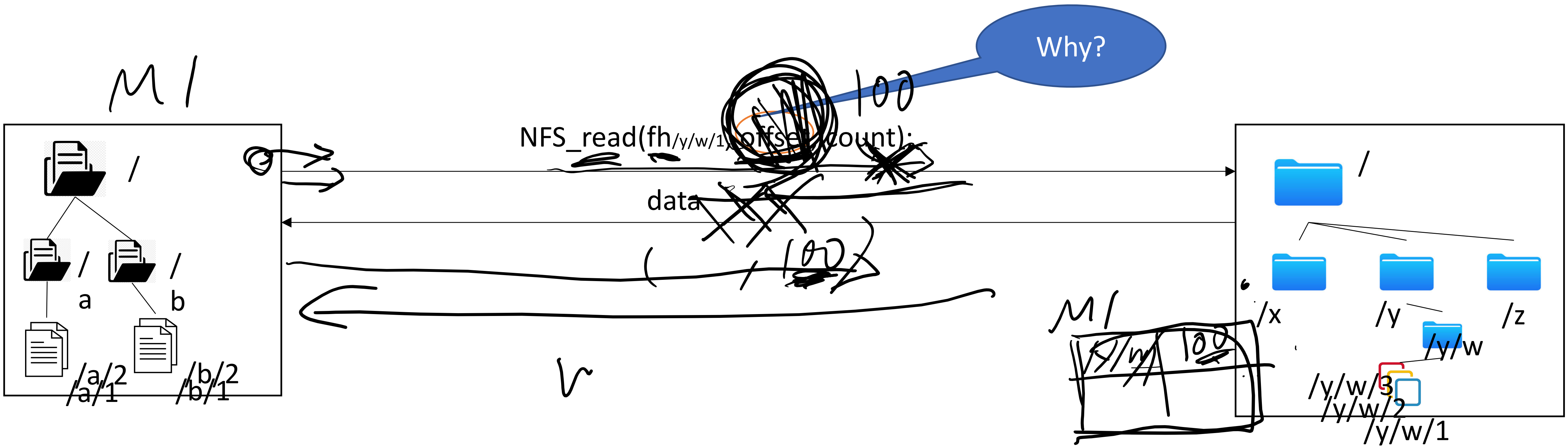
100

read (fd, void* buf, count)



Open file table

| i-node | offset |
|---|---|
| <M2> fh$_{/y/w/1}$ | 0 |
| | |



NFS_read(fh$_{/y/w/1}$, offset, count);

data

Why?

# NFS server is stateless

- Not keeping information about files opened by another machine
    - Easy for failure recovery

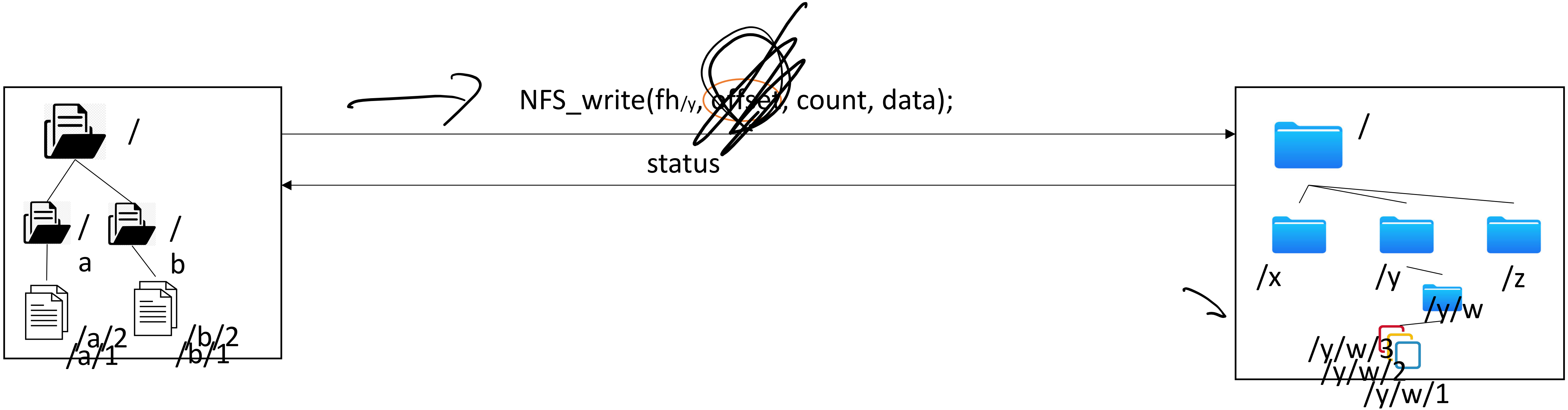
- Read request
    - has to include offset information
    - Read request is idempotent

**Idempotent operation O:**
doing O once is equivalent with doing O many times

# write (fd, void* buf, count)

Open file table

| i-node | offset |
|---|---|
| <M2> $fh_{/y/w/1}$ | 0 |
| | |



NFS_write($fh_{/y}$, ~~offset~~, count, data);

status

# NFS server is stateless

- Not keeping information about files opened by another machine
  - Easy for failure recovery
- Read request
  - has to include offset information
  - Read request is idempotent
- Write request
  - Has to include offset information
  - Write request is idempotent

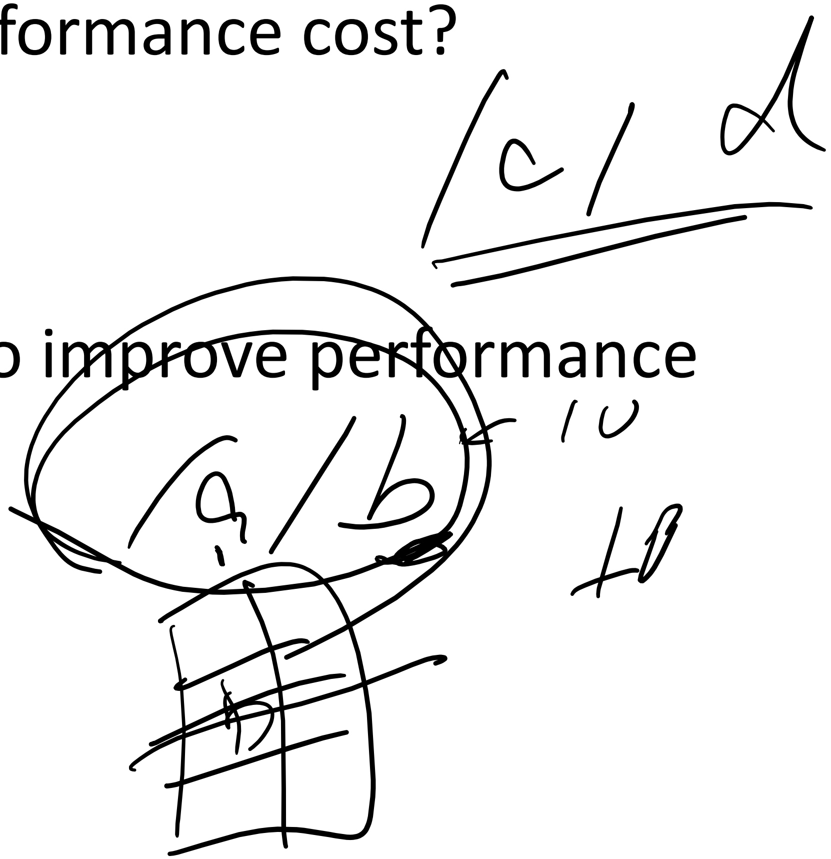
```
y=10;
x=y+1;
```

```
y=x;
x=y+1;
```

**Idempotent operation O:**
doing O once is equivalent with doing O many times

# Performance

- What is the performance cost?
- Using caching to improve performance



# Poll questions in today’s lecture

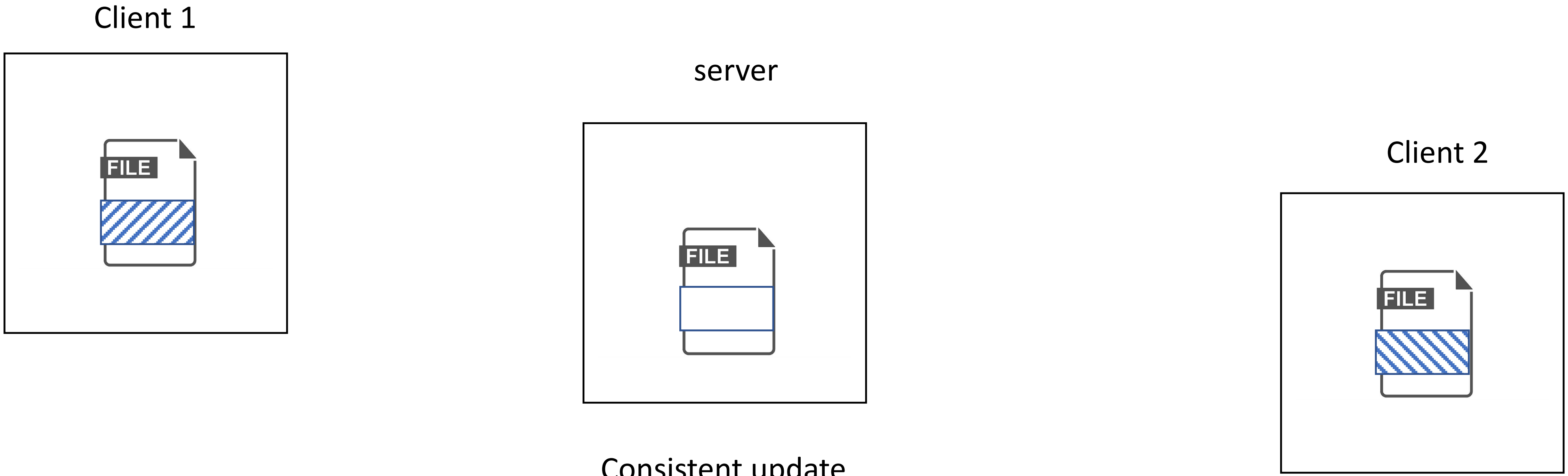
1. What is the file handle in NFS? ( Single Choice)
Answer 1: symbolic path and file name
Answer 2: i-node
**Answer 3: i-node and some other information**

2. what is returned by the open system call when I am opening a remote file through NFS ( Single Choice)
**Answer 1: file descriptor that points to the process' open file table**
Answer 2: file handle of the remote file
Answer 3: i-node content of the remote file

3. does the remote machine store the file handle of /y/w/1 into its kernel open file table? ( Single Choice)
Answer 1: yes
**Answer 2: no**

# Concurrent updates

- What if multiple machines are updating a file?



Client 1

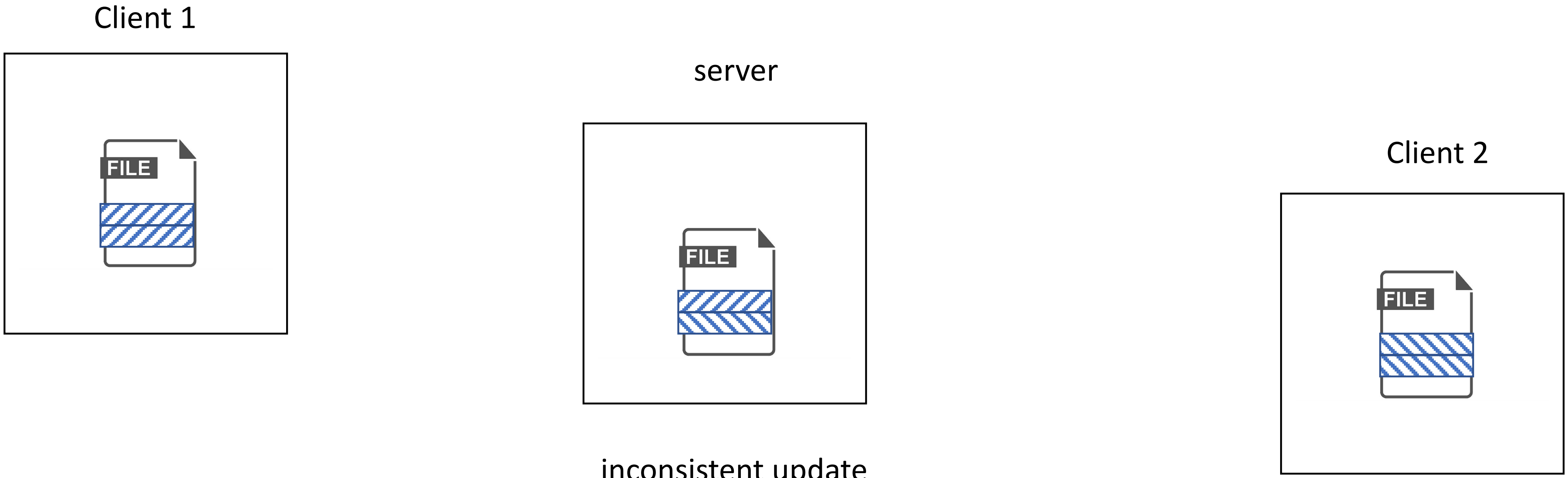
server

Client 2

Consistent update

# Concurrent updates

- What if multiple machines are updating a file?



Client 1

server

Client 2

inconsistent update

# Something about networked systems

- Using networked machines to improve capacity
- Paying attention to failure tolerance!
- Paying attention to concurrency/consistency control!
- RPC (remote procedure call)