CS1004: Intro to CS in Java, Spring 2005

Lecture #26: OS and networks

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW4 returned today
- HW6 due next Monday
- Solutions for 4, 5 coming by end of this week
- Forgot to give a bonus yesterday, make sure I give one today
- I've received three requests for exam rescheduling; will deal with them individually this week

Black-Box Testing

- In *black-box testing*, test cases are developed without considering the internal logic
- They are based on the input and expected output
- Input can be organized into *equivalence categories*
- Two input values in the same equivalence category would produce similar results
- Therefore a good test suite will cover all equivalence categories and focus on the boundaries between categories

White-Box Testing

- *White-box testing* focuses on the internal structure of the code
- The goal is to ensure that every path through the code is tested
- Paths through the code are governed by any conditional or looping statements in a program
- A good testing effort will include both black-box and white-box tests

Segue

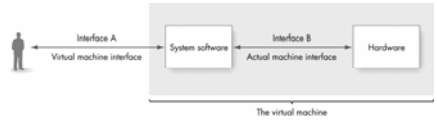
- We now know how to write code
- But how do we actually *run it* on a computer?
- A Von Neumann computer is a "naked machine"
  - Hardware without any helpful user-oriented features
  - Extremely difficult for a human to work with
- An interface between the user and the hardware is needed to make a Von Neumann computer usable
  - The *operating system*

Goals of an operating system/ "system software"

- Hide details of the underlying hardware from the user
- Present information in a way that does not require in-depth knowledge of the internal structure of the system
- Allow easy user access to the available resources
- Prevent accidental or intentional damage to hardware, programs, and data

The "Virtual Machine"

- System software
  - Acts as an intermediary between users and hardware
  - Creates a *virtual environment* for the user that hides the actual computer architecture
- Virtual machine (or virtual environment)
  - Set of services and resources created by the system software and seen by the user



Types of System Software

- System software is actually a collection of many different programs
- Operating system
  - Controls the overall operation of the computer
  - Communicates with the user
  - Determines what the user wants
  - Activates system programs, applications packages, or user programs to carry out user requests

S/G figure 6.2
Types of System Software

Types of System Software, cont'd.

- User interface
- Language services
  - Assemblers, compilers, and interpreters
  - Allow you to write programs in a high-level, user-oriented language, and then execute them
- Memory managers: allocate and retrieve memory space
- Information managers: handle the organization, storage, and retrieval of information on mass storage devices

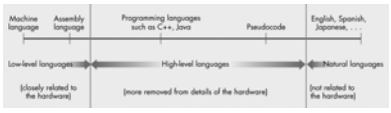
System software, cont'd.

- I/O systems: allow the use of different types of input and output devices
- Scheduler: keeps a list of programs ready to run and selects the one that will execute next
- Utilities: collections of library routines that provide services either to user or other system routines

- Given these, how do we run (machine) code on the machine?

Assembly Language

- Machine language poses a problem
  - Clumsy and difficult to change things like memory addresses
  - Makes it difficult to run a program twice, or run multiple programs
- Therefore, we use an *assembly language*
  - Designed to overcome shortcomings of machine languages
  - Create a more productive, user-oriented environment
  - Still a low-level programming language, similar to machine language



Assembly Language (continued)

- Main differences between assembly and machine language
  - Use of symbolic operation codes rather than numeric (binary) ones
  - Use of symbolic memory addresses rather than numeric (binary) ones
  - Pseudo-operations that provide useful user-oriented services such as data generation
- Various examples in the book; don't worry about them

Translation and Loading

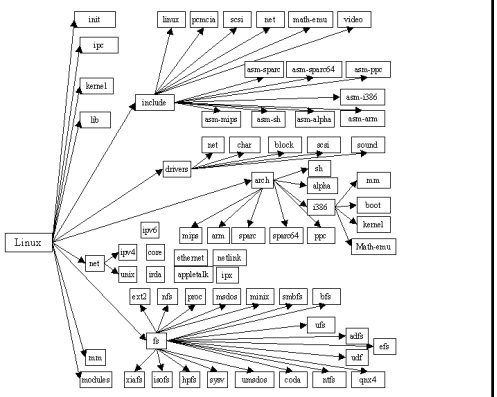
- Before a source program can be run, an assembler and a loader must be invoked
- Assembler: translates a symbolic assembly language program into machine language
- Loader: reads instructions from the object file and stores them into memory for execution
- Once the program is in memory, the operating system can *schedule* individual commands for execution

Functions of an Operating System

- Five most important responsibilities of the operating system
  - Program scheduling and activation
  - Control of access to system and files
  - Efficient resource allocation
  - Deadlock detection and error detection
  - User interface management
- The *kernel* handles the first four; the *shell* handles the fifth

The Linux kernel

- Popular learning kernel, since it's open source
- You can grab your own copy from www.kernel.org, if you want to take a look
- A Linux operating system distribution (like Red Hat) consists of the *Linux kernel* and a bunch of tools (including GNU tools)
- Here's a somewhat dated image of the directory structure of Linux source



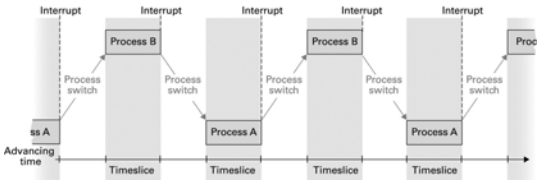
System Security And Protection

- The operating system must prevent
  - Non-authorized people from using the computer
    - User names and passwords
  - Legitimate users from accessing data or programs they are not authorized to access
    - Authorization lists

Efficient Allocation Of Resources

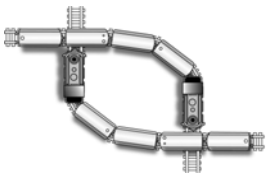
- The operating system ensures that
  - Multiple tasks of the computer may be underway at one time
  - Processor is constantly busy
    - Keeps a "queue" of programs that are ready to run
    - Whenever processor is idle, picks a job from the queue and assigns it to the processor
  - Modern OSes *timeslice* multiple processes so that no one process waits forever; gives perception of simultaneous execution

Multitasking



The Safe Use Of Resources

- Deadlock
  - Two processes are each holding a resource the other needs
  - Neither process will ever progress
- The operating system must handle deadlocks
  - Deadlock prevention
  - Deadlock recovery



The User Interface

- Operating system
  - Waits for a user command
  - If command is legal, activates and schedules the appropriate software package
- User interfaces
  - Text-oriented: command-line
  - Graphical

Generation	Approximate Dates	Major Advances
First	1945–1955	No operating system available Programmers operated the machine themselves
Second	1955–1965	Batch operating systems Improved system utilization Development of the first command language
Third	1965–1985	Multiprogrammed operating systems Time-sharing operating systems Increasing concern for protecting programs from damage by other programs Creation of privileged instructions and user instructions Interactive use of computers Increasing concern for security and access control First personal computer operating systems
Fourth	1985–present	Network operating systems Client-server computing Remote access to resources Graphical user interfaces Real-time operating systems Embedded systems
Fifth	??	Multimedia user interfaces Massively parallel operating systems Distributed computing environments

S/G Figure 6.24 (Some of the Major Advances in Operating Systems Development)

The next step

- Scale up from one machine to a multitude of machines, or a *computer network*
- Computer network
  - Set of independent computer systems connected by telecommunication links
  - Just about any kind of binary information can be exchanged – instead of writing it to disk, you send it over the wire
- Nodes, hosts, or end systems – individual computers on a network

Communication Links

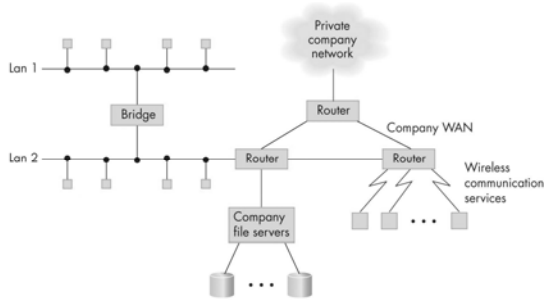
- PAN – *Personal* Area Network
  - IR, Bluetooth (10kbps-1mbps)
- LAN – *Local* Area Network
  - Ethernet (10-1000mbps)
  - WiFi (10-100mbps)
- WAN – *Wide* Area Network
  - Switched, dial-up telephone line (via *modem*; 56kbps)
  - Broadband (digital encoding, always-on)
    - Consumer: DSL, cable (256kbps-10mbps)
    - Enterprise: T1 (1.544mbps), T3 (45mbps), OC3 (150mbps), OC12 (622mbps)
    - Columbia has a 300mbps Internet and 200mbps Internet2 connection
  - Wireless (cellular/radio, microwave) – 9.6kbps to ~ 100mbps

Line Type	Speed	Time to Transmit 8 Million Bits (one compressed image)
Dial-up phone line	56 Kbps	2.4 minutes
DSL line, cable modem	2 Mbps	4 seconds
Ethernet	10 Mbps	0.8 second
Fast Ethernet	100 Mbps	0.08 second
Gigabit Ethernet	1 Gbps	0.008 second

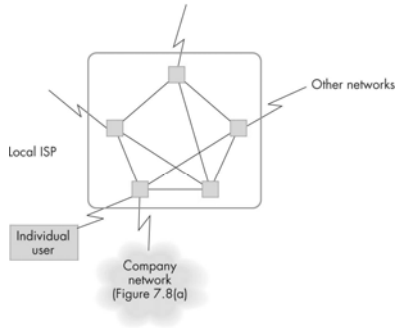
S/G Figure 7.3
Transmission Time of an Image at Different Transmission Speeds

Overall Structure of the Internet

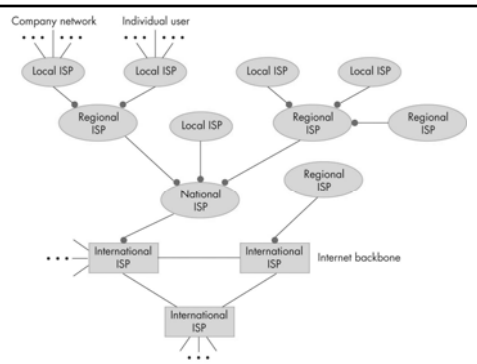
- All real-world networks, including the Internet, are a mix of LANs and WANs
- LAN commonly deployed *within* a company
  - One or more LANs connecting its local computers
  - Individual LANs interconnected into a wide-area "company network"
- Internet Service Provider (ISP) enables WAN communication
  - Provides a pathway from a specific network to other networks, or from an individual to other networks
- ISPs are hierarchical
  - Interconnect to each other in multiple layers to provide greater geographical coverage



S/G Figure 7.8(a): Structure of a Typical Company Network



S/G Figure 7.8(b): Structure of a Network Using an ISP



S/G Figure 7.8(c):
Hierarchy of Internet Service Providers

Communication Protocols

- *Protocol*: A mutually agreed upon set of rules, conventions, and agreements for the efficient and orderly exchange of information
- IP: "Internet protocol"
  - Governs the operation of the Internet (and LANs!)
  - Five "layers" (some people view it as 7)

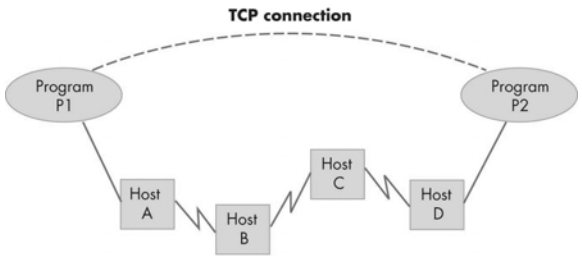
Layer	Name	Examples	
5	Application	HTTP, SMTP, FTP	
4	Transport	TCP, UDP	
3	Network	IP	
2b	Logical Link Control	PPP, Ethernet	Data Link Layer
2a	Medium Access Control	Ethernet	
1	Physical	Modem, DSL, Cable Modem	

RFCs

- Protocols are all documented as part of the Internet Engineering Task Force (IETF)
- http://www.ietf.org
- RFC == "Request For Comment"
- All of the basic protocols, like TCP, IP, etc. are all documented
- Many of them were invented by Jon Postel in 1981

The layers

- Physical: actually move the bits around
- Medium/logical link: define a physical (or dial-up) connection between computers
- Network: define how computers are named and *reached* across lots of medium/logical links
- Transport: how do we reliably exchange information across the network?
- Application: how do we tell the network what info we want to take or give?



S/G Figure 7.15: Logical View of a TCP Connection

Network addressing

- IPv4 specifies the idea of a *32-bit address* for a node
  - Theoretical maximum of 2^{32} computers, practical a lot less (IPv6 increases to 2^{128})
  - "Dotted quad" notation (e.g., 128.59.16.20)
- Subnet mask used to determine if the other computer is local or not, using bitwise AND
- DNS, or *Domain Name Service*, maps a *hostname* to an IP address

Common Application Protocols

- Needed to implement the end-user services provided by a network
- There are many application protocols, including:
  - HTTP (Hypertext Transfer Protocol)
  - SMTP (Simple Mail Transfer Protocol)
  - POP3 (Post Office Protocol v3)
  - IMAP (Internet Mail Access Protocol)
  - FTP (File Transport Protocol)
  - SSH (Secure SHell)
- All of these use a TCP *port* to offer service

Application-Layer Addressing

- Either just a hostname or a URL (Uniform Resource Locator)
  - The latter lets you specify both the hostname and an item (e.g., webpage) on that host
  - Form **`protocol://host address/page`**
  - The most common Web page format is hypertext information, accessed using the **`HTTP`** protocol
  - Most browsers also support **`FTP`** URLs, however

A Brief History of the Internet

- August 1962: first proposal for building a global computer network (J. C. R. Licklider of MIT)
- ARPANET built by the Advanced Research Projects Agency (ARPA) in the 1960s
  - Grew quickly during the early 1970s
- NSFNet: A national network built by the National Science Foundation (NSF)
- October 24, 1995: Formal acceptance of the term "Internet"
- Internet service providers start offering Internet access once provided by the ARPANET and NSFNet

History of the WWW

- Development completed in May 1991
- Designed and built by Tim Berners-Lee
- Hypertext: a collection of documents interconnected by pointers called links
- HTML: common format for creating hypertext documents
  - Don't confuse HTTP and HTML!

Internet Security

- Encompasses various problems
- How do we *encrypt* traffic so people in the middle can't read it?
- How do we design software so that it doesn't crash (*denial-of-service*) or get hacked (*vulnerabilities*)?
- Turns out these are very hard problems to solve
- The `tcpdump` program is insightful…

Next time

- Computing theory
  - What's a computer?
- Artificial intelligence
  - Is it the future?
- Wrapup