Network Organisation and Management

Lecture #11

Department of Computer Science and Technology
University of Bedfordshire

Written by David Goodwin,
based on the lecture series of Dayou Li
and the book *Understanding Operating Systems* 4^{th} *ed.*
by *I.M.Flynn and A.McIver McHoes* (2006).

Operating Systems, 2012

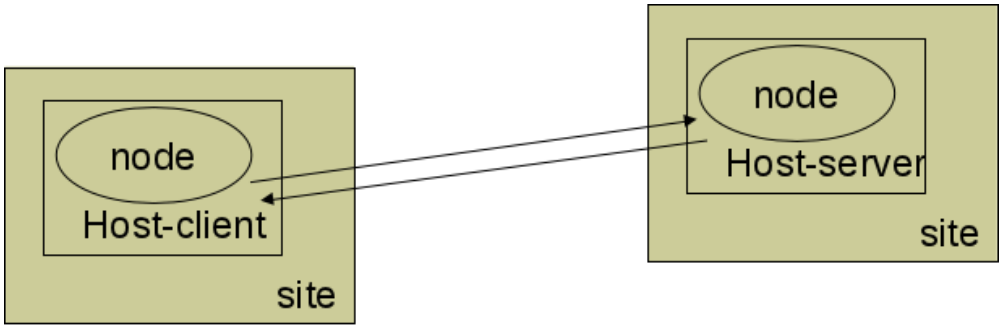
TERMINOLOGY

- Types of network operating systems
  - Network operating system (NOS) – a traditional single-user operating system
  - Distributed operating system (D/OS) – multi-user system comprised of
    - Memory manager
    - Processor manager
    - Device or I/O manager
    - File manager
    - Network manager
- Remote and local
  - Remote – a processor in a networked system regards other processors and their resources as remote
  - Local – it considers its own resources as local

TERMINOLOGY

University of Bedfordshire

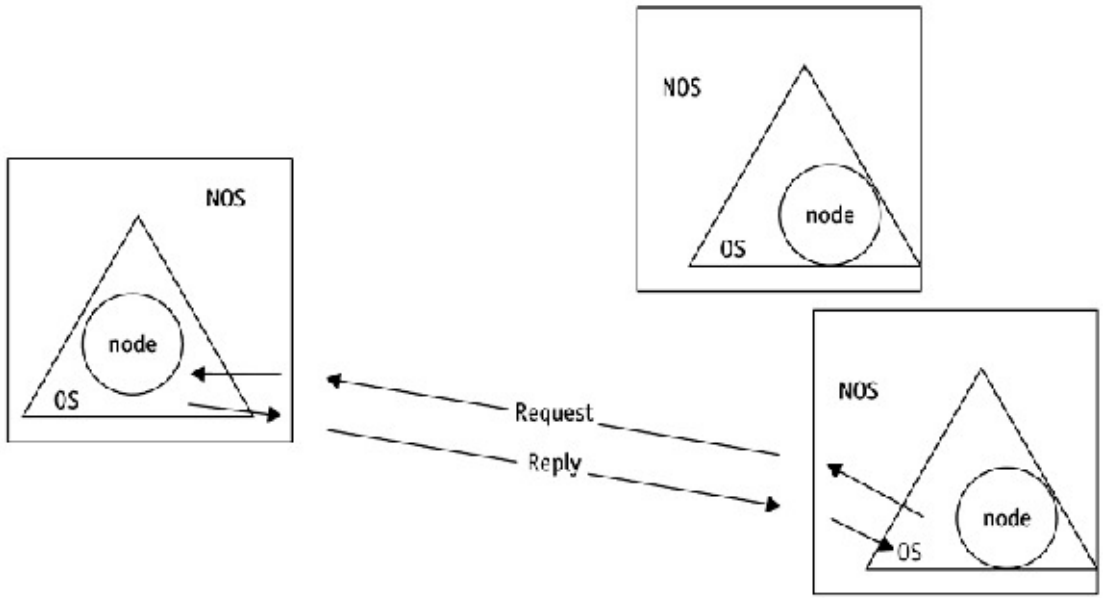
- Site, host and node
  - Site – a location in a networked system containing one or more computers
  - Host – a specific computer of a site whose resources can be used from remote locations
  - Node – name assigned to a computer within a networked system
  - Typically, a host on one site called server owns resources and a host at another site called client wants to use the resources



DISTRIBUTED OPERATING SYSTEMS

University of Bedfordshire

- General descriptions
  - Gives users global access to resources
  - Allows network processes being managed globally
  - Makes the network completely transparent to users and their local operating systems



NETWORK OPERATING SYSTEMS

- NOS typically runs on a computer called a server and performs services for network workstations called clients
- Network management functions come into play only when the system needs to use the network
- Focus is on sharing resources instead of running programs
- Best NOS choice depends on following factors:
  - Applications to be run on the server
  - Technical support required
  - User's level of training

DISTRIBUTED OPERATING SYSTEMS

- Features:
  - Resources owned by local nodes
  - Local recourses managed by local OSs
  - Access performed by local OSs
  - Requests passed from a local OS to another through NOS
  - NOS handles interfacing details and coordinates remote processing
  - It coordinates communications between local operating systems
  - Limitation: Doesn't take global control over memory management, process management, device management, or file management

NOS DEVELOPMENT

- Principles
  - To provide support for standard local area network technologies and client desktop operating systems
  - To have a robust architecture that adapts easily to new technologies
    - Must provide strong support for every operating system in the corporate information network
  - To be able to operate wide range of third-party software applications and hardware devices
  - To supports software for multiuser network applications

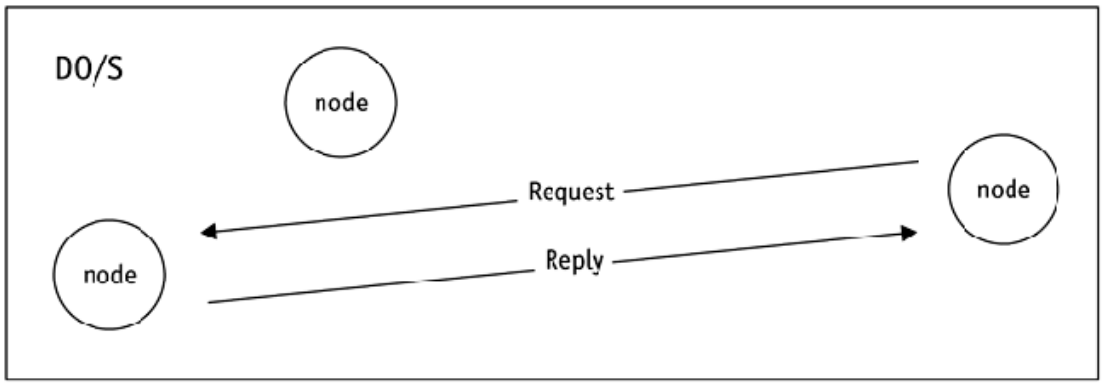
NOS DEVELOPMENT

- To blend efficiency with security
- To allows users to access hardware or software at a remote site
  - Example: Internet's telnet command
- Security is a critical function of the NOS
  - To verify every attempt to log in and have policies in place to handle unsuccessful attempts
- Throughout the telnet session, NOS handles the networking functions
- To let users transfer files from one computer to another

DISTRIBUTED OPERATING SYSTEMS

University of Bedfordshire

- DO/S
  - Need for global control of assets by OS led to the development of DO/S
  - Provide a unified environment designed to optimize operations for the network as a whole
  - Typically constructed with replicated kernel OS
  - Network and intricacies are hidden from users so they can use network as single logical system



DISTRIBUTED OPERATING SYSTEMS

- Features:
  - Resources owned by local OSs
  - Local resources managed by a global DO/S
  - Access performed by DO/S
  - Requests passed from node to node through DO/S
  - Advantage: Ability to support file copying, e- mail, and remote printing without installation of special server software on local machines

DO/S DEVELOPMENT

- Memory management
  - Memory Manager uses a kernel with a paging algorithm to track the amount of available memory
  - Memory allocation and deallocation depend on scheduling and resource-sharing schemes
  - Memory Manager accepts requests for memory from both local and global sources
  - Functions of Memory Manager in DO/S:
    - Allocates pages based on the local policy (on a local level)
    - Receives requests from the Process Manager to provide memory to new or expanding client or server processes (on a global level)

DO/S DEVELOPMENT

- Uses local resources to perform garbage collection in memory, perform compaction
- Decide which are most and least active processes
- Determine which processes to preempt to provide space for others
- To control demand, it handles requests to allocate & deallocate space based on network's usage patterns
- Automatically brings requested page into memory
- Examines the total free memory table before allocating space
- Manages virtual memory
  - Allocates and deallocates virtual memory
  - Reads and writes to virtual memory
  - Swaps virtual pages to disk
  - Locks virtual pages in memory, and protects the pages that need to be protected

DO/S DEVELOPMENT

Access Allowed	Protection
Read/write	Allows users to have full access to the page's contents, giving them the ability to read and write.
Read-only	Allows users to read the page but they're not allowed to modify it.
Execute-only	Allows users to use the page but they're not allowed to read or modify it. This means that, although a user's process can't read or write to the page, it can jump to an address within the page and start executing. This is appropriate for shared application software, editors, and compilers.
Guard-page	Used to facilitate automatic bounds-checking on stacks and other types of data structures.
No access	Prevents users from gaining access to the page. This is typically used by debugging or virus protection software to prevent a process from reading from or writing to a particular page.

DO/S DEVELOPMENT

- Process Management
  - Provides policies and mechanisms to create, delete, abort, name, rename, find, schedule, block, run, and synchronize processes, and to provide real-time priority execution if required
  - Manages the states of execution: READY, RUNNING, and WAIT
    - Each CPU in the network is required to have its own run-time kernel

DO/S DEVELOPMENT

- Kernel:
  - Each kernel assumes the role of helping the system reach its operational goals
  - Kernel's states are dependent on the global system's process scheduler and dispatcher
  - System's scheduling function has three parts:
    - Decision mode: determines which policies to use when scheduling a resource, such as Preemptive, nonpreemptive, round robin etc
    - Priority function: gives scheduling algorithms the policy that's used to assign an order to processes in the execution cycle, for example, Most time remaining (MTR), LTR
    - Arbitration rule: used to resolve conflicts between jobs of equal priority, for example, Last-in first-out (LIFO), FIFO

DO/S DEVELOPMENT

- Advances in job scheduling rely on:
  - Queuing theory
  - Statistical decision theory
  - Estimation theory:
    Maximizes system's throughput by using durations to compute and schedule optimal way to interleave process chunks
- Processes are created, located, synchronized and deleted using specific procedures

DO/S DEVELOPMENT

- Process-based DO/S
  - Network resources are managed as a large heterogeneous collection
  - Provides for process management via client/server processes synchronized and linked together through messages & ports (channels or pipes)
  - Emphasizes processes and messages and how they provide basic features essential to process management
  - Processes can be managed from single OS copy, from multiple cooperating peers, or some combination of two
  - High level of cooperation and sharing of actions & data
  - Synchronization is a key issue in network process management
  - Interrupts represented as messages sent to proper process for service

DO/S DEVELOPMENT

- Object-based DO/S
  - A system is viewed as a collection of objects
    - Example: Hardware (CPUs, memory), software (files, programs), or a combination of the two
  - Objects are viewed as abstract entities
    - Objects have a set of unchanging properties
  - The states of objects can transformed from one to another when operations are performed on the objects
  - Process management becomes object management, with processes acting as discrete objects
  - Two components of process management:
    - Kernel level and process manager

DO/S DEVELOPMENT

- The Kernel Level
  - Provides basic mechanisms for building OS by dynamically creating, managing, scheduling, synchronizing, and deleting objects
  - Maintains network's capability lists
  - Responsible for process synchronization and communication support
  - Communication between distributed objects can be in the form of shared data objects, message objects, or control interactions
  - Must have a scheduler with a consistent and robust mechanism for scheduling objects

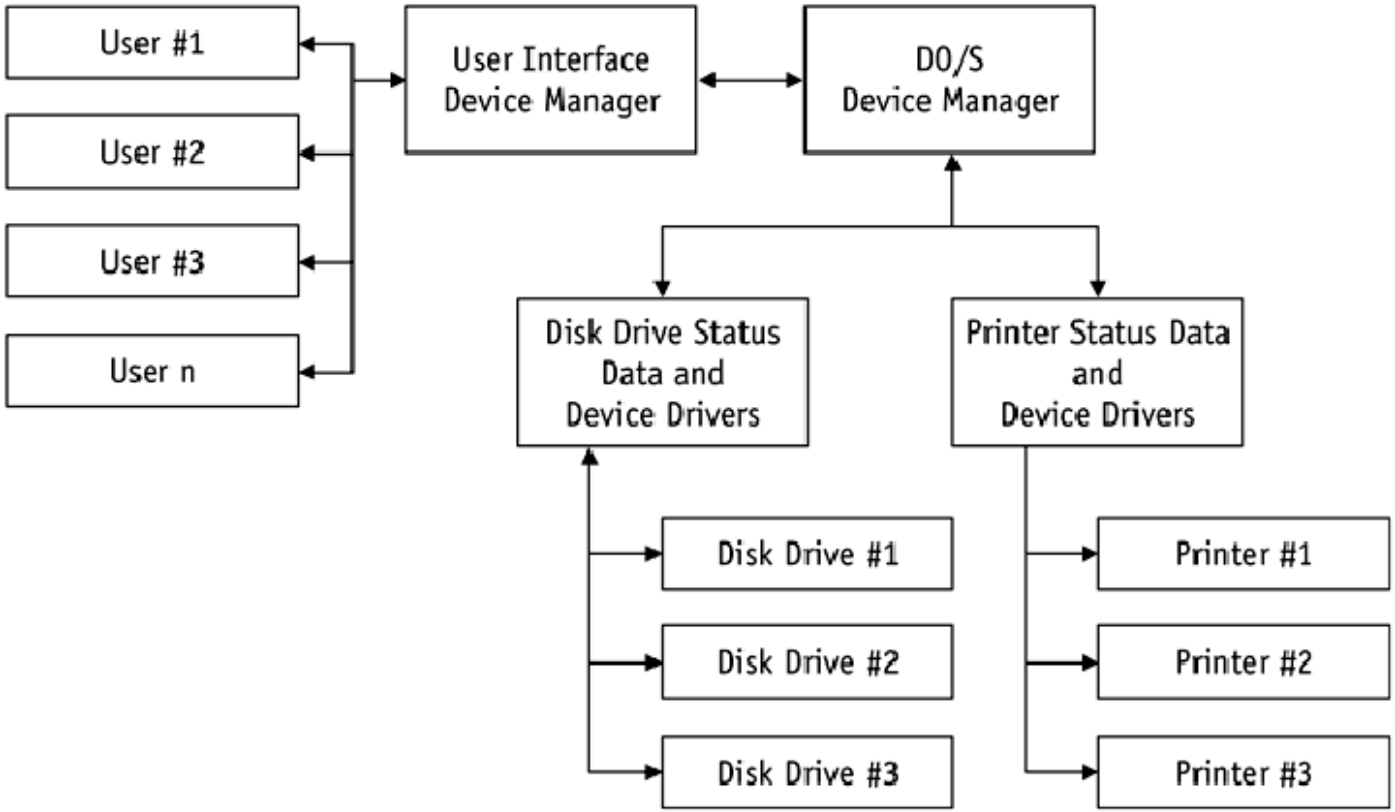
DO/S DEVELOPMENT

- The Process Manager
  - Creates its own primitives if kernel doesn't already have primitives (test and set, P and V)
  - Responsible for:
    - Creating, dispatching, and scheduling objects
    - Synchronizing operations on objects
    - Communicating among objects and deleting objects
  - Uses kernel environment to perform above tasks
  - Objects contain all of their state information

DO/S DEVELOPMENT

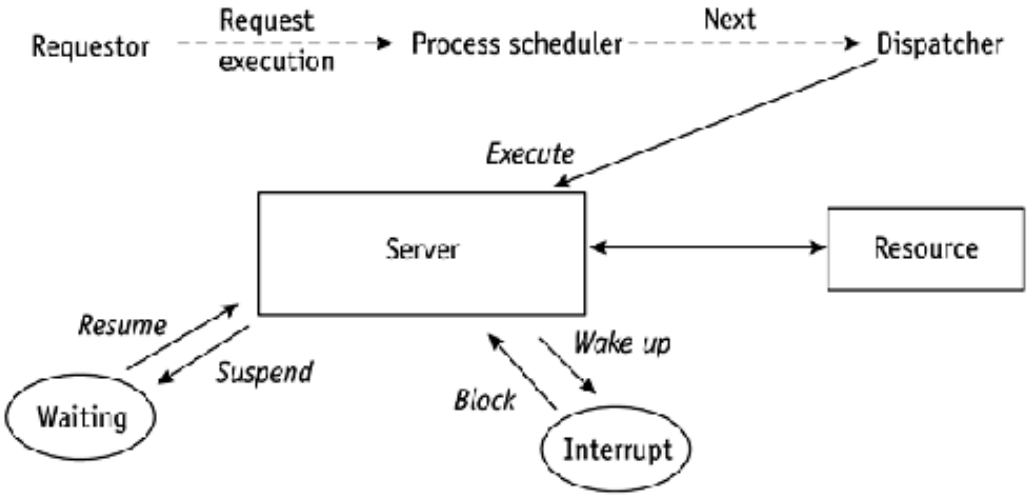
- Device Management
  - Devices must be opened, read from, written to, and closed
  - Device parameters must be initialized and status bits must be set or cleared
    - Can be done on a global, cluster, or localized basis
  - Allocates and deallocates devices to users
    - Only when a process issues OPEN and CLOSE command
  - Keeps a global accounting of each network device and its availability

DO/S DEVELOPMENT



DO/S DEVELOPMENT

- Process-Based DO/S
  - All resources in process-based DO/S are controlled by servers called "guardians" or "administrators", which are responsible for:
    - Accepting requests for service on the individual devices they control
    - Processing each request fairly
    - Providing service to the requestor, and returning to serve others



DO/S DEVELOPMENT

- Many systems have clusters of resources
  - To control these clusters as a group, most process-based systems are configured around complex server processes
- The administrator process is configured as a Device Manager and includes software needed to
  - Accept local and remote requests for service
  - Decipher their meaning, and act on them
- A server process is made up of one or more device drivers, a Device Manager, and a network server component

DO/S DEVELOPMENT

- Object-based DO/S
  - Each device is managed the same way throughout the network
  - Physical device is considered as an object, surrounded by a layer of software
  - Physical device is manipulated by a set of operations, that mobilize the device to perform its designated functions
  - Objects can be assembled to communicate and synchronize with each other
    - If local device manager can't satisfy user's request, the request is sent to another device manager

DO/S DEVELOPMENT

-
  - Users don't need to know if the network's resources are centralized or distributed
  - Device Manager object at each site needs to maintain a current directory of device objects at all sites

DO/S DEVELOPMENT

- File Management
  - To provide transparent mechanisms to find and open, read, write, close, create, and delete files
  - Subset of database managers; implemented as distributed database management systems as part of LANs
  - Tasks involve:
    - Concurrency control
    - Data redundancy
    - Location transparency and distributed directory
    - Deadlock resolution or recovery
    - Query processing

DO/S DEVELOPMENT

Desired File Function	File Manager's Action
Find and Open	It uses a master directory with information about all files stored anywhere on the system and sets up a channel to the file.
Read	It sets up a channel to the file and attempts to read it using simple file access schemes. However, a read operation won't work if the file is currently being created or modified.
Write	It sets up a channel to the file and attempts to write to it using simple file access schemes. To write to a file the requesting process must have exclusive access to it. This can be accomplished by locking the file, a technique frequently used in database systems. While a file is locked, all other requesting processes must wait until the file is unlocked before they can write to or read the file.

Desired File Function	File Manager's Action
Close	It sends a command to the remote server to unlock that file. This is typically accomplished by changing the information in the directory at the file's storage site.
Create	It creates a unique file identifier in the network's master directory and assigns space for it on a storage device.
Delete	It erases the unique file identifier in the master directory and deallocates the space reserved for it on the storage device.

DO/S DEVELOPMENT

- Concurrency Control
  - Gives the system the ability to perform concurrent reads and writes, provided these actions don't jeopardize database
  - Provides a serial execution view on a database
- Data Redundancy
  - Makes files much faster and easier to read
  - Allows a process to read the copy that's closest or easiest to access
  - Read request can be split into several different requests for a larger file
  - Advantage: Disaster recovery easy
  - Disadvantage: Task of keeping multiple copies of the same file up-to-date at all times
    - Updates to be performed at all sites

DO/S DEVELOPMENT

- Location transparency and Distributed directory:
  - Users not concerned with physical location of their files, deal with the network as a single system
  - Provided by mechanisms and directories that map logical data items to physical locations
  - Distributed directory manages transparency of data location and enhances data recovery for users and contains:
    - Definitions dealing with the physical and logical structure for the stored data
    - Policies and mechanisms for mapping between the two
    - Systemwide names of all resources and addressing mechanisms for locating and accessing them

DO/S DEVELOPMENT

- Deadlock resolution
  - Most important function is to detect and recover from a circular wait
    - Complex and difficult to detect because it involves multiple processes and multiple resources
  - Detection, prevention, avoidance, and recovery are all strategies used by a distributed system
    - To recognize circular waits, system uses directed resource graphs and looks for cycles
    - To prevent circular waits, system tries to delay the start of a transaction until it has all the resources
    - To avoid circular waits, system tries to allow execution only when it knows that the transaction can run to completion
    - To recover, system selects the best victim, kills the victim, reallocates its resources to the waiting processes

DO/S DEVELOPMENT

- Query Processing
  - Function of processing requests for information
  - Tries to increase the effectiveness of global query execution sequences, local site processing sequences, and device processing sequences
  - To ensure consistency of the entire system's scheduling scheme
    - Query processing strategy must be an integral part of the processing scheduling strategy

DO/S DEVELOPMENT

- Network Management
  - Network Manager provides policies to provide intrasite and intersite communication
  - Network Manager's responsibilities include:
    - Locate processes in the network
    - Send messages throughout the network, and track media use
    - Reliably transfer data
    - Code and decode messages, retransmit errors
    - Perform parity checking, do cyclic redundancy checks, establish redundant links
    - Acknowledge messages and replies, if necessary
    - Links processes or objects together through a port when they need to communicate with each other
    - Provides routing functions
    - Keeps statistics on network use
    - Provides mechanisms to aid process time synchronization

DO/S DEVELOPMENT

- Process-based DO/S:
  - Interprocess communication is transparent to users
  - Network Manager assumes full responsibility for:
    - Allocating ports to the processes
    - Identifying every process in the network
    - Controlling flow of messages
    - Guaranteeing transmission and acceptance of messages without errors
  - Routinely acts as interfacing mechanism for every process in the system
  - As traffic operator, it accepts and interprets each process's commands to send and receive

DO/S DEVELOPMENT

- Object-based DO/S:
  - Network Manager object makes both intermode and intramode communications among cooperative objects easy
  - User doesn't need to know the location of receiver
    - Only needs to know the receiver's name
  - Provides the message's proper routing to the receiver
  - A process can also invoke an operation that's part of its local object environment
  - Network Manager services are usually provided at the kernel level

DO/S DEVELOPMENT

Function	Purpose
Send	Allows objects to send a message with operations to any object in the system.
Receive	Warns objects of incoming communications from an object in the system.
Request	Provides a mechanism for objects to ask for particular services. For example, they can request that a message be sent.
Reply	Allows objects to do one of three things: • Respond to requests for communications from another object • Respond to a send command that they aren't prepared for • Indicate that they're ready to accept a send command; a message can now be sent knowing that the receiver is awaiting it