Network File Systems (NFS): Case Study

- **NFS (Network File System)**
  - Developed by Sun Microsystems (in 1985)
  - Most popular, open, and widely used.
  - NFS protocol standardized through IETF (RFC 1813)

NFS Architecture

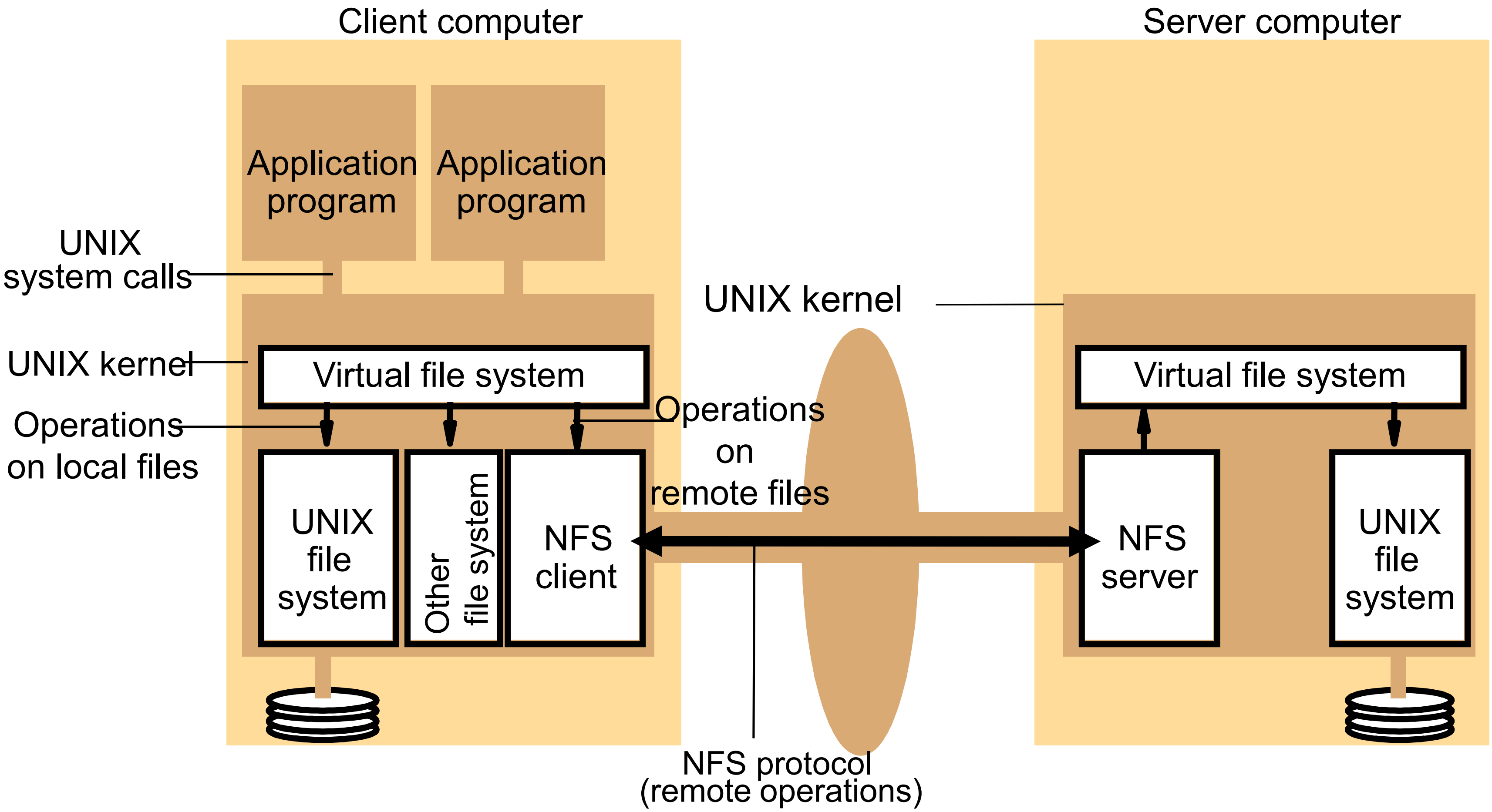
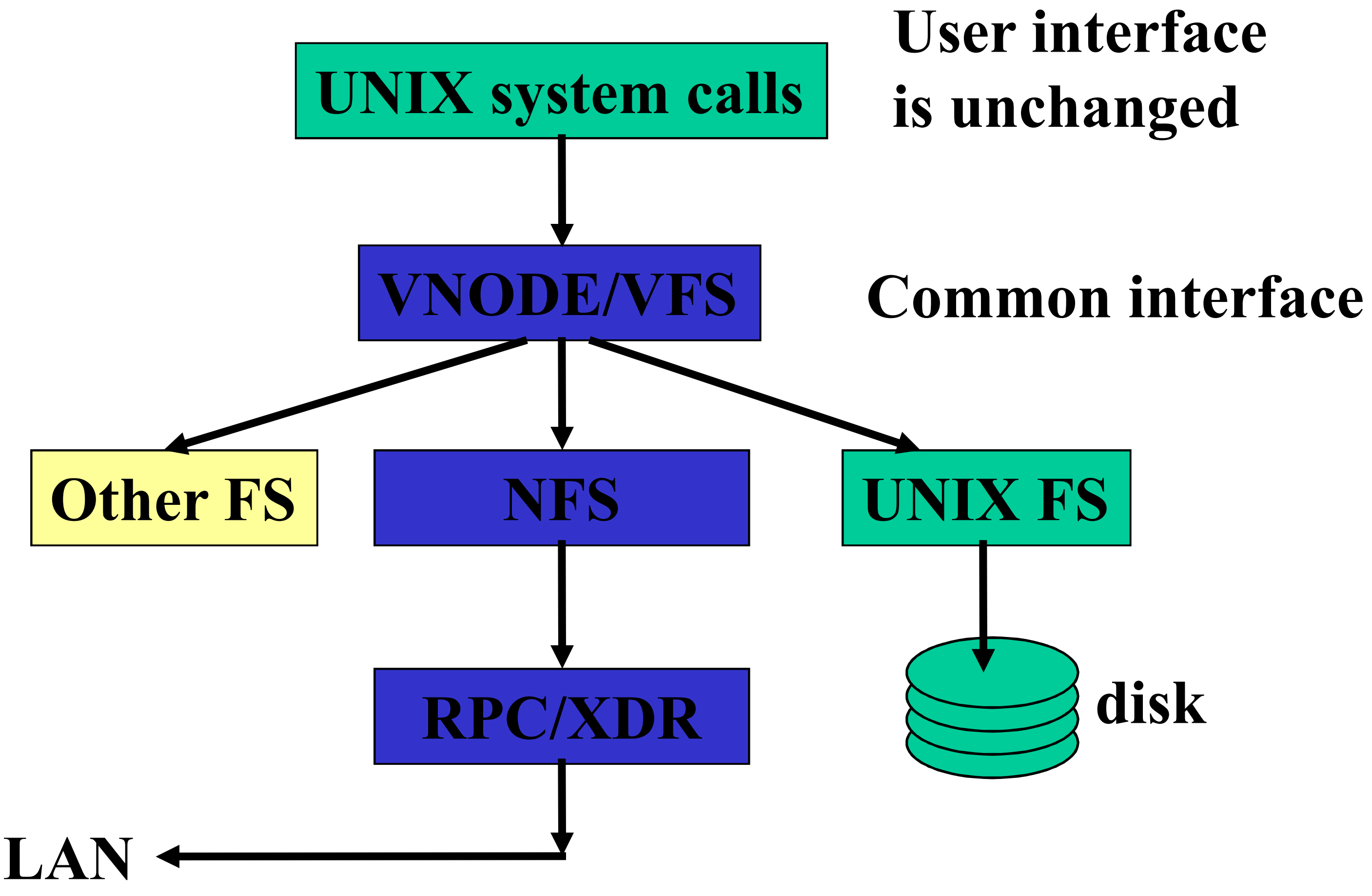


Figure: NFS architecture

Client side (III)



UNIX system calls

User interface is unchanged

VNODE/VFS

Common interface

Other FS

NFS

UNIX FS

RPC/XDR

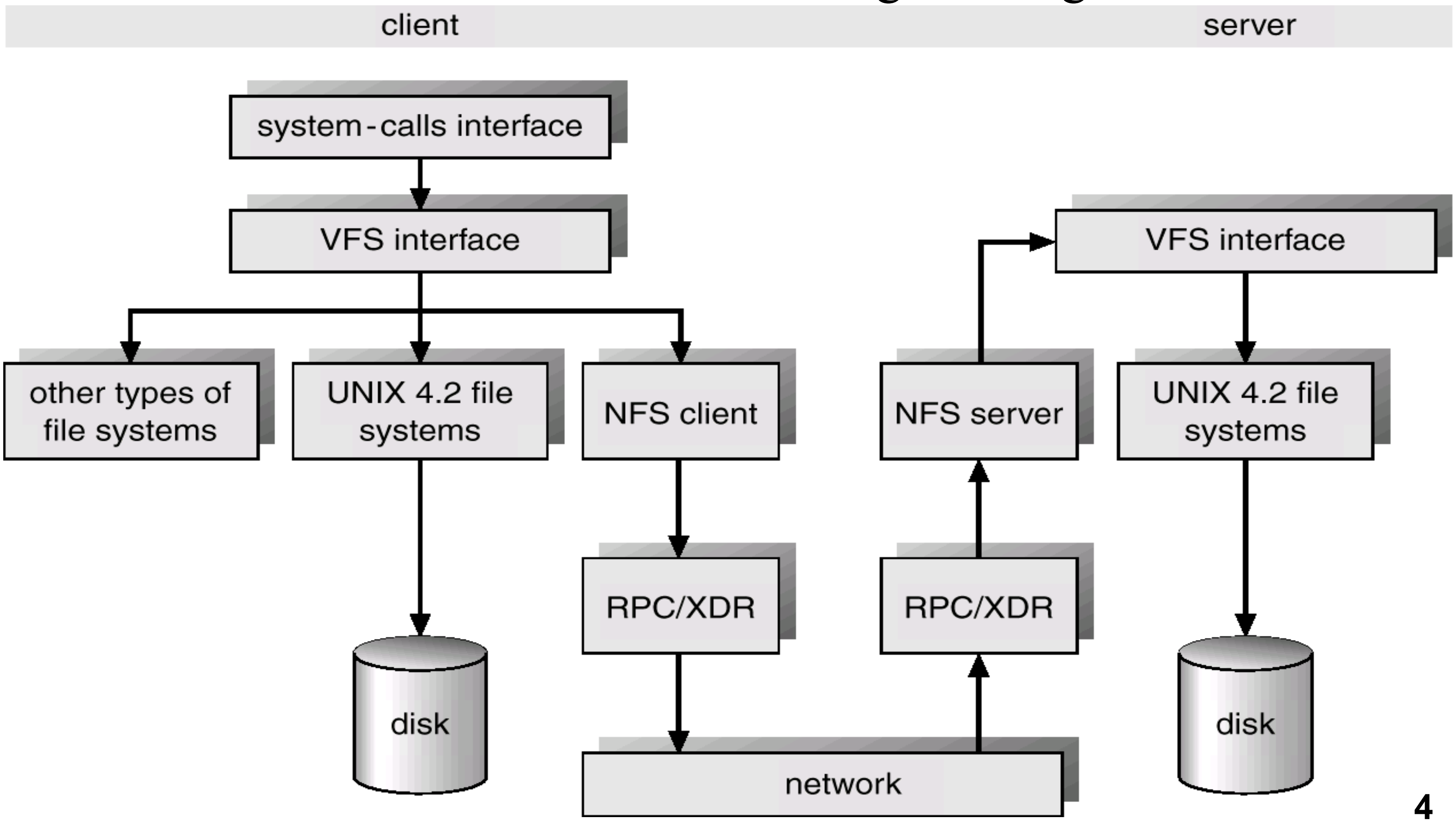
disk

LAN

DISTRIBUTED FILE SYSTEMS
SUN Network File System

NFS ARCHITECTURE:

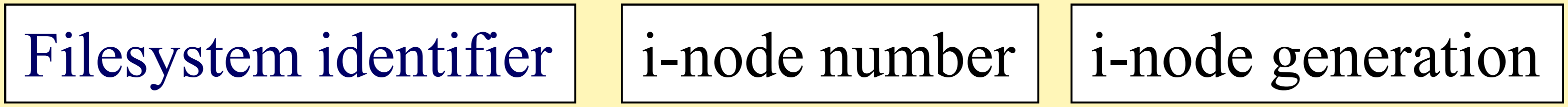
Follow local and remote access through this figure:



Sun NFS

- The file identifiers used in NFS are called file handles.

fh = file handle:


Filesystem identifier | i-node number | i-node generation

Server side

- ***File handle*** consists of
  - ***Filesystem id*** identifying disk partition
  - ***I-node number*** identifying file within partition
  - ***Generation number*** changed every time. i-node number is i-node number is reused after a file is removed (to store a new file)
  - **V-node-** contains an indicator to show whether a file is local or remote. If file is local, the v-node contain reference to the index of local file(i-node). If the file system is remote, it contains the file handle to remote file.
- ***Filesystem id*** in filesystem **superblock(**File system type, Size, Status, Information about other metadata structures**)**

DISTRIBUTED FILE SYSTEMS

SUN Network File System

NFS ARCHITECTURE:

1. UNIX filesystem layer - does normal open / read / etc. commands.
2. Virtual file system (VFS) layer –
   a) Gives clean layer between user and filesystem.
   b) Acts as deflection point by using global vnodes.
   c) Understands the difference between local and remote names.
   d) Keeps in memory information about what should be deflected (mounted directories) and how to get to these remote directories.
3. System call interface layer -
   a) Presents sanitized validated requests in a uniform way to the VFS.

Sun NFS

- Mount service
  - Mount operation:

    mount(remotehost, remotedirectory, localdirectory)
  - Server maintains a table of clients who have mounted filesystems at that server.
  - Each client maintains a table of mounted file systems holding:

    < IP address, port number, file handle>
  - Remote file systems may be hard-mounted (In case of failure, it will repeatedly retry to connect the server) or soft-mounted (In case of failure, it will report an error) in a client computer.
  - Figure 10 illustrates a Client with two remotely mounted file stores.

Sun NFS

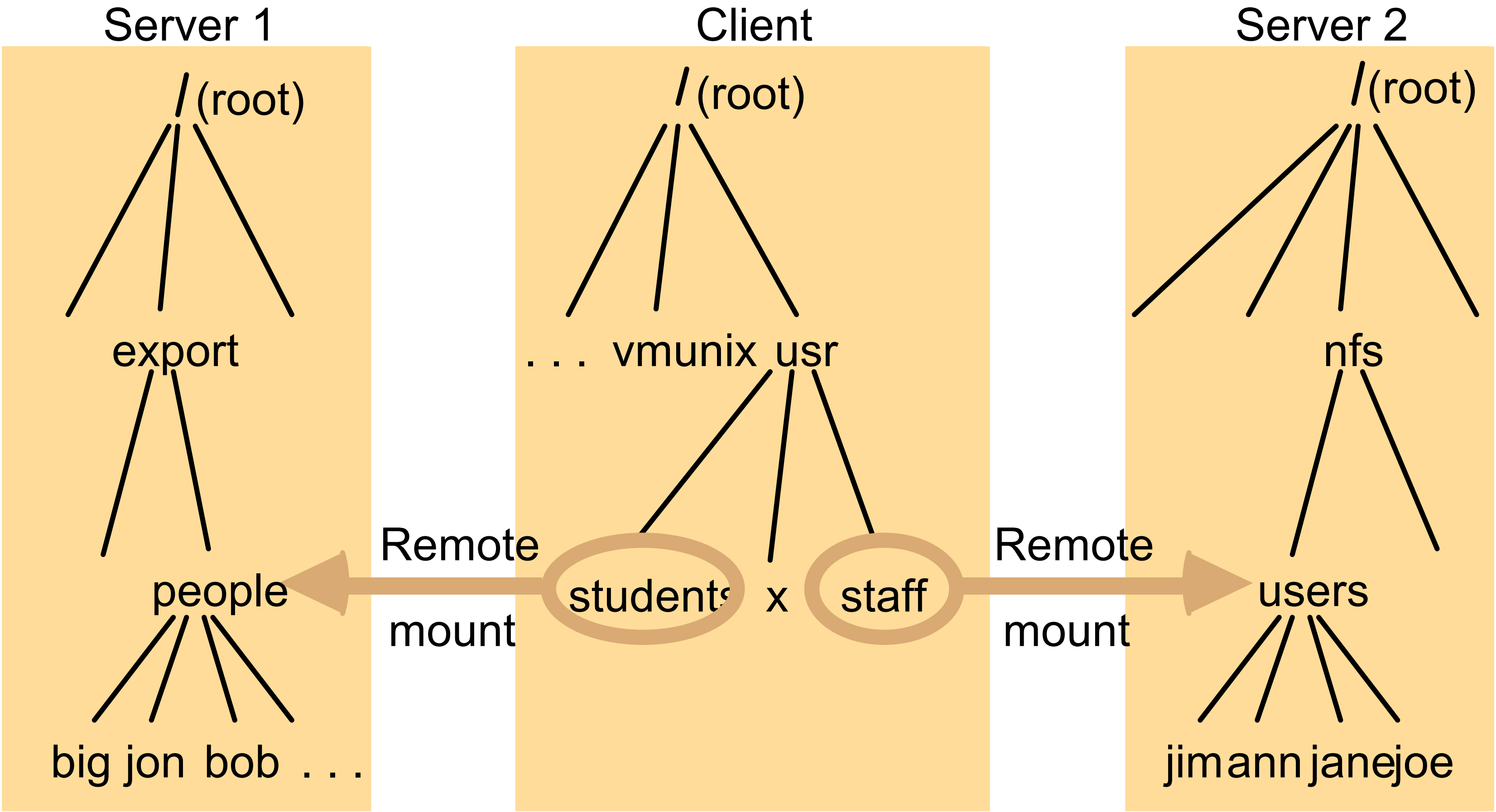


Figure : Local and remote file systems accessible on an NFS client

Note: The file system mounted at */usr/students* in the client is actually the sub-tree located at */export/people* in Server 1; the file system mounted at */usr/staff* in the client is actually the sub-tree located at */nfs/users* in Server 2.

Sun NFS

- NFS summary
  - NFS is an excellent example of a simple, robust, high-performance distributed service.
  - Achievement of transparencies are other goals of NFS:
    - Access transparency:
      - Enables local and remote resources to be accessed using identical operations.
      - The API is the UNIX system call interface for both local and remote files.
    - Location transparency:
      - NFS provides the location transparency i.e name does not hint at its physical storage location.

Sun NFS

- Mobility transparency:
  - Hardly achieved; relocation of filesystems is possible, but requires updates to client configurations.
- Scalability transparency:
  - NFS does not scale well as it is limited to LAN
- Replication transparency:
  - Limited to read-only file systems; for writable files, the SUN Network Information Service (NIS) runs over NFS and is used to replicate essential system files.
- Hardware and software operating system heterogeneity:
  - NFS has been implemented for almost every known operating system and hardware platform and is supported by a variety of filling systems.

Sun NFS

- Fault tolerance:
    - Limited but effective; service is suspended if a server fails. Recovery from failures is aided by the simple stateless design.

Sun NFS

CACHES OF REMOTE DATA:

- The client keeps:
  File block cache - (the contents of a file)
  File attribute cache - (file header info (inode in UNIX)).

- The local kernel hangs on to the data after getting it the first time.

- On an open, local kernel, it checks with server that cached data is still OK.

- Cached attributes are thrown away after a few seconds.