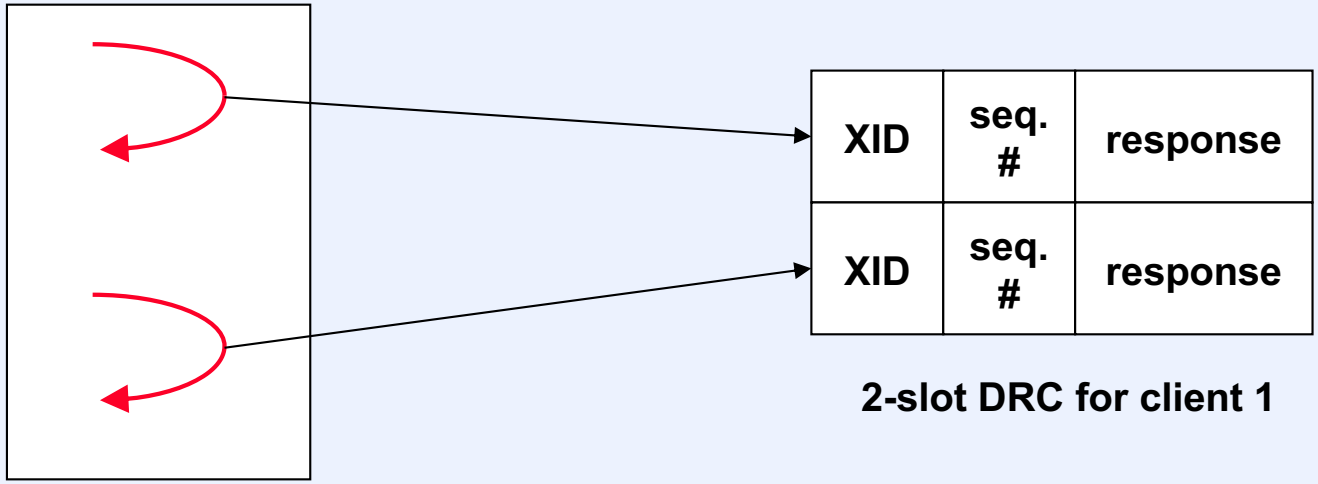
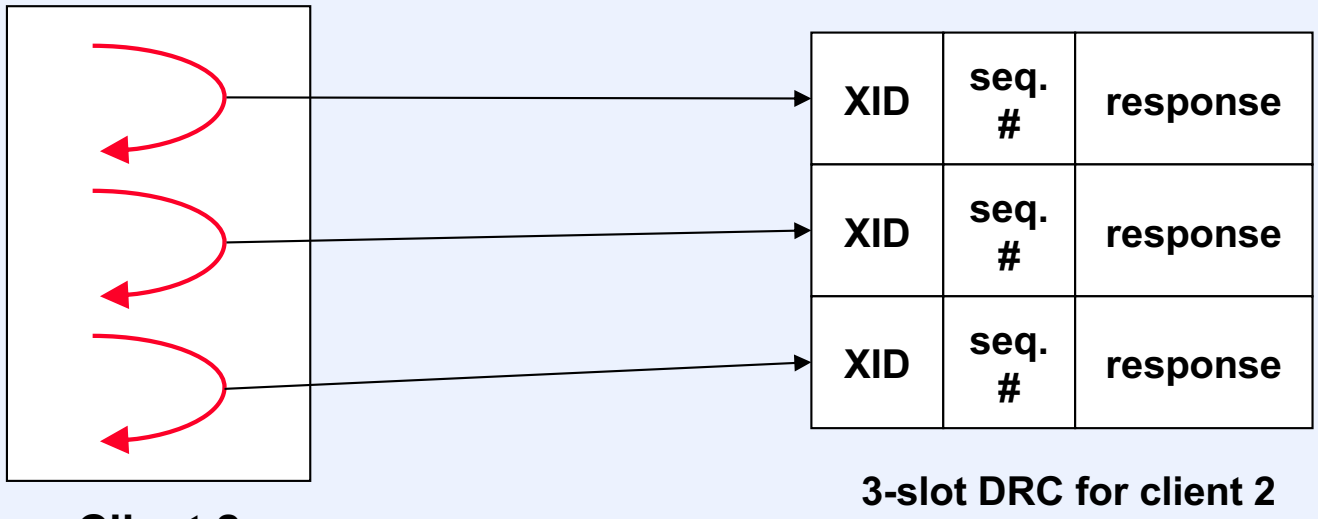
Remote Procedure Call Protocols (2)

What's Wrong?

- The problem is the duplicate request cache (DRC)
  - it's necessary
  - but when may cached entries be removed?

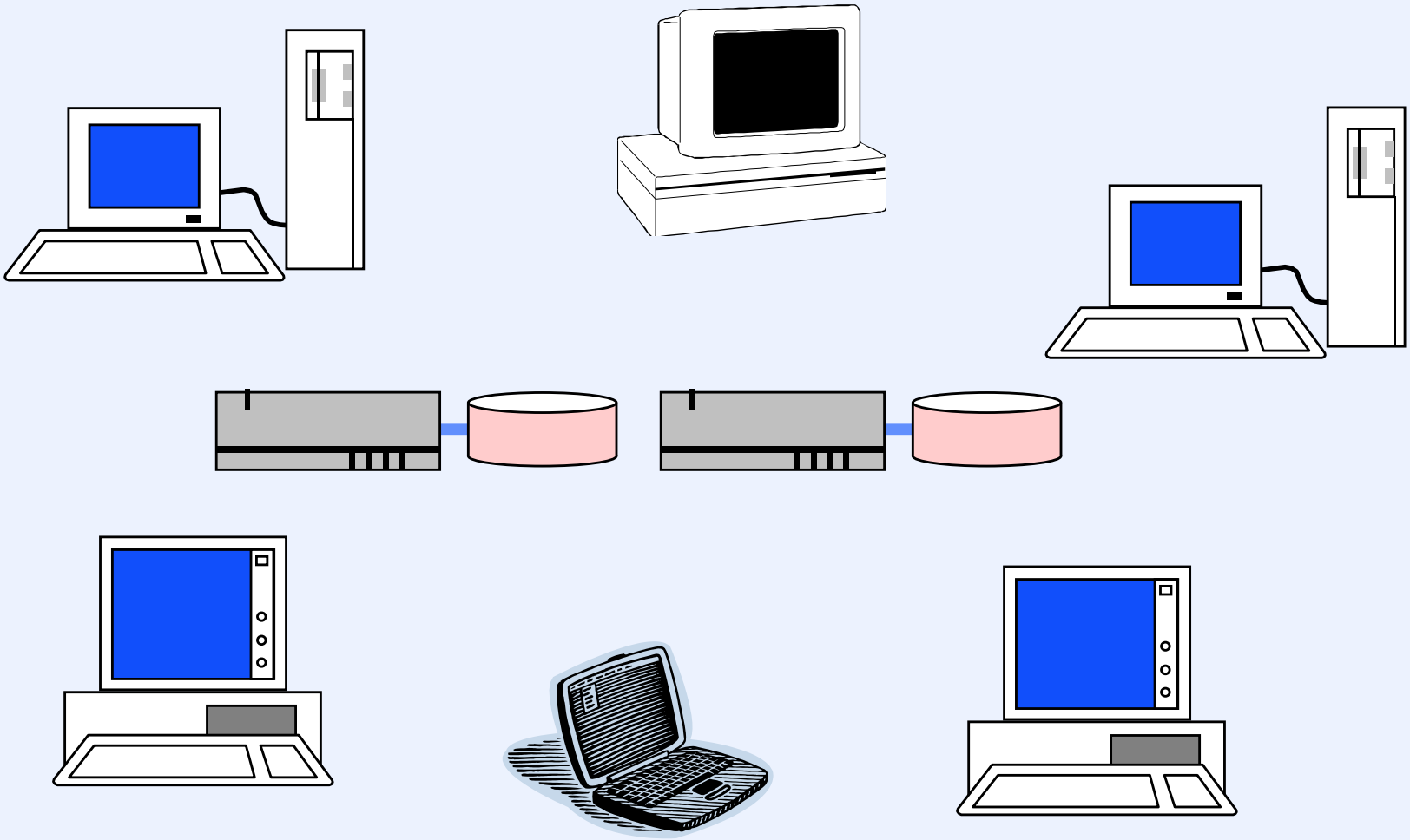
Session-Oriented RPC





Distributed File Systems Part 1

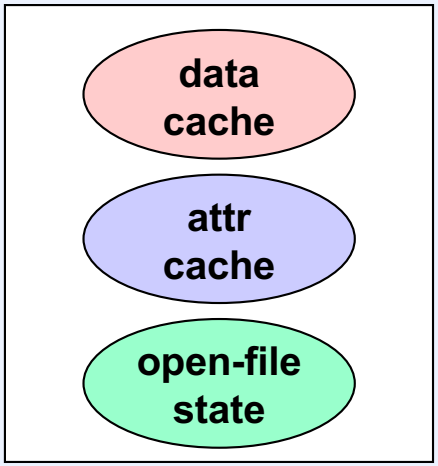
Distributed File Systems



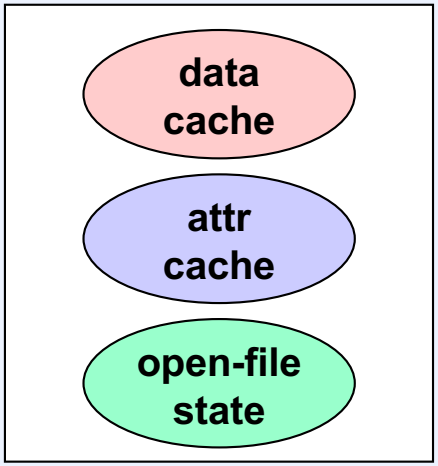
DFS Components

- Data state
  - file contents
- Attribute state
  - size, access-control info, modification time, etc.
- Open-file state
  - which files are in use (open)
  - lock state

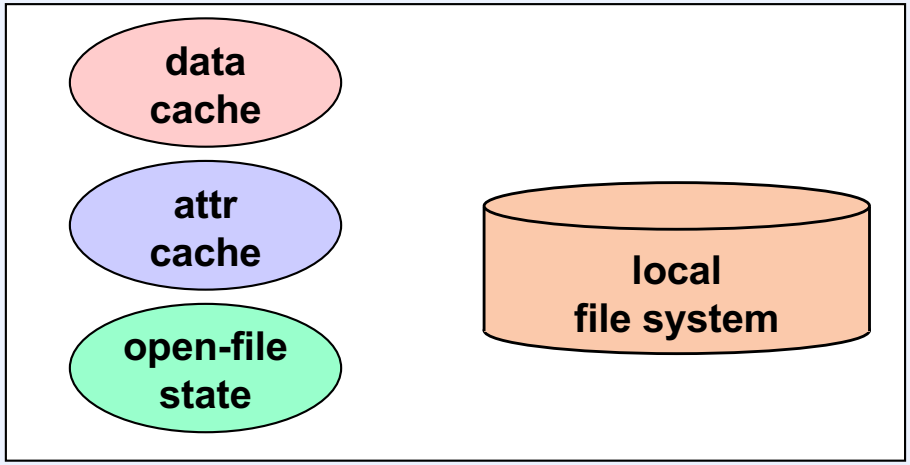
Possible Locations



Client



Client



Server

Quiz 1

We'd like to design a file server that serves multiple Unix client computers. Assuming no computer ever crashes and the network is always up and working flawlessly, we'd like file-oriented system calls to behave as if all parties were on a single computer.

a) It can't be done

b) It can be done, but requires disabling all client-side caching

c) It can be done, but sometimes requires disabling client-side caching

d) It can be done, irrespective of client-side caching

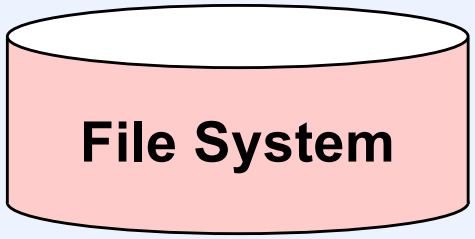
Guiding Principle

Principle of least astonishment (PLA)

– people don’t like surprises, particularly when they come from file systems

Single-Thread Consistency

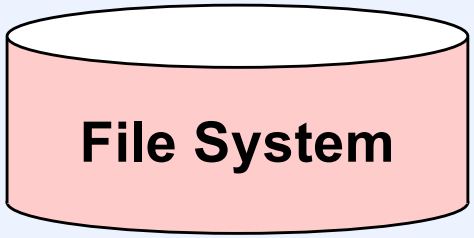
```
write(fd, buf1, size1);

read(fd, buf2, size2);


// no surprises if
// single-thread consistent

// Operations are time-ordered
```



Single-Client Consistency

```
% cp x y
%
```

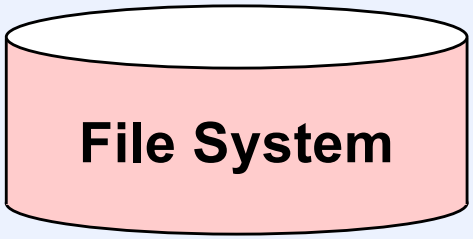
```
% cmp x y
%
```



Distributed Consistency

Ted's Computer

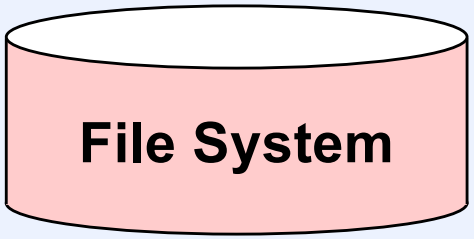
Alice's Computer



Strict Consistency

Ted's Computer

```
write(fd1, "A", 2);
write(fd2, "B", 2);
```



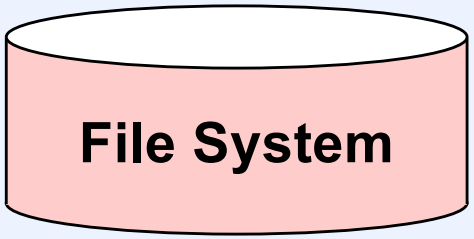
Alice's Computer

```
// an instant later …
read(fd1, buf1, 2);
read(fd2, buf2, 2);
// buf1 contains "A"
// buf2 contains "B"
```

Weak Consistency

Ted's Computer

```
write(fd1, "A", 2);
write(fd2, "B", 2);
```



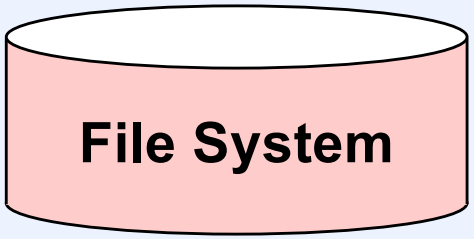
Alice's Computer

```
// a while later …
read(fd1, buf1, 2);
read(fd2, buf2, 2);
// maybe buf1 contains "A"
// maybe buf2 contains "B"
```

Sequential Consistency

Ted's Computer

```
write(fd1, "A", 2);
write(fd2, "B", 2);
```



Alice's Computer

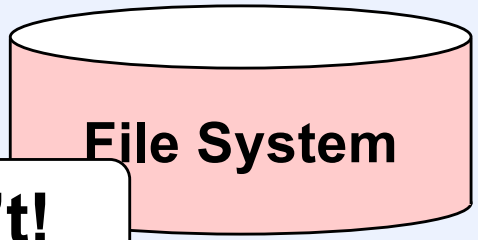
```
// an instant later …
read(fd1, buf1, 2);
read(fd2, buf2, 2);
// if buf2 contains "B"
// then buf1 contains "A"
```

Sequential Consistency

Ted's Computer

I just updated the file!

```
write(fd1, "A", 2);
write(fd2, "B", 2);
```



No you didn't!

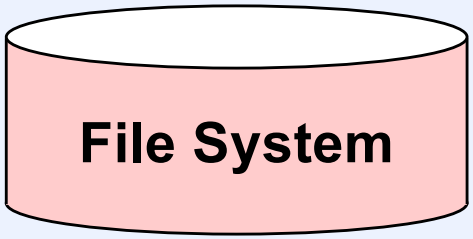
Alice's Computer

```
// an instant later …
read(fd1, buf1, 2);
read(fd2, buf2, 2);

// buf1 and buf2 contain "X"
```

Entry Consistency

Ted's Computer

```
writelock(fd);
write(fd, "B", 2);
unlock(fd);
```



Alice's Computer

```
// an instant later …

readlock(fd);
read(fd, buf, 2);
unlock(fd);

// buf now contains "B"
```

In Practice …

- Data state
  - NFS
    - single-client consistent
    - weakly consistent
  - SMB
    - strictly consistent
- Lock state
  - must be strictly consistent

Thursday morning, November 17th
At 7:00 a.m.

Maytag, the department's central file server, will be taken down to kick off a filesystem consistency check.

Linux machines will hang.

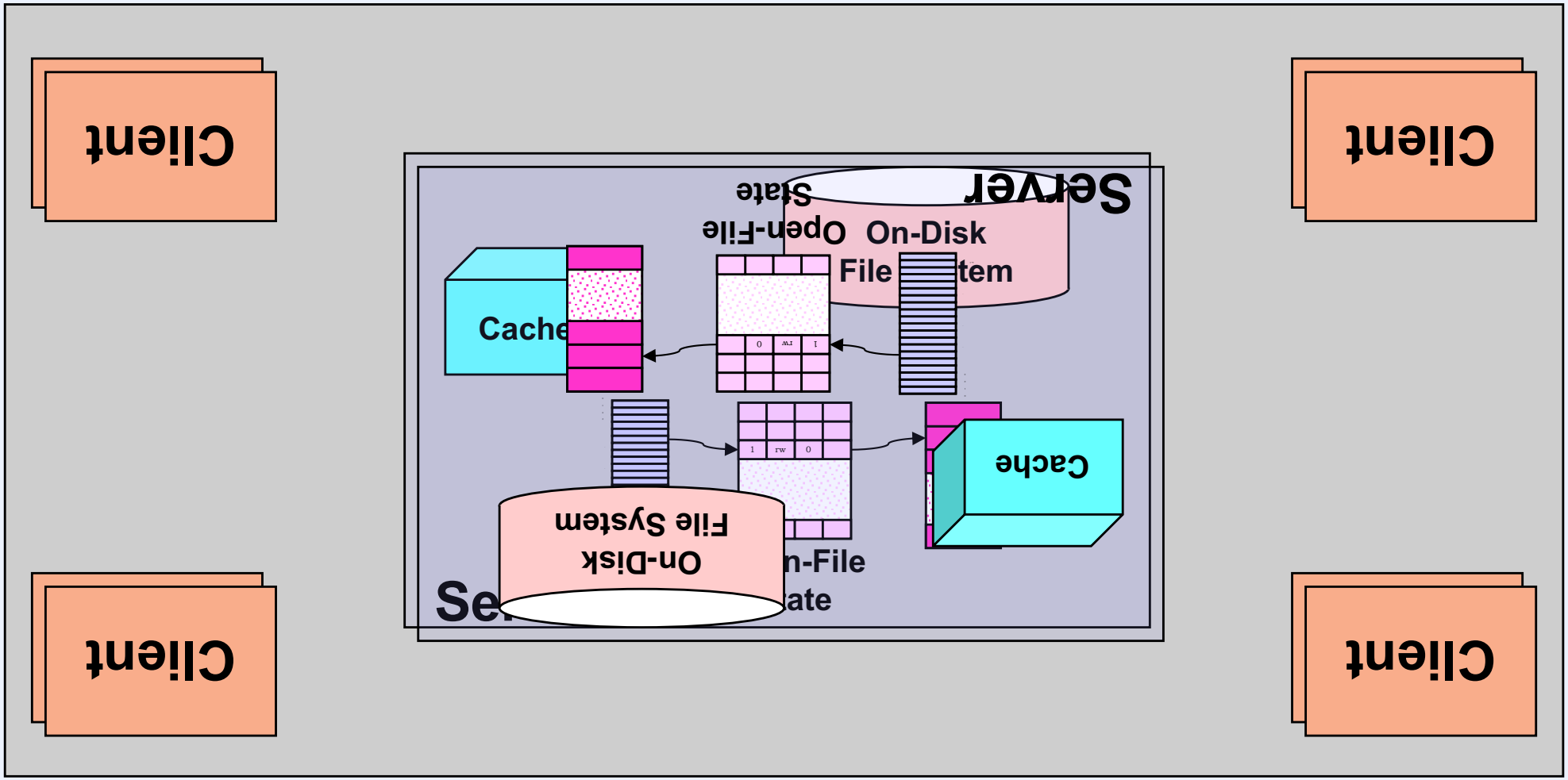
All Windows users should log off.

Normal operation will resume by 8:30 a.m. if all goes well.

All windows users should log off before this time.

Questions/concerns to problem@cs.brown.edu

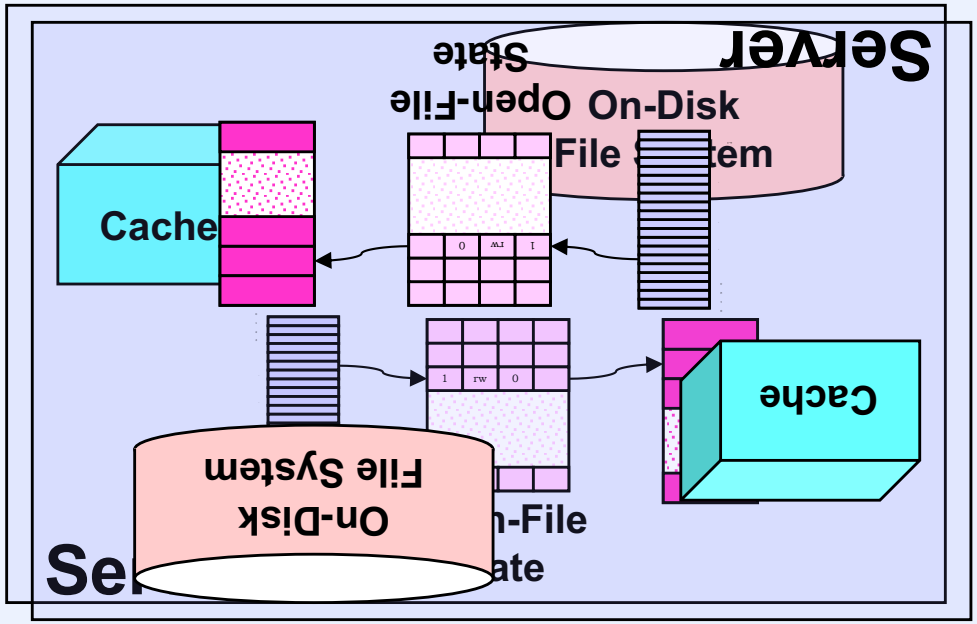
Failures in a Local File System



Distributed Failure

Client

Client



Client

Client

Quiz 2

We'd like to design a file server that serves multiple Unix client computers, but we now realize we must cope with failures. Which one of the following is not true?

a) At least one of the following statements is false

b) If we relax Unix system-call semantics a bit, this is easy

c) If we don't relax Unix system-call semantics, it's doable, but we need to introduce some new error messages for certain situations

d) There are failure modes that can't possibly occur if all parties are on the same computer

In Practice …

- NFS version 2
  - relaxed approach to consistency
  - handles failures pretty well
- SMB
  - strictly consistent
  - intolerant of failures

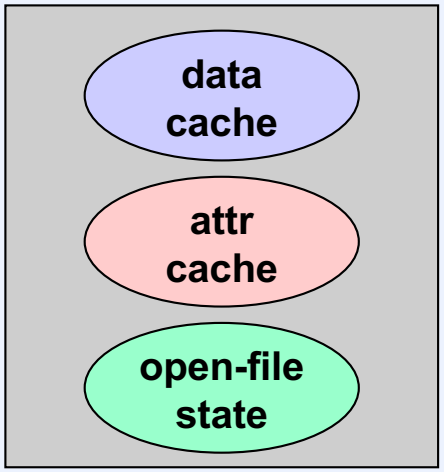
NFS Version 2

- Released in mid 1980s
- Three protocols in one
  - file protocol
  - mount protocol
  - network lock manager protocol

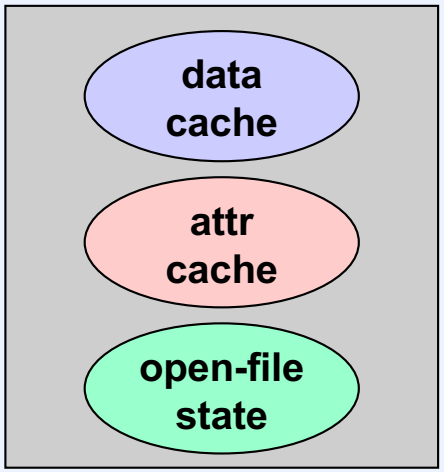
Basic NFS

Extended NFS

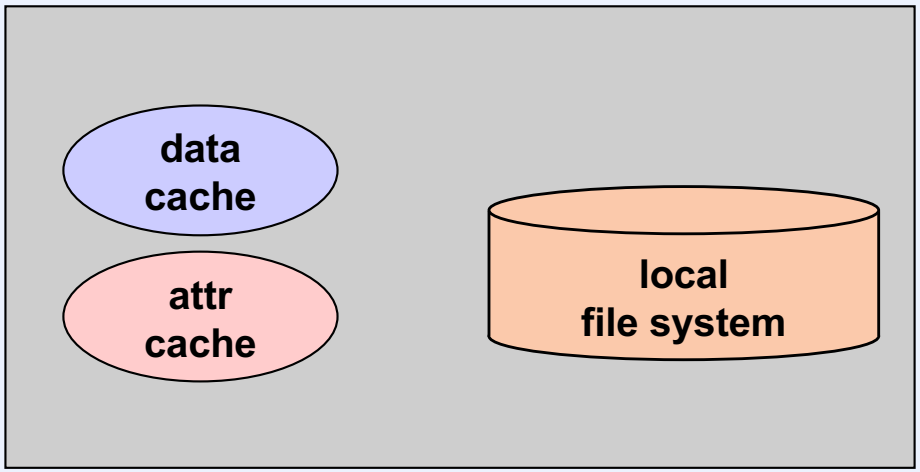
Distribution of Components



NFSv2 client



NFSv2 client

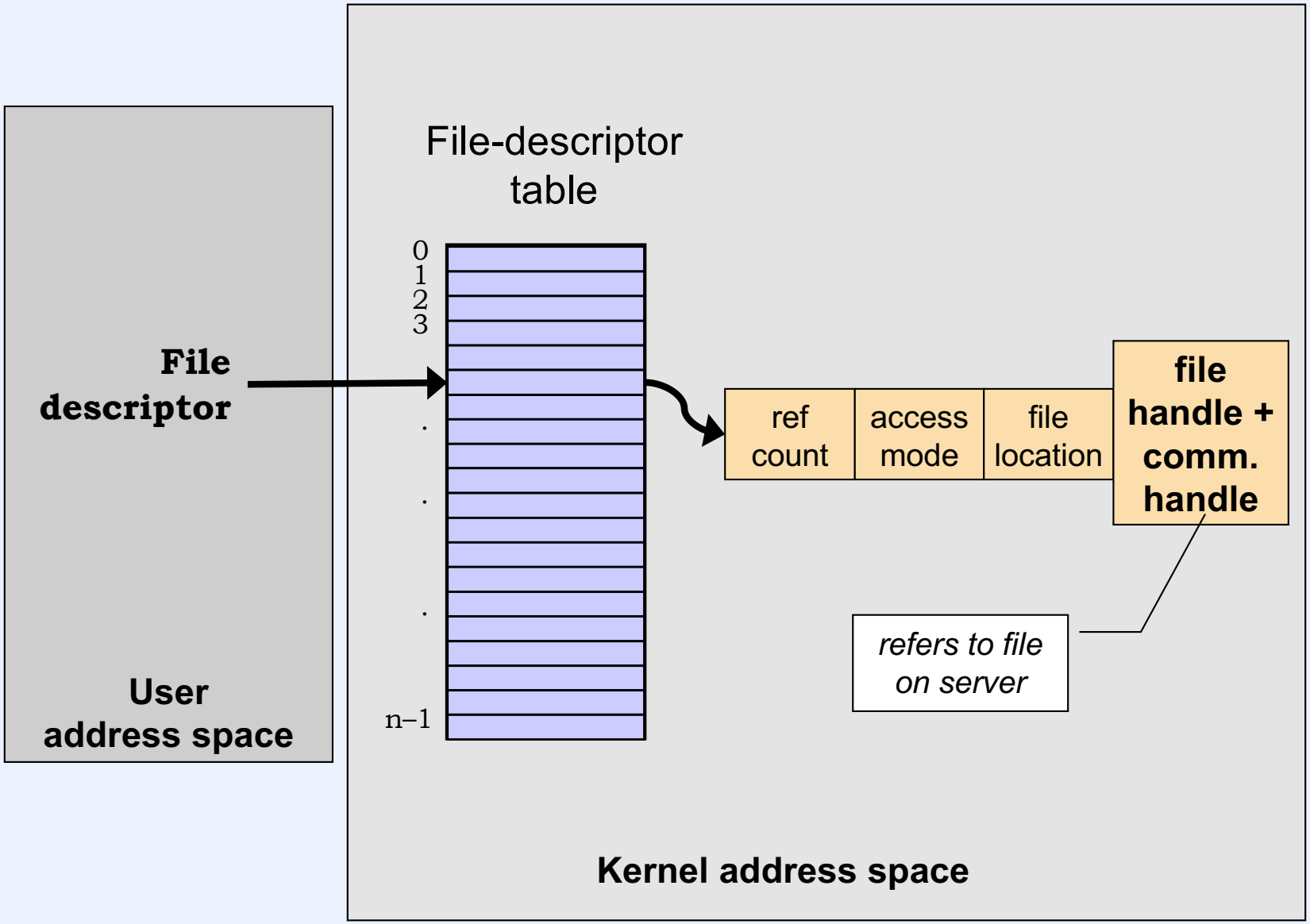


NFSv2 server

NFS in Action

```
char buffer[100];
int fd = open("/home/twd/dir/fileX", O_RDWR);
read(fd, buffer, 100);
…
lseek(fd, 0, SEEK_SET);
write(fd, buffer, 100);
```

Open-File Data Structures (Client)



However …

```
int fd = creat("/home/twd/dir/tempfile", 0600);
char buf[1024];
unlink("/home/twd/dir/tempfile");
…
write(fd, buf, 1024);
…
lseek(fd, 0, SEEK_SET);
read(fd, buf, 1024);
close(fd);
```

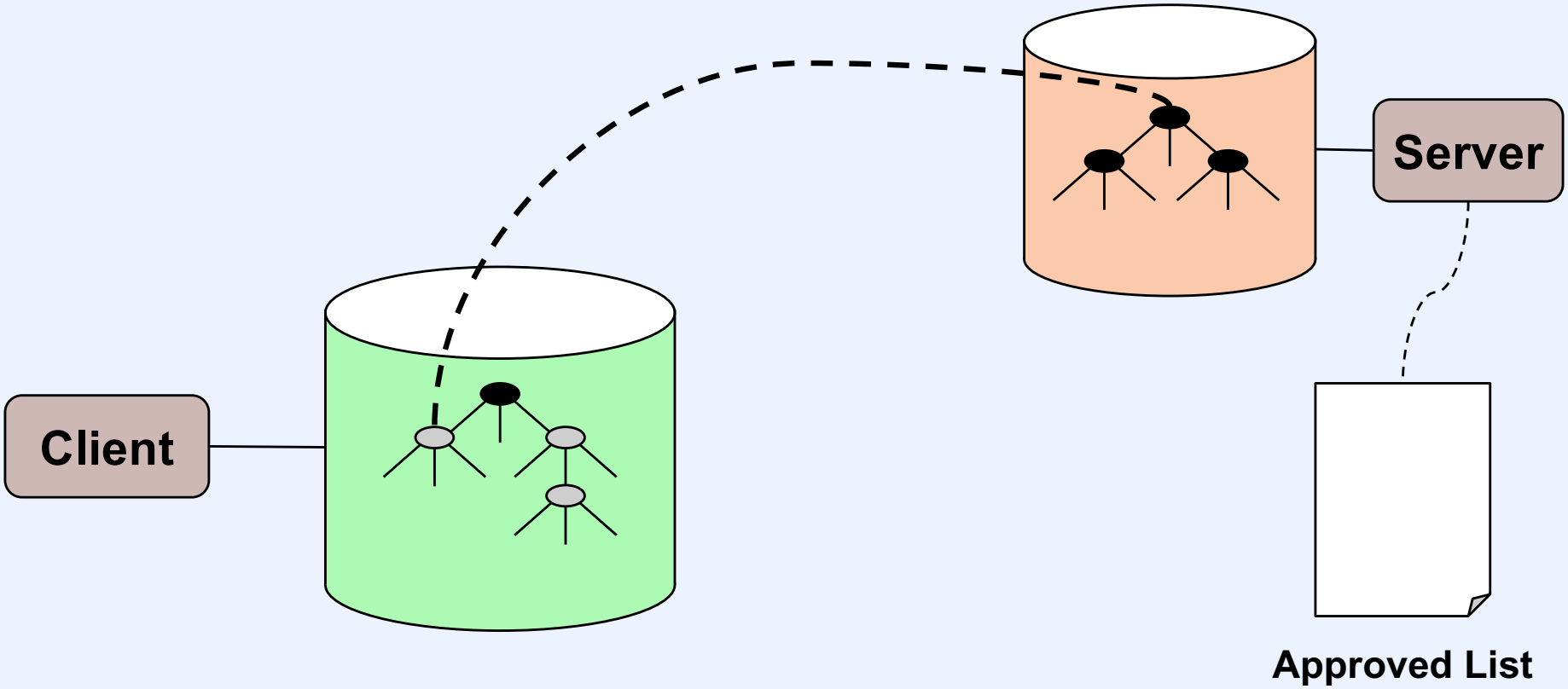
And …

```
int fd = creat("/home/twd/dir/permfile", 0600);
char buf[1024];
chmod("/home/twd/dir/permfile", 0400)
…
write(fd, buf, 1024);
…
```

RPC Semantics

- All requests done with ONC RPC
- Most are idempotent
- A few aren't
  - e.g. unlink
- Made *reasonably* reliable with DRC
  - susceptible to Byzantine routers and poorly timed crashes
    - crashes affect ability to handle retransmitted requests correctly

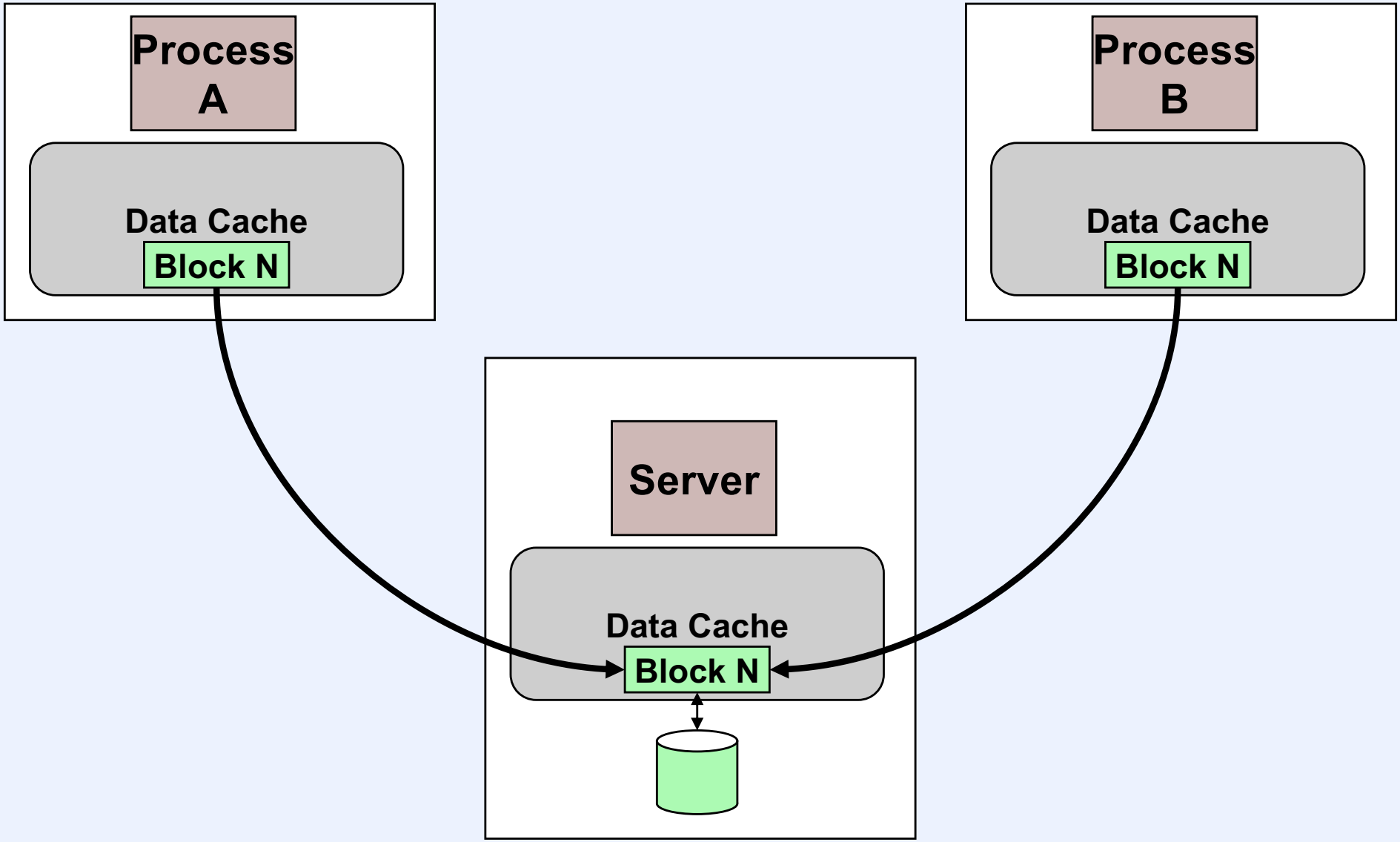
NFS Mount Protocol



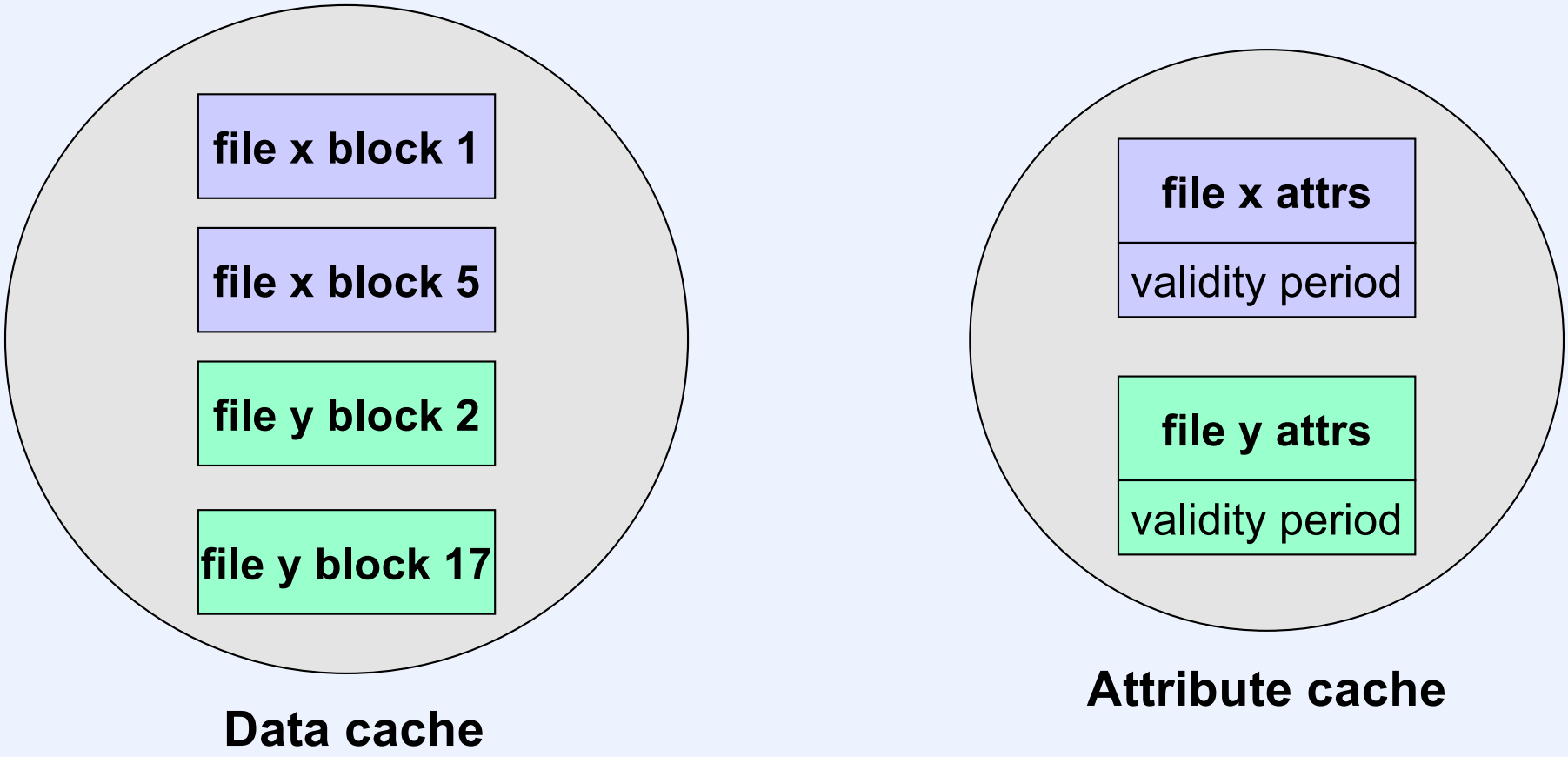
NFSv2 Assumptions about Sharing

1) Most writable files are private
2) Most shared files are read-only

NFS Consistency



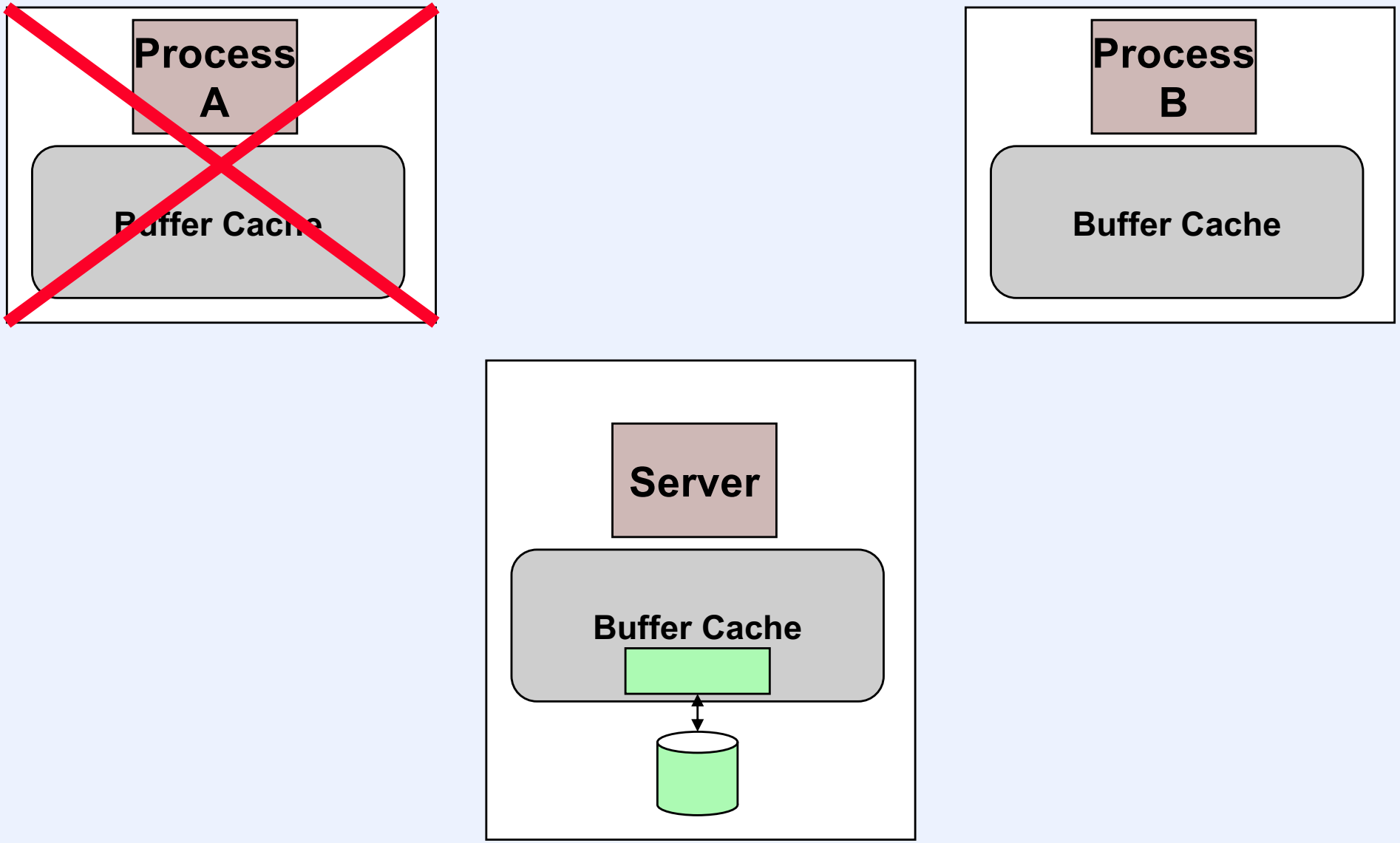
The Attribute Cache



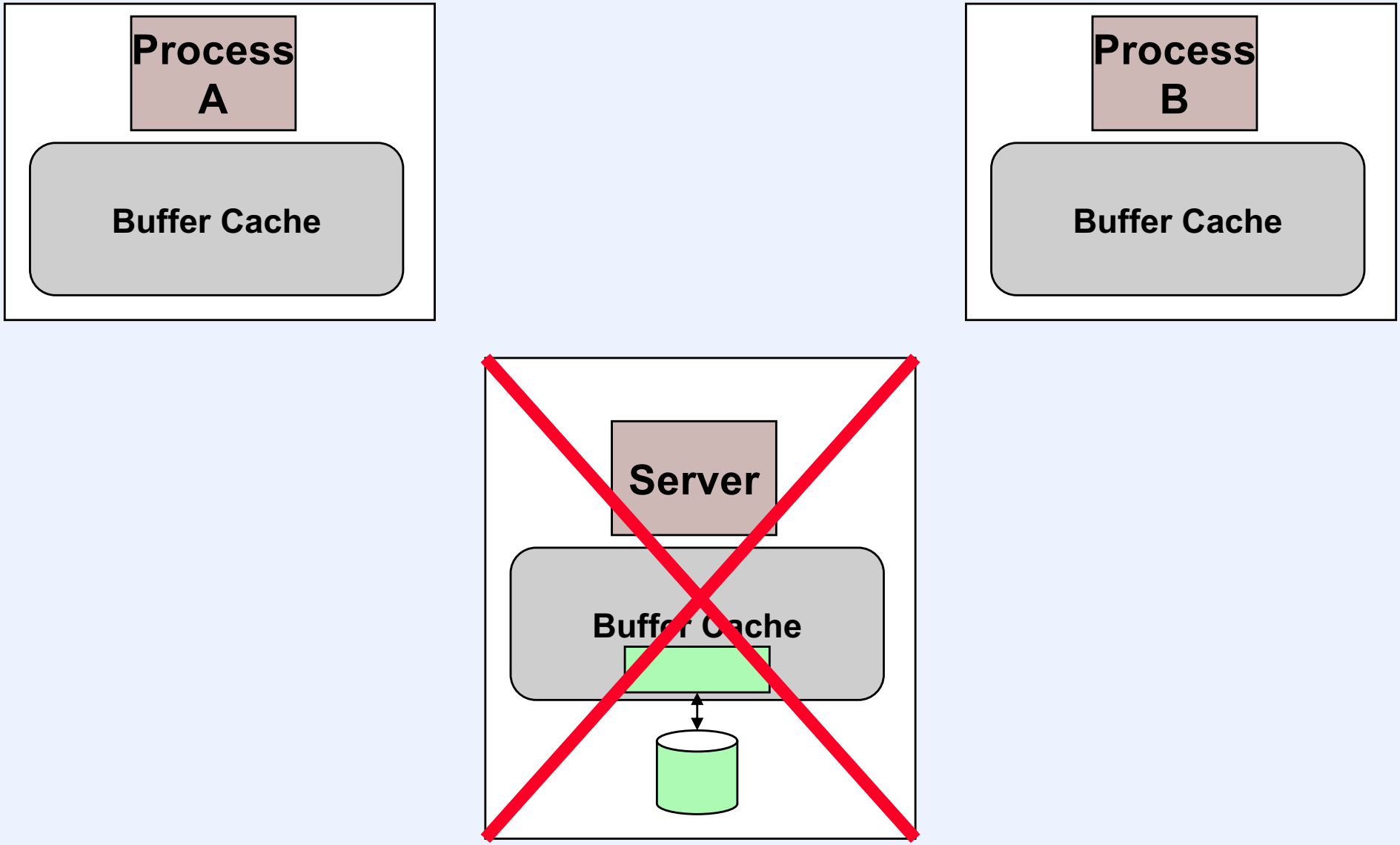
More …

- All write RPC requests must be handled synchronously on the server
- Close-to-Open consistency
  - client writes back all changes on close
  - flushes cached file info on open

Client Crash Recovery



Server Crash Recovery



Quiz 3

A client is modifying a file on the server, using a write RPC call (that specifies which file and where in the file). The file system is "hard-mounted". The server crashes, then, in a few minutes, comes back up.

a) It is not clear to the client application if its most recent write (before the crash) took place on the server

b) It is clear to the client application that its most recent write took place on the server

c) Its most recent write did not take place on the server

File Locking

- State is required on the server!
  - recovery must take place in the event of client and server crashes

Quiz 4

Can it be determined by a server that one of its clients has crashed and rebooted (assuming some cooperation from the client)?

a) no

b) yes with high probability

c) yes with certainty