CONCEPTS OF DISTRIBUTED FILE SYSTEMS

Network Distributed Computing
Dr. Christina Thorpe

https://www.youtube.com/watch?v=A1W2nqVjaOA

–Network File System

NAMING SCHEMES

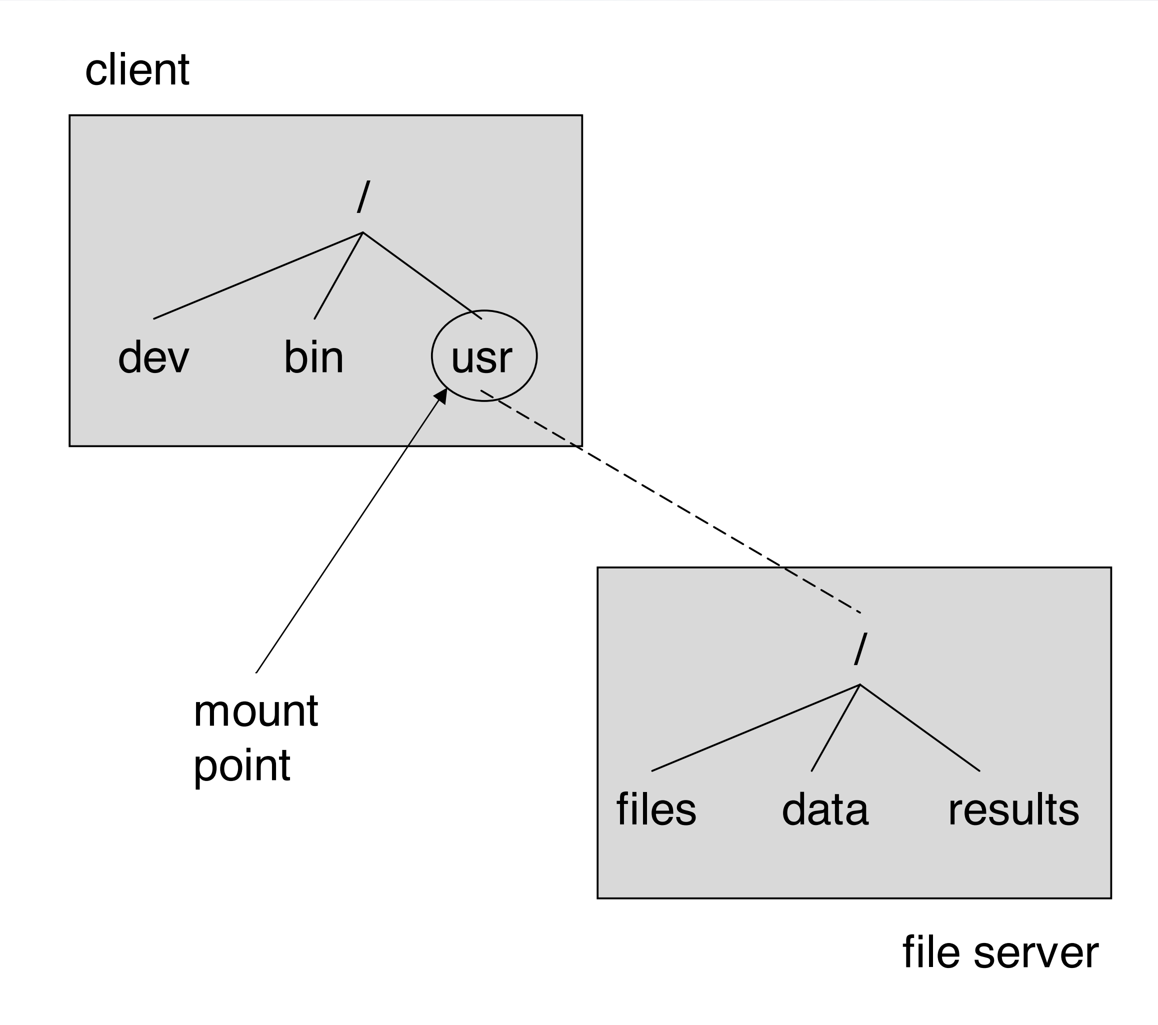
- Location evident: host-name::local-name
- Mounting: assigning the root of a remote file system to an already accessible directory (e.g., NFS)
- Single image: all users see the same integrated name structure for all files (e.g., Sprite)

GOALS AND CRITERIA

- Goal: present to a user a coherent, efficient, and manageable system for long-term data storage in a distributed environment.
- Criteria:
  - Transparency: the degree to which the user is aware of the existence of the underlying distribution of data (naming schemes)
  - Performance: the difference in time between access to local vs remote data (caching vs. remote operations)
  - Fault tolerance: the ability of the system to provide acceptable service in the presence of failures to clients, servers, and the network (stateful vs. stateless; replicas)
  - Scaleability: the ability of the system to exhibit sustained performance against increases in the number of users and the volume of data
  - Security: a guarantee that data access conforms to stated policies

MOUNTING

- Creates names such as /usr/data
- Location transparent
- Allows different users to see different name structures
- Potential administrative costs
- Client maintains "mount table"



TRANSPARENCY

- Network: the same interface is presented for access to local and non-local files
- Access: the user has the same view of the file system regardless of the physical point of access
- Naming:
  - Location transparency (the name conveys no information about the location of the data)
  - Location independence (the name of a file need not be changed if/when the location of the file is changed)

SEMANTICS

- "Unix" semantics:
    - reflects familiar semantics of a non-distributed file system
    - Allows existing applications to be run without change value read is the value stored by last write
    - writes to an open file are visible immediately to other that have this file opened concurrently
    - easy to implement if one server and no caching

SEMANTICS

- Session semantics
  - Acknowledges difficulty in reflecting changes immediately to other readers
  - Write to an open file are not immediately visible to remote readers (are visible to local readers)
  - Changes are visible to those readers who open the file after the file is closed by the writer (not visible to those reading concurrently with the writer)

DISK VS MEMORY CACHE

- Disk caches
  - More reliable (survive failures)
  - Avoids reloading on recovery
- Memory caches
  - Allow diskless workstations
  - Faster access on client machine
  - Since servers use memory caching, allows a single uniform mechanism

SEMANTICS

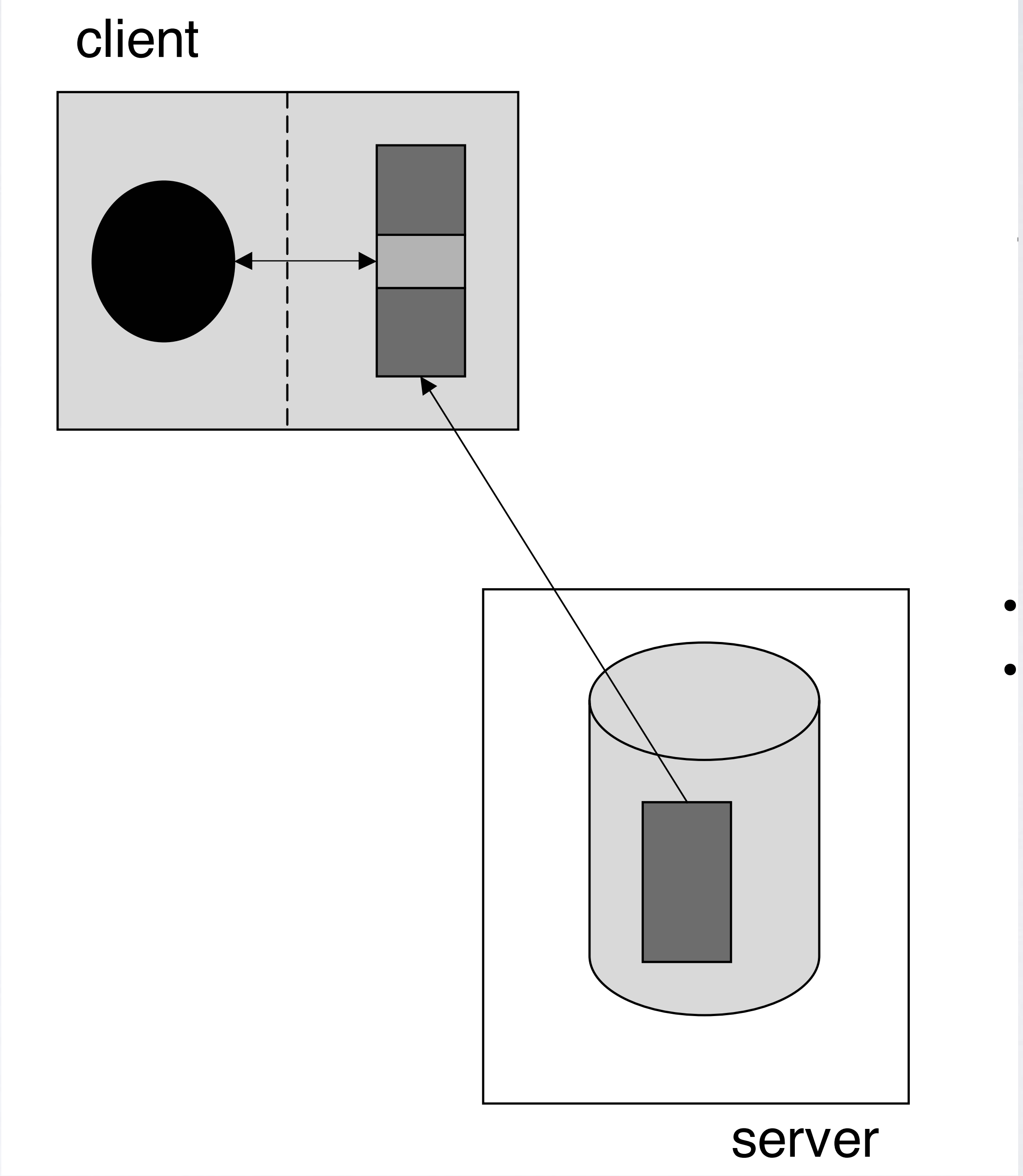
- Immutable shared files
  - A shared file cannot be changed – File names cannot be reused
  - Simple to implement
- Transaction:
  - Operations conform to ACID properties
  - Requires greater system support

UPDATE POLICY

- Write-through
  - reliable: little loss of information in the event of a client failures
  - slow: defeats purpose of cache
- Delayed-write
  - Optimizes network traffic for successive writes to same/nearby blocks
  - Avoids overhead for data that will be overwritten (20- 30% of data is deleted within 30 seconds)
- Write-on-close
  - Works best for files open for a short period
  - Susceptible to loss of data for files in long use

CACHING

- Caching vs. remote service
- Units of caching: block or file
- Local cache: disk or memory
- Update policy:
  - Write through
  - Delayed write
  - Write-on-close
- Consistency
  - Client initiated validity check
  - Server-initiated callback
  - Leases



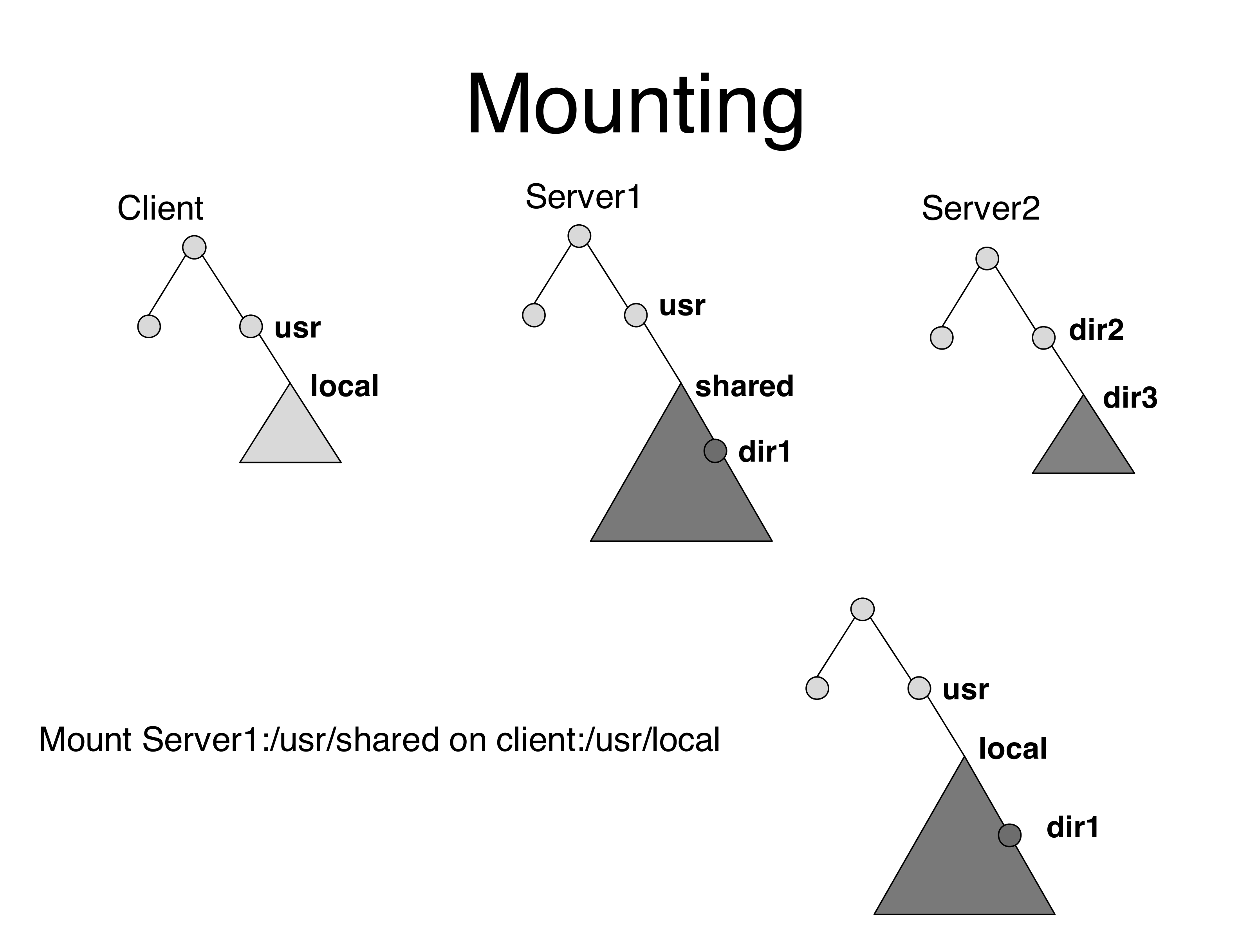
FAULT TOLERANCE: STATEFUL VS. STATELESS SERVERS

- Stateful
  - Server maintains information about a file opened by a client (e.g., file pointer, mode)
  - Mechanism: on open, the server provides a "handle" to the client to use on subsequent operations
- Stateless
  - Server maintains no information about client access to files
  - Mechanism: each client operation must provide context information for that operation

SUN NFS

- File system sharing among networked workstations in a client-server model
- Each workstation may be both a client and a server (no dedicated role)
- Services defined for implementation on heterogeneous architectures and file systems using machine-independent protocol
- • Key protocols:
    - Mount (define hierarchical structure)
    - NFS (read/write operations)
- Employs stateless operations(untilV4)

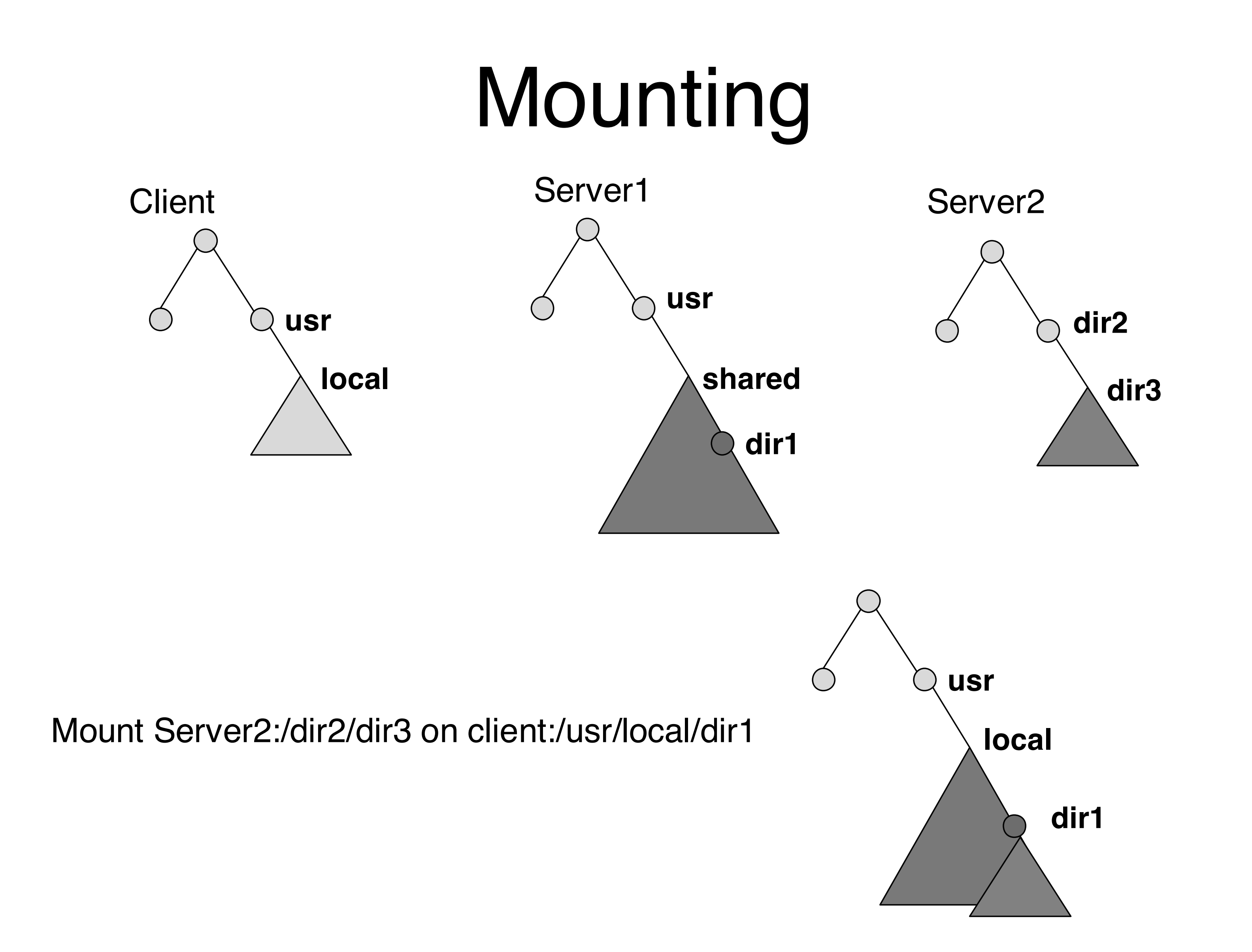
MOUNTING



Mounting

Mount Server1:/usr/shared on client:/usr/local

MOUNTING



Mounting

Mount Server2:/dir2/dir3 on client:/usr/local/dir1

MOUNT PROTOCOL

- Mount operation specifies remote file system and local directory mount point
  - Request translated to RPC and forwarded to server
  - Server maintains export list: local file systems it will allow to be mounted and clients that can mount them
- Server returns file handle that uniquely identifies the exported file system to the server.
- Mount operation does not change server's view of the file system – only the clients view is changed.

NFS PROTOCOL

- Provides a set of RPCs for name translation and file manipulation (reading and writing)
- Path-name translation:
  - Separate NFS lookup performed on each component of path name
  - Client side cache used to speed-up lookup operation
- Uses remote service paradigm