ECE437/CS481

M06C: VIRTUAL FILE SYSTEM & NETWORK FILE SYSTEM

CHAPTER 12.8

Xiang Sun

The University of New Mexico

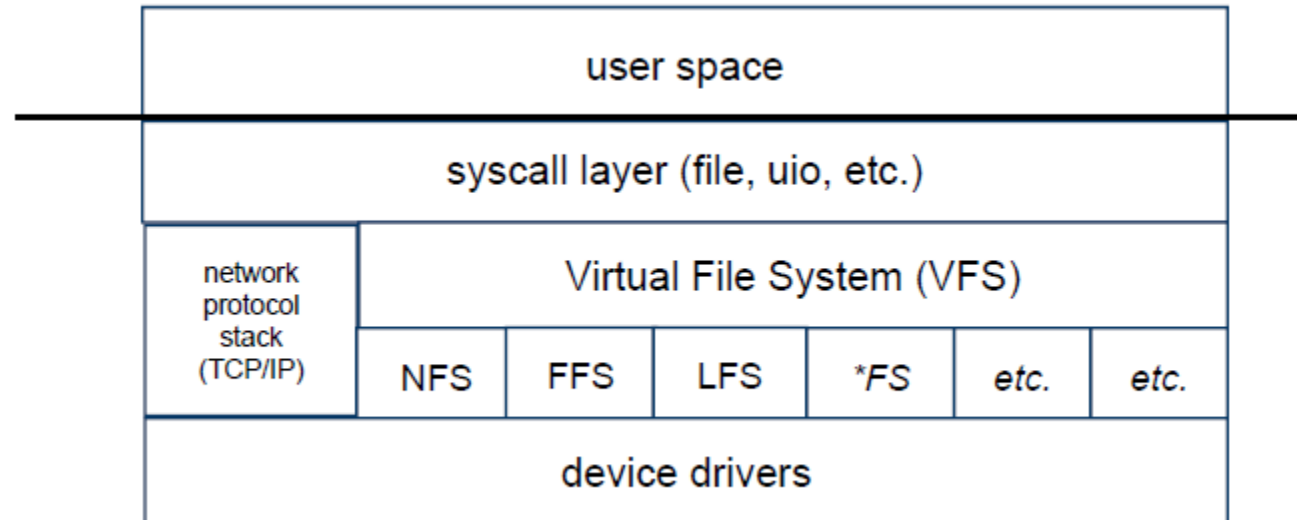# Virtual File System

❑ Lots and lots of file system types

➢ Different file systems may have different directory structures, block sizes, etc.
➢ Is it possible to support more than one file system types on a single OS?

File System	Windows XP	Windows 7/8/10	macOS (10.6.4 and earlier)	macOS (10.6.5 and later)	Ubuntu Linux	Playstation 4	Xbox 360/One
NTFS	Yes	Yes	Read Only	Read Only	Yes	No	No/Yes
FAT32	Yes	Yes	Yes	Yes	Yes	Yes	Yes/Yes
exFAT	Yes	Yes	No	Yes	Yes (with ExFAT packages)	Yes (with MBR, not GUID)	No/Yes
HFS+	No	(read-only with Boot Camp)	Yes	Yes	Yes	No	Yes
APFS	No	No	No	Yes (macOS 10.13 or greater)	No	No	No
EXT 2, 3, 4	No	Yes (with third-party software)	No	No	Yes	No	Yes

Virtual File System

❑ Virtual file system

➢ A kernel software layer that handles all system calls related to file systems. Its main strength is providing a **common interface** to file systems.

user space					
syscall layer (file, uio, etc.)					

network protocol stack (TCP/IP)	Virtual File System (VFS)				
	NFS	FFS	LFS	*FS	etc.	etc.

device drivers

✓ Processes make system calls to VFS.
✓ VFS separates file-system generic operations from implementation details, and then dispatches operations to appropriate file system implementation routines.

Virtual File System

❑ VFS system calls (~100)

➢ **filesystem** operations – mounting, info, chroot, pivot_root

➢ **directory** operations – chdir, getcwd, link, unlink, rename, symlink

➢ **file** operations – open/close, read/write, stat, permissions, seek
  • chmod, chown, stat, creat, umask, dup, fcntl, truncate
  • read/write, readv/writev, pread/pwrite

➢ **memory mapping** files – mmap, munmap, madvise, mlock

➢ **wait** for input – poll, select

➢ **flushing** – synch, fsync, msync, fdatasync

➢ **file locking** – flock

Network File System

❑ Purpose: access disks attached to other computers

❑ NFS is a protocol for remote access to a file system
  ➢ Does not implement a file system per se
  ➢ NFS is a network protocol over TCP/UDP
    ✓ Original implementations use UDP a transport layer protocol for low overhead.
    ✓ Some newer implementations use TCP as a transport layer protocol for reliable communications.
  ➢ Remote access is transparent to applications

❑ File system, OS, and architecture independent

❑ Implemented as a client/server architecture
  ➢ Local file system (i.e., client) requests are forwarded to a remote server
  ➢ Requests are implemented as remote procedure calls (RPCs)
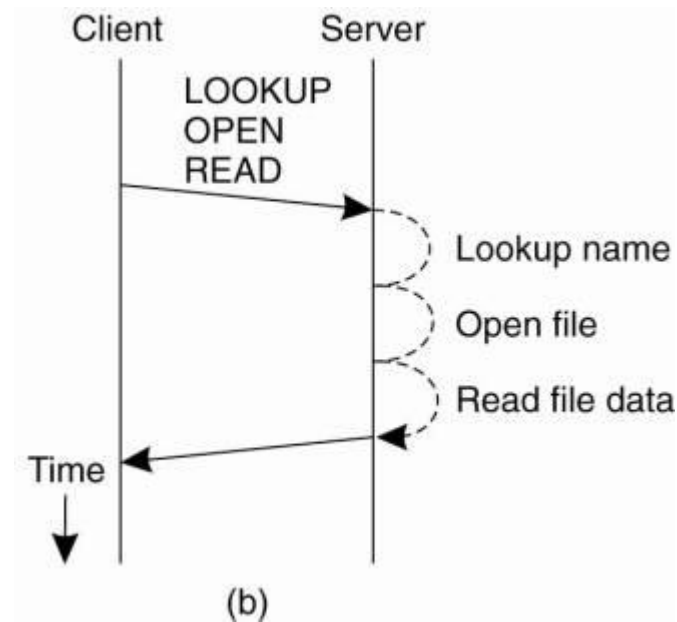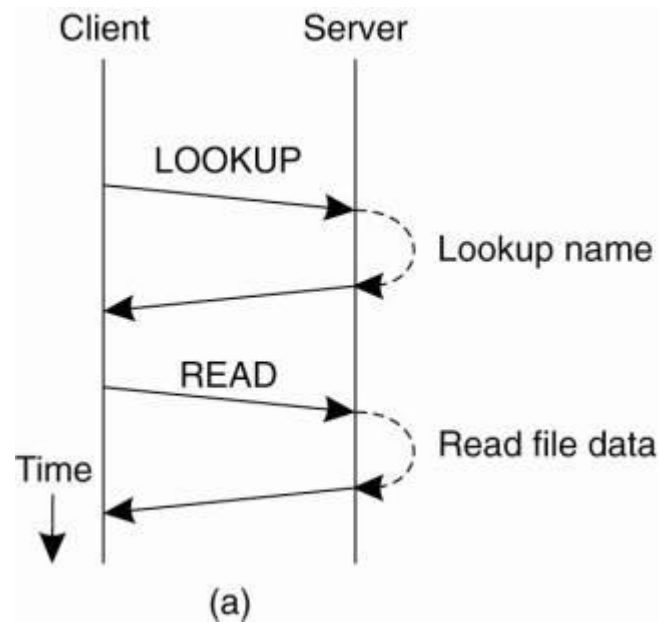
Network File System
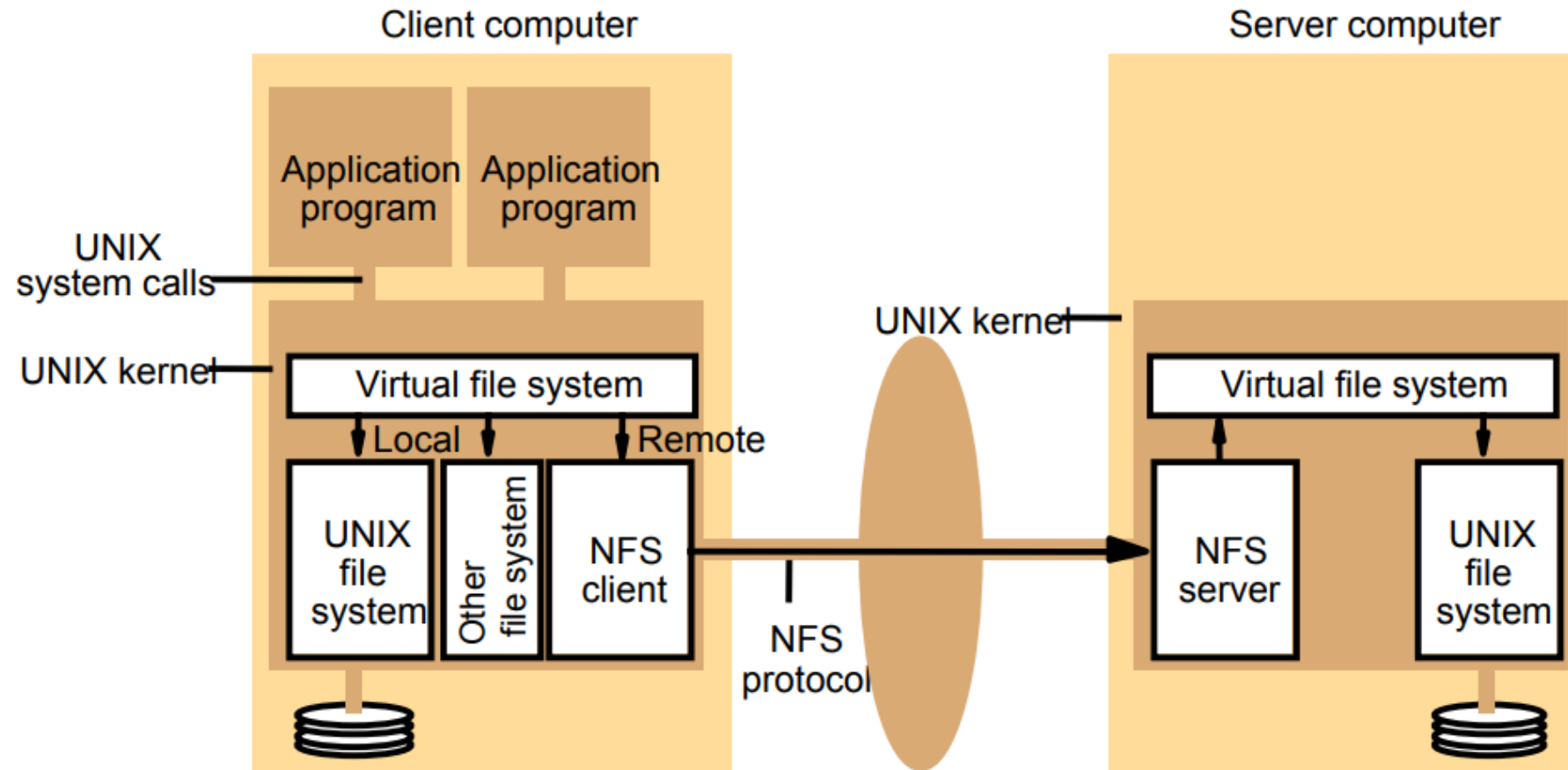
- ❑ NFS server check identity (e.g., uid, gid) for each access.
- ❑ NFS server is **stateless**, doesn't keep open files for clients
  - ✓ In NFS v4, **stateful** server are used.
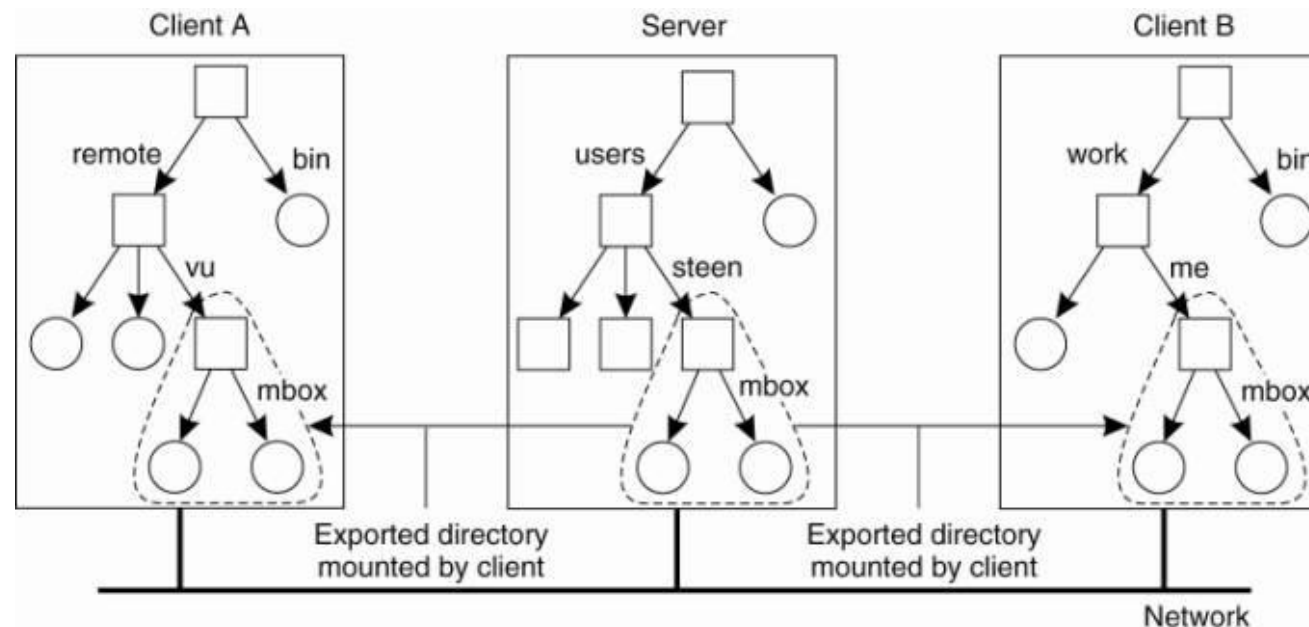
Network File System

❏ Communications in NFS

   ✓ Every NFS operation (e.g., open(), read(), write()) can be implemented as a single remote procedure call (RPC) to a file server.

   ✓ In order to reduce the communications overhead (i.e., reduce the number of RPCs), compound procedures is applied (in NFS v4), where several RPCs can be grouped into a single request.
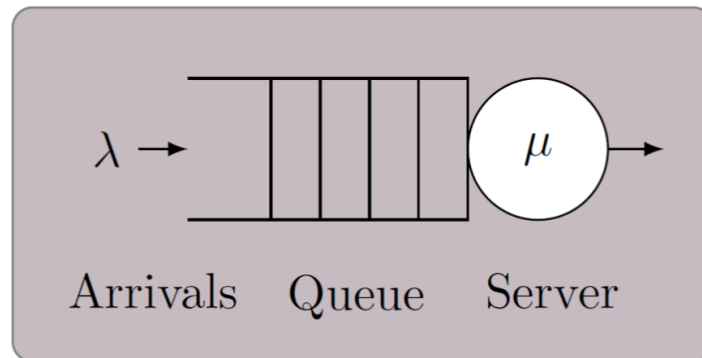
Network File System

❑ Naming in NFS

➢ The fundamental idea underlying the NFS naming model is to provide clients complete transparent access to a remote file system as maintained by a server. This transparency is achieved by letting a client be able to mount (part of) a remote file system into its own local file system.

➢ A server exports a directory to a client, and the client mount the exported directory to its local name space.

© by Dr. X. Sun

Network File System

□ Synchronization in NFS

➢ Different clients mount the same part of file system in the server → files are shared among the clients.

➢ How to achieve synchronization if many clients try to read/write the same file?

➢ If every read/write go directly to the file server, the server can easily achieve the synchronization by processing them strictly sequentially.

➢ However, enabling the server to process every read and write may degrade the performance (in terms of throughput) of NFS, involving too many RPCs.

Network File System

□ Synchronization in NFS

➢ In order to improve the performance, clients are allowed to maintain local copies of heavily-used files in their private local caches. Thus, clients can read/write their local copies, and flush back the cached data to server when files are closed.

➢ If a client locally modifies a cached file and shortly thereafter another client reads the file from the server, the second client will get an obsolete file.

➢ There is no way to guarantee clients can read the up-to-date file in NFS.