LECTURE- 22
DFS Design and Implementation

File Concept

- OS abstracts from the physical storage devices to define a logical storage unit: File
- Types:
  - Data: numeric, alphabetic, alphanumeric, binary
  - Program: source and object form

Logical components of a file

- File name: symbolic name
  - When accessing a file, its symbolic name is mapped to a unique file id (ufid or file handle) that can locate the physical file
    - Mapping is the primary function of the directory service
- File attributes – next slide
- Data units
  - Flat structure of a stream of bytes of sequence of blocks
  - Hierarchical structure of indexed records

File Attributes

- **File Handle** – Unique ID of file
- **Name** – only information kept in human-readable form
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file (and the maximum allowable) size
- **Protection** – controls who can read, write, execute
- **Time**, **date**, **and user identification** – data for protection, security, and usage monitoring.
- Information about files are kept in the directory structure, which is maintained on the physical storage device.

Access Methods

- Sequential access: information is processed in order
  - *read next*
  - *write next (append to the end of the file)*
  - *reset to the beginning of file*
  - *skip forward or backward n records*
- Direct access: a file is made up of fixed length logical blocks or records
  - *read n*
  - *write n*
  - *position to n*
  - *read next*
  - *write next*
  - *rewrite n*

Access Methods (Cont.)

- Indexed sequential access
  - Data units are addressed directly by using an index (key) associated with each data block
  - Requires the maintenance of an search index on the file, which must be searched to locate a block address for each access
  - Usually used only by large file systems in mainframe computers
  - Indexed sequential access method (ISAM)
    - A two-level scheme to reduce the size of the search index
    - Combine the direct and sequential access methods

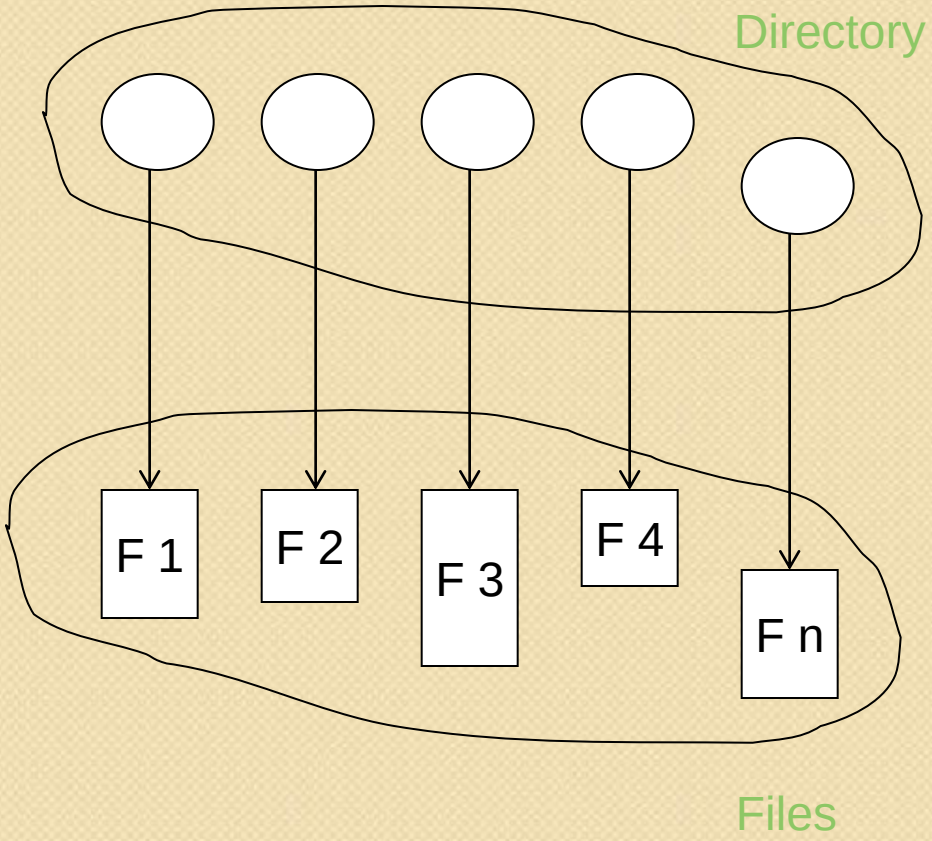
Major Components in A File System

A file system organizes and provides access and protection services for a collection of files

Directory Service		Name resolution, add and deletion of files
Authorization Service		Capability and /or access control list
File Service	Transaction	Concurrency and replication management
	Basic	Read/write files and get/set attributes
System Service		Device, cache, and block management

Directory Structure

- Access to a file must first use a directory service to locate the file.
- A collection of nodes containing information about all files.
- Both the directory structure and the files reside on disk.



Information in a Directory

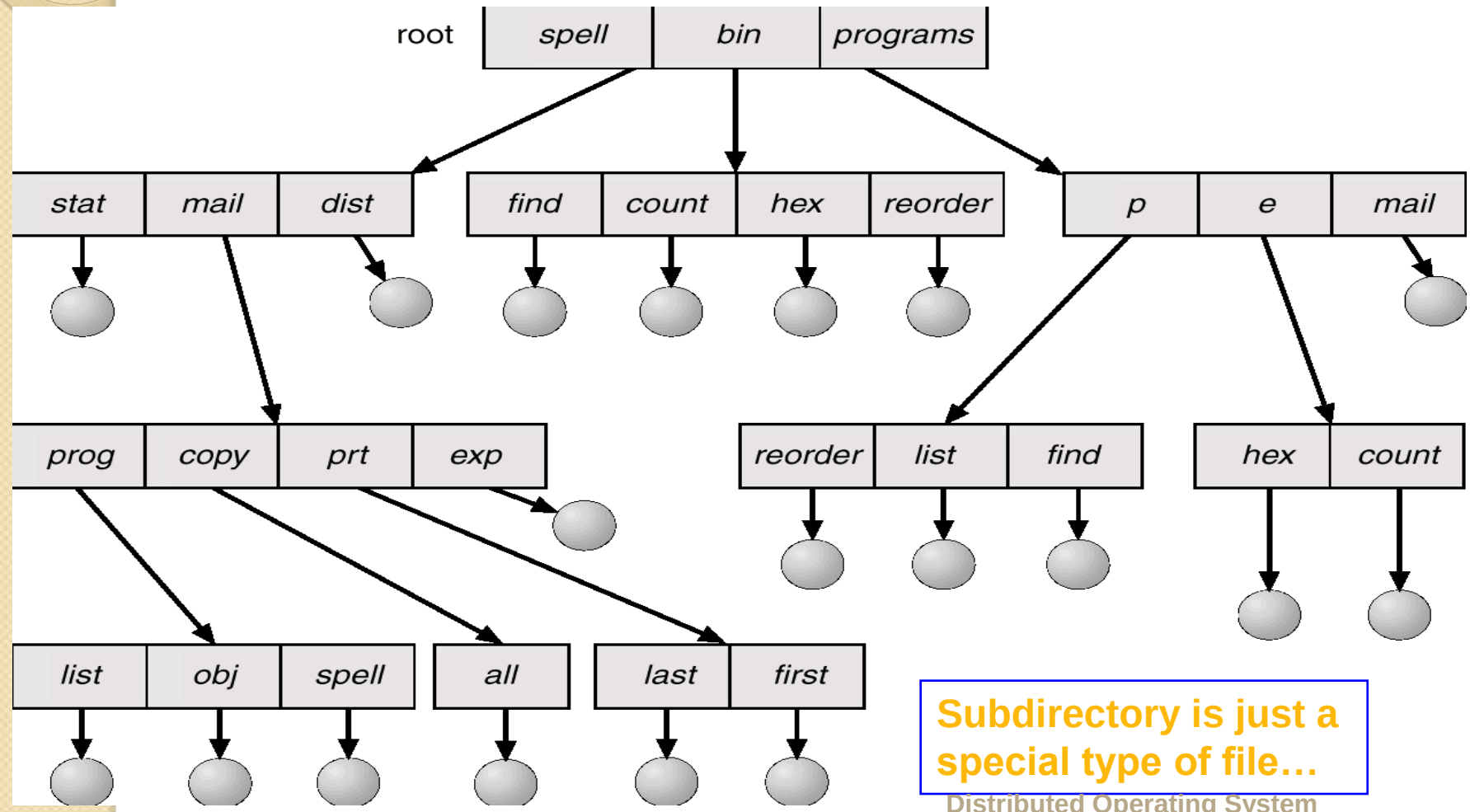
- Name
- Type: file, directory, symbolic link, special file…
- Address: device blocks to store a file
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID
- Protection information

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Some kind of name service

Tree-Structured Directories – Hierarchical Structure of A File System



Subdirectory is just a special type of file…

Authorization Service

- File access must be regulated to ensure security
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

File Service – File Operations

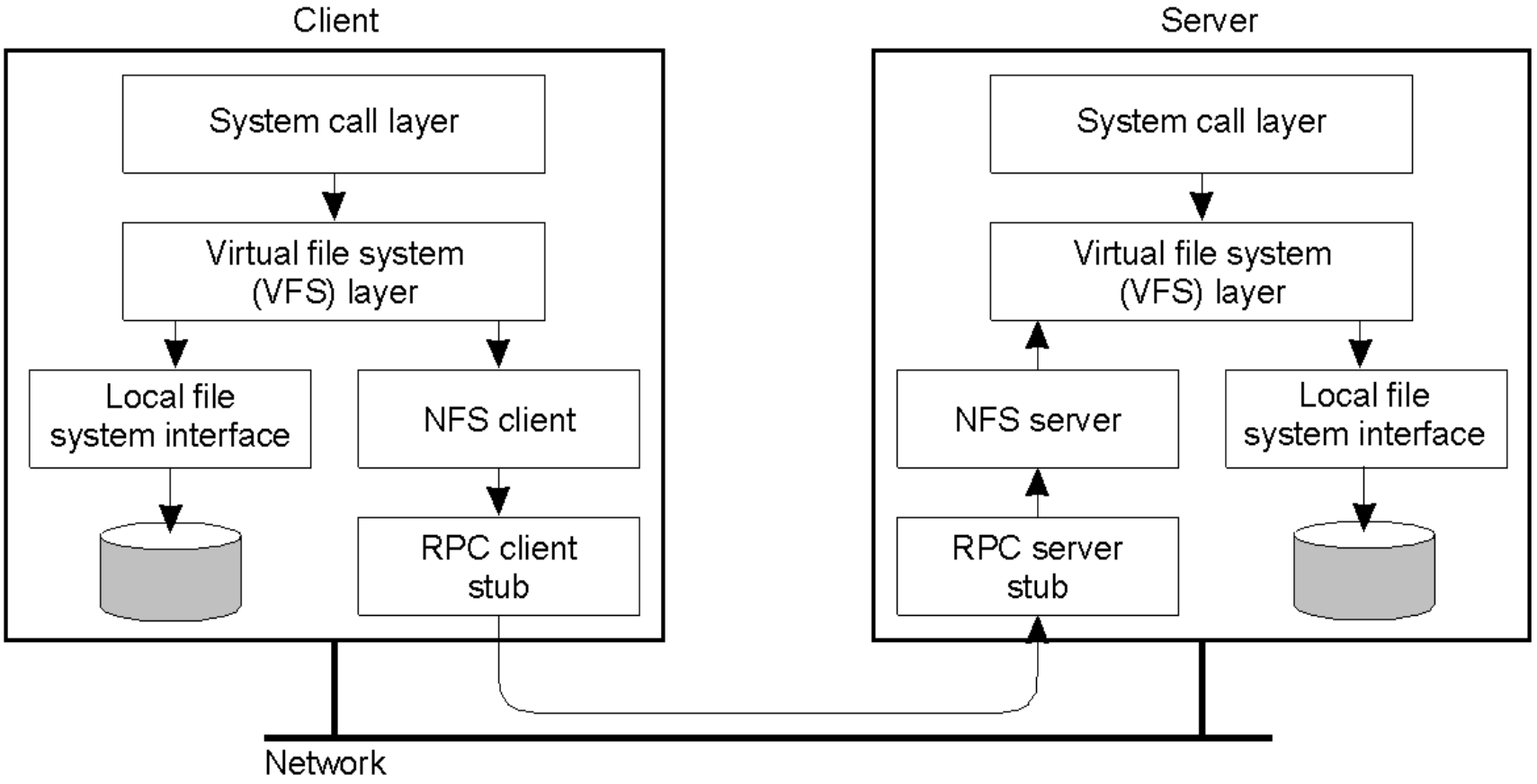
- Create
  - Allocate space
  - Make an entry in the directory
- Write
  - Search the directory
  - Write is to take place at the location of the write pointer
- Read
  - Search the directory
  - Read is to take place at the location of the read pointer
- Reposition within file – file seek
  - Set the current file pointer to a given value

- Delete
  - Search the directory
  - Release all file space
- Truncate
  - Reset the file to length zero
- Open(Fi)
  - Search the directory structure
  - Move the content of the directory entry to memory
- Close(Fi)
  - move the content in memory to directory structure on disk
- Get/set file attributes

System Service

- Directory, authorization, and file services are user interfaces to a file system (FS)
- System services are a FS's interface to the hardware and are transparent to users of FS
  - Mapping of logical to physical block addresses
  - Interfacing to services at the device level for file space allocation/de-allocation
  - Actual read/write file operations
  - Caching for performance enhancement
  - Replicating for reliability improvement

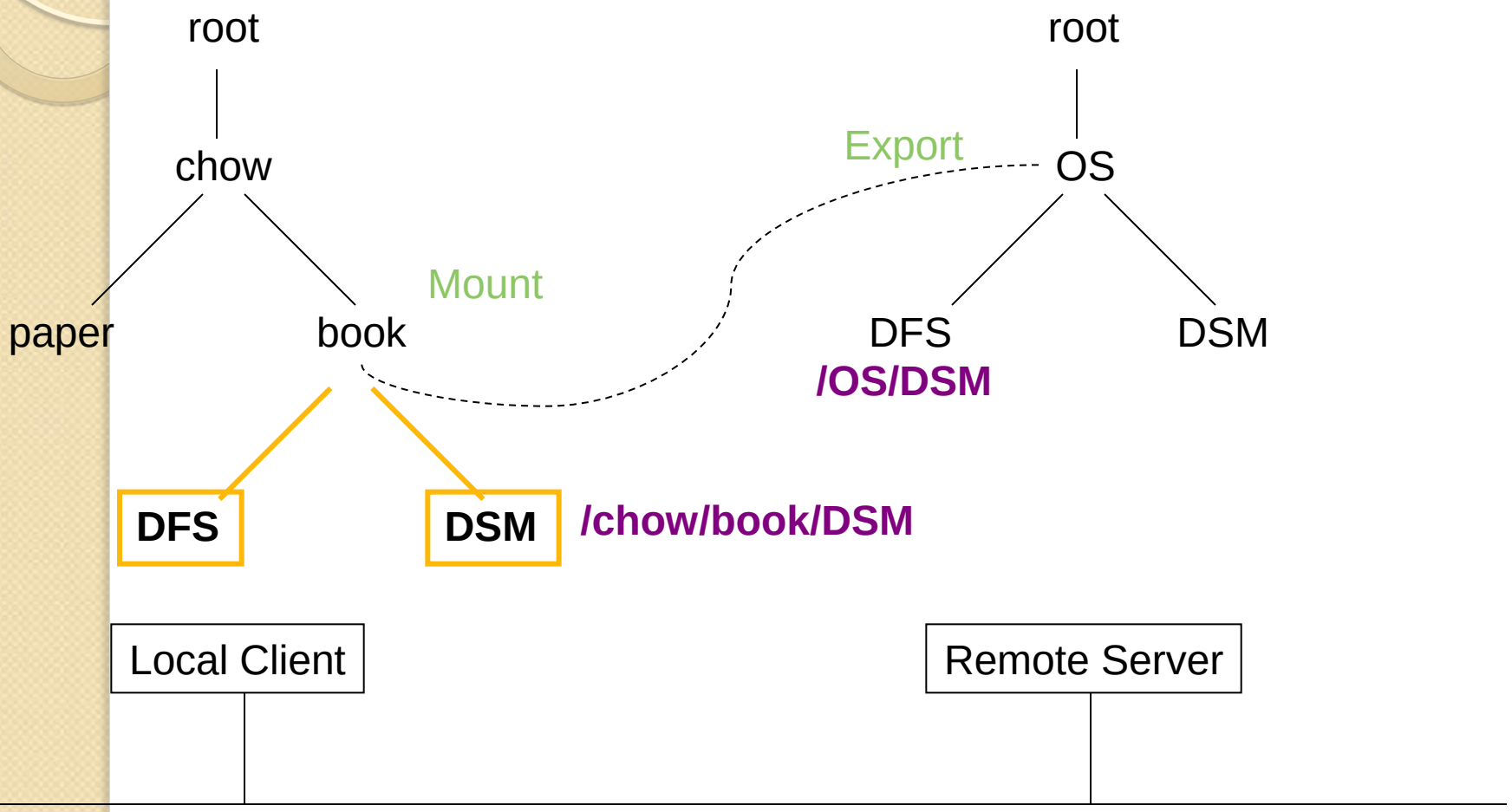
DFS Architecture – NFS Example



File Mounting

- A useful concept for constructing a large file system from various file servers and storage devices
- Attach a remote named file system to the client's file system hierarchy at the position pointed to by a path name (mounting point)
  - A mounting point is usually a leaf of the directory tree that contains only an empty subdirectory
  - mount claven.lib.nctu.edu.tw:/OS /chow/book
- Once files are mounted, they are accessed by using the concatenated logical path names without referencing either the remote hosts or local devices
  - Location transparency
  - The linked information (mount table) is kept until they are unmounted

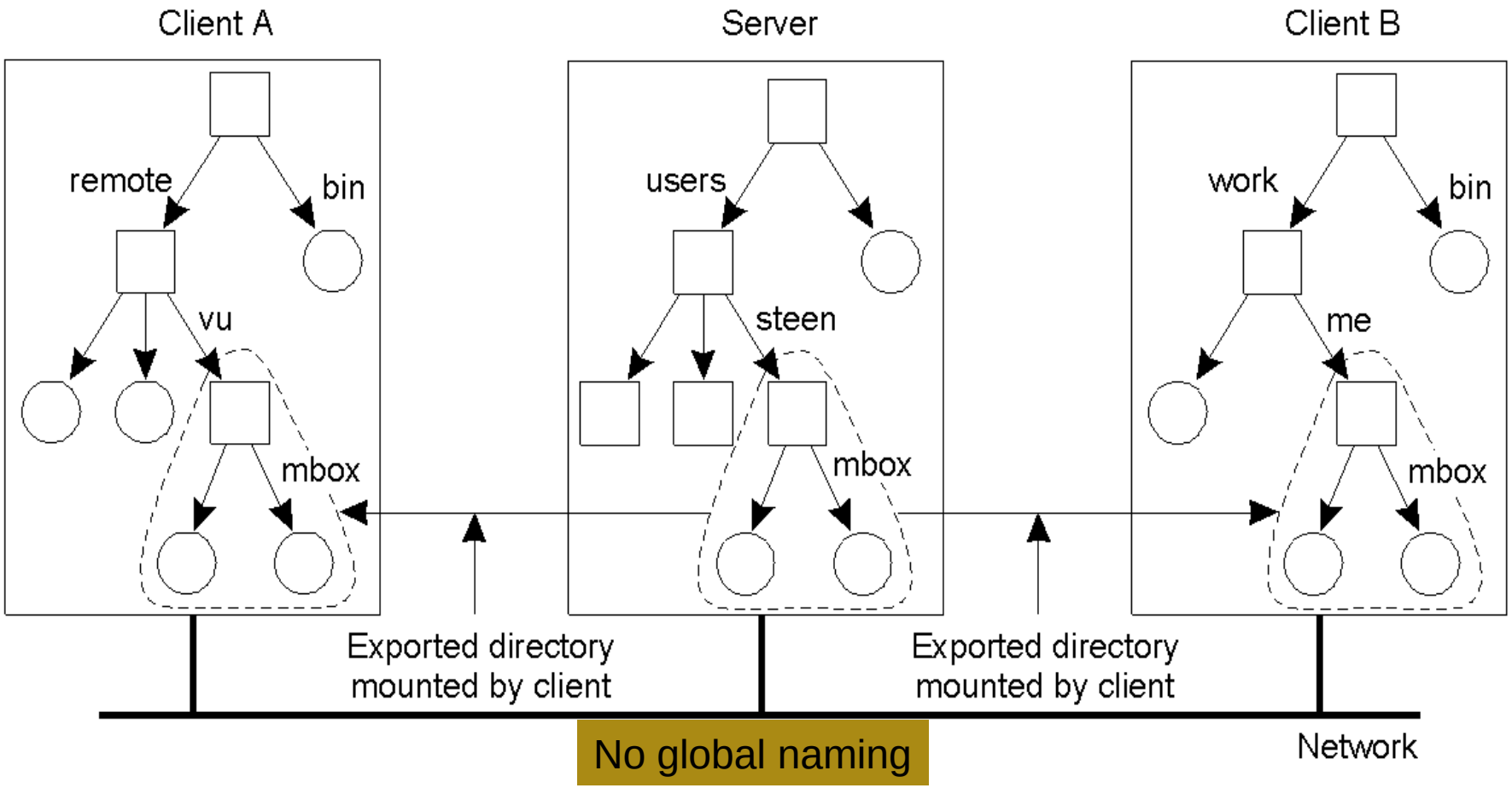
File Mounting Example



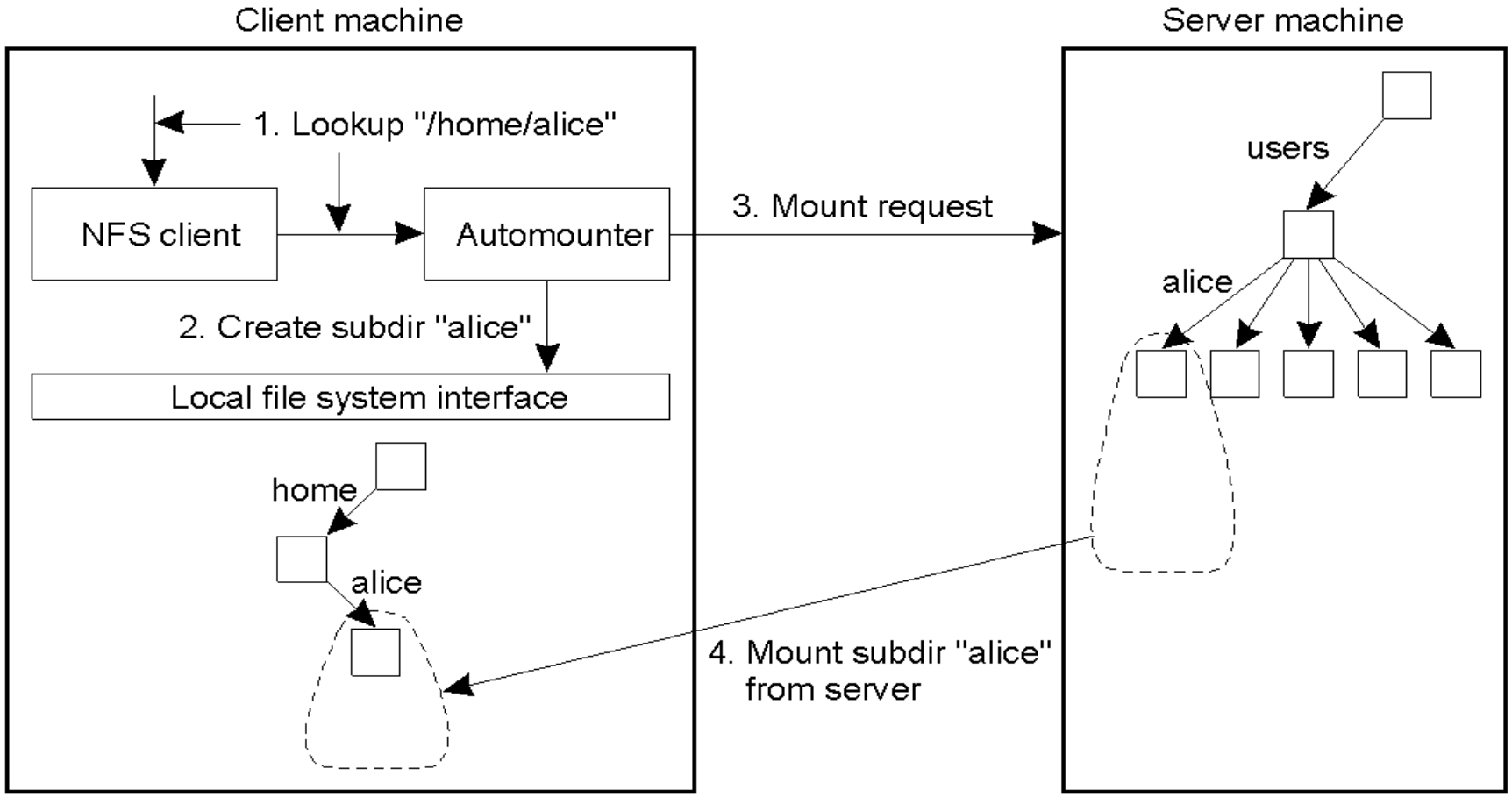
File Mounting (Cont.)

- Different clients may perceive a different FS view
  - To achieve a global FS view – SA enforces mounting rules
- Export: a file server restricts/allows the mounting of all or parts of its file system to a predefined set of hosts
  - The information is kept in the server's *export* file
- File system mounting:
  - Explicit mounting: clients make explicit mounting system calls whenever one is desired
  - Boot mounting: a set of file servers is prescribed and all mountings are performed the client's boot time
  - Auto-mounting: mounting of the servers is implicitly done on demand when a file is first opened by a client

Location Transparency



A Simple Automounter for NFS



Server Registration

- The mounting protocol is not transparent – require knowledge of the location of file servers
- When multiple file servers can provide the same file service, the location information becomes irrelevant to the clients
- Server registration → name/address resolution
  - File servers register their services with a registration service, and clients consult with the registration server before mounting
  - Clients broadcast mounting requests, and file servers respond to client's requests

Stateful and Stateless File Servers

- Stateless file server – when a client sends a request to a server, the server carries out the request, sends the reply, and then remove from its internal tables all information about the request
  - Between requests, no client-specific information is kept on the server
  - Each request must be self-contained: full file name and offset…
- Stateful file server – file servers maintain state information about clients between requests
- State information – may be kept in servers or clients
  - Opened files and their clients
  - File descriptors and file ha
  - Current file position pointe
  - Mounting information
  - Lock status
  - Session keys
  - Cache or buffer

Session: a connection for a sequence of requests and responses between a client and the file server

A Comparison between Stateless and Stateful Servers

Advantages of Stateless Server	Advantages of Stateful Server
No OPEN/CLOSE calls needed	Better performance
Fault tolerance	Shorter request messages
No server space wasted on tables	Read-ahead possible
No limits on number of open files	Idempotency easier
No problems if a client crashes	File locking possible
Easy to implement	More flexible

Issues of A Stateless File Server

- Idempotency requirement
  - Is it practical to structure all file accesses as idempotent operations?
- File locking mechanism
  - Should locking mechanism be integrated into the transaction service?
- Session key management
  - Can one-time session key be used for each file access?
- Cache consistency
  - Is the file server responsible for controlling

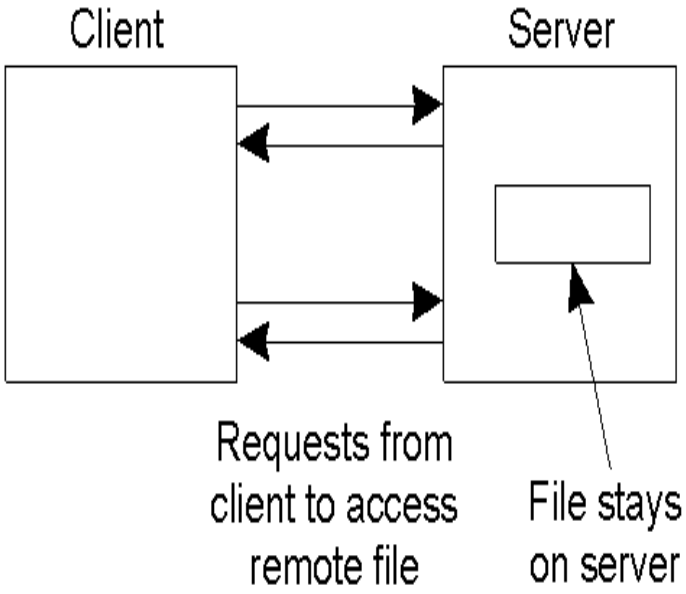
File Sharing

- Overlapping access: multiple copies of the same file
  - Space multiplexing of the file
  - Cache or replication
  - Coherency control: managing accesses to the replicas, to provide a coherent view of the shared file
  - Desirable to guarantee the atomicity of updates (to all copies)
- Interleaving access: multiple granularities of data access operations
  - Time multiplexing of the file

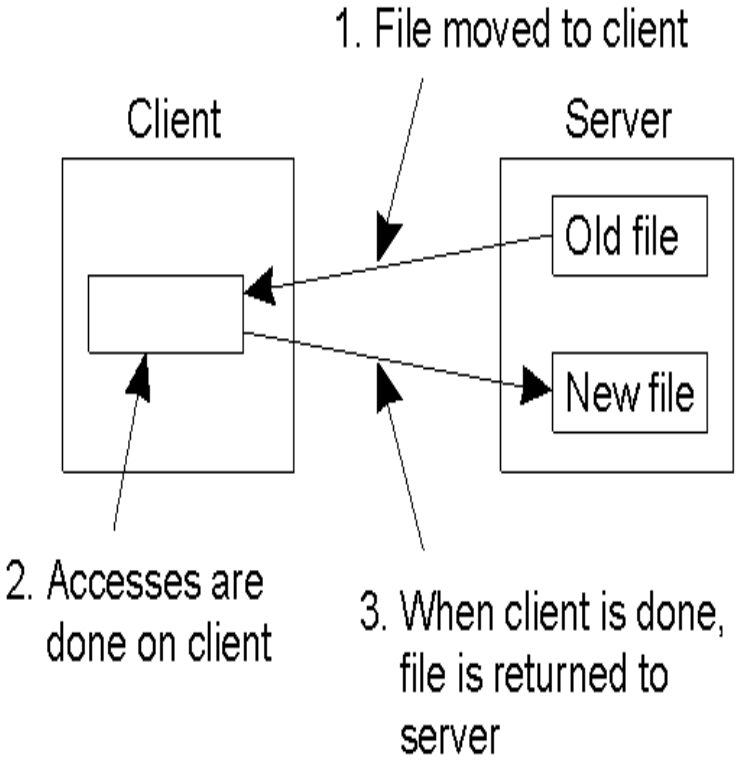
Space Multiplexing

- Remote access: no file data is kept in the client machine. Each access request is transmitted directly to the remote file server through the underlying network.
- Cache access: a small part of the file data is maintained in a local cache. A write operation or cache miss results a remote access and update of the cache
- Download/upload access: the entire

Remote Access VS Download/Upload Access

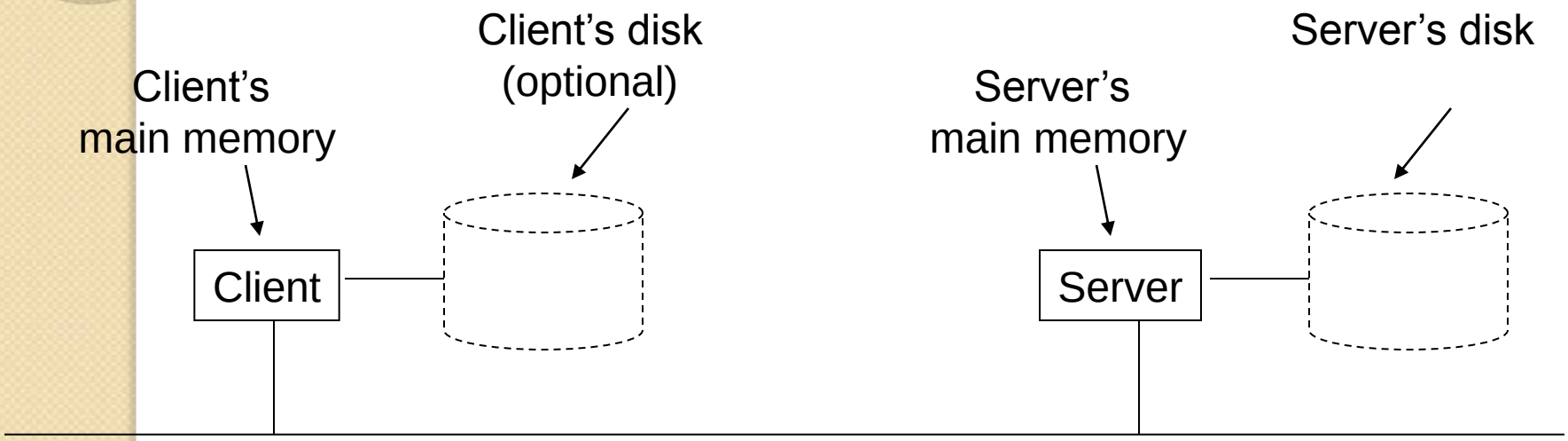


Remote Access



Download/Upload Access

Four Places to Caching



Coherency of Replicated Data

- Four interpretations:
  - All replicas are identical at all times
    - Impossible in distributed systems
  - Replicas are perceived as identical only at some points in time
    - How to determine the good synchronization points?
  - Users always read the “most recent” data in the replicas
    - How to define “most recent”?
      - Based on the “completion” times of write operations (the effect of a write operation has been reflected in all copies)

Time Multiplexing

- Simple RW: each read/write operation is an independent request/response access to the file server
- Transaction RW: a sequence of read and write operations is treated as a fundamental unit of file access (to the same file)
  - ACID properties
- Session RW: a sequence of transaction and simple RW operations

Space and Time Concurrencies of File Access

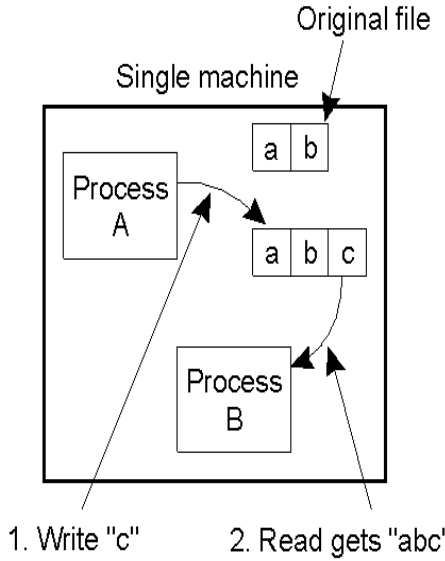
Space / Time	Remote Access	Cache Access	Download/Upload Access
Simple RW	No true sharing	Coherency Control	Coherency Control
Transaction	Concurrency Control	Coherency and Concurrency Control	Coherency and Concurrency Control
Session	Not applicable	Not applicable	Ignore sharing

Semantics of File Sharing

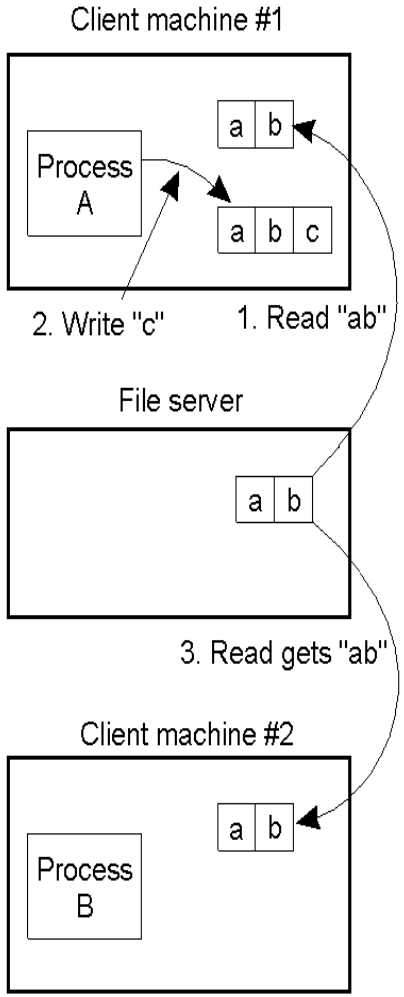
a) On a single processor, when
write, the value returned by t
value just written (Unix Sema

b) In a distributed system with c
values may be returned.

Solution to coherency and concurrency control problems depends on the semantics of sharing required by applications



(a)



(b)

Semantics of File Sharing (Cont.)

Unix Semantics (Currency)	Every operation on a file is instantly visible to all processes. File accesses with a write-through cache and write-invalidation
Transaction Semantics (Consistency)	All changes have the all-or-nothing property. Update the server at the end of a transaction.
Immutable Files	No updates are possible; simplify sharing and replication
Session Semantics (Efficiency)	No changes are visible to other processes until the file is closed. Update the server at the end of a session.

Version Control

- Version control under immutable files
  - Implemented as a function of the directory service
  - Each file is attached with a version number
  - An open to a file always returns the current version
  - Subsequently read/write operations to the opened files are made only to the local working copy
  - When the file is closed, the local modified version (tentative version) is presented to

Version Control (Cont.)

- Action to be taken if based on an older version…
  - Ignore conflict: a new version is created regardless of what has happened (equivalent to session semantics)
  - Resolve version conflict: the modified data in the tentative version are disjoint from those in the new current version
    - Merge the updates in the tentative version with the current version to yield to a new version that combines all updates
  - Resolve serializability conflict: the

Windows 2003 Server R2

The purpose of distributed file system is to minimize network traffic due to file replication and optimize the administration of shared folder

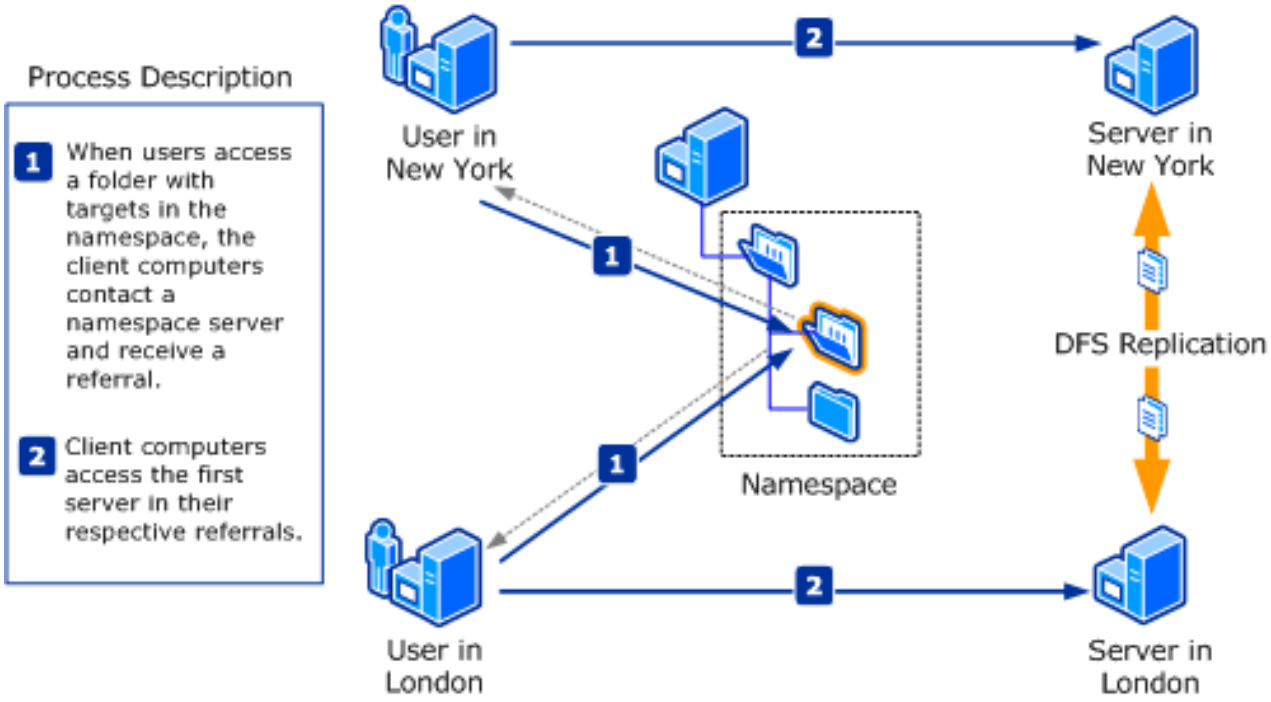
DFS Replication

- DFS Replication is a state-based, multimaster replication engine that supports replication scheduling and bandwidth throttling. DFS Replication uses a new compression protocol called Remote Differential Compression (RDC), which can be used to efficiently update files over a limited-bandwidth network. RDC detects insertions, removals, and re-arrangements of data in files, thereby

Namespaces

- DFS Namespaces, formerly known as Distributed File System, allows administrators to group shared folders located on different servers and present them to users as a virtual tree of folders known as a namespace. A namespace provides numerous benefits, including increased availability of data, load sharing, and simplified data migration.

DFS in Win2003 R2



ASSIGNMENT

- Q: Explain DFS design and implementation.