Advanced Distributed Systems

Haibo Chen
Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University
http://ipads.se.sjtu.edu.cn/haibo_chen

Credits: some slides from Jinyang Li (NYU) and Robert Morris (MIT)

Outline

Course Information

Intro to distributed systems

Intro to Frangipani

Lab1: implementing a lock server

Faculty Information

Teacher

CHEN Haibo 陈海波 & CHEN Rong 陈榕

Office: Room 1-307, Software Build Building

Contact: 34202949

Course site: http://ipads.se.sjtu.edu.cn/courses/ads

TA

Chenning Xie

xie.chenny@gmail.com

Course Topics

Building distributed systems

- big ideas, useful techniques, implementation, case studies
- lectures on ideas, papers for case studies you'll build a real system (yfs, inspired by Frangipani)

Pre-requisite

- Advanced Operating Systems or equivalent
  - See http://ipads.se.sjtu.edu.cn/courses/aos if you haven't done a course like this
- We need to build a real distributed system

Difference with Distributed Computing (Courses)

Distributed Computing Courses (By Prof. Jinyu Zhou)

- Focus more on principles and algorithms

This course

- More practically-oriented
- Studies state-of-the-art (may not in textbook) systems
  - Big data and cloud computing systems
- Accumulate knowledge to build a real yet large one

Why take this course?

Synthesize many different areas in order to build working systems

Internet has made area much more attractive and timely

Hard/unsolved: not many deployed sophisticated distrib systems

Course Goals

Introduce you to (distributed) system design concepts and principles

Cover important systems design principles in general

Cover the design and evolutions of many important (distributed) computer systems

- Share both the lessons and experiences

Tentative Grading

In-classroom exam

Questions from papers and presentation from the class

Grade =

30% Exam +

40% Labs +

10% Lab presentation +

10% Hand-in + attendance +

10% Course participation

Lectures

Basic concepts + paper discussion + idea exchange

Two papers per week

Must hand in answers to paper question before class

Outline

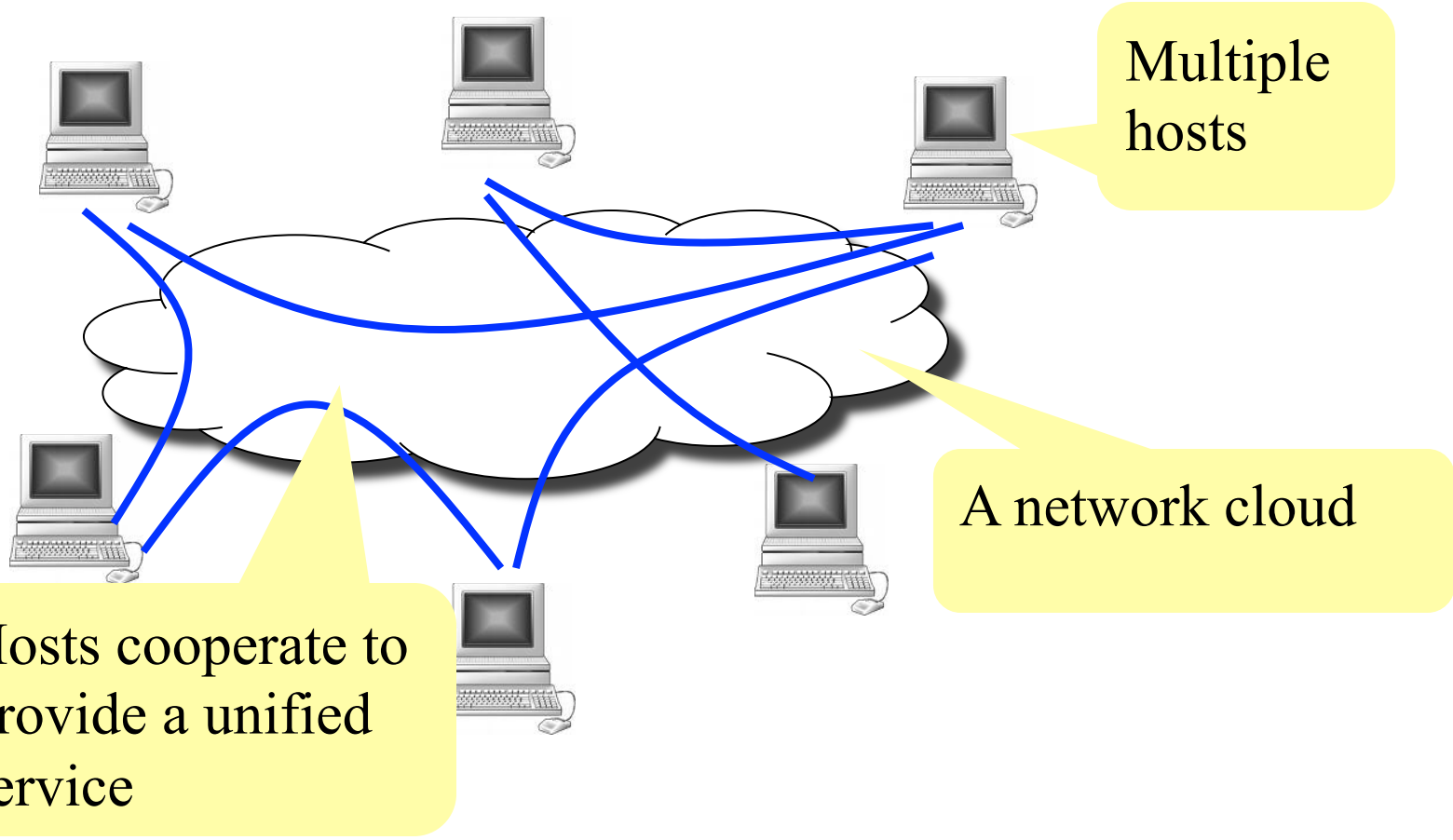
Course Information

Intro to distributed computer systems

Frangipani (yfs): a distributed file system

Introduction to Lab1

Review: what are distributed systems?



- Examples?

Why distributed systems? for ease-of-use

Handle geographic separation

Provide users (or applications) with location transparency:

- Web: access information with a few "clicks"
- Network file system: access files on remote servers as if they are on a local disk, share files among multiple computers

Why distributed systems? for availability

Build a reliable system out of unreliable parts

- Hardware can fail: power outage, disk failures, memory corruption, network switch failures…
- Software can fail: bugs, mis-configuration, upgrade …
- To achieve 99.999% availability, replicate data/ computation on many hosts with automatic failover

Why distributed systems? for scalable capacity

Aggregate resources of many computers

- CPU: Dryad, MapReduce, Grid computing
- Bandwidth: Akamai CDN, BitTorrent
- Disk: Frangipani, Google file system

Why distributed systems? for modular functionality

Only need to build a service to accomplish a single task well.

- Authentication server
- Backup server

Challenges

System design

- What is the right interface or abstraction?
- How to partition functions for scalability?

Consistency

- How to share data consistently among multiple readers/ writers?

Fault Tolerance

- How to keep system available despite node or network failures?

Challenges (continued)

- Different deployment scenarios
  - Clusters
  - Wide area distribution
  - Sensor networks
- Security
  - How to authenticate clients or servers?
  - How to defend against or audit misbehaving servers?
- Implementation
  - How to maximize concurrency?
  - What's the bottleneck?
  - How to reduce load on the bottleneck resource?

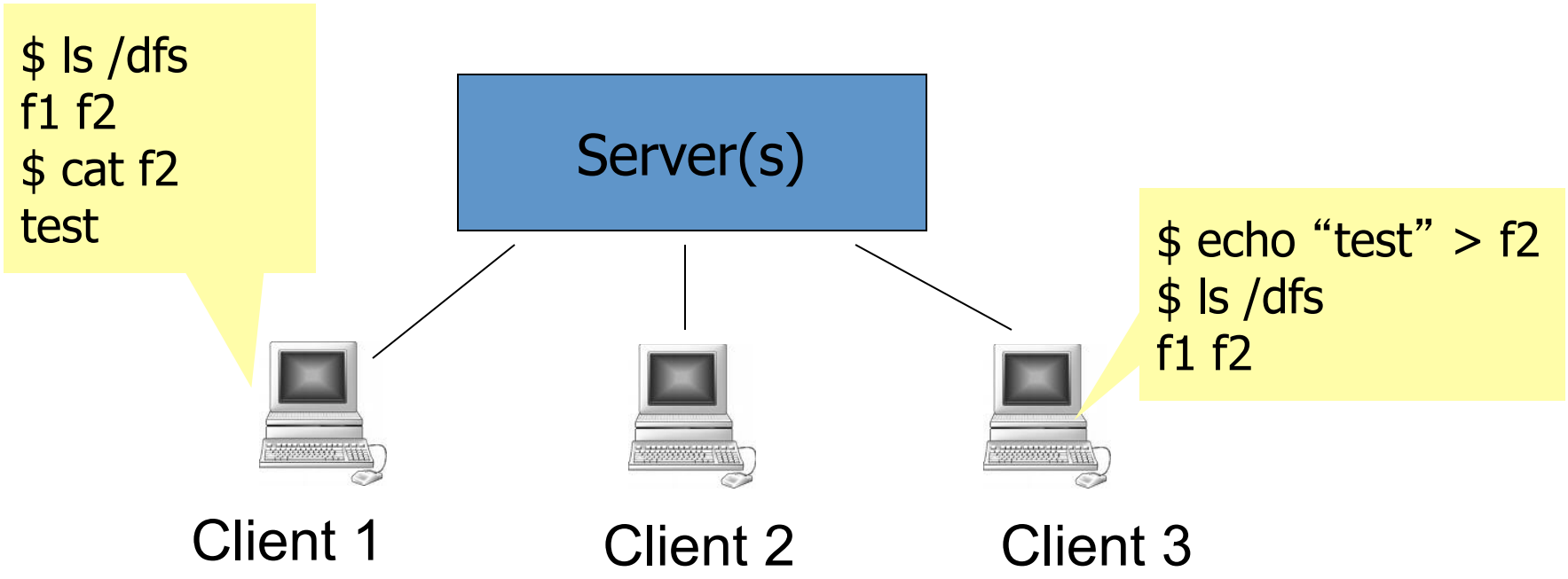
A word of warning

A distributed system is a system in which I can't do my work because some computer that I've never even heard of has failed."

-- Leslie Lamport

Topics in this course

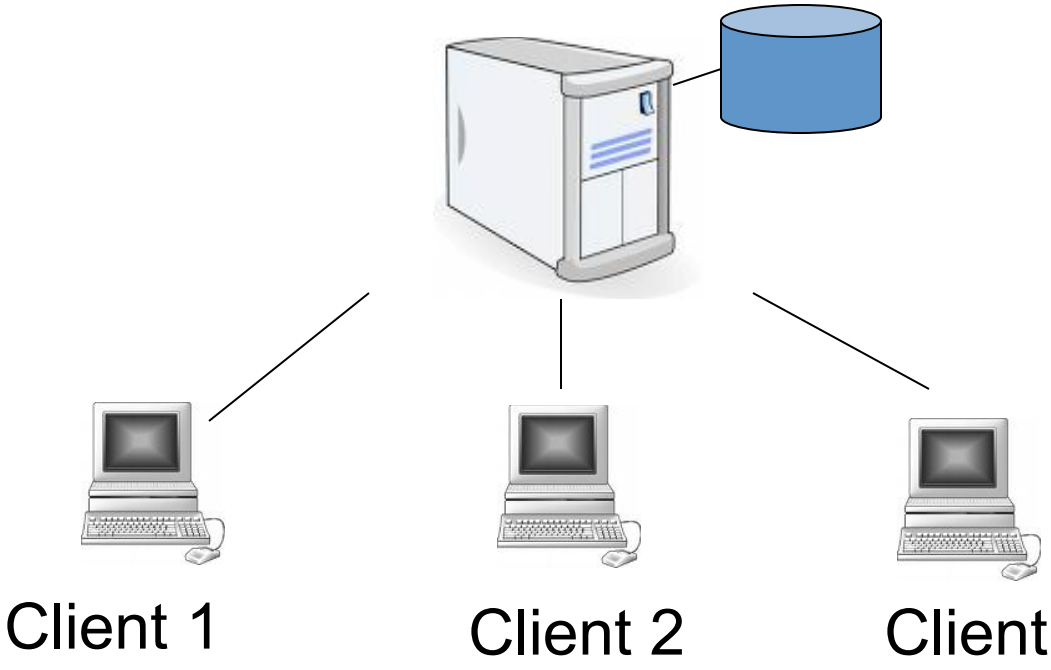
Case Study: Distributed file system



A distributed file system provides:

- location transparent file accesses
- sharing among multiple clients

A Simple Distributed FS design



A single server stores all data and handles clients' FS requests.

Topic: System Design

What is the right interface?

- possible interfaces of a storage system
  - Disk
  - File system
  - Database

Can the system handle a large user population?

- E.g. all SJTU students and faculties

How to store peta-bytes of data?

- Has to use more than 1 server
- Idea: partition users across different servers

Topic: Consistency

When C1 moves file f1 from /d1 to /d2, do other clients see intermediate results?

What if both C1 and C2 want to move f1 to different places?

To reduce network load, cache data at C1

- If C1 updates f1 to f1', how to ensure C2 reads f1' instead of f1?

Topic: Fault Tolerance

How to keep the system running when some file server is down?

- Replicate data at multiple servers

How to update replicated data?

How to fail-over among replicas?

How to maintain consistency across reboots?

Topic: Security

Adversary can manipulate messages

- How to authenticate?

Adversary may compromise machines

- Can the FS remain correct despite a few compromised nodes?
- How to audit for past compromises?

Which parts of the system to trust?

- System admins? Physical hardware? OS? Your software?

Topic: Implementation

The file server should serve multiple clients concurrently

- Keep (multiple) CPU(s) and network busy while waiting for disk

Concurrency challenge in software:

- Server threads need to modify shared state:
  - Avoid race conditions
- One thread's request may dependent on another
  - Avoid deadlock and livelock

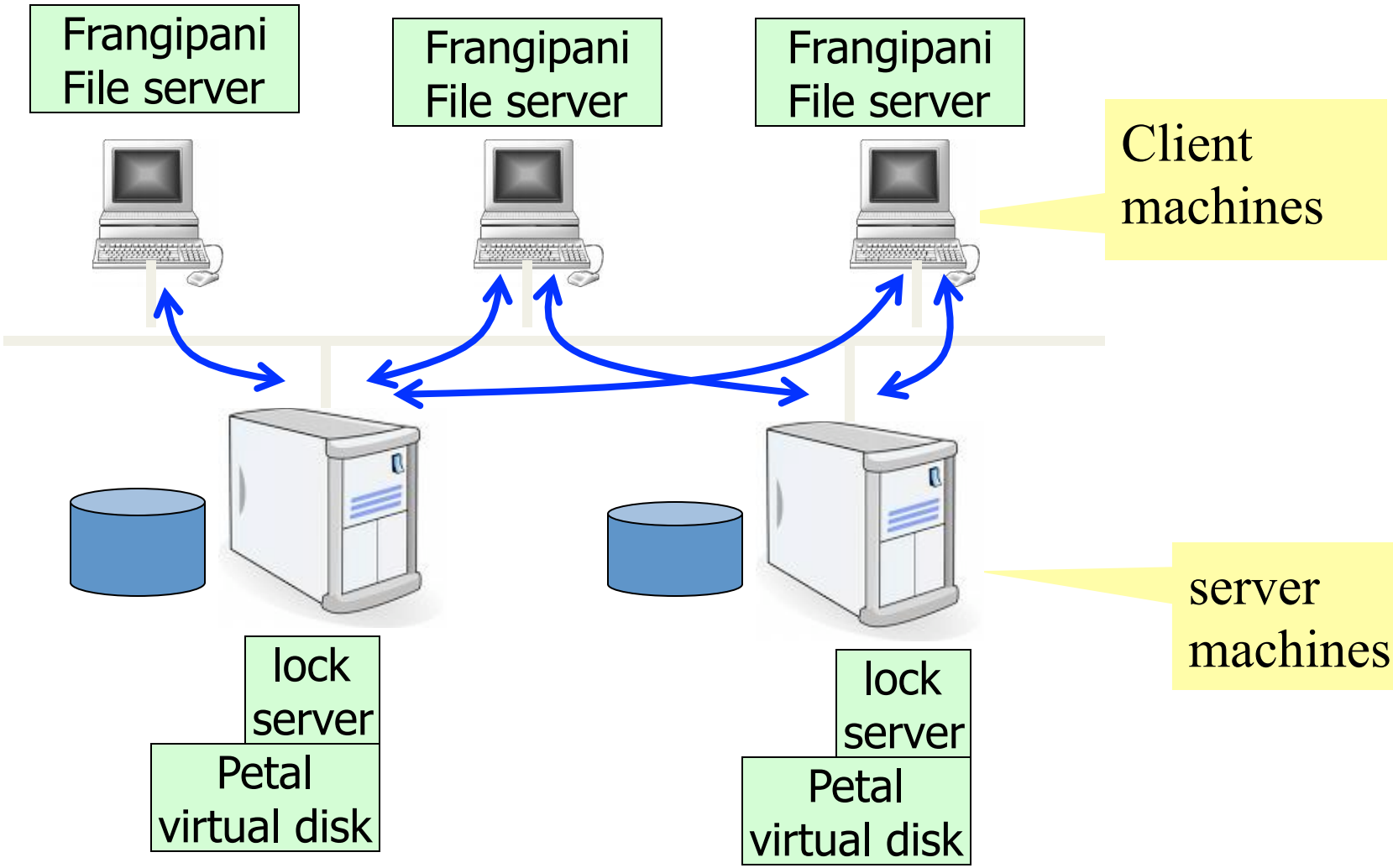
Intro to programming Lab: Yet Another File System (yfs)

YFS is inspired by Frangipani

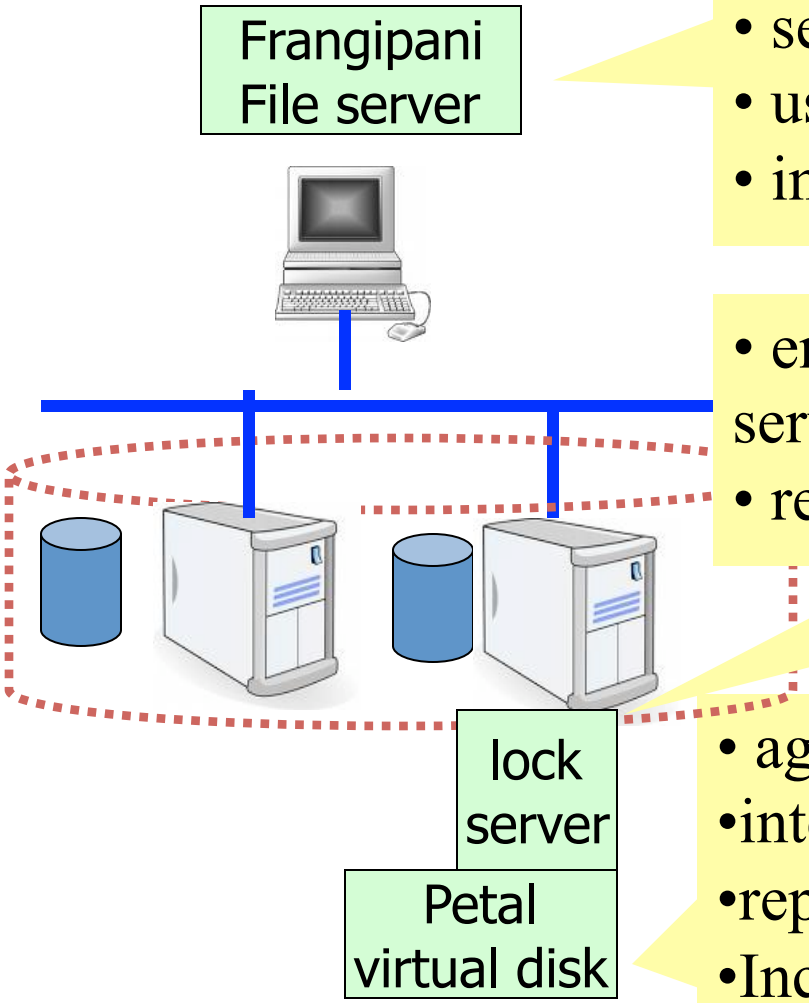
Frangipani goals:

- Aggregate many disks from many servers
- Incrementally scalable
- Automatic load balancing
- Tolerates and recovers from node, network, disk failures

Frangipani Design



Frangipani Design



- serve file system requests
- use Petal to store data
- incrementally scalable with more servers

- ensure consistent updates by multiple servers
- replicated for fault tolerance

- aggregate disks into one big virtual disk
- interface: put(addr, data), get(addr)
- replicated for fault tolerance
- Incrementally scalable with more servers

Frangipani security

Simple security model:

- Runs as a cluster file system
- All machines and software are trusted!

Frangipani server implements FS logic

- Application program:

  creat(“/d1/f1”, 0777)

- Frangipani server:

  1. GET root directory’s data from Petal
  2. Find inode # or Petal address for dir “/d1”
  3. GET “/d1”s data from Petal
  4. Find inode # or Petal address of “f1” in “/d1”
  5. If not exists

     alloc a new block for “f1” from Petal

     add “f1” to “/d1”’s data, PUT modified “/d1” to Petal

Concurrent accesses cause inconsistency

time

App: creat("/d1/f1", 0777)
Server S1:

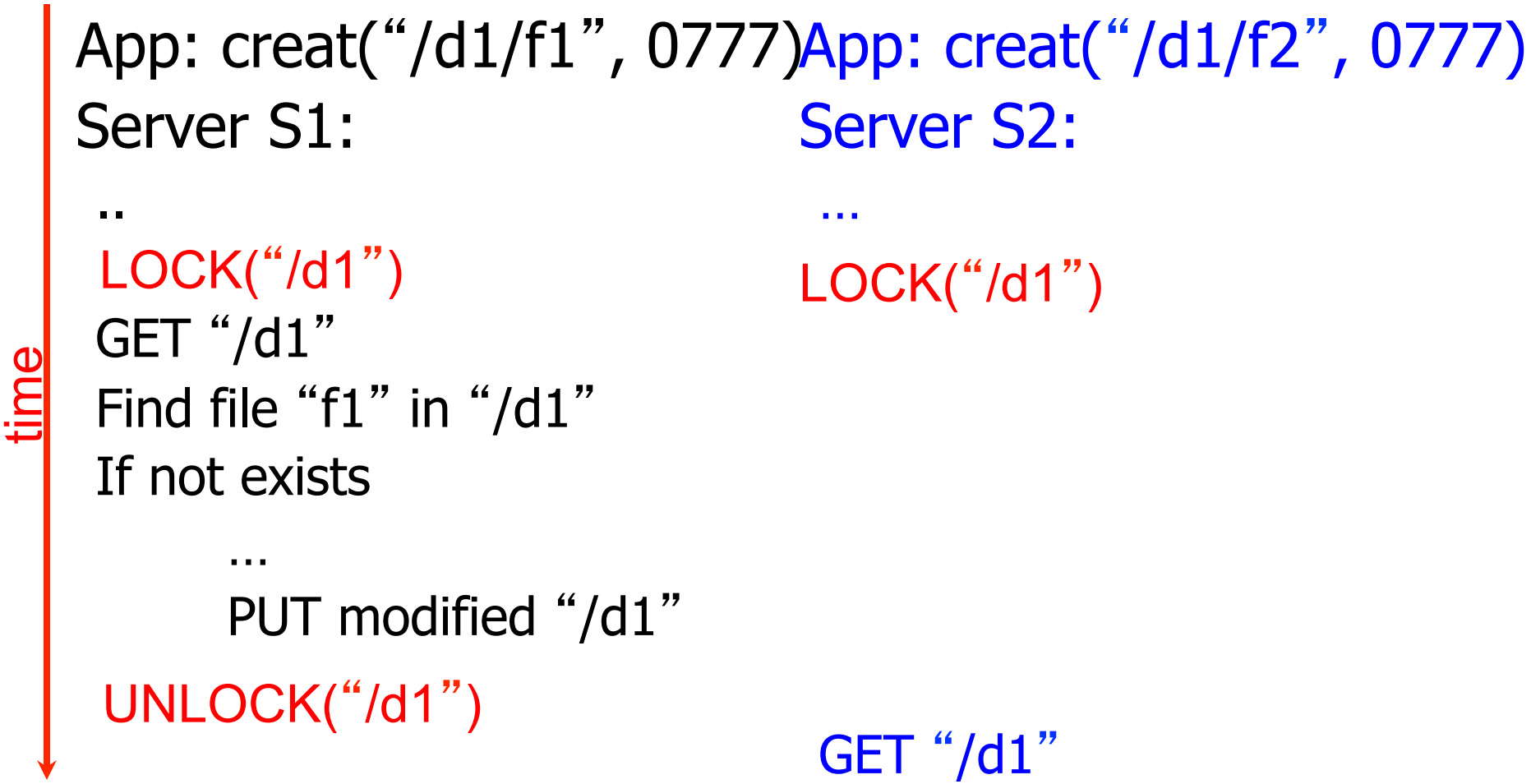
…
GET "/d1"
Find file "f1" in "/d1"
If not exists
…
PUT modified "/d1"

App: creat("/d1/f2", 0777)
Server S2:

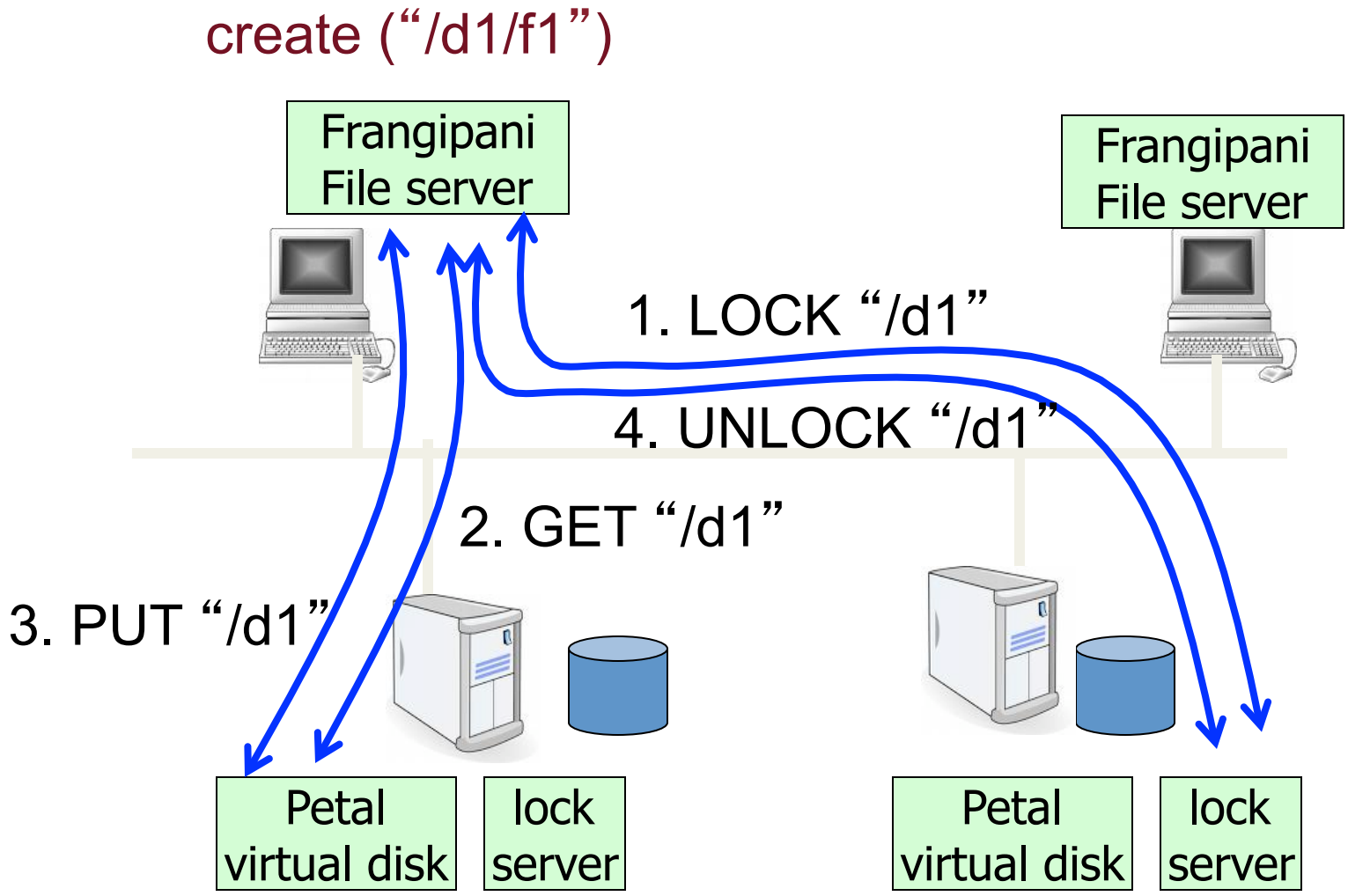
…
GET "/d1"
Find file "f2" in "/d1"
If not exists
…
PUT modified "/d1"

What is the final result of "/d1"? What should it be?

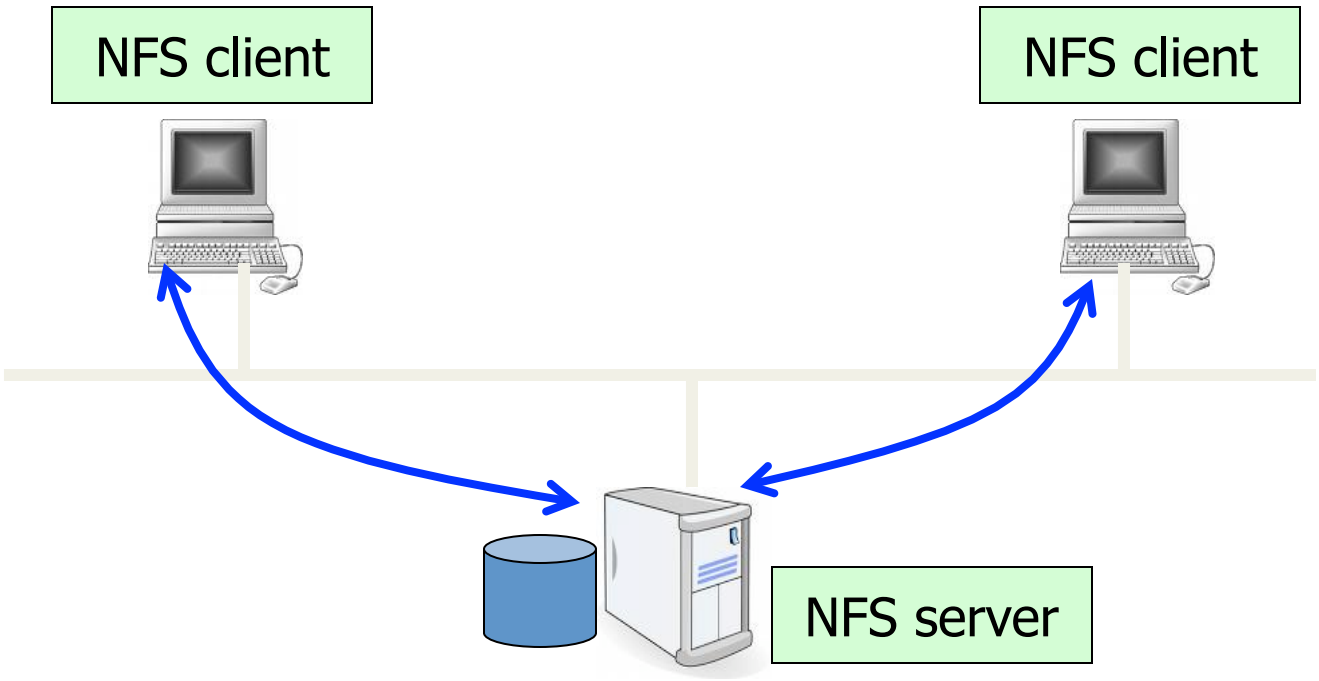
Solution: use a lock service to synchronize access



Putting it together

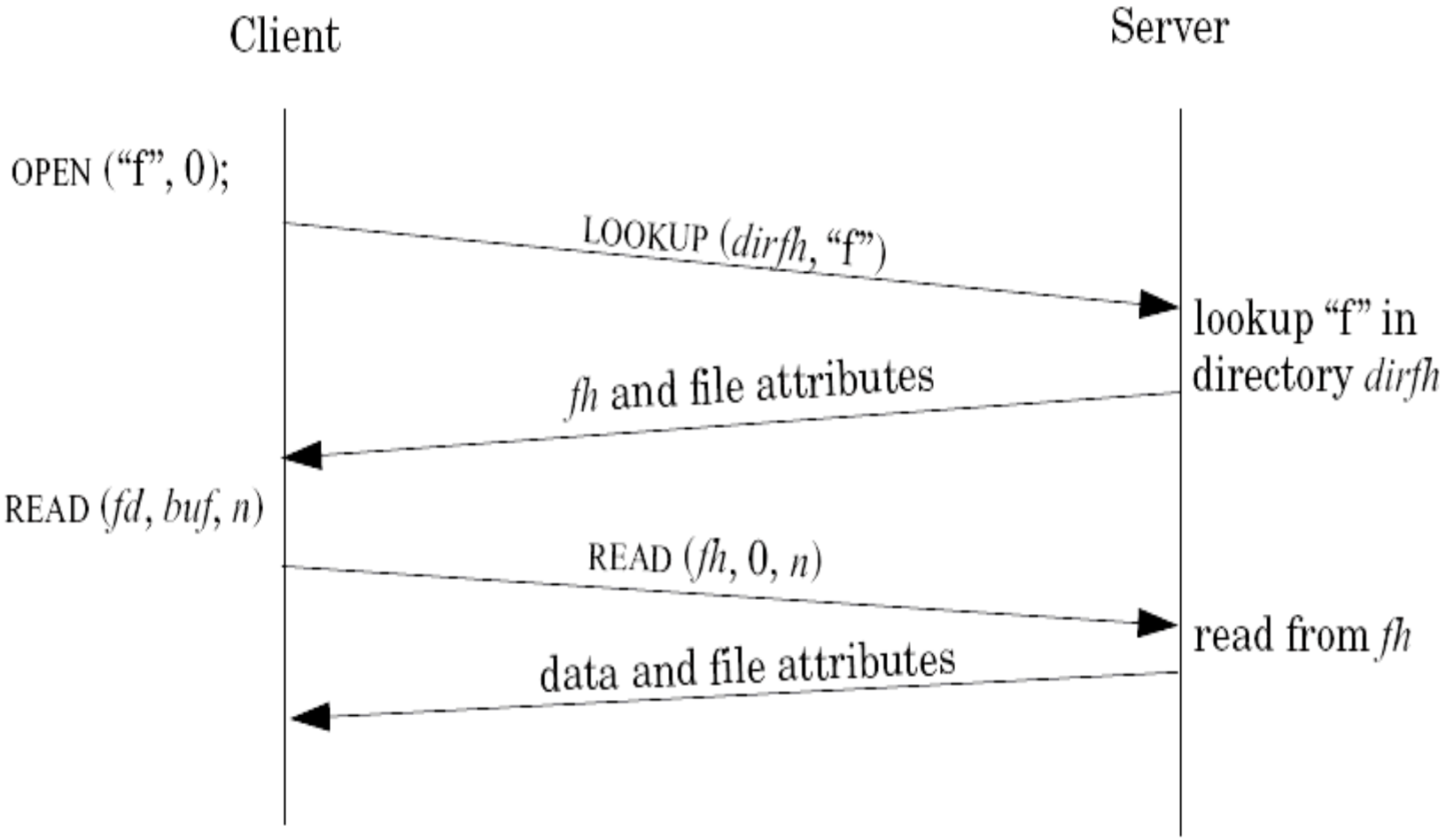


NFS (or AFS) architecture

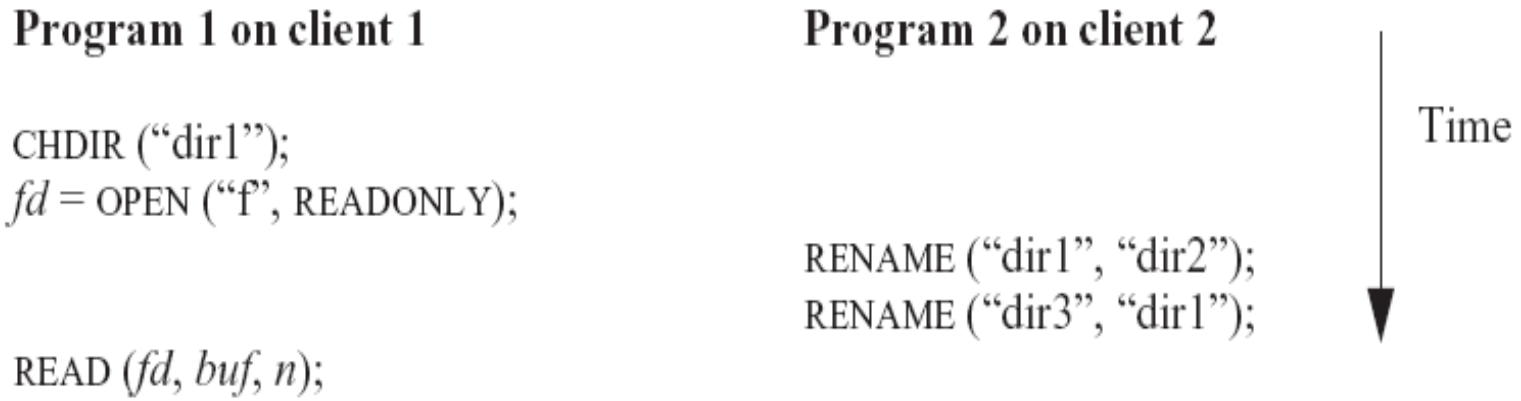


- Simple clients
  - Relay FS calls to the server
  - LOOKUP, CREATE, REMOVE, READ, WRITE …
- NFS server implements FS functions

NFS messages for reading a file



Why use file handles in NSF msg, not file names?



What file does client 1 read?

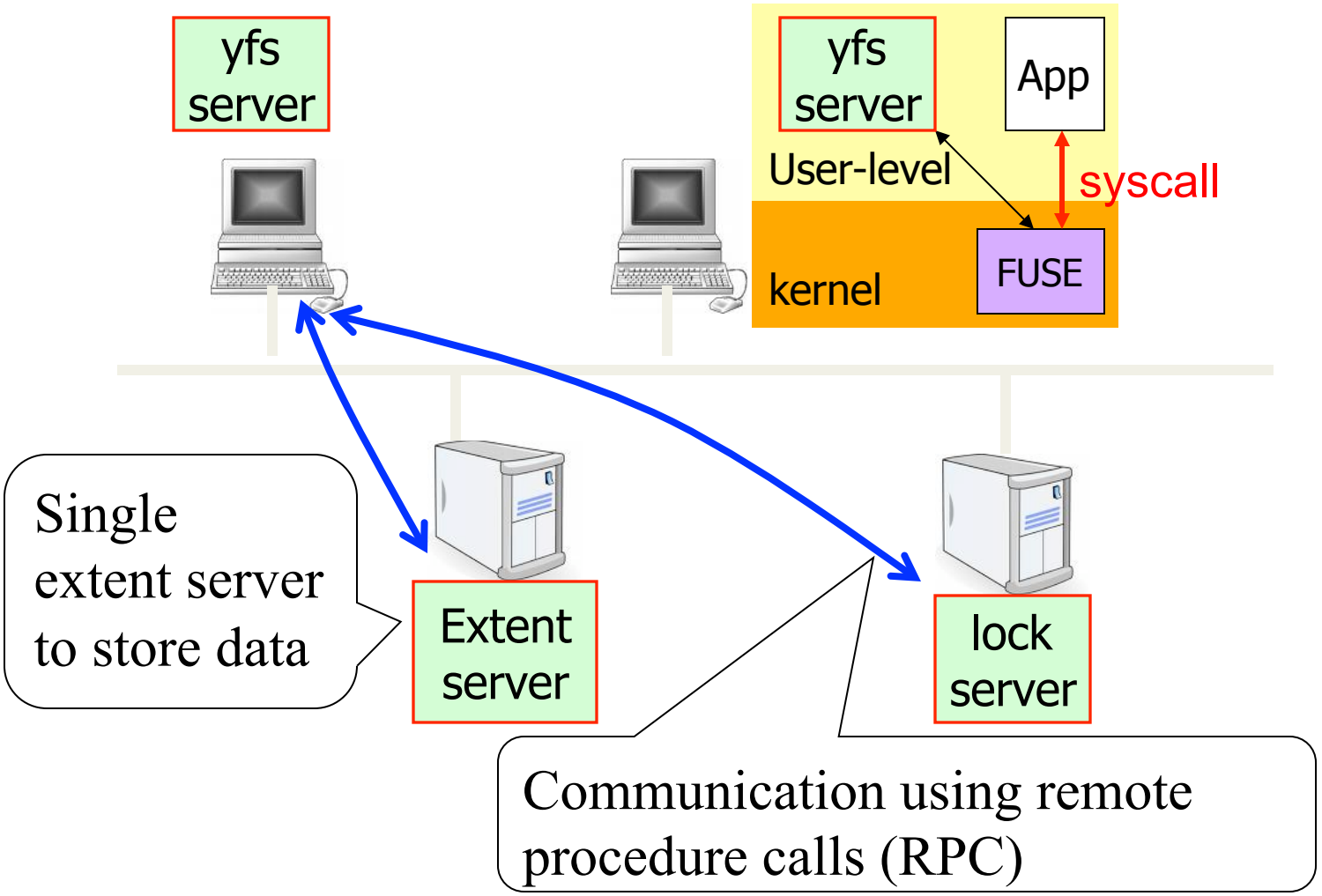
- Local UNIX fs: client 1 reads dir2/f
- ~~NFS using filenames: client 1 reads dir1/f~~
- NFS using file handles: client 1 reads dir2/f

File handles refer to actual file object, not names

Frangipani vs. NFS

	Frangipani	NFS
Scale storage	Add Petal nodes	Buy more disks
Scale serving capacity	Add Frangipani servers	Manually partition FS namespace among multiple servers
Fault tolerance	Data is replicated On multiple Petal Nodes	Use RAID

YFS: simplified Frangipani



Lab schedule

L1: lock server
- Programming w/ threads
- RPC semantics

L2: basic file server
L3: Sharing of files
L4: Locking
L5: Caching locks
L6: Caching extend server+consistency
L7: Paxos
L8: Replicated lock server
L9: Project: extend yfs

L1: lock server

Lock service consists of:

- Lock server: grant a lock to clients, one at a time
- Lock client: talk to server to acquire/release locks

Correctness:

- At most one lock is granted to any client

Additional Requirement:

- acquire() at client does not return until lock is granted

Next Class

Implementing RPC

Read the RPC paper and hand in paper questions before class

The End

THANKS