50.005 Computer System Engineering

Tey Siew Wen

15 Oct 2019

Contents

1 Networking

1.1 The Internet - A nuts & bolts view

1. Hosts (End Systems) running network apps
2. Communication Links
3. Packet Switches

1.2 The Packet

- Packet = Header + Payload
  - Header data is used by networking hardware to direct the packet to its destination
  - Payload is extracted and used by application software.
- Usually, the mode of transmission of packets is: PC → Switch → Router → Modem → Modem → Switch → Servers
- When the packets arrive at the router, the packet arrival rate to link temporarily exceeds the output link capacity. Hence they would queue in the router buffers, causing a delay.
  - If there are no free buffers for them to get into queue, then the packets will be dropped. This is known as packet loss.

1.2.1 Single Router Packet Delay Sources

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

1. Nodal Processing (d_{nodal})
   - Examine packet header
   - Check for bit errors
   - Determine output link (destination IP address in packet header)
2. Queueing (d_{queue})
   - Waiting time for packet to get to front of queue for output link
   - Depends on congestion level (how much other users are also sending data)
3. Transmission (d_{trans})
   - Time to push the whole packet (all the bits) from router to link

$$d_{trans} = \frac{\mathrm{L}}{\mathrm{R}} = \frac{\text{packet length (bits)}}{\text{link bandwith (bps)}}$$

4. Propagation (d_{prop})
   - Time for packet to move from beginning to end of the link, $d_{prop} = \frac{\text{length of physical link (m)}}{\text{propagation speed in medium}}$
   - Propagation speed in medium is $\approx 2 \times 10^8$ m/s ($\frac{2}{3}$ speed of light in vacuum)

$$d_{prop} = \frac{\mathrm{d}}{\mathrm{s}}\mathrm{m/s}$$

1.2.2 End-to-end Delay for many hosts

- If not given space-time diagram

$$d_{end} = \text{no. of hops} \times \text{nodal delay}$$

- If given space-time diagram, we can approximate d_{end} from the graph.

Calculating Throughput
Throughput: File Receival rate (bits/s)

- Instantaneous Throughput : at any instant of time
- Average Throughput : over a longer period
  - for a file consisting of F bits
  - and the transfer takes T seconds for Host B to receive all F bits, then the calculation for average throughput is as shown below.

$$\frac{\mathrm{F}}{\mathrm{T}} \text{ bits/s}$$

- Effective throughput is determined by the slowest bandwidth in the route, the ***bottleneck link***.

1.2.3 Average Link Utilization

Also known as *Traffic Intensity*, given by the formula

$$\text{Traffic Intensity} = \frac{\mathrm{La}}{\mathrm{R}}$$

where:

- L is the packet length (bits)
- a = average packet arrival rate
- R = link bandwidth (bits/s)

If $\frac{\mathrm{La}}{\mathrm{R}} > 1$,

- the average rate at which bits arrive at the queue exceeds the rate at which the bits can be transmitted from the queue.
- the queue will tend to increase without bound and the queuing delay will approach infinity.
- so system must be designed that traffic intensity should be ≤ 1.

If $\frac{\mathrm{La}}{\mathrm{R}} \leq 1$,

- If packets arrive individually periodically, then every packet will arrive at an empty queue and there will be no queueing delay
- If the packets arrive periodically in bursts (bursty data), then the nth packet transmitted will have a queueing delay of

$$(n-1) \times \frac{\mathrm{L}}{\mathrm{R}} \text{ seconds}$$

Do note that the cases stated here are more academic examples than reality. In reality, typically, the arrival process to a queue is random; that is, the arrivals do not follow any pattern and the packets are spaced apart by random amounts of time. $\frac{La}{R}$ is not usually sufficient to fully characterize the queueing delay statistics.

1.2.4 Modems

- Carries out Modulation & Demodulation
- Modulation: A process to add text/img/vid/sound etc into a carrier signal. Demodulation is vice versa.
    - Types of broadbands:
        - Dial-up: audio freq $\leftrightarrow$ digital
        - DSL: copper telephone lines
        - Coaxial Cables: faster than DSL
        - Fibre Optic Cables: Light pulse

      s Wifi signals: EM waves
- But for modern modems, there is no longer a need for conversion of signals from audio freq $\leftrightarrow$ digital. Their primary role is for converting electrical signals over long connections, into some signal form that is understood by router & computers.

1.3 The Internet - A service view

1. Protocols for communication
    - Define format, order of messages sent & received
    - Appropriate actions to be taken on message transmission and receipts.
    - Set by RFC (Request for Comment) documents, published by IETF (Internet Engineering Task Force).
    - e.g. TCP connection request $\rightarrow$ connection response $\rightarrow$ Get request to a server URI $\rightarrow$ receive packet for a file
2. Network API for infrastructures to **provide services to user applications**
3. **Sharing of info among many devices** via circuit switching & packet switching
4. **Scalability of systems** via the Hierarchy of ISPs

1.4 Internet Protocol Stack (Layering)

1. Application: messages
2. Transport: segments
3. Network: datagrams
4. Link: frames
5. Physical: bits

The stack is separated into 5 layers so that we can **avoid complex interactions between components**.

- The packet's header & payload is relative to the layer.
- E.g. Transport layer's header is network layer's payload.
- Number of possible interactions with N layers is N-1.
- Without layers, there are N^2 possible interactions instead.
    - Difficult to debug
    - Might lead to emergent & undesirable behaviour

1.4.1 Internet Control Message Protocol (ICMP)

ICMP is used by hosts and routers to communicate network-layer information to each other. ICMP is often considered part of IP but architecturally it lies just above IP, as **ICMP messages are carried inside IP datagrams**. (**is also carried as IP payload**) *Similar to TCP/UDP segments being carried as IP payload.*

When a host receives an IP datagram with ICMP specified as the upper-layer protocol, it demultiplexes the datagram's contents to ICMP.

ICMP messages have a type and a code field, and contain the header and the first 8 bytes of the IP datagram that caused the ICMP message to be generated in the first place.The sender can then determine the datagram that caused the error.

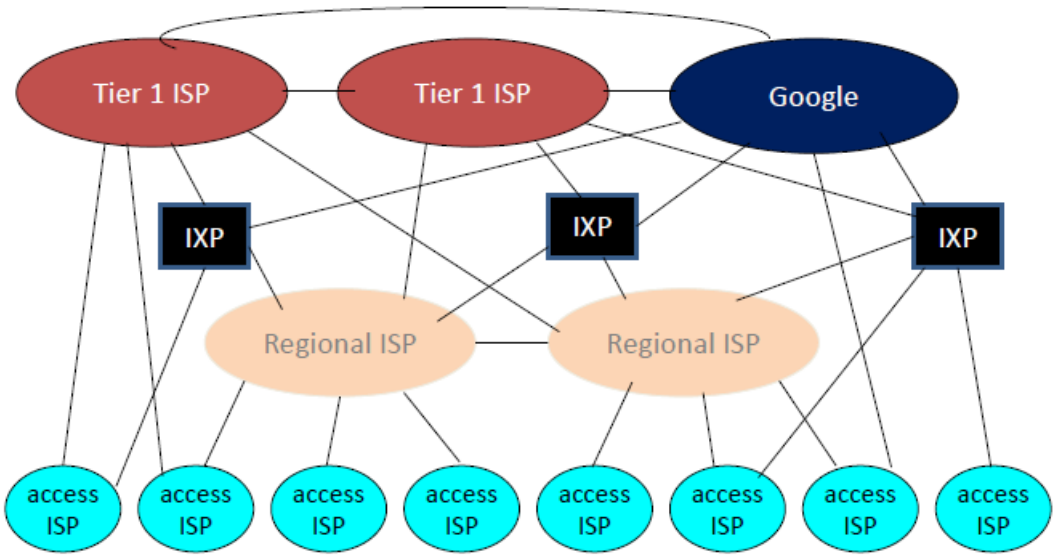
Types of ICMP Messages:

- Source Quench: perform congestion control - congested routers send this message to a host to force that host to reduce its transmission rate
- Port Unreachable
- Ping Echo request/reply

1.4.2 Internet Service Provider (ISP)

An organization that provides services for accessing and using the internet. It is also a network of packet switches & comm links.

The hierarchy of ISPs:



- IXP connects the various global ISPs together.
- Then the small individual access ISP subscribe to commercial global Tier 1 ISPs.
- Google here is not a global ISP, but rather a Content Delivery Network (CDN), which is a private network that carries data to & from Google servers only, but connects to everyone.

1.5 Sharing of Information via Switching

1.5.1 Circuit Switching

Circuit switching can performed via Time Domain Multiplexing (TDM) & Frequency Domain Multiplexing (FDM).

Multiplexing is the process of combining multiple signals into one, in such a manner that each individual signal can be retrieved at the destination. These multiple signals are also travelling in the same channel.

TDM	FDM
Bandwidth = bits/second	Bandwidth = frequency range
Divides and allocates certain time periods to each channel in an alternating manner.	Divides the channel into two or more frequency ranges that do not overlap.
Greater flexibility as it can dynamically allocate more time periods to the signals that need more of the bandwidth, while reducing the time periods to those signals that do not need it.	Less flexible since it preallocates use of transmission link regardless of demand. Cannot dynamically change the width of the allocated frequency. So there could be allocated but unneeded link time going unused.
Each signal uses all of the bandwidth some of the time.	Each signal uses a small portion of the bandwidth all of the time.

While there are advantages & disadvantages to each method, FDM and TDM are *often used in tandem*, to create even more channels in a given frequency range.

A common practice for telecoms to allow a huge number of users to use a certain frequency band:

- **divide the channel with FDM**, so that you have a dedicated channel with a smaller frequency range.
- Each of the FDM channels is then **occupied by multiple channels that are multiplexed using TDM**.

Circuit switching allows a **fixed, dedicated fraction of the link for each user**. Better for continuous, streamline usage.

1.5.2 Packet Switching

On the other hand, Packet switching occupy link on demand, **better for bursty data**. e.g. A link shared among many users, but only a low proportion of users are active at the same time.

2 Operating System (OS)

Resource allocation

- Policy: Manages all resources
- Decides between conflicting requests (efficiency and fairness)

Control program

- Mechanism: Controls execution of programs (prevent errors and misuse)
- Modern operating system is interrupt driven. The operating system preserves the state of the CPU by storing registers and the program counter.

2.1 Kernel

- runs with special privileges – it can do what normal user code cannot do (e.g., access to special instructions & special memory regions)
- Hardware support required – CPU has (at least) dual mode operation: user (unprivileged) mode vs. kernel (privileged) mode
- System calls let (unprivileged, untrusted) user programs access (privileged, trusted) kernel services
- System call changes mode to kernel, return from call (via return-from-trap or RETT instruction) resets it to user

Kernel type	Idea	Benefits	Detriments
Monolithic	Simple layered structure	?	?
Microkernel	Move services into user space	Easier development, more reliable	Performance overhead of communication between user and kernel space
Hybrid	?	?	?

Dual-mode Operation

- Software error or request creates exception or trap (software interrupt)
- Dual-mode operation: User mode and (privileged) kernel mode
- User-mode program is usually limited to its own address space so that it cannot access or modify other running programs or the operating system itself, and is usually prevented from directly manipulating hardware devices
- Mode bit provided by hardware provides ability to distinguish when system is running user code or kernel code
- System program is in user mode!
- System call: API to services provided by OS kernel
- System calls are made available by the operating system to provide well-defined, safe implementations for operations that require access to e.g. hardware devices

OS Services

- Communications between processes (IPC)
- Error and misuse detection: Owners of information stored in a multiuser or networked computer system may want to control use of that information; concurrent processes should not interfere (logically) with each other

2.1.1 Computer-system operation

- One or more CPUs, device controllers connect through common bus providing access to shared memory and other resources

2.2 Processes

- Process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity which requires resources to complete a task
- Process contains a PC, stack, data section and allocated memory in heap
- Process defines a private address space
- Process couples abstractions such as concurrency and protection
- Process identified and managed via `int pid` process identifier
- Single-threaded process has one program counter specifying location of next instruction to execute, while multi-threaded process has one program counter per thread

Process State	Explanation
new	being created
running	executing on CPU
waiting/blocked	wait for I/O or event
ready	queue for execution
terminated/stop	finished execution

Process control block (PCB) is a data structure that contains all information about active processes.

- Job queue: set of all processes
- Ready queue: processes in RAM that are ready to execute
- Device queue: waiting for I/O

2.2.1 Process Management

- Cooperating processes may affect each other, mainly through sharing data
- Two basic problems: **mutual exclusion** and **condition synchronization**
- Two basic models of IPC: Shared memory and Message passing
- Socket as an example of message passing
- Thread: line of execution, has registers + stack

- Many threads can run within a process, sharing the address space
- Kernel Thread
  - known to OS and scheduled by kernel
  - represented within a kernel-specific data structure
- User Thread
  - not known to OS kernel
  - scheduled by user-mode thread scheduler (such as `pthread`)
  - must belong to a process

Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity.

Distribution Systems

Single-core

- Concurrency: CPU performs context switching so frequently that it gives the illusion of multi-tasking and users can interact with each job while it is running

Multi-processor

- Multiprocessing overheads like context switching, IPC, or inter-process synchronization. Hence the speed-up in reality will never be the theoretical maximum of `N`
- Context switching in user threads us faster because it doesn't have to trap to the kernel (no system call overhead)

2.2.2 Producer-Consumer Problem

- producer puts a new item of work into a shared buffer
- consumer takes an item of work from the buffer
- buffer can store a fixed number of items (bounded buffer)

2.2.3 Critical Section Problem

- Mutual Exclusion - If process is executing in its critical section (CS), then no other processes can be executing in their critical sections
- Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely
- Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

Peterson's Solution

- require busy waiting, generally bad for uniprocessors
- busy waiting (spinlock) means executing instructions (polling the lock) without doing anything useful

Synchronization Hardware

- Many modern machines provide special atomic hardware instructions
- Test original value of memory word and set its value
- Swap two memory words

Semaphore

- Semaphore is a high-level synchronization primitive
- No busy waiting
- Two atomic operations on the `int` **state variable**
- `acquire()` ++ and `release()`
- These are critical sections, hence require hardware instructions or use of spinlock
- Binary (0/1) semaphore or Counting semaphore (`int`)

Java Synchronization

- Each Java object has an associated binary lock
- Lock is acquired/released by invoking a synchronized method
- Mutual exclusion is guaranteed for this object's method – at most only one thread can be inside it at any time
- Threads waiting to acquire the object lock are placed in the **entry set** for the object lock
- A synchronized method of an object can successfully call another synchronized method of the same object although both synchronized methods are guarded by the same binary lock

2.2.4 Deadlocks

- Each process utilizes a resource as follows: request, use, release
- **Circular wait**: set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set
- Single lane bridge crossing example: if a deadlock occurs, it can be resolved if one car backs up (pre-empt resources and rollback)
- Cycle in resource allocation graph: if resource only has one instance, then **deadlock**, else, possibility of deadlock

Necessary, but not sufficient, for deadlock: all four conditions hold **simultaneously**:

Condition	Elaboration
Mutual exclusion	Only one process can use a resource
Hold and wait	One process holds a resource while waiting for other resources
No preemption	Resources are only release voluntarily
Circular wait	There exists a set of waiting processes such that one is waiting for another that is held by yet another process

Pre-emptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Non-pre-emptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

Deadlock avoidance

- Before granting a resource request (even if request is valid and the requested resources are now available), check that the request will leave system in a **safe state**
- By right not possible for system to enter an unsafe state, but if it does there is possibility of deadlock
- Requires advance knowledge of future resource needs (e.g. Banker's algorithm)

Deadlock detection and recovery

- Actively detect deadlock after the fact, then recover from it (e.g. pre-empting held resources and rolling back processes)

Deadlock prevention

- Impose conditions on resource requests to ensure that a valid request can never cause the system to enter a deadlock state by design without having to check the system's detailed resource allocation state
    - basically disallow any of the four necessary conditions for deadlock
- No resource hold and wait
    - Must get all resources before process execution
    - Only allow request for resources if the process has none
- Disallow circular wait
- Enforces pre-emption
    - Process must release all resources its already holding if it needs another resources that require wait
    - Restart process, must wait for every resources again

2.2.5 Banker's Algorithm

Data structure

- Available: 1D array
- Max: `n*m` 2D array
- Allocation: `n*m` 2D array
- Need: `n*m` 2D array

Safety Algorithm

1. Let Work and Finish be vectors of length `m` and `n`, respectively
   - `n` = number of processes, `m` = number of resource types
   - `work = available`
   - `finish[i] false for i in 0 .. n-1`
2. Find `i` such that
   - `finish[i] == false`
   - `need[i]` $\leq$ work
   - else: skip to Step 4
3. `work = work + allocation[i]; finish[i] = true; GOTO step 2`
4. if `finish[i] == true` for all `i`, then only system is in a safe state

2.2.6 Interrupt Handling

- Interrupt handling is done in **kernel mode**, by the service routine
- Interrupts transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction. Incoming interrupts are disabled while another interrupt (of same or higher priority) is being processed to prevent a lost interrupt or reentrancy problems
- **Context switching**: need to save/restore into PCB all the CPU hardware state that keeps track of the process's execution, e.g., registers like SP, PC, PSR, etc. to/from main memory
- OS Scheduler interrupts periodically running processes give control to the OS, which can then force rescheduling even if the process doesn't give up the CPU voluntarily
- **trap** is software-generated interrupt caused either by an error or a user request to allow a user program to invoke an OS function (system call) and run it in kernel mode
- Polling interrupt checks each I/O devices one by oneon whether this is the correct device that requests for I/O interrupt
- Vectored interrupt identifies the device that sent the request

2.3 Storage Hierarchy

System	Cost	Speed	Capacity
Registers	highest	highest	< KB
L(X) Cache			> MB
RAM			> GB
NVRAM			> GB
Disk	lowest	slowest	> TB

- Caching: Information in use copied from slower to faster storage temporarily
- If processes don't fit in memory, swapping moves them in and out to run

2.3.1 File System

UNIX cmd	function
`cat`	display file contents
`rm`	remove file
`mkdir/rmdir`	make/remove dir
`cd`	change current dir
`find`	traverse and find
`ln source link`	create hard link
`ln -s source link`	create symbolic link

Files

- Files are stored on persistent (non-volatile) storage, and in UNIX are memory cached for performance:
  - regular files
  - special files:
    - folders
    - `.` refers to current directory
    - `..` refers to parent directory
  - file links:
    - symbolic link (shortcut link)
    - hard link (both path point to the same file)
- file is an uninterpreted sequence of words/bytes
- can be viewed as an abstract data type, like a class of objects in the OOP sense
  - state: file data and metadata (file attributes)
  - interface (set of methods e.g. open/rw/del to be used)
- format is interpreted by some application (or OS kernel in case of special files)
- UNIX doesn't allow hard links to directories. This restriction ensures that we can't form cycles with hard links

File Descriptor Tables

- 2 file descriptors `fd1` and `fd2` reference same open file entry, they share usage (e.g., cp) of the file – read/write through fd1 will advance cp seen by fd2

Filesystem Structure

- Directory Structure is meta-data that organizes files in a structured name space
- Both the directory structure and the files themselves reside on disk (and cached in memory)
- There are three types of directory structures:
  i. Tree - no links, one location in filesystem for every inode location
  ii. Acyclic graph (UNIX) - possible to have multiple links/path to one inode location
  iii. General graph - possible to have cycles of links