Building a Scalable NFS

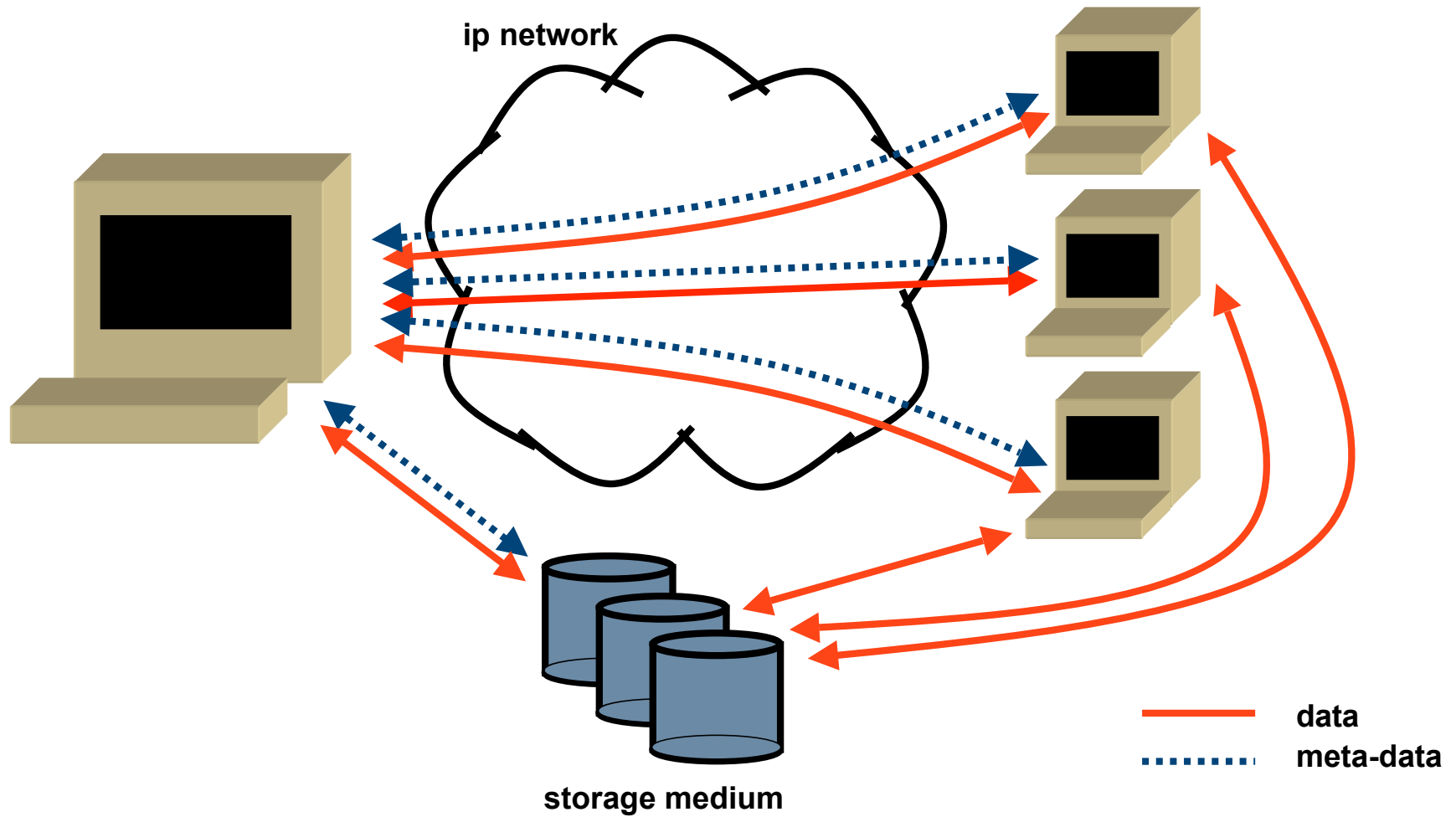**Breaking the Single Server Bottleneck**

Stephen Fridella, Xiaoye Jiang, David Black

Goals

Extend NFS in a way which:

- Breaks the single server bottleneck

- Supports multiple underlying storage models

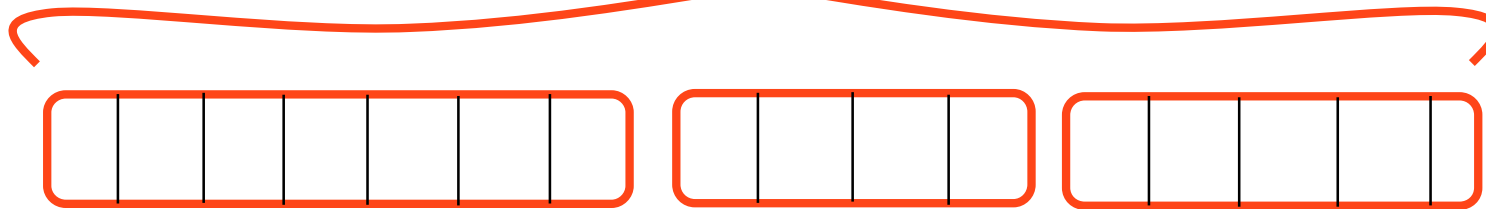- Achieves scalable sharing for file systems *and* individual files

Single Server Bottleneck

What Storage Medium?
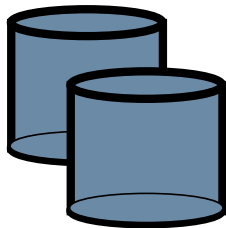
All of them…

File System

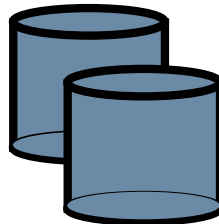"logical volumes" ---- block containers

abstraction

realization

SCSI/ FC device

OSD

NFS/ Samba
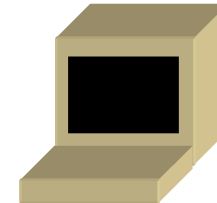
NFS Client Architecture

Local File System

NFS

TCP/IP

Medium-specific driver

← to server

network

to storage →

storage fabric
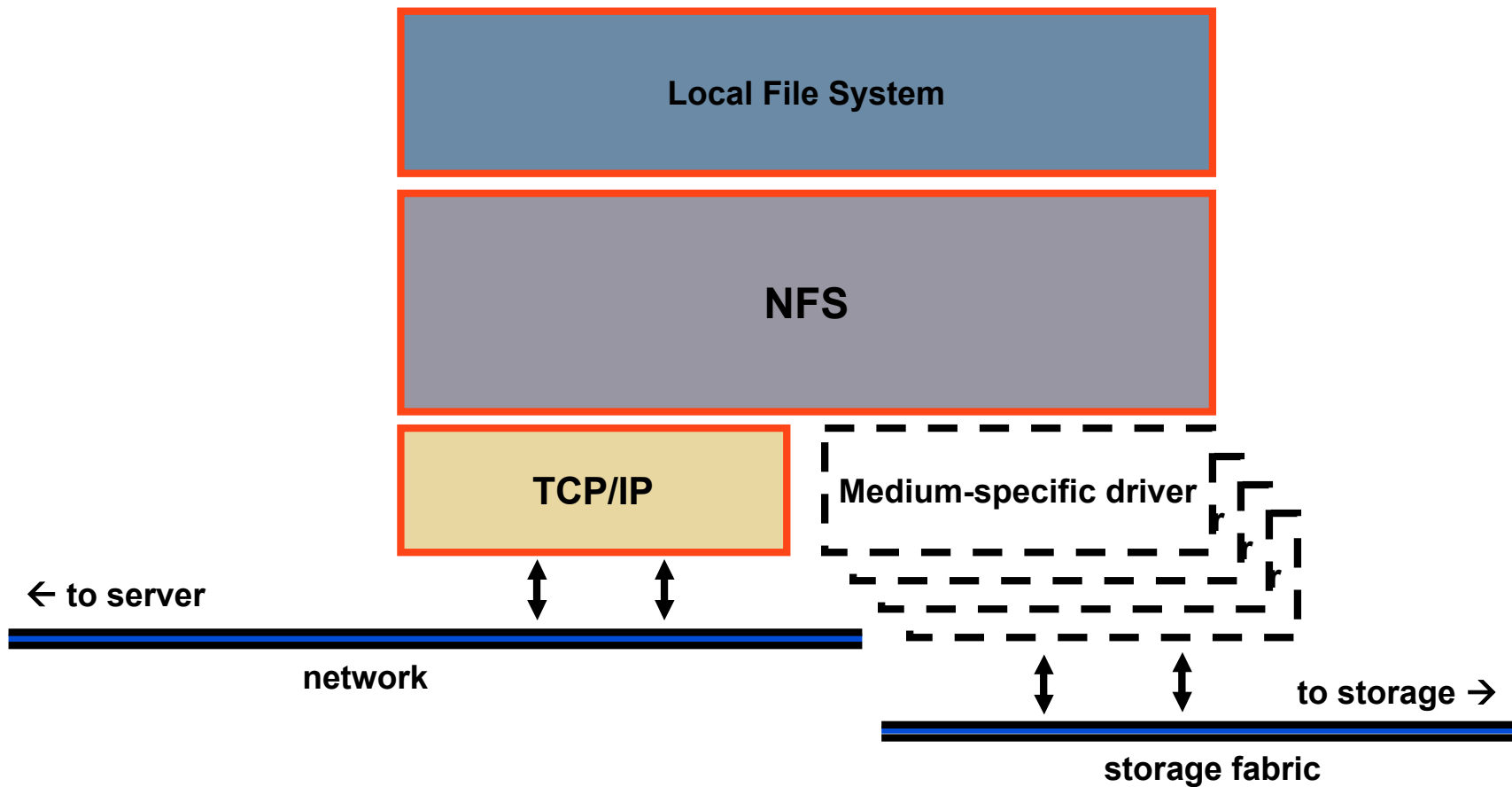
Volume Discovery

Need volume discovery phase

- Client and server agree on what storage medium is used

- Server communicates to client a mapping of logical volume ids to appropriate storage space

- Future communications are in terms of logical volume addresses

Meta-Data Management Protocol

Client needs protocol elements which allow it to…

■ fetch block mappings for files

■ allocate new blocks to files

■ coordinate access to files with other clients

Meta-Data Management (cont)
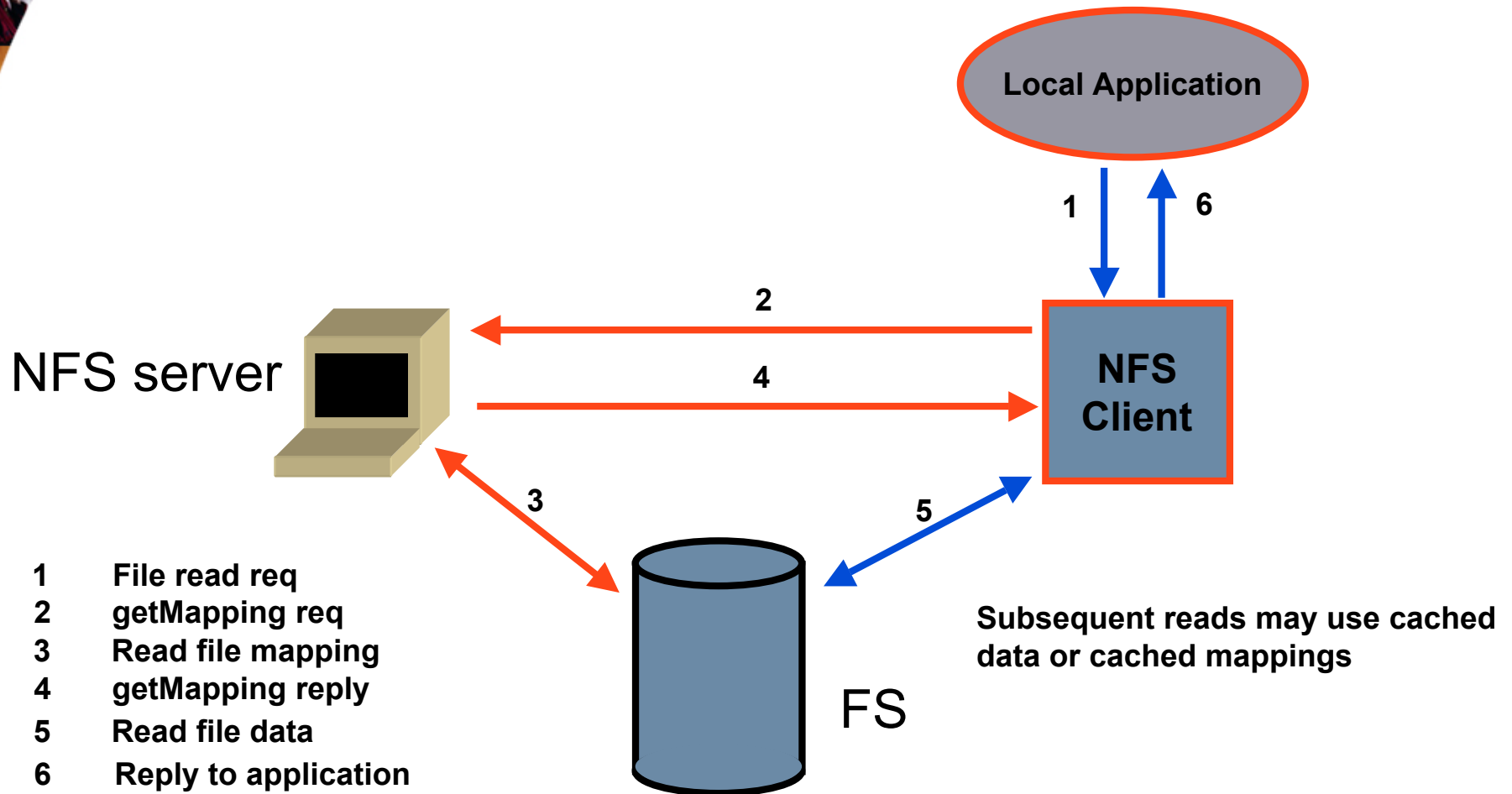
Our proposal:

- Block-range read and write delegations are implicitly granted on successful mapping/allocation requests

- Allocations are initially provisional---they require an explicit commit request after data is written

- Ideas are based on EMC's File Mapping Protocol

Reading Data (example)

EMC²
where information lives

Local Application

1   6

2

NFS server

4

NFS
Client

3   5

1   File read req
2   getMapping req
3   Read file mapping
4   getMapping reply
5   Read file data
6   Reply to application

FS

Subsequent reads may use cached
data or cached mappings

Writing Data (example)

Local Application

1   5

NFS server

2

4

7

NFS
Client

3

8

6

FS

1   File write req
2   allocSpace req
3   Blocks pre-allocated
4   allocSpace reply
5   File write reply
6   Write cached data to disk
7   commitBlock req
8   Blocks added to file mapping
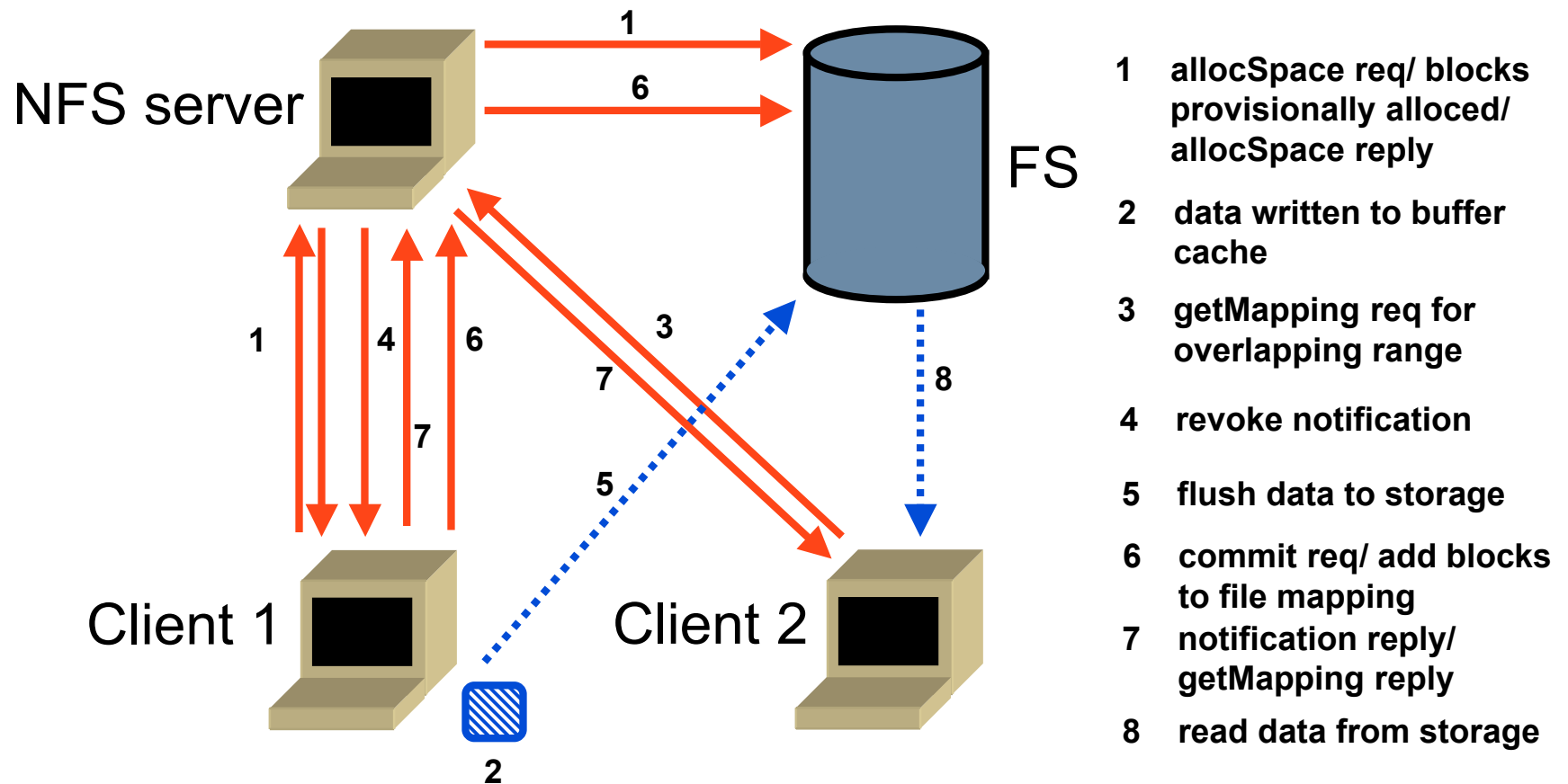
**Subsequent writes can use
cached pre-allocated blocks**

Block Range Delegations

- Shared and exclusive

- Lease-based

- Can be revoked when server detects conflicting delegation request

- Can be revoked when server detects operation that will change mapping (write, truncate)

Conflicting Requests (Example)

1. allocSpace req/ blocks provisionally alloced/ allocSpace reply
2. data written to buffer cache
3. getMapping req for overlapping range
4. revoke notification
5. flush data to storage
6. commit req/ add blocks to file mapping
7. notification reply/ getMapping reply
8. read data from storage

Summary

- Break the single server bottleneck by providing a meta-data management interface for clients

- Support multiple underlying storage models by using a generic logical volume abstraction

- Provide for read/write sharing of files as well as file-systems by using block-range delegations