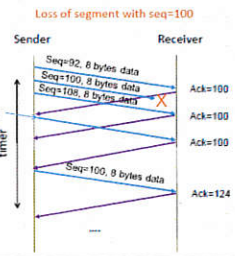
Performance of TCP can be improved by retransmitting lost segments prior to timeout.

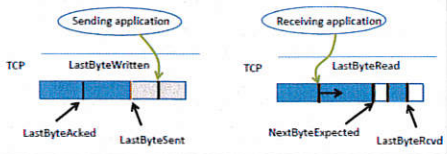


Retransmit segment if same ack received 3 times.

TCP Flow Control: Overview

•Receiver may not process data as fast as sender transmits data:
–receiver's buffer may fill up
–receiver has to drop data that waste bandwidth in network traffic
•Receiver needs a mechanism to tell the sender how much buffer space is still available.
–helps to synchronize receiver's data processing with sender
–avoids receiver dropping data
–may also tell sender overall buffer space
•The sender has a send buffer
–Application puts data into the send buffer
–TCP transmits data from the send buffer
•The receiver has a corresponding receive buffer
–Application reads data from the receive buffer
–TCP places data into the receive buffer

Rules:
LastByteAcked <= LastByteSent
LastByteSent <= LastByteWritten
LastByteRead < NextByteExpected
NextByteExpected <= LastByteRcvd + 1



TCP Congestion Control

•Use Congestion Window to limit rate of sender:
LastByteSent - LastByteAcked <= CongestionWindow
amount of unacknowledged data
•Congestion Window is increased and decreased depending on traffic.
–linear increase: increase congestion window by 1 MSS (maximum segment size) every RTT
–exponential increase: double congestion window every time ACK received, start with 1 MSS
–multiplicative decrease: cut congestion window in half after loss occurs

TCP Congestion Control Events

Sender transmits data at a rate

$$R = \frac{Congestion\ Window}{RTT} * Bps$$

•Sender receives Acks for sent packets.
-Assume no congestion. Increase size of congestion window.
•Sender times out or receives 3 or more duplicate AcksS:
-Congestion is inferred: Decrease size of congestion window.

- 4000 byte datagram (3980 bytes in payload)
- MTU = 1500 bytes

Original datagram:

....	length= 4000	ID= x	fragflag= 0	offset= 0	....

Fragmented datagrams:

1480 bytes in data field	....	length= 1500	ID= x	fragflag= 1	offset= 0	....
1480 bytes in data field	....	length= 1500	ID= x	fragflag= 1	offset= 185	....
1020 bytes in data field	....	length= 1040	ID= x	fragflag= 0	offset= 370	....

multiple of 8

IP Address Examples

•Addresses are typically written using the dotted quad notation
–Example: 143.88.1.208
•Addresses in class A networks range from:
0.0.0.0 to 127.255.255.255
•Addresses in class B networks range from:
128.0.0.0 to 191.255.255.255
•Addresses in class C networks range from:
192.0.0.0 to 223.255.255.255
•Some IP addresses are reserved for a special class D used for multicasting:
224.0.0.0 to 239.255.255.255
•The class A address 127.0.0.0 is used for a special **loopback** network
In particular, 127.0.0.1 always refers to the current system (also referred to as **localhost**)

TCP: Establishing Connection (cont.)

3-way handshaking:
Step 1: client host sends TCP SYN segment to server
–specifies initial sequence number
–no data
Step 2: server host receives SYN, replies with SYNACK segment
–server allocates buffers
–specifies server initial sequence number
Step 3: client receives SYNACK, replies with ACK segment, which may contain data

TCP Flow Control – Receiver Side

•On receiver's side, TCP must enforce:
LastByteRcvd - LastByteRead <= MaxRcvBufferSize
•Receiver computes the free buffer space and advertises the space to the sender:
AdvertisedWindow = MaxRcvBufferSize – (LastByteRcvd - LastByteRead)
•This AdvertisedWindow parameter is sent by the receiver in each packet sent back to the sender.

TCP Flow Control – Sender Side

•On the sender's side, TCP must enforce:
LastByteSent - LastByteAcked <= AdvertisedWindow
amount of unacknowledged data
•Sender computes an effective window which determines how much data can be sent:
EffectiveWindow = AdvertisedWindow – (LastByteSent - LastByteAcked)
•If the sender is faster than the receiver, the *AdvertisedWindow* gradually shrinks limiting the speed of the sender.

Network Layer – What does it do?

COP4635, Spring 2013
• transport segments from sending to receiving host
• on sending side, encapsulates segments into datagrams,
• on receiving side, delivers segments to transport layer
• network layer protocols in *every* host, router
• router examines header fields in all IP datagrams passing through it

Network Layer Functions

•Forwarding: move packets from router's input to appropriate router output.
Analogy: Forwarding: process of going through an intersection an selecting the street to follow next.
•Routing: determine route taken by packets from source to destinations.
Analogy: Routing: process of selecting a route for a trip from place A to place B.
routing algorithms take care of determining routes

Internet Protocol (IP) – What is it?

•One of the core protocols in the TCP/IP protocol suite.
•Used throughout the current Internet.
•Operates at the network layer (above the media access layer)
•Handles routing of packets from any host on any network to any other host on any other network (provided a path exists)
•Two major goals:
–Handle a collection of **heterogeneous** networks
–Handle a very large number of networks (scalability)

IP Network Address Classes (Classful Addressing)

•Three classes of addresses available (for different sized networks)
–Class A
•7 bits for network address,
•24 bits for host address
•Allowed up to16.7 million hosts
•Only a maximum of 128 networks
–Class B
•14 bits for network address
•16 bits for host address
•Allows up to 65534 hosts and 16384 such networks
–Class C
•21 bits for network address
•8 bits for host address
•Allows up to 2 million networks each up to 254 hosts

DHCP Client-Server Message Exchange

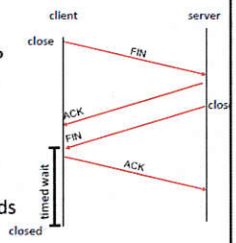
1.Host broadcasts "DHCP discover" message.
•Client sends a UDP packet to port 67 on the network.
2.DHCP server responds with "DHCP offer" message.
•DHCP server listens for incoming discovery messages from host.
•DHCP server looks up available IP addresses in a table
3.Host requests IP address: "DHCP request" message.
•Host selects offer from among DHCP servers responding to its initial inquiry.
•Host sends request echoing back the suggested configuration parameters.
4.DHCP server sends address: "DHCP ack" message.
•Server sends acknowledgment to host.
•Server updates its table of available IP addresses in network to note allocation.

TCP: Closing a Connection

- Client closes socket:

```
close(sockfd);
```

- Step 1: client sends TCP FIN control segment to server
- Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.



TCP Congestion Control Summary

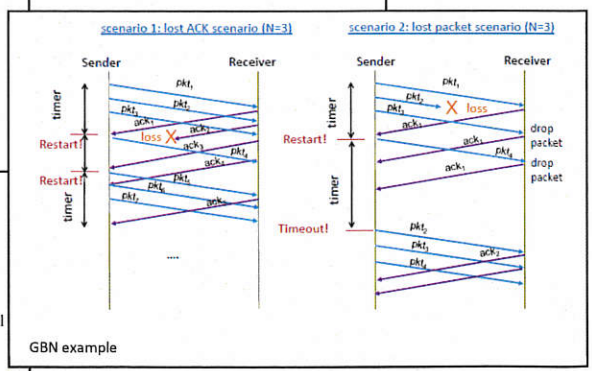
•*CongestionWindow* is below threshold:
–grow window exponentially (slow-start).
•*CongestionWindow* is above threshold:
–grow window linearly (congestion-avoidance).
•When triple duplicate Ack occurs:
–set *CongestionWindow* to threshold [Reno].
–set *CongestionWindow* to 1 MSS [Tahoe].
•When timeout occurs:
–set threshold to ½ CongestionWindow.
–set *CongestionWindow* to 1 MSS.

Connection Oriented & Connection-Less Services

•Two network services exist:
–Connection-oriented services (Virtual Circuit Network)
–Connection-less services (Datagram Network)
•Virtual Circuit Network
–Examples: Asynchronous Transfer Mode (ATM) networks, frame relay, telephone networks
–Path is established from source to destination..
–Packets travels on established path.
•Datagram Network
–Examples: Internet
–No connection needed to transmit packets (datagrams).
–Packets may travel on separate path from source to destination.

Internet Protocol Features

•Provides a uniform addressing scheme and uniquely identifies every host connected to the internetwork
•Provides a datagram (connectionless) delivery service to protocols above the IP layer
•Handles all routing of packets through the internetwork
–A host can send datagrams to any other host on any network
•Datagram delivery service is a **best-effort** or **unreliable** service
–Datagrams may be lost
–Datagrams may be delivered out of order
–Datagrams may be duplicated
•Providing the simple, best-effort delivery service enables IP to work on many different network types and topologies



GBN example

Comparison VC and Datagram Network

Datagram Network (Internet)

•end system is "smart" device
–can perform error checking
–can adjust to traffic
–can recover from errors
–network is simple
•reliable & unreliable service
•different link types and device characteristics
–different hardware components form network

VC Network (Telephone)

•end system is "dumb" device
–simple conversion only
–complexity is with network
•reliable service
–strict timing limits
–must guarantee delivery
•uniform link types and switch devices
–same characteristic of hardware components building network

Problems with Class-based Addressing

•A two level hierarchy is not sufficient (other enhancements do exist)
–Class A supports many hosts and few networks.
–Class B supports fewer hosts and more networks.
–Class C supports many network addresses and few hosts.
•Classes will result in the address space being exhausted before all addresses are allocated.
–any organization with more than 255 hosts requires a class B.
–Internet runs out of class B addresses.
–but IPv6 (128 bits instead of 32) fixes this problem, widening the address space significantly

IP Subnet Mask

• Without subnetting, a class-based IP address could easily be separated into network and host parts.
• With subnetting, a subnet mask distinguishes between the host and network portions of an IP address.
• Typically, the portion of the IP address that represents the network id has corresponding 1s in the subnet mask

•Patch Panel
–serves as junction points between cables coming from switch/hub and wall
–no signal strengthening or filtering

•Hubs
–distribute signal across multiple connected ports
–regenerates signal to original signal level (signal strengthening)
–filters any noise
–operates at physical layer in OSI reference model

switches
•similar to hub, strengthens signal and filters noise
•selects destination port instead of broadcasting on all ports
–reduces overall traffic on network compared to hub
–may broadcast on all ports to learn destination port
•operates at data link layer in OSI reference model (layer 2)

Network Devices: Routers
•similar to switch, strengthen and filters signal
•connects different networks to build larger networks
•directs traffic: routing tables in each router determines the flow of network traffic
•operates at network layer in OSI reference model (layer 3)
•may bridge different types of networks

TCP Handshake
•Done by connect()/accept()
•Client
–Sends "CONN REQ" to server socket
•Server
–Accepts request
–Creates new socket (dynamic port binding)
–Randomly generates 1st S->C sequence number
–Sends new socket & 1st S->C sequence number to client ("CONN ACK")
•Client
–Accepts message
–Randomly generates 1st C->S sequence number
–Sends 1st C->S sequence number to server ("CONNECT")

Socket Types
TCP
–Connection-oriented, reliable
-Also called *stream*

UDP
–Connectionless, unreliable

Also called *datagram*

RAW
–Bypasses niceties of above

–Functions at IP level

sending data through socket
1.Create a socket
2.Get information about *dest* host.
3.Fill in *dest addr* structure.
4.Connect to remote host.
5.Send /receive data to *dest.*
6.Close connection.

Sockets
–Communication endpoints for processes
–Can be on different machines
–Can be used over network

Ports
–Identify process on machine
–Must be known before receiving message
–Two ways of establishing

FTP via UDP
Datagram is limited in size (i.e., 10K)
How can we transfer large file (i.e., 100K)?
UDP Sender
–Partition file into 10 datagrams
–Send datagram 1, then datagram 2, …
UDP Receiver
–Receive all datagrams
–Reconstruct file from datagrams
FTP/UDP Problem
Some datagrams could be lost
Datagrams could arrive out of order
Or Both

UDP Solution?
Add a number to each datagram
–Sequence number (start at 0)
–Receiver can reorder datagrams
–Receiver can ask for lost datagram to be resent
–Receiver can send acknowledgement (ACK) of receipt for datagrams

Application Layer Protocol
•May be proprietary or public:
–proprietary examples: Skype
–public examples: HTTP, SMTP
•Specifies message syntax and semantic:
–syntax defines the specific fields in the message and how they are delineated
–semantic defines the meaning of the individual fields
•Specifies rules for what messages need to be send and how to respond to a message.

TCP Overview
Designed for client/server model
Server is at well-known place
Usually only one server
Clients (multiple) are anywhere
Client contacts server for *service*
Client and server communicate to facilitate service
Client and server disconnect
Server continues to wait for next client contact
Client and Server have "1-to-1" connection
No other process can use connection
Connection is bidirectional
–Client can send to server

–Server can send to client
Requires buffers, segments, handshakes, …

Packet-switched Networks
• No path is setup prior to data transmission.
• Packets between same source-destination pair may take different paths.
• Network uses best-effort to deliver packets.
• Routers use store-and-forward to deliver packets into network.
– each packet uses full link-bandwidth

Circuit-switched Networks
•Sets up a dedicated path in the network prior to data transmission
–dedicated path must be torn down after end of data transmission, to relinquish network resources
•Communication link must be shared by other senders
–sharing can be done by multiplexing signals onto same link
–bandwidth is divided by shared simultaneous connections

Types of Transmission Delays
1.Node Processing Delay
•Time it takes for the node to:
–checks for bit errors
–determines output link using routing table
2.Queuing Delay
•Time it takes for packet to wait in queue until it can be sent on target link.
3.Transmission delay: **L/R**
• Time it takes to send bits into link
–depends on bandwidth (bps) of link (R) and packet length (L)
4.Propagation Delay: **d/s**
•Time it takes for bits to travel across physical link
–depends on length of physical link (d) and propagation speed in medium (s).
Traffic Intensity = L·a / R

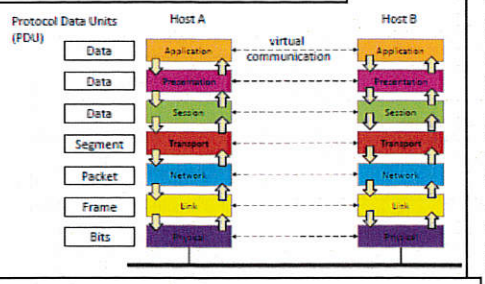
Transport Service Requirements
What requirements for transport does an application need?
•Data Loss
–no data loss (reliable transfer) or data loss may be tolerated
•Timing
–low latency may be needed for application to be effective
•Bandwidth
–high or low throughput for data transport
•Encryption
–data must be encrypted to ensure integrity and security

TCP Server
bind()
–Makes receipt before send possible
listen()
–Expresses willingness to accept incoming connections
–Sets connection parameters (queue limit, etc.) for incoming connections
accept()
–Accepts a connection on a listening socket
–Returns file descriptor for newly created 1-to-1

•TCP Transport Service
–connection-oriented service: send after connection has been setup
–reliable transport between sender and receiver
–flow and congestion control: sender will not send more data than receiver or network can accept
–no timing guarantees or security services
•UDP Transport Service
–connection-less service: send when ready
–no flow, congestion control: sender will transmit at sender's own pace
–no timing guarantees or security services

HTTP
•HTTP uses TCP.
–Client initiates TCP connection on port 80.
–Server accepts request from client.
–Client & server exchange HTTP message.
–Server and client closes connection.
•HTTP is a stateless protocol.
–Server & client don't maintain information about previous message exchanges.
–statelessness keeps protocol simple
HTTP Connection
COP4635 Spring 2012
•Non-tent:
–At most one object is sent over a TCP connection.
Example: Get a Web page but no embedded images.
•Persistent:
–Several objects may be sent over single TCP connection between client and server.
Example: Get a Web page and all embedded images.

Transport Layer – What does it do?
• Provides cost-effective data transport service from source to destination machine across a network.
• Transport protocols run in end systems, not routers or switches.
– Sender: splits messages into segments, passes them to network layer.
– Receiver side: reassembles segments into messages, passes it to application layer.
• Connection-oriented and connection-less transport protocol available to applications:
– Internet: TCP and UDP



DNS(Domain Name System) Services
•Name to address translation
–translate hostname to IP address and vice-versa
•Host aliasing
–Hosts with complicated hostnames may may be given an alias to simplify name for outside users.
–Example: www.dell.com is alias for www1.ins.dell.com. The hostname www1.ins.dell.com is the canonical name.
•Mail server aliasing
–Host name portion of Email address maps to the hostname of the server.
–Example: uwf.edu maps to delta.uwf.edu and gamma.uwf.edu.
•Load balancing
–Instead of a single host, multiple hosts provide services to balance workload.
–Client receives multiple IP addresses for single hostname.
–DNS server rotates IP addresses in list to ensure that clients contact different hosts

Performance Bottlenecks of rdt 3.0
•Poor performance of rdt 3.0 (stop-and-wait) protocol.
–sender waits for acks before sending again
•Utilization of sender:

$$U_{sender} = \frac{d\_trans}{RTT + d\_trans} = \frac{L/R}{RTT + L/R}$$

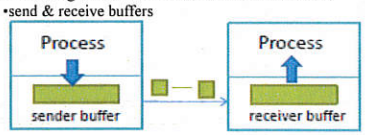
•Example:
1 Gb link, 10000 bits per packet, estimated RTT = 50msec

Sliding Window Protocols
•Sender may have up to N packets in pipeline.
Go-Back-N:
-receiver sends cumulative acks
-sender has timer for oldest unacked packet
-timeout triggers retransmission of all unacked packets
Selective Repeat:
-receiver acks individual packets
-sender has timer for each sent packet
-timeout triggers retransmission of unacked packet for selected timer

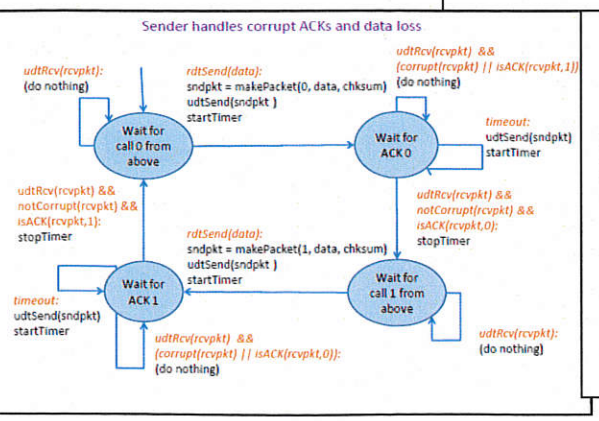
User Datagram Protocol (UDP)
•A best-effort protocol:
–UDP segments may be lost or delivered out of order to application.
•Connection less:
–no handshaking between UDP sender, receiver
–each UDP segment handled independently of others
•Benefits:
–simple because no connection state at sender/receiver needed
–no congestion control allowing it to be as fast as possible
–no overhead with connection setup/tear down
•Usages:
–streaming multi-media applications
–discovery of services on LAN
•Checksum:
–may be used for error detection
•Reliability:
–may be added at application layer for application specific uses

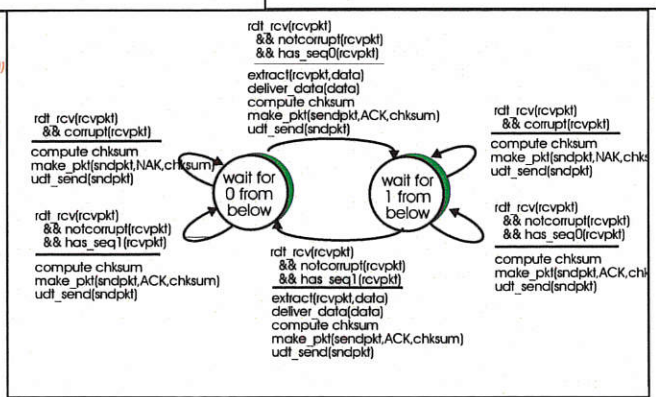
rdt 3.0 reliable transfer over channel with loss & bit error
•Communication channel may transmit packets with errors.
–acks and seq # provide sufficient info to address problem
•Communication channel may loose packets
–acks and seq # alone insufficient
–timer needed to keep track of lost packets
•Additional requirements:
–retransmit if no ack within a time period
–must handle delayed packets
–requires count down timer that triggers interrupts
– sender must handle timer interrupt, stop & start timer

Transmission Control Protocol (TCP)
•point-to-point:
–one sender, one receiver
•reliable, in-order byte stream:
–no "message boundaries"
•pipelined:
–TCP congestion and flow control set window size
•send & receive buffers

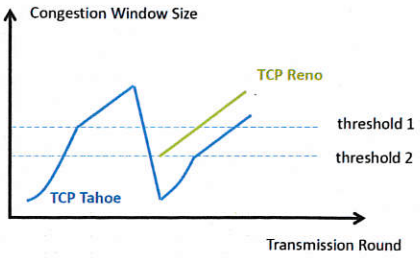


•full duplex data:
–bi-directional data flow in same connection
–MSS: maximum segment size
•connection-oriented:
–handshaking (exchange of control msgs) init's sender, receiver state before data exchange
•flow controlled:
–sender will not overwhelm receiver

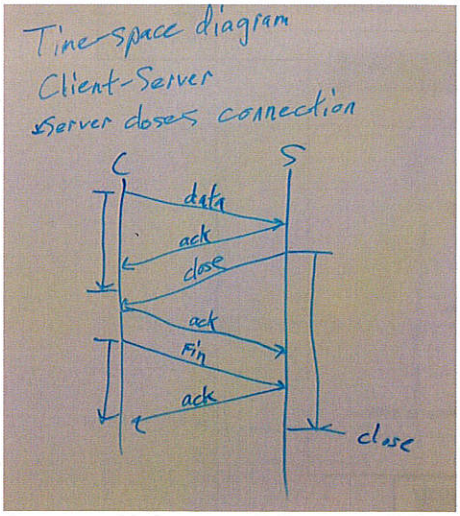




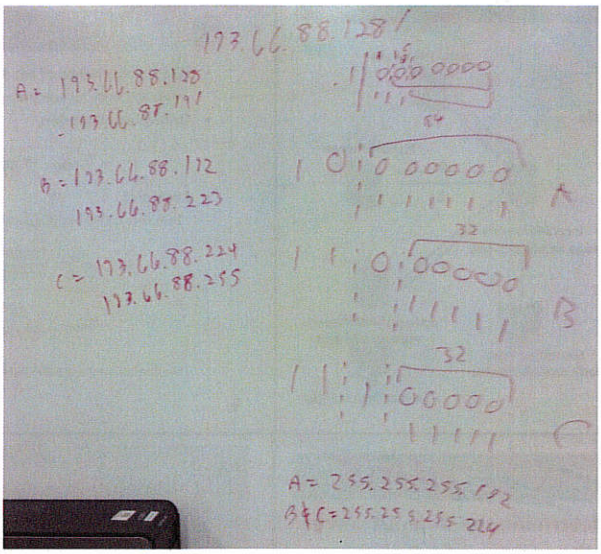
- Threshold is set to ½ value of CongestionWindow when congestion was last encountered
  - prevents that sender is growing window too fast



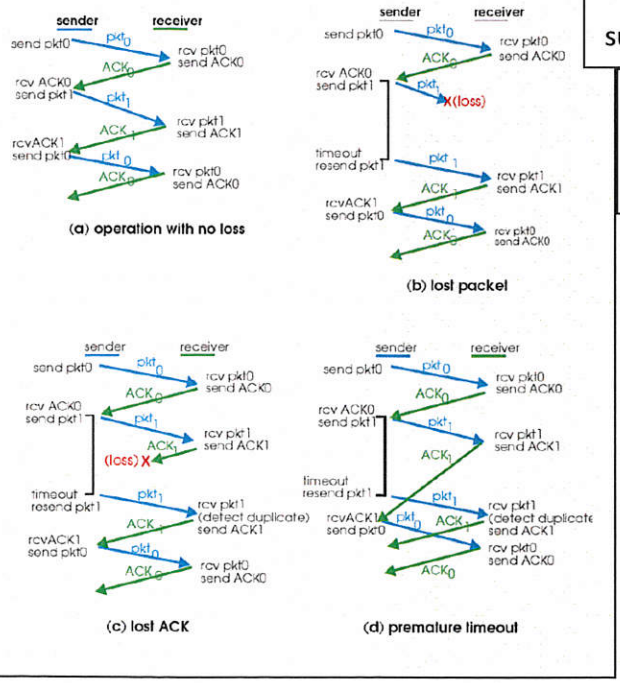
Criteria	G-B-N	Selective Repeat	TCP
Acks	Cumulative Acks	Individual acks	Cumulative acks
Timer	Can't receive out of order b/c recv has no buffer	Can receive out of order with recv buffer	If timeout before ack, packet resent
Window	Sender entire window has timer re send all constant	Each packet has timer, resend only unpacked constant	



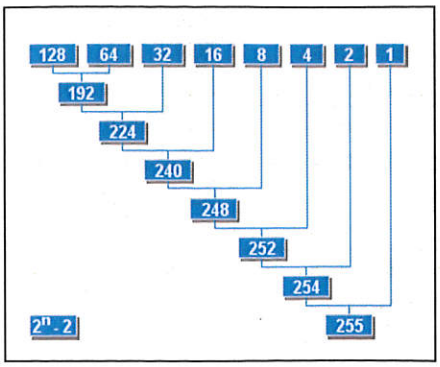
TCP



subnetting 3 offices: 60, 30, 31



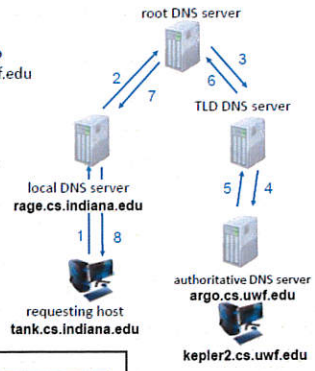
- Unshielded twisted pair :: host->switch or switch to router
- cross over :: host -> host or switch -> switch
- serial cable :: router -> router
- roll over :: host -> router



DNS Name Resolution – Recursive Example

Host tank.cs.indiana.edu wants to know IP address of kepler2.cs.uwf.edu

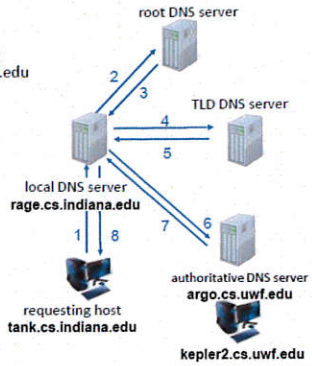
- Host contacts local DNS server.
- If local DNS has no entry for destination host, server contacts root DNS server (if not cached)
- root DNS server contact TLD DNS server,
- TLD DNS server contacts authoritative DNS server,
- ...



DNS Name Resolution – Iterative Example

Host tank.cs.indiana.edu wants to know IP address of kepler2.cs.uwf.edu

- Host contacts local DNS server.
- If local DNS has no entry for destination host, server contacts root DNS server (if not cached)
- root DNS server may direct to TLD DNS server (I don't know but ask this server...)
- TLD DNS server may direct to authoritative DNS server
- ...



Performance Bottlenecks of rdt 3.0

- **Poor performance of rdt 3.0 (stop-and-wait) protocol.**
  - sender waits for acks before sending again
- **Utilization of sender:**

$$U_{sender} = \frac{d_{trans}}{RTT + d_{trans}} = \frac{L/R}{RTT + L/R}$$

- **Example:**

1 Gb link, 10000 bits per packet, estimated RTT = 50msec

$$d_{trans} = \frac{L}{R} = \frac{10000\ bits}{10^9\ bps} = 10^{-5} sec = 10^{-2} msec$$

$$U_{sender} = \frac{10^{-2} msec}{(50 + 10^{-2}) msec} \approx \frac{10^{-2}}{50} = 0.0002$$

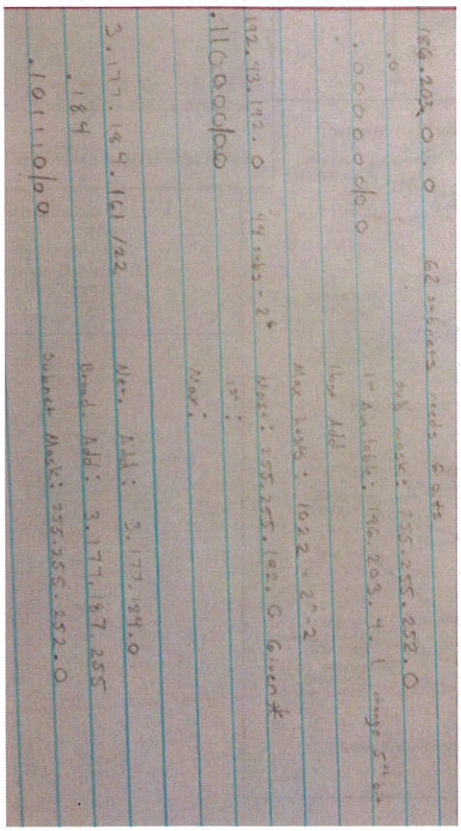
Sender transmits approximately 1 KB (8,192 bits) every 50msec or 20KB/sec.

TCP Estimating Timeout

- Question: How do we determine the timeout value?
- Answer: use round trip time as an estimate.
  - But round-trip-time may vary for different segments.
- Challenges:
  - timeout set too short: host generates retransmission due to premature timeout
  - timeout set too long: host responds too slow to lost segment
- Solution:
  - use previous measured RTT to estimate expected RTT

EstimatedRTT = (1- α)• EstimatedRTT + α• SampleRTT

TimeoutInterval = EstimatedRTT + "safety margin"



Wifi Network Challenges

Hidden node problem:
–two nodes may be sending but they cannot "hear" each other's transmission due to obstacles or distance

IEEE 802.11 MAC Protocol

Uses CSMA/CA as random access protocol.
Uses Acks at link layer to handle errors in transmission.

•Sender:
1.if channel idle, send after initial waiting period (distributed inter-frame space = DIFS)
2.if channel busy, backoff for a random time and transmit when time expires
3.if no Ack received after transmission, increase timeout period and return to 2.

•Receiver
–if frame received without error, return Ack after a short period of time (short inter-frame space = SIFS).

IEEE 802.11 : Solving Hidden Node Problem

•Avoid collisions by assigning sender channel rather than allowing for random access.
•Sender transmits Request to Send (RTS) frame to access point (base station)
–RTS is short to avoid collisions
•Base station broadcasts Clear-to-Send (CTS) frame in response to RTS
–clear the channel for specific sender
–sender is allowed to transmit

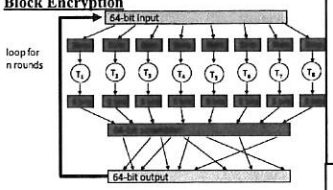
What is network security?

•Confidentiality: only sender & receiver should "understand" the message transmitted.
–sender must be able to encrypt message
–receiver must be able to decrypt message
•Authentication: sender & receiver want to confirm identity of each other.
•Message integrity: sender & receiver want to ensure message was not altered while in transit or afterwards.
•Access and availability: encryption, authentication, and detection services must be accessible and available to users.
•symmetric key encryption: sender & receiver have identical keys.
•public-key encryption: a public key is used for encryption, a private for decryption.

Network Intrusion: What can happen?

•eavesdropping: someone intercepts messages transmitted over communication channel.
•message insertion: someone actively inserts messages into a connection between two hosts.
•impersonation: someone faking (spoof) source address in packet (or any field in packet).
•hijacking: someone "taking over" ongoing connection by removing sender or receiver, inserting himself in place.
•denial of service: someone preventing service from being used by others (e.g., by overloading resources).

Block Encryption



•Encrypts a block of characters instead of individual characters.
•One pass: one input bit affects eight output bits.
•Multiple passes: each input bit affects all output bits.

Encryption / Decryption

•Public key is *(n,e)*. Private key is *(n,d)*.
•To encrypt bit pattern *m* compute:
$c = m^e \bmod n$
"c is remainder of division of me with n"
•To decrypt bit pattern *c* compute:
$m = c^d \bmod n$
"m is remainder of division of cd with n"
$m = (m^e \bmod n)^d \bmod n$
$-(m^e \bmod n)^d$ is "c"

Public Key Requirements

1.Given a public key KR+ (.) and a private key KR- (.)
KR- (KR+ (m)) = m
2.It is impossible to compute a private key from a given public key.
RSA: Rivest, Shamir, Adleman algorithm

RSA Approach to Key Creation

1.Choose two large prime numbers p, q. (e.g., 1024 bits each).
2.Compute $n = pq$, $z = (p-1)(q-1)$.
3.Choose e (with e<n) that has no common factors with z. (e, z are "relatively prime").
4.Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5.Public key is *(n,e)*. Private key is *(n,d)*.
-(n,e) is K_{R+} (.) and *(n,d) is* K_{R-} (.)

Public Key Certification

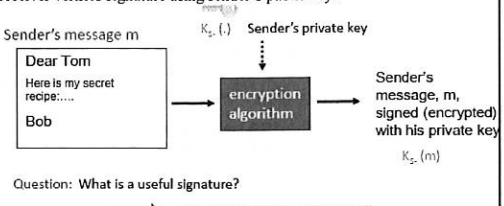
Problem with Digital Signature:
How do we know that the public key of the sender is really the sender's public key and not someone else's key?
Solution:
–Trusted certification authority (CA) manages public keys.
–Sender registers public key with a CA, using some form of identification upon registration.
–Receiver obtains public key from sender.

SSL: Step 2 – Key Generation

•Sender and receiver use shared secret (MS) to generate 4 different keys:
–ES: Sender-to-Receiver data encryption key
–ER: Receiver-to-Sender data encryption key
–MS: Sender-to-Receiver Message

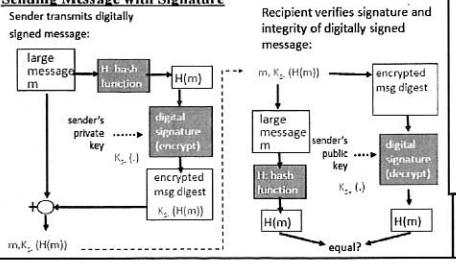
Digital Signature

•Sender encrypts a digital signature with his private key.
–receiver verifies signature using sender's public key.



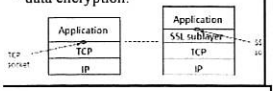
Question: What is a useful signature?
Hash value of message m itself!

Sending Message with Signature



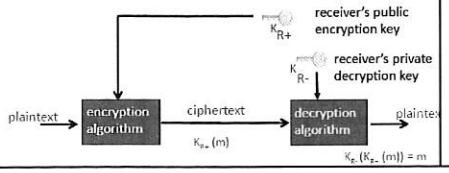
Secure Socket Layer (SSL)

•SSL establishes a secure communication between two hosts.
–works at the Transport layer
•Security Services provides:
–server authentication,
–data encryption.

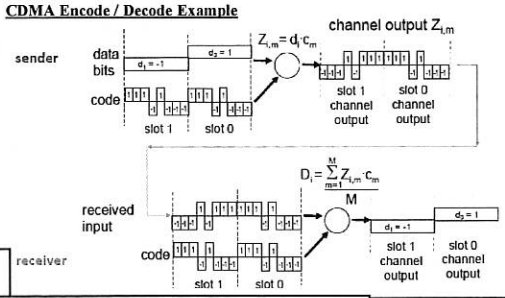


Public Key Encryption

•Sender & receiver do not share secret key.
•Public encryption key is known to all.
•Private decryption key known only to receiver.

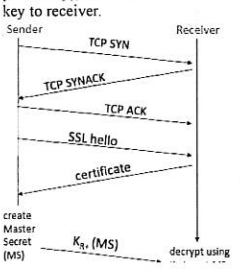


CDMA Encode / Decode Example

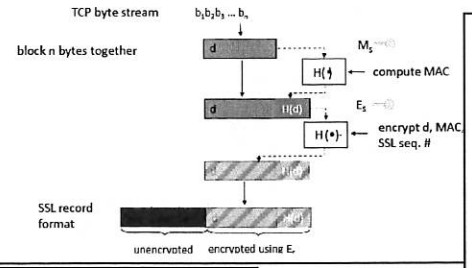


SSL: Step 1 Handshake

•Sender establishes TCP connection to receiver.
•Sender authenticates receiver via CA signed certificate.
•Creates, encrypts (using receiver's public key), sends master secret key to receiver.



SSL: Step 3 – Sending



Types of Transmission Delays

1.Node Processing Delay
•Time it takes for the node to:
–checks for bit errors
–determines output link using routing table
2.Queuing Delay
•**Traffic Intensity = L·a / R**
3.Transmission delay: **L/R**
• L = packet size,
–depends on bandwidth (bps) of link (R) and packet length (L)
4.Propagation Delay: **d/s**
–depends on length of physical link (d) and propagation (s).

RSA Example

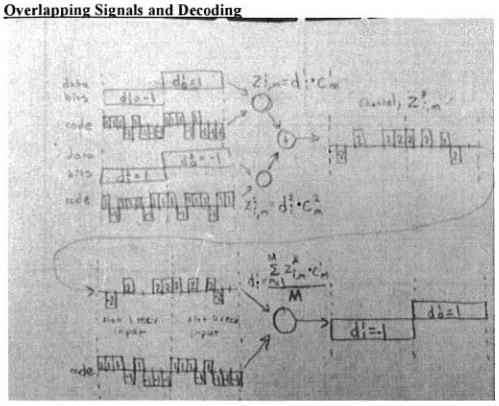
•Assume $p=5$, $q=7$. Then $n=35$ and $z=24$.
•Choose $e=5$ and $d=29$.
–e is relative prime with z (they don't have a common factor).
–$e \cdot d$ -1 is devisible by z ($5 \cdot 29-1 = 144$ is divisible by 24).

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	r	9	59049	4

	c	c^d	$m = c^d \bmod n$	letter
decrypt:	4	2.8823E+17	9	r

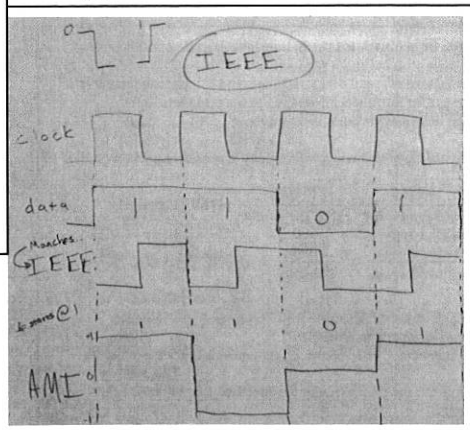
n = p*q
z = (p - 1)(q - 1)
e < n && has no common factors w/ z
choose d so (ed - 1) is divisible by z

Overlapping Signals and Decoding



-Explain why a minimum frame size is required for Ethernet. For example, 10Base Ethernet imposes a minimum of 64 bytes. Suppose the distance between two nodes on a bus network is *d*. Derive a formula that determines the minimum frame size for an Ethernet frame based on the distance, the propagation speed, and the transmission speed for sending bits.

a) $2d * \frac{Trans}{prop}$ = packet size.



Performance Bottlenecks of rdt 3.0

•Poor performance of rdt 3.0 (stop-and-wait) protocol.
–sender waits for acks before sending again
•Utilization of sender:

$$U_{sender} = \frac{d\_trans}{RTT + d\_trans} = \frac{L/R}{RTT + L/R}$$

•Example:
1 Gb link, 10000 bits per packet, estimated RTT = 50msec
$-d_{trans} = \frac{L}{R} = \frac{10000 bits}{10^9 bps} = 10^{-5}$sec $= 10^{-2}$msec

Dynamic Host Configuration Protocol (DHCP)

•Assigning hosts an IP address manually in networks may not be feasible.
–number of hosts in wired and wireless network may change frequently
•DHCP provides services to acquire IP address in a network.
–Allows reuse of addresses (only hold address while connected an "on")
–Support for mobile users who want to join network (more shortly)
–Implemented by a client-server service model.

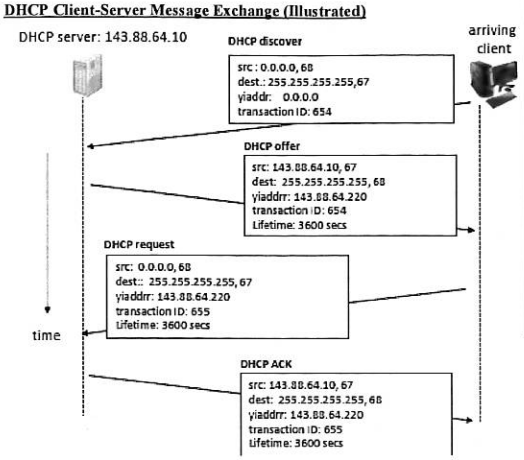
DHCP Client-Server Message Exchange

1.Host broadcasts "DHCP discover" message.
•Client sends a UDP packet to port 67 on the network.
2.DHCP server responds with "DHCP offer" message.
•DHCP server listens for incoming discovery messages from host.
•DHCP server looks up available IP addresses in a table
3.Host requests IP address: "DHCP request" message.
•Host selects offer from among DHCP servers responding to its initial inquiry.
•Host sends request echoing back the suggested configuration parameters.
4.DHCP server sends address: "DHCP ack" message.
•Server sends acknowledgment to host.
•Server updates its table of available IP addresses in network to note allocation.

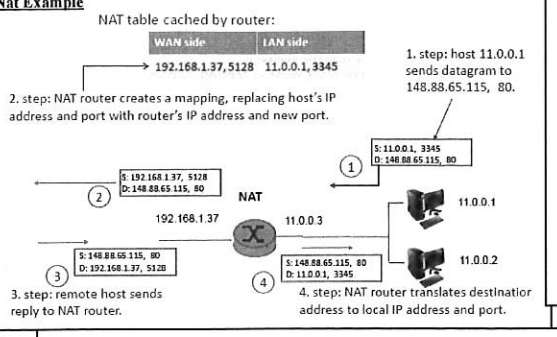
Application Layer Security

•Applications may use public-private key encryption to encrypt communication
•Examples:
–HTTPS is an encrypted version of HTTP
•uses a SSL to encrypt HTTP messages
–denyhosts is a Python program that analyzes log data to prevent intrusion for SSH servers
•if a remote host attempts to log in for more than a fixed number of times, the host will be blocked from future attempts to connect to the server

DHCP Client-Server Message Exchange (Illustrated)



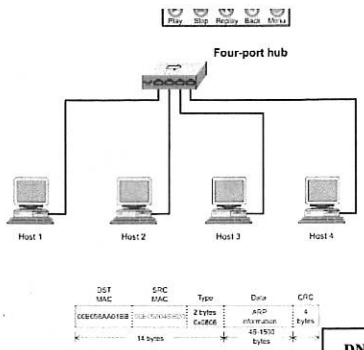
Nat Example



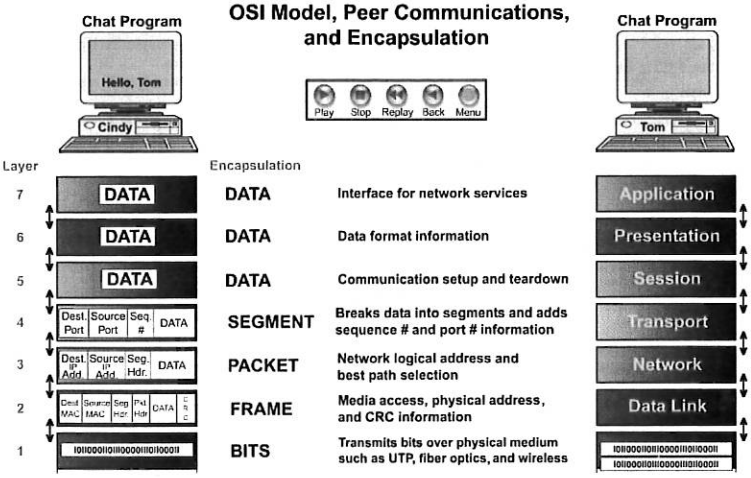
subnet 193.66.88.128/25
=193.66.88.(1|0000000) splint on 25th bit
-split 3 ways network with 62, 28, 30 hosts
-use 2 bits to differentiate the 3 networks (1|00|00000)
-i.g. 00, 10, 11
-64 host network range 193.66.88.129-190
need an extra bit for 62 hosts
min(1|0|000000) =128, max(1|0|111111)=191
subnet mask = 255.255.255.128
-28 host network range 193.66.88.193-222
min(1|10|00000) =192, max(1|10|11111)=223
subnet mask = 255.255.255. 224
-32 host network range = 193.66.88.225-254
min (1|11|00000) =224, max(1|11|11111)=255
subnet mask = 255.255.255.224

How ARP works

1. Host 1 wants to send a frame to Host 4
2. In order to send the frame, Host 1 needs the MAC address for Host 4.
3. Host 1 tries to look up the MAC address in its ARP cache and finds no entry for Host 4.
4. Host 1 sends an ARP request broadcast frame to all Hosts on the segment looking for the Host 4 MAC address by using the IP address for Host 4.
5. Each of the Hosts receives the frame
6. Hosts 2 and 3 discard the frame because their IP addresses do not match the request.
7. Host 4 receives the frame and replies with an ARP response frame telling Host 1 its MAC address.
8. Host 1 receives the ARP response frame and fills in the missing MAC address it received from Host 4 and sends the frame.
9. The frame is delivered to Host 4.

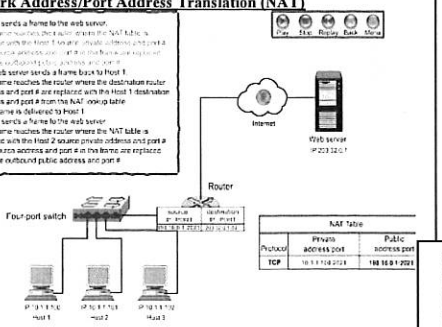


OSI Model, Peer Communications, and Encapsulation



Network Address/Port Address Translation (NAT)

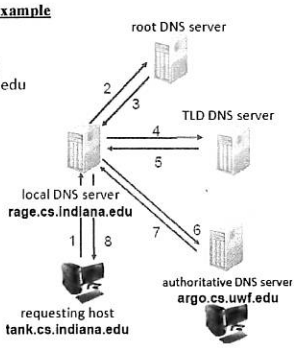
1. Host 1 sends a frame to the web server.
2. The frame reaches the router where the NAT table is updated with the Host 1 source private address and port # The source address and port # in the frame are replaced with the outbound public address and port #
3. The web server sends a frame back to Host 1
4. The frame reaches the router where the destination router address and port # are replaced with the Host 1 destination address and port # from the NAT lookup table
5. The Frame is delivered to Host 1
6. Host 2 sends a frame to the web server
7. The frame reaches the router where the NAT table is updated with the Host 2 source private address and port # The source address and port # in the frame are replaced with the outbound public address and port #

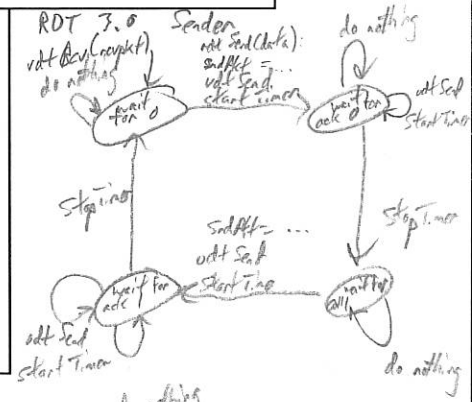


DNS Name Resolution – Iterative Example

Host tank.cs.indiana.edu wants to know IP address of kepler2.cs.uwf.edu

- Host contacts local DNS server.
- If local DNS has no entry for destination host, server contacts root DNS server (if not cached)
- root DNS server may direct to TLD DNS server (I don't know but ask this server...)
- TLD DNS server may direct to authoritative DNS server
- ...

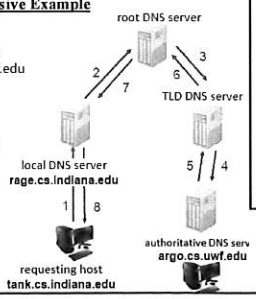




DNS Name Resolution – Recursive Example

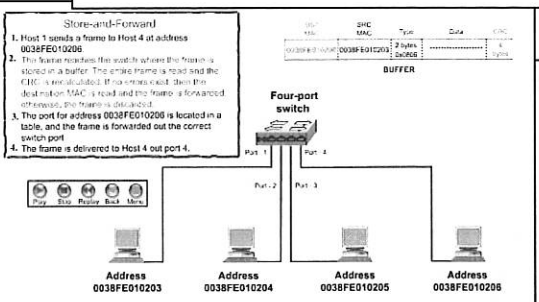
Host tank.cs.indiana.edu wants to know IP address of kepler2.cs.uwf.edu

- Host contacts local DNS server.
- If local DNS has no entry for destination host, server contacts root DNS server (if not cached)
- root DNS server contact TLD DNS server,
- TLD DNS server contacts authoritative DNS server,
- ...



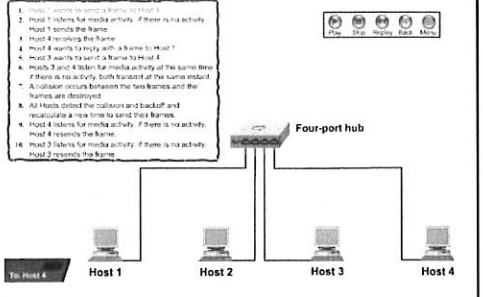
Store-and-Forward

1. Host 1 sends a frame to Host 4 at address 0038FE010206
2. The frame reaches the switch where the frame is stored in a buffer. The entire frame is read and the CRC is recalculated. If no errors exist, then the destination MAC is read and the frame is forwarded, otherwise, the frame is discarded.
3. The port for address 0038FE010206 is located in a table, and the frame is forwarded out the correct switch port
4. The frame is delivered to Host 4 out port 4.



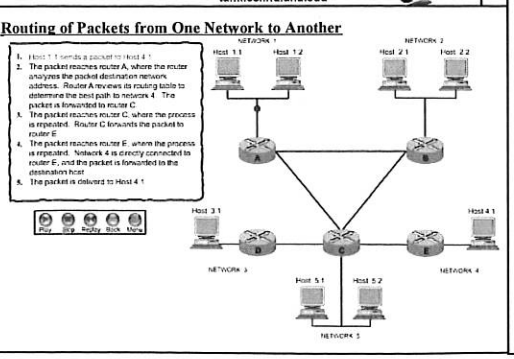
10BaseT Ethernet Operation

1. Host 1 wants to send a frame to Host 4
2. Host 1 listens for media activity. If there is no activity Host 1 sends the frame.
3. Host 4 receiving the frame
4. Host 4 wants to reply with a frame to Host 1
5. Host 3 wants to send a frame to Host 4
6. Hosts 3 and 4 listen for media activity at the same time if there is no activity, both transmit at the same instant
7. A collision occurs between the two frames and the frames are destroyed
8. All Hosts detect the collision and backoff and recalculate a new time to send their frames.
9. Host 4 listens for media activity. If there is no activity, Host 4 resends the frame.
10. Host 3 listens for media activity. If there is no activity, Host 3 resends the frame.



Routing of Packets from One Network to Another

1. Host 1.1 sends a packet to Host 4.1
2. The packet reaches router A, where the router analyzes the packet destination network address. Router A reviews its routing table to determine the best path to network 4. The packet is forwarded to router C.
3. The packet reaches router C, where the process is repeated. Router C forwards the packet to router E
4. The packet reaches router E, where the process is repeated. Network 4 is directly connected to router E, and the packet is forwarded to the destination host
5. The packet is delivered to Host 4.1



1.5 MB file from host A to host B, transmits at a rate of 500Kbps? stop-and-wait protocol and a segment size of 2 KB. RTT is fixed at 150ms (milliseconds) and that 5 segments and 1 acknowledgment is lost. time-out for lost/unacknowledged packages160ms. Ignore processing delays at the hosts and routers.

a. Compute the time it takes without any loss of data packets on the network.

$$\frac{1.5MB}{2KB/segment} = \frac{1536KB}{2KB/segment} = \mathbf{768\ segments}$$

trans time for 1 seg

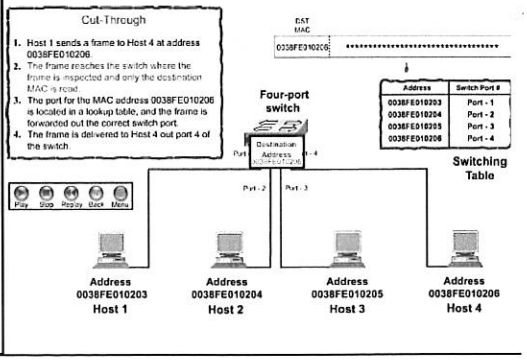
$$\frac{2*1024*8bits}{500*1000bits/sec} = \frac{16384}{500000} = 0.032768\ \text{sec} = \mathbf{32.768msec}$$

Time for ack to arrive: 150msec + 32.768msec = 182.768msec.
Time for all segments to transmit:
768 segments * 182.768 msec/segment = 140365.824 msec = **140.365 sec**

b. Compute the time it takes with the specified loss of data packets on the network.
Delay for 6 lost transmissions: 960 msec.
Total time: 140.365 sec + 0.96 sec = 141.325 sec.

Cut-Through

1. Host 1 sends a frame to Host 4 at address 0038FE010206
2. The frame reaches the switch where the frame is inspected and only the destination MAC is read.
3. The port for the MAC address 0038FE010206 is located in a lookup table, and the frame is forwarded out the correct switch port.
4. The frame is delivered to Host 4 out port 4 of the switch



Wifi Network Challenges

Hidden node problem:

–two nodes may be sending but they cannot "hear" each other's transmission due to obstacles or distance

IEEE 802.11 MAC Protocol

Uses CSMA/CA as random access protocol.

Uses Acks at link layer to handle errors in transmission.

•Sender:

1.if channel idle, send after initial waiting period (distributed inter-frame space = DIFS)

2.if channel busy, backoff for a random time and transmit when time expires

3.if no Ack received after transmission, increase timeout period and return to 2.

•Receiver

–if frame received without error, return Ack after a short period of time (short inter-frame space = SIFS).

IEEE 802.11 : Solving Hidden Node Problem

•Avoid collisions by assigning sender channel rather than allowing for random access.

•Sender transmits Request to Send (RTS) frame to access point (base station)

–RTS is short to avoid collisions

•Base station broadcasts Clear-to-Send (CTS) frame in response to RTS

–clear the channel for specific sender

–sender is allowed to transmit

What is network security?

•Confidentiality: only sender & receiver should "understand" the message transmitted.

–sender must be able to encrypt message

–receiver must be able to decrypt message

•Authentication: sender & receiver want to confirm identity of each other.

•Message integrity: sender & receiver want to ensure message was not altered while in transit or afterwards.

•Access and availability: encryption, authentication, and detection services must be accessible and available to users.

•symmetric key encryption: sender & receiver have identical keys.

•public-key encryption: a public key is used for encryption, a private for decryption.

Network Intrusion: What can happen?

•eavesdropping: someone intercepts messages transmitted over communication channel.

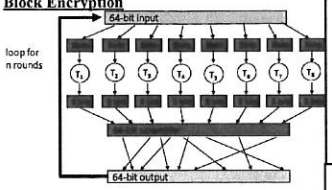
•message insertion: someone actively inserts messages into a connection between two hosts.

•impersonation: someone faking (spoof) source address in packet (or any field in packet).

•hijacking: someone "taking over" ongoing connection by removing sender or receiver, inserting himself in place.

•denial of service: someone preventing service from being used by others (e.g., by overloading resources).

Block Encryption



•Encrypts a block of characters instead of individual characters.

•One pass: one input bit affects eight output bits.

•Multiple passes: each input bit affects all output bits.

Encryption / Decryption

•Public key is *(n,e)*. Private key is *(n,d)*.

•To encrypt bit pattern *m* compute:

$c = m^e \bmod n$

"c is remainder of division of me with n"

•To decrypt bit pattern *c* compute:

$m = c^d \bmod n$

"m is remainder of division of cd with n"

$m = (m^e \bmod n)^d \bmod n$

$-(m^e \bmod n)^d$ is "c"

Public Key Requirements

1.Given a public key KR+ (.) and a private key KR- (.)

KR- (KR+ (m)) = m

2.It is impossible to compute a private key from a given public key.

RSA: Rivest, Shamir, Adleman algorithm

RSA Approach to Key Creation

1.Choose two large prime numbers p, q. (e.g., 1024 bits each).

2.Compute $n = pq$, $z = (p-1)(q-1)$.

3.Choose e (with e<n) that has no common factors with z. (e, z are "relatively prime").

4.Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).

5.Public key is *(n,e)*. Private key is *(n,d)*.

-(n,e) is K_{R+} (.) and *(n,d) is* K_{R-} (.)

SSl: Step 2 – Key Generation

•Sender and receiver use shared secret (MS) to generate 4 different keys:

–ES: Sender-to-Receiver data encryption key

–ER: Receiver-to-Sender data encryption key

–MS: Sender-to-Receiver Message

Secure Socket Layer (SSL)

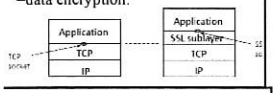
•SSL establishes a secure communication between two hosts.

–works at the Transport layer

•Security Services provides:

–server authentication,

–data encryption.

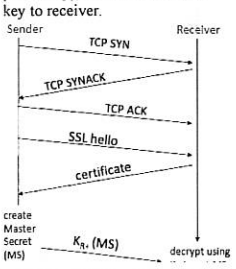


SSL: Step 1 Handshake

•Sender establishes TCP connection to receiver.

•Sender authenticates receiver via CA signed certificate.

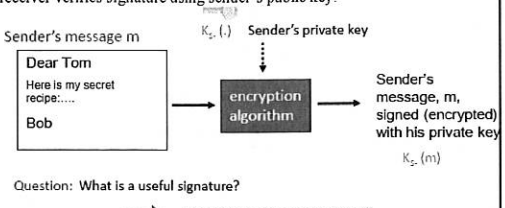
•Creates, encrypts (using receiver's public key), sends master secret key to receiver.



Digital Signature

•Sender encrypts a digital signature with his private key.

–receiver verifies signature using sender's public key.



Question: What is a useful signature?

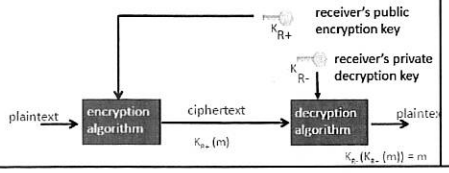
Hash value of message m itself!

Public Key Encryption

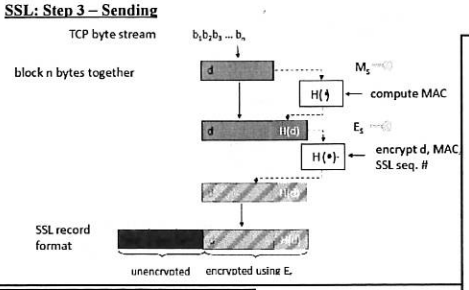
•Sender & receiver do not share secret key.

•Public encryption key is known to all.

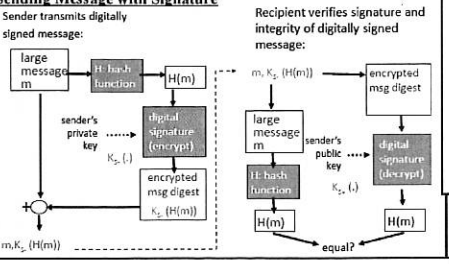
•Private decryption key known only to receiver.



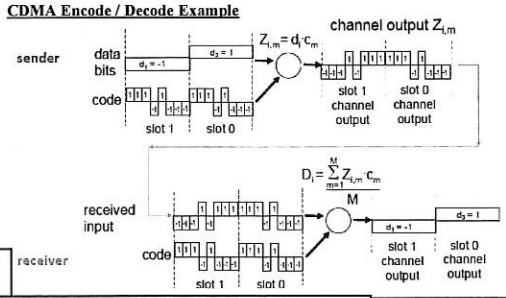
SSL: Step 3 – Sending



Sending Message with Signature



CDMA Encode / Decode Example



Public Key Certification

Problem with Digital Signature:

How do we know that the public key of the sender is really the sender's public key and not someone else's key?

Solution:

–Trusted certification authority (CA) manages public keys.

–Sender registers public key with a CA, using some form of identification upon registration.

–Receiver obtains public key from sender.

Application Layer Security

•Applications may use public-private key encryption to encrypt communication

•Examples:

–HTTPS is an encrypted version of HTTP

•uses a SSL to encrypt HTTP messages

–denyhosts is a Python program that analyzes log data to prevent intrusion for SSH servers

•if a remote host attempts to log in for more than a fixed number of times, the host will be blocked from future attempts to connect to the server

Types of Transmission Delays

1.Node Processing Delay

•Time it takes for the node to:

–checks for bit errors

–determines output link using routing table

2.Queuing Delay

•**Traffic Intensity = L·a / R**

3.Transmission delay: **L/R**

• L = packet size,

–depends on bandwidth (bps) of link (R) and packet length (L)

4.Propagation Delay: **d/s**

–depends on length of physical link (d) and propagation (s).

RSA Example

•Assume $p=5$, $q=7$. Then $n=35$ and $z=24$.

•Choose $e=5$ and $d=29$.

–e is relative prime with z (they don't have a common factor).

–$e \cdot d - 1$ is devisible by z ($5 \cdot 29 - 1 = 144$ is divisible by 24).

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	r	9	59049	4

	c	c^d	$m = c^d \bmod n$	letter
decrypt:	4	2.8823E+17	9	r

n = p*q

z = (p - 1)(q - 1)

e < n && has no common factors w/ z

choose d so (ed - 1) is divisible by z

Performance Bottlenecks of rdt 3.0

•Poor performance of rdt 3.0 (stop-and-wait) protocol.

–sender waits for acks before sending again

•Utilization of sender:

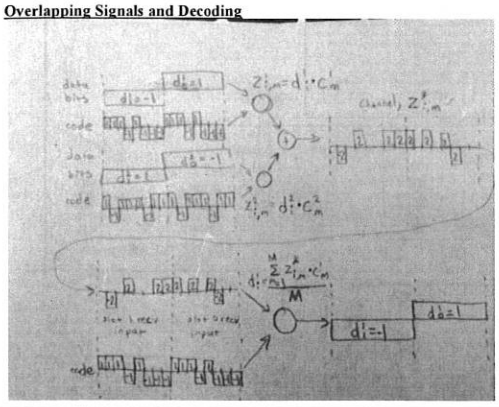
$$U_{sender} = \frac{d\_trans}{RTT + d\_trans} = \frac{L/R}{RTT + L/R}$$

•Example:

1 Gb link, 10000 bits per packet, estimated RTT = 50msec

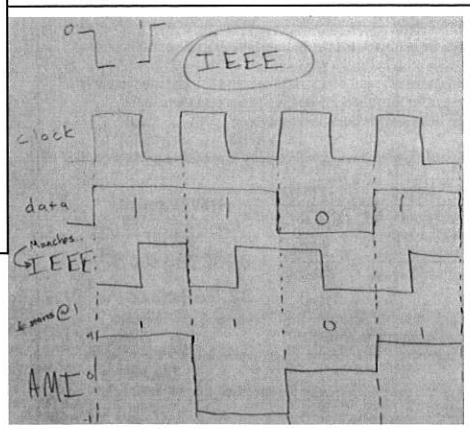
$-d_{trans} = \frac{L}{R} = \frac{10000bits}{10^9 bps} = 10^{-5}\text{sec} = 10^{-2}\text{msec}$

Overlapping Signals and Decoding



-Explain why a minimum frame size is required for Ethernet. For example, 10Base Ethernet imposes a minimum of 64 bytes. Suppose the distance between two nodes on a bus network is *d*. Derive a formula that determines the minimum frame size for an Ethernet frame based on the distance, the propagation speed, and the transmission speed for sending bits.

a) $2d * \frac{Trans}{prop}$ = packet size.



Dynamic Host Configuration Protocol (DHCP)

•Assigning hosts an IP address manually in networks may not be feasible.

–number of hosts in wired and wireless network may change frequently

•DHCP provides services to acquire IP address in a network.

–Allows reuse of addresses (only hold address while connected an "on")

–Support for mobile users who want to join network (more shortly)

–Implemented by a client-server service model.

DHCP Client-Server Message Exchange

1.Host broadcasts "DHCP discover" message.

•Client sends a UDP packet to port 67 on the network.

2.DHCP server responds with "DHCP offer" message.

•DHCP server listens for incoming discovery messages from host.

•DHCP server looks up available IP addresses in a table

3.Host requests IP address: "DHCP request" message.

•Host selects offer from among DHCP servers responding to its initial inquiry.

•Host sends request echoing back the suggested configuration parameters.

4.DHCP server sends address: "DHCP ack" message.

•Server sends acknowledgment to host.

•Server updates its table of available IP addresses in network to note allocation.

Mulitple Access Protocol Overview

•Types of links:
- –Point-to-Point (PPP)
- –broadcast

•Point-to-Point
- –original dial-up (PPP)
- –over Ethernet (PPPoE)
- –segment link between host and switch
- –router-to-router

•Broadcast (shared wired or wireless medium)
- –old-fashioned Ethernet
- –wireless LAN (803.11, RF)

Shared Broadcast Channel: multiple nodes may access simultaneously channel for transmission

Solution:
- –time- or frequency-division multiplexing
- –distributed / centralized system that determines when node can transmit

•requires channel for coordination

Media Access Control (MAC) Addressing

• Internet:
- – logical network-layer address (e.g. 32 or 64 bits)
- – used in datagram to designate destination subnet and host
- – address hierarchical and location-specific

• LAN:
- – physical, data-link layer address (e.g. 48 bits)
- – address assigned permanently to NIC
- – flat address, none-location specific

Ethernet

•Implementations
- –10Base5: Thicknet
- –10Base2: Thinnet
- –100BaseT, 1000BaseT: Ethernet over UTP
- –1000BaseLX, 1000BaseSX: Ethernet over fiber
- –10GBaseSR, 10GBaseLR, … : Ethernet over fiber

Ethernet CSMA/CD

1.NIC receives datagram from network layer implemented by OS, to create a frame.
2.NIC listens on channel.
- –if channel idle, start transmitting frame
- –if channel busy, wait until channel is idle

before transmitting
3.NIC checks if transmission of frame was successful.
- –NIC detects no transmission of another signal

4.NIC's transmission is not successful if another transmission was detected.
- –aborts own transmission and sends jamming signal

5.After aborting, NIC waits for a randomly chosen time before retrying.
- –after m-th attempt to send fails, choose k from {1, …, 2m-1} and wait k·512 bit times
- –repeat with step 2

Data Transport in Hubs

•Hub is a plug-and-play device.
- –requires no manual configuration

•Frames sent across the network may collide with each other.

•No frame buffering.

Data Transport in Switches

•Switch is a plug-and-play, self-learning device.
- –requires no manual configuration
- –forwards Ethernet frames using physical address in the frame
- –identifies the destination port by observing traffic flow

Self-Learning Switches

Switch sets up a table to forward frames to destination host.

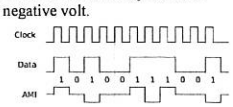
•unknown destination flood

•known destination select

MAC address	Interface	TTL
A	3	60
C	2	60

Alternative Mark Inversion (AMI)

• 0 are encoded as 0 volt, 1 is encoded as alternate positive or negative volt.



• Problems: long sequence of 0 may cause receiver to become out of sync with sender.

IEEE 802.11b/g Wireless LAN (WiFi) Configuration

•Access point (AP) must be assigned a channel name and number.
- –Each access point is given a MAC address
- –85 MHz band (2.4-2.485) can be divided into 11 channels of partially overlapping spectrum of airwaves
- –channels separate different APs for better wireless access

Multiple Access Protocol Taxonomy

Over shared media:
•Channel partitioning
- –divide channel based on time slots, frequency, etc.
- –each node is given a designated slot, frequency, etc.

•Random Access
- –channel may be accessed at any time by node
- –may cause collisions that must be addressed

•Taking Turns
- –nodes take turns to access shared media

Slotted Aloha Efficiency

-Assume N nodes sends a frame in a slot with probability p.
-What is the probability for a given node to succeed
sending in a slot?
-P(success of a node) = p(1-p)(N-1)
-What is the probability for any of the N nodes to succeed sending in a slot?
-P(success of a node) = Np(1-p)(N-1)
-Maximum efficiency: find p so that
-max Np(1-p)(N-1)
-for N going to infinity.
-p = 1/e = 0.37 **37% Efficiency at best.**

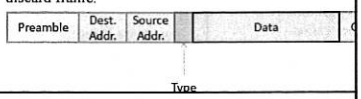
ARP: Steps for determining MAC address.

Task: Host A wants to send Host B a frame.
Assumption: Host A does not have Host B's MAC address, only its IP address.
•Host A broadcasts ARP query containing B's IP address.
-destination MAC address is FF-FF-FF-FF-FF-FF
-all hosts on LAN receive the query
•Host B responds to ARP query.
-sends a frame to A with its MAC address.
-A stores MAC address in ARP table along with TTL

Ethernet Frame Structure

•Destination and Source address is 6 bytes long.
- –Adapter passes frame into memory if
  - •destination address matches adapter's destination address
  - •destination address is broadcast address

•Type in Ethernet Frame indicates higher layer protocol such as IP, Novell, AppleTalk, etc.
•CRC checksum is used to detect error in frame, to discard frame.



Ethernet Efficiency

•Collisions reduce efficiency.
•Approximation of efficiency:

$$efficiency = \frac{1}{1 + 5 \cdot \frac{t_{prop}}{t_{trans}}} \quad \rightarrow 1 \text{ for } t_{prop} \rightarrow 0, \; t_{trans} \rightarrow \infty$$

–*tprop* is propagation delay between 2 nodes in LAN
–*ttrans* is transmission time for complete frame

Switching Methods

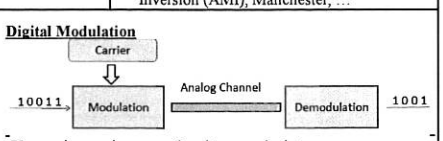
•store-and-forward switching
- –switch stores frame
- –forwards frame based on address
- –checks for errors and only forwards error-free and complete frames
- –slowest method of switching

•cut-Through switching
- –does not store frame but instead looks at address information to forward frame to output port
- –fastest method of switching

Baseband Transmission

- Apply pulse modulation to binary symbol for transmission over digital/baseband channel.
- Pulse modulation uses regular sequence of pulses to encode / decode data.
- Line coding examples: Alternate Mark Inversion (AMI), Manchester, …

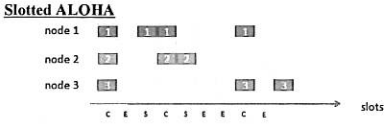
Digital Modulation



•Uses a sine carrier wave signal to encode data.
- –sine carrier wave has low frequency spectrum compared to square waves

•Used for transmission over long distances.
•Forms of modulation:
- –amplitude, frequency, phase-shift

Slotted ALOHA



- all frames are of same size
- time is divided into equal size slots
- a time slot permits a node to send a frame
- nodes are synchronized
- nodes start transmitting at slot beginning
- if 2 or more nodes transmit in a slot, all nodes detect collision
- when node obtains fresh frame, it transmits the frame in next slot
  - *if no collision:* node successfully transmitted frame,
  - *if collision:* node must retransmit frame in each subsequent slot with probability *p* until success

Carrier Sense Multiple Access (CSMA)

• Implemented by Ethernets
• Node send whenever ready and no signal detected
- – may cause collision when two nodes send simultaneously
- – requires method to handle collisions

Carrier Sense Multiple Access/Collision Detection (CSMA/CD)
• **Step 1:** Listen on the network before transmitting.
• **Step 2:**
- – If no signal detected (channel is idle), transmit frame.
- – If signal detected (channel is busy), wait.

• **Step 3:** After frame has been transmitted, check if a collision occurred.
- – If no collision detected, frame was successfully transmitted.
- – If collision has been detected, wait some time before retransmitting.

CSMA/CD Summary
• Collision must be detected within short time.
• Collision detection:
- – easy in wired networks by comparing sent and received signal
- – difficult in wireless network due to signal attenuation

Ethernet Services

•**Connectionless**: sender and receiver do not need to establish a connection prior to sending (hand-shake)

•**Unreliable**: receiver does not send an acknowledgment to sender.

•Media Access:
- –CSMA/CD for wired
- –CSMA/CA for wireless
  - -**Collision Avoidance**: The sender must signal intent to transmit data before it is allowed to access channel and ransmit any data. All other stations may not transmit until they signal their intent.

Data Transport Devices

•Repeater
- –connects two segments
- –regenerates signal to enable data to be transmitted across combined single segment
- –operates at physical layer

•Hubs
- –connects multiple host into a star configuration
- –operates at physical layer

•Bridge
- –connects two LANs
- –directs frames between connected LANs to behave like a single LAN
- –operates at data link layer

Manchester Encoding (line code)

•Transmits the exclusive-OR of the clock and signal.
- –each bit is transmitted in a fixed time
- –a 1 is encoded by a transition from high to low and a 0 by a transition from low to high (G.E. Thomas convention)

•Results in a transition every clock cycle.

Wireless Channel Access

•Code Division Multiple Access (CDMA)
- –cellular networks and satellite networks

•Carrier Sense Multiple Access / Collision Avoidance (CSMA /CA)
- –Wireless LAN (WIFI)

Code Division Multiple Access
•All wireless nodes share same frequency.
- –wireless node is given a "code" to encode data
- –multiple users can transmit simultaneously with minimal interference
- –data can be extracted from combined signal

•Divides one-bit slot into multiple smaller slots.
•Signal Encoding: constant product of bit with chipping sequence.
•Signal Decoding: inner product of signal with chipping sequence.

Random Access Protocols

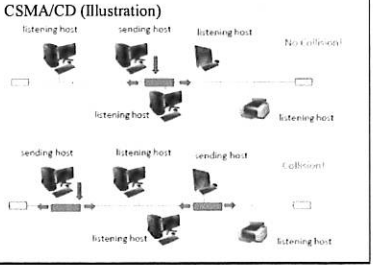
•When network nodes have data to send:
- –nodes transmit at full channel data rate
- –no coordination among nodes on who sends first

•When network nodes transmit collision may happen.
- –requires technology to detect collisions
- –requires protocol to address collisions and recover from them

Random Access MAC Protocol Examples include:
- •(slotted) ALOHA

•CSMA, CSMA/CD, CSMA/CA

CSMA/CD (Illustration)



Ethernet Architecture

Bus Ethernet

Star Ethernet

Types of Signal Transmissions

•Broadband transmission
- –transmission over a broad spectrum of frequencies
- –use analog techniques to encode binary 1s and 0s across a continuous range of values
- –used over long distances
- –supports multiplexing of signals at different frequencies

•Baseband transmission
- –signals take the form of discrete pulses of electricity or light
- –uses a digital encoding scheme (line coding) to encode bit stream
- –used by Ethernet for data transmission

Types of Wireless Network

•Wireless local area network (WLAN)
- –easy to setup and maintain
- –provides ad-hoc network access in buildings
- –requires wireless network card and access point device
- –access point translates radio to electrical signal and vice-versa

•Extended wireless LANs
- –moderately difficult to setup and maintain
- –provides more coverage beyond a typical LAN configuration

–higher equipment cost than WLAN
•Mobile computing
–uses cell-phone technology for data transfer
–provides immediate network access anywhere within reach of a radio tower

Wireless Networking Benefits
•Users can create temporary connections to existing wired networks.
•Allows easy setup of separate subnet with different security profile.
•Provides backup or contingency connectivity for existing wired networks.
•Extend a network's span beyond the reach of wire-based or fiber-optic cabling
- –attractive for buildings were wiring would be too expensive or impractical

•Enable users to roam with their machines within certain limits (called "mobile networking")

Wireless LAN Transmission
•Uses electromagnetic waves for data transmission.
•Frequency of the wave forms is measured in Hz.
- –low frequency waves carry data slower over longer distances than high frequency waves
- –low frequency waves emit less energy than high frequency waves

•Spectrum of electromagnetic waves for data transmission is divided into three ranges:
- –Radio: 10 KHz to 1 GHz
- –Microwave: 1 GHz to 500 GHz
- –Infrared: 500 GHz to 1 THz

Routing Algorithms

- Global
  - –router in network must have complete topological and cost info on network before computing routing table
  - –link state algorithms
- Decentralized
  - –routers "know" connected neighbors and cost
  - –router iteratively updates table using neighbor info from other neighbors
  - –distance vector algorithms
- Static Routing
  - –routes in network change slowly
  - –route change may be due to manual change of routing table entries
- Dynamic Routing
  - –routes in network change quickly
  - –route change may be due to traffic load change or topological change
  - –algorithm runs periodically or in response to changes in network topology or cost

Bellman-Ford Equation:

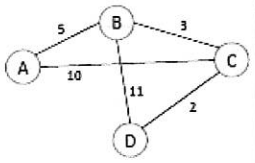
Let d(x, y) = cost of least-cost path from x to y.

Formula: d(x,y) = min {c(x,v) + d(v,y)}

-(x,v) is direct cost from x to neighbor v

-min is minimum coputed over all neighbors v of x

Variation on Dijkstra's Least Cost Path Algorithm

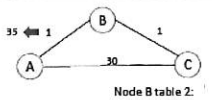


D(B) = min(D(B), D(C) + c(C,B)

1	(D,0,-)	
2	(D,0,-)	(B,11,B)
3	(D,0,-) (C,2,C)	
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)	

Distance Vector Algorithm

1. Each node maintains a table that specifies the distances to all other nodes.
   a. distance may be infinity if path to destination node is unknown
2. Each node sends its distance vector to all of its surrounding nodes periodically.
3. If a node receives a distance vector from an adjoining node:
   a. node may replace its own entry to use shorter path through adjoining node
4. If a node does not get an update from an adjoining node for a while:
   a. adjoining node is considered dead
   b. new routes will be calculated.



Node B table 1:

From	Distance to Reach Node A	B	C
A	0	1	2
B	1	0	1
C	2	1	0

Node B table 2:

From	Distance to Reach Node A	B	C
A	3	1	2
B	3	0	1
C	2	1	0

- Cost between A and B changes to 35. B computes new route to A.
- B "thinks" that reaching A through C is cheaper because cost of C to A is only 2.
  - loop: B forwards packets to C, C forwards packets to B
  - new cost to A is updated and forwarded to C and A (see table 2)
- C recomputes distance to A using information received from B
  - C updates its table and forwards its distance vector back to B.
- B updates its table considering the new distance vector from C
  - new cost to A is updated and forwarded to C and A
- ...

Link State Algorithm

- Uses a technique called **reliable flooding** to broadcast link-state information.
  - – information broadcast by each node is called the link-state packet (LSP)
- Each LSP contains
  - – ID of the creator node
  - – List of directly connected neighbors with link costs
  - – Sequence number
  - – Time to live (TTL)
- Reliable flooding works as follows:

a. Each node sends its LSP to every directly connected node

b. When a node receives information from another node, it checks if the LSP has a newer sequence number than the last one received from that node.

c. If the sequence number is newer, then the LSP is stored and also sent to all of the new node's direct neighbors excluding the original sender.

- Consider two lists, **tentative** and **confirmed**, that contain entries of the form (Destination, Cost, NextHop)
- Algorithm:

1. Initialize **confirmed** to an entry to itself (cost = 0).
2. Pick the entry just added to the **confirmed** list and call it **next**.
3. For each neighbor of **next**, calculate the cost to reach the neighbor.
4. If the neighbor is neither on the **confirmed** nor **tentative** list, add (Neighbor, Cost, NextHop) to the tentative list.
5. If the neighbor is on the **tentative** list, replace it if the new cost is lower.
6. If the **tentative** list is empty, stop. Otherwise, pick the lowest cost entry from the **tentative** list and move it to the **confirmed** list and return to step 2.

Distance Vector vs Link State

Link State:

- with n nodes requires O(n2) steps to compute table
- with E links, generates O(nE) messages in network
- each node computes its own table
- sabotaged node may report incorrect link cost to other nodes

Distance Vector

- iterative update of tables converging on final table
- – may suffer from routing loops
- nodes inform immediate neighbors only
- sabotaged node may advertise incorrect path to neighboring nodes

RIP "Routing Information Protocol"

-uses distance vector **NO COST INVOLVED** ONLY # hops

-Router sends response messages (advertisements) every 30 seconds.

-If no advertisement received for time:

- –link declared dead
- –routes via dead neighbor are invalidated
- –new advertisements sent to remaining neighbors
- -15 hops is max and 16 is infinity

-**Split Horizon**: node won't forward packet to node it recieved it from

- -**poison**: sets path to 16 to say route is dead
- -**poison reverse**: senders sends poisoned route to reciever and receiver resends poisoned route to sender so sender keeps the route as poisoned

-doesnt prevent issue of RIP certain graph structures can still result in false updates. Only deals with Classful IP addresses.

Job of Network Interface Card

- Creates and maintains network connection.
  - –Fundamental unit for sending/receiving data is a frame.
- Translates signal into data (bits) and vice-versa.
- Performs error checking on incoming data from network.
  - –handles incomplete or unintelligible frames
  - –handles errors in data
- Buffers data from network and memory alike for transmission onto the network and transport into main memory.
  - –buffer is a temporary storage area.
  - –driver transports data from and to main memory.
- Checks if received frame is targeted for host machine.

NIC as a Gatekeeper

- Each network card is given a unique identifier known as the Media Access Control (MAC) address.
- MAC address of recipient is stored in data frame.
- NIC determines if frame should be kept because
  - –the MAC address of recipient matches the MAC address of the NIC and
  - –the frame is in a correct format and no transmission error have been detected
- Gatekeeper function may be disabled to receive any frame it recognizes
  - –maybe useful for data sniffing on network

Data Transfer between NIC and Memory

- Direct Memory Access (DMA)
  - –allows direct transfer of data from adapter to main memory without CPU intervention
  - –improves data transfer over interrupt-driven I/O
- Shared Adapter Memory
  - –adapter's buffers map directly into main memory
  - –no overhead in data transfer; CPU treats adapter's memory as own memory
- Shared System Memory
  - –adapter's onboard processor selects region in memory to read/write data
  - –adapter uses main memory as its own memory

Autonomous Systems (AS)

Even with subnetting, far too many networks for routing exists.

-Increase levels of hierarchy by grouping organizations into larger entities called Autonomous Systems (AS)

- An AS is a group of networks and routers controlled by a single administrative authority.
- Each AS appears as a single network to any entity outside the AS.
- AS may be a large corporation or a regional ISP
- Each AS has one or more designated routers that communicate with other Ass
  - –a backbone is also considered an AS).

Broadcast Routing

- Send packets from source to all other nodes in network.
- –sender duplication (N-way unicast):
- requires knowledge of nodes in network
- inefficient because same packets travel multiple times over same link
- –flooding (controlled):
- send copy of packet to all neighbors (if not already seen)
- –spanning tree
- send copy of packet along spanning tree edges
- requires construction of spanning tree before broadcast

Multicasting

- Send packets from source to a subset of nodes in network.
  - –useful for file sharing, media streaming, etc.
- Requires a single identifier labeling subset (multicast group)
  - –group is addressed via class D address
- Internet Group Management Protocol (IGMP)
  - –manages communication between host and router
  - –allows hosts to join a multicast group

Multicast Routing

- Connect nodes that are members of the multicast group in a tree structure.
  - –tree contains routers that are members of the multicast group and in-between routers
  - –multicast packets will be broadcast along tree path
- Trees connecting nodes can be
  - –source-based: different trees from each sender to receiver
  - –shared: same tree used by all group members

BGP "Border Gateway Protocol"

- BGP route selection steps
  - –obtains subnet reachability from neighboring AS (CDIRized prefixes)
  - –propagates reachability info to internal routers
  - –internal router determines "good" route to subnets
- selects shortest AS-PATH info for prefix
- selects closets next-hop router in AS for prefix
- applies additional policies that must be enforced for route selection

Multiple Access Protocol Taxonomy

Over shared media:

- Channel partitioning
  - –divide channel based on time slots, frequency, etc.
  - –each node is given a designated slot, frequency, etc.
- Random Access
  - –channel may be accessed at any time by node
  - –may cause collisions that must be addressed
- Taking Turns
  - –nodes take turns to access shared media

OSPF "Open Shortest Path First"

- Employed in upper-tier ISPs
- Steps for Computing Routing Table:
  - –LS packets are disseminated in OSPF message
  - –topological map of AS is computed from received LS packets
  - –shortest distance routes are computed from map
- Uses authentication for messages exchanged.
  - –prevent malicious attacks on routing tables
- Allows for multiple same-cost paths.
- Uses hierarchical OSPF in large domains.

Link Layer Services

- Framing and network access
  - –encapsulating data into frame, adding header and footer
  - –addressing frame using the **MAC** address
  - –accessing communication channel
- Reliable deliver between adjacent nodes
  - –seldom used on low bit-error link (fiber, some twisted pair)
  - –wireless links: high error rates
- Flow Control
  - –adjusting data flow between adjacent nodes in network
- Error Detection
  - –errors may be caused by signal attenuation, noise, etc.
  - –receiver detects presence of errors in frame
  - –receiver may request new transmission of erroneous frames
- Error Correction
  - –receiver detects bit error in frame and repairs the frame
- Half-duplex, Full-duplex
  - –two nodes in a ongoing communication may send at same time (full-duplex) or one at a time (half-duplex)
- Implemented in every Network Interface Card on every host

Network Address Translation

- –range of addresses not needed from ISP to build private network,
- –outside world only sees network as a single IP address,
- –single IP address is sufficient to communicate with outside world,
- –network can shrink or expand in size arbitrarily without ISP assistance,
- –network devices in private network can be configured with specific IP address as needed

NAT Implementation

- For outgoing datagrams, replace IP address and port of source in outgoing datagram.
  - –new IP address becomes IP address of router based on NAT
  - *–new port number is port number chosen by router based on NAT*
  - *–remote host will respond to host with the new IP address and port*
- For incoming datagrams, replace IP address and port of NAT router with IP address and port of destination host.
  - –NAT table holds destination IP address and port
- Router implementing NAT stores mapping of IP address and port between external and internal network.

Cyclic Redundancy Check (CRC)

- |G| = r + 1
  - -G is generator, r is # of 0's padding message

ex) G = 10101, message = 10111011, r = 4

this is from a sender

-XOR the bits

10101 | 101110110000

10101

00010011 -grab more #'s for 0's out front

10101

0011000

10101

011010

10101

011110

10101

01011 = remainder (R)

- for receiver XOR the sent message |M|R| -> 10110111011
  - -remainder **MUST** be 0 else there is an error

Reliable Transmission in Data Link Layer

- Relies on two basic mechanisms
  - –Acknowledgments (ACKs)
    - A small control frame that notifies the other end about frames received
  - –Timeouts
    - A mechanism that allows the sender or receiver to wait for a specified amount of time and then take corrective action if the awaited event does not occur. Also known as Automatic Repeat Request (ARQ).

Criteria	Go-Back-N	Selective Repeat	TCP
Acks	Cum. acks	Indiv. acked	Cum. acks
Timer	1 timer per unacked seq	1 timer per unacked seq	one timer
T/O action	Send all unacked	Send unacked, t/o'ed	Send unacked
Seq. #	counter for seq sent	counter for seq sent	byte stream

Delays: Processing | Transmission (packet length L / trans speed R) | Propagation (distance d / prop speed S) | Queuing

Mulitple Access Protocol Overview

•Types of links:
- –Point-to-Point (PPP)
- –broadcast

•Point-to-Point
- –original dial-up (PPP)
- –over Ethernet (PPPoE)
- –segment link between host and switch
- –router-to-router

•Broadcast (shared wired or wireless medium)
- –old-fashioned Ethernet
- –wireless LAN (803.11, RF)

Shared Broadcast Channel: multiple nodes may access simultaneously channel for transmission

Solution:
- –time- or frequency-division multiplexing
- –distributed / centralized system that determines when node can transmit

•requires channel for coordination

Media Access Control (MAC) Addressing

• Internet:
- – logical network-layer address (e.g. 32 or 64 bits)
- – used in datagram to designate destination subnet and host
- – address hierarchical and location-specific

• LAN:
- – physical, data-link layer address (e.g. 48 bits)
- – address assigned permanently to NIC
- – flat address, none-location specific

Ethernet

•Implementations
- –10Base5: Thicknet
- –10Base2: Thinnet
- –100BaseT, 1000BaseT: Ethernet over UTP
- –1000BaseLX, 1000BaseSX: Ethernet over fiber
- –10GBaseSR, 10GBaseLR, … : Ethernet over fiber

Ethernet CSMA/CD

1.NIC receives datagram from network layer implemented by OS, to create a frame.
2.NIC listens on channel.
- –if channel idle, start transmitting frame
- –if channel busy, wait until channel is idle

before transmitting
3.NIC checks if transmission of frame was successful.
- –NIC detects no transmission of another signal

4.NIC's transmission is not successful if another transmission was detected.
- –aborts own transmission and sends jamming signal

5.After aborting, NIC waits for a randomly chosen time before retrying.
- –after m-th attempt to send fails, choose k from {1, …, 2m-1} and wait k·512 bit times
- –repeat with step 2

Data Transport in Hubs

•Hub is a plug-and-play device.
- –requires no manual configuration

•Frames sent across the network may collide with each other.

•No frame buffering.

Data Transport in Switches

•Switch is a plug-and-play, self-learning device.
- –requires no manual configuration
- –forwards Ethernet frames using physical address in the frame
- –identifies the destination port by observing traffic flow

Self-Learning Switches

Switch sets up a table to forward frames to destination host.

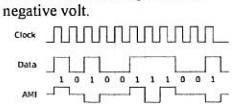
•unknown destination flood

•known destination select

MAC address	Interface	TTL
A	3	60
C	2	60

Alternative Mark Inversion (AMI)

• 0 are encoded as 0 volt, 1 is encoded as alternate positive or negative volt.



• Problems: long sequence of 0 may cause receiver to become out of sync with sender.

IEEE 802.11b/g Wireless LAN (WiFi) Configuration

•Access point (AP) must be assigned a channel name and number.
- –Each access point is given a MAC address
- –85 MHz band (2.4-2.485) can be divided into 11 channels of partially overlapping spectrum of airwaves
- –channels separate different APs for better wireless access

Multiple Access Protocol Taxonomy

Over shared media:
•Channel partitioning
- –divide channel based on time slots, frequency, etc.
- –each node is given a designated slot, frequency, etc.

•Random Access
- –channel may be accessed at any time by node
- –may cause collisions that must be addressed

•Taking Turns
- –nodes take turns to access shared media

Slotted Aloha Efficiency

-Assume N nodes sends a frame in a slot with probability p.
-What is the probability for a given node to succeed sending in a slot?
- -P(success of a node) = p(1-p)(N-1)

-What is the probability for any of the N nodes to succeed sending in a slot?
- -P(success of a node) = Np(1-p)(N-1)

-Maximum efficiency: find p so that
- -max Np(1-p)(N-1)

-for N going to infinity.
- -p = 1/e = 0.37 **37% Efficiency at best.**

ARP: Steps for determining MAC address.

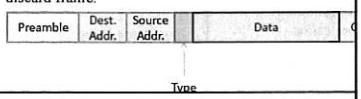
Task: Host A wants to send Host B a frame.
Assumption: Host A does not have Host B's MAC address, only its IP address.
•Host A broadcasts ARP query containing B's IP address.
- -destination MAC address is FF-FF-FF-FF-FF
- -all hosts on LAN receive the query

•Host B responds to ARP query.
- -sends a frame to A with its MAC address.
- -A stores MAC address in ARP table along with TTL

Ethernet Frame Structure

•Destination and Source address is 6 bytes long.
- –Adapter passes frame into memory if
  - •destination address matches adapter's destination address
  - •destination address is broadcast address

•Type in Ethernet Frame indicates higher layer protocol such as IP, Novell, AppleTalk, etc.
•CRC checksum is used to detect error in frame, to discard frame.



Ethernet Efficiency

•Collisions reduce efficiency.
•Approximation of efficiency:

$$efficiency = \frac{1}{1 + 5 \cdot \frac{t_{prop}}{t_{trans}}}$$

→ 1 for $t_{prop} \to 0$ or $t_{trans} \to \infty$

–*tprop* is propagation delay between 2 nodes in LAN
–*ttrans* is transmission time for complete frame

Switching Methods

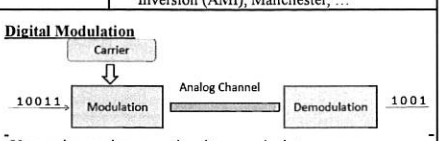
•store-and-forward switching
- –switch stores frame
- –forwards frame based on address
- –checks for errors and only forwards error-free and complete frames
- –slowest method of switching

•cut-Through switching
- –does not store frame but instead looks at address information to forward frame to output port
- –fastest method of switching

Baseband Transmission

- Apply pulse modulation to binary symbol for transmission over digital/baseband channel.
- Pulse modulation uses regular sequence of pulses to encode / decode data.
- Line coding examples: Alternate Mark Inversion (AMI), Manchester, …

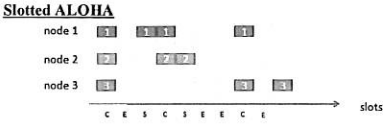
Digital Modulation



•Uses a sine carrier wave signal to encode data.
- –sine carrier wave has low frequency spectrum compared to square wave

•Used for transmission over long distances.
•Forms of modulation:
- –amplitude, frequency, phase-shift

Slotted ALOHA



- all frames are of same size
- time is divided into equal size slots
- a time slot permits a node to send a frame
- nodes are synchronized
- nodes start transmitting at slot beginning
- if 2 or more nodes transmit in a slot, all nodes detect collision
- when node obtains fresh frame, it transmits the frame in next slot
  - *if no collision:* node successfully transmitted frame,
  - *if collision:* node must retransmit frame in each subsequent slot with probability *p* until success

Carrier Sense Multiple Access (CSMA)

• Implemented by Ethernets
• Node send whenever ready and no signal detected
- – may cause collision when two nodes send simultaneously
- – requires method to handle collisions

Carrier Sense Multiple Access/Collision Detection (CSMA/CD)
• **Step 1:** Listen on the network before transmitting.
• **Step 2:**
- – If no signal detected (channel is idle), transmit frame.
- – If signal detected (channel is busy), wait.

• **Step 3:** After frame has been transmitted, check if a collision occurred.
- – If no collision detected, frame was successfully transmitted.
- – If collision has been detected, wait some time before retransmitting.

CSMA/CD Summary
• Collision must be detected within short time.
• Collision detection:
- – easy in wired networks by comparing sent and received signal
- – difficult in wireless network due to signal attenuation

Ethernet Services

•**Connectionless**: sender and receiver do not need to establish a connection prior to sending (hand-shake)

•**Unreliable**: receiver does not send an acknowledgment to sender.

•Media Access:
- –CSMA/CD for wired
- –CSMA/CA for wireless
  - -**Collision Avoidance**: The sender must signal intent to transmit data before it is allowed to access channel and ransmit any data. All other stations may not transmit until they signal their intent.

Data Transport Devices

•Repeater
- –connects two segments
- –regenerates signal to enable data to be transmitted across combined single segment
- –operates at physical layer

•Hubs
- –connects multiple host into a star configuration
- –operates at physical layer

•Bridge
- –connects two LANs
- –directs frames between connected LANs to behave like a single LAN
- –operates at data link layer

Manchester Encoding (line code)

•Transmits the exclusive-OR of the clock and signal.
- –each bit is transmitted in a fixed time
- –a 1 is encoded by a transition from high to low and a 0 by a transition from low to high (G.E. Thomas convention)

•Results in a transition every clock cycle.

Wireless Channel Access

•Code Division Multiple Access (CDMA)
- –cellular networks and satellite networks

•Carrier Sense Multiple Access / Collision Avoidance (CSMA /CA)
- –Wireless LAN (WIFI)

Code Division Multiple Access
•All wireless nodes share same frequency.
- –wireless node is given a "code" to encode data
- –multiple users can transmit simultaneously with minimal interference
- –data can be extracted from combined signal

•Divides one-bit slot into multiple smaller slots.
•Signal Encoding: constant product of bit with chipping sequence.
•Signal Decoding: inner product of signal with chipping sequence.

Random Access Protocols

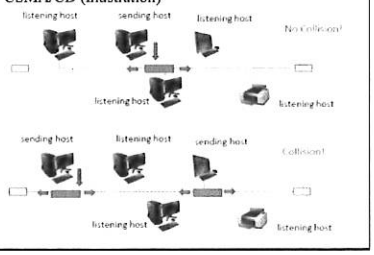
•When network nodes have data to send:
- –nodes transmit at full channel data rate
- –no coordination among nodes on who sends first

•When network nodes transmit collision may happen.
- –requires technology to detect collisions
- –requires protocol to address collisions and recover from them

Random Access MAC Protocol Examples include:
- •(slotted) ALOHA

•CSMA, CSMA/CD, CSMA/CA

CSMA/CD (Illustration)



Ethernet Architecture

Bus Ethernet

Star Ethernet

Types of Signal Transmissions

•Broadband transmission
- –transmission over a broad spectrum of frequencies
- –use analog techniques to encode binary 1s and 0s across a continuous range of values
- –used over long distances
- –supports multiplexing of signals at different frequencies

•Baseband transmission
- –signals take the form of discrete pulses of electricity or light
- –uses a digital encoding scheme (line coding) to encode bit stream
- –used by Ethernet for data transmission

Types of Wireless Network

•Wireless local area network (WLAN)
- –easy to setup and maintain
- –provides ad-hoc network access in buildings
- –requires wireless network card and access point device
- –access point translates radio to electrical signal and vice-versa

•Extended wireless LANs
- –moderately difficult to setup and maintain
- –provides more coverage beyond a typical LAN configuration

–higher equipment cost than WLAN
•Mobile computing
–uses cell-phone technology for data transfer
–provides immediate network access anywhere within reach of a radio tower

Wireless Networking Benefits

•Users can create temporary connections to existing wired networks.
•Allows easy setup of separate subnet with different security profile.
•Provides backup or contingency connectivity for existing wired networks.
•Extend a network's span beyond the reach of wire-based or fiber-optic cabling
- –attractive for buildings were wiring would be too expensive or impractical

•Enable users to roam with their machines within certain limits (called "mobile networking")

Wireless LAN Transmission

•Uses electromagnetic waves for data transmission.
•Frequency of the wave forms is measured in Hz.
- –low frequency waves carry data slower over longer distances than high frequency waves
- –low frequency waves emit less energy than high frequency waves

•Spectrum of electromagnetic waves for data transmission is divided into three ranges:
- –Radio: 10 KHz to 1 GHz
- –Microwave: 1 GHz to 500 GHz
- –Infrared: 500 GHz to 1 THz

– Implementation - Too much overhead Time of access, Sort by access, etc.
• Pro
– Good ways to approximate LRU

From Synchronization

Critical Sections and Threads
• Threads must proceed through an entry and exit sections.
• Threads must not die or quit inside a critical section.
• Threads may be context switched inside a critical section.
• A context switch is different from an exit because another thread may not be allowed to enter their critical section.

Semaphores

```
int sem_wait( sem ) {
        Disable_Interrupts();
        sem.count--;
        if (sem.count < 0) {
                waitlist.enqueue(self);
                Enable_Interrupts();
                block();
                Disable_Interrupts();
        }
        Enable_Interrupts();
        return 0;
}
int sem_post( sem ) {
        Disable_Interrupts();
        sem.count++;
        if (sem.count <= 0) {
                proc = waitlist.dequeue();
                readyList.insert(proc);
        }
        Enable_Interrupts();
        return 0;
}
```

Monitors
• Monitors are a high-level synchronization construct.
• Monitors allow safe sharing of abstract data types among concurrent processes.
• Monitors provide synchronized methods that can only be entered by one process or thread at a time.
• Monitors provide condition variables with two operations
– wait: puts process or thread asleep
– signal: triggers a waiting process or thread to resume operation exactly at the position where process or thread called wait

Scheduling
• Allocate CPU to the next process ready to execute.
• CPU scheduling is needed when:
– 1. Process switches from running to waiting state.
– 2. Process switches from running to ready state.
– 3. Process switches from waiting to ready state.
– 4. Process terminates.
• Switching a process from running to ready state or from waiting to ready state is carried out preemptively.
– Switch is not triggered by process but by the OS instead.

MLFQ Algorithm
Scheduler picks process from highest priority nonempty queue
– If Qi is empty, try Qi+1
Process goes to CPU from Qi
If quantum expires
– Add process to Qi+1
If I/O initiated
– Move process to blocked list
– Return process to either Qi-1 or Q1

Threads
• Definition: A thread (= job) is a lightweight process, representing a single unit of program execution within a process.
– shares code, data & file descriptor table
– has individual stack, registers & PCB (TCB)
• A process may run multiple threads concurrently.
– allows program to handle several tasks simultaneously (e.g. spell check a document while allowing the user to edit it)
• Two types of threads
– User-level threads: managed by library in user space; library performs context switch
– Kernel threads: managed by kernel; kernel performs context switch
• Thread creation similar to process creation except
– Processes begins execution at main()
– Thread begins execution at specified function
– Creation call identifies initial function

Benefits
• More responsive: even if parts of the program are blocked, other parts may still be running.
• Resource sharing: threads within a process share the same address space and global variables.
• Economy: because threads share resources within a process, creating new threads is less costly in terms of time and memory consumption than creating new processes.

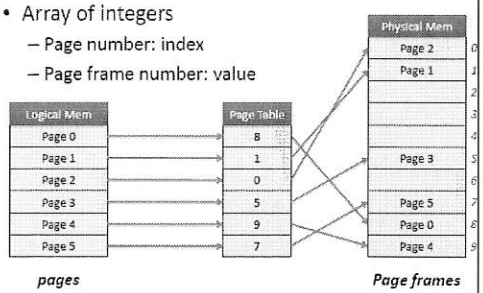
Thread I/O
• Library threads
– kernel schedules process
– thread initiates I/O, process blocks
– no other thread can run
• Kernel threads
– kernel schedules threads
– thread initiates I/O, thread blocks
– doesn't effect other threads

Processes

Layout
Process consists of code: a sequence of instructions
data: global variables
stack: local variables, function parameters generated during execution
heap: dynamic memory allocated during run-time

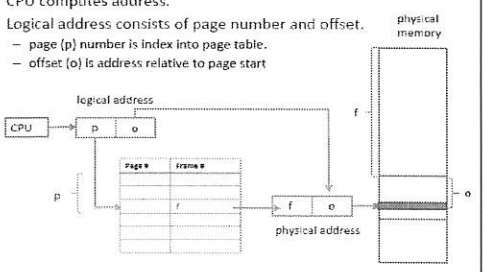
Transitions to kernel mode
1. Software
• System call
• Called a "trap" into the kernel
• Jump to well-known function
• Hardware switches to kernel mode
• Example: call to printf() or read()
2. Exception
• Error state or debugging
• Similar to system call without return (error)
• Jump to well-known function
• Hardware switches to kernel mode
• Example: division by zero or segmentation violation
• Result: core dump
3. Hardware
• Called an "interrupt"
• Communication between kernel & devices
• Can occur between any two instructions
• Similar to system call without call
• Jump to well-known function
• Hardware switches to kernel mode
• Example: clock tick or I/O complete

- Logical → Physical (page → page frame)
- Array of integers
  - Page number: index
  - Page frame number: value

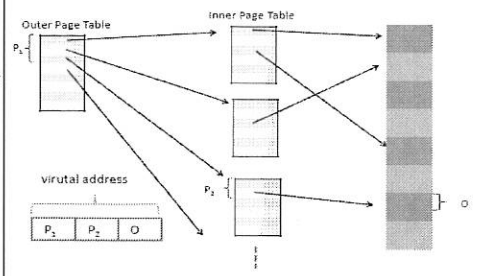


Address Translation for Paging

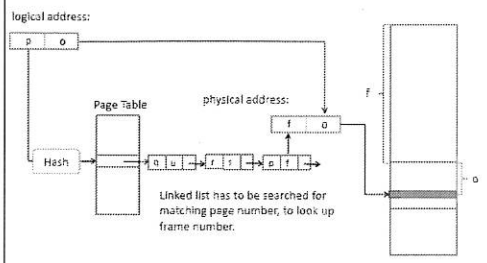
CPU computes address.
Logical address consists of page number and offset.
- page (p) number is index into page table.
- offset (o) is address relative to page start



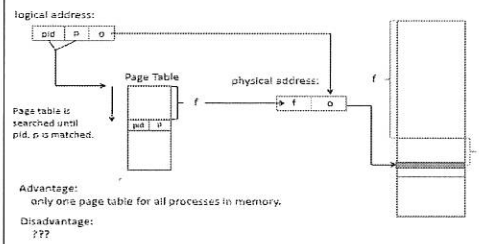
Hierarchical Paging



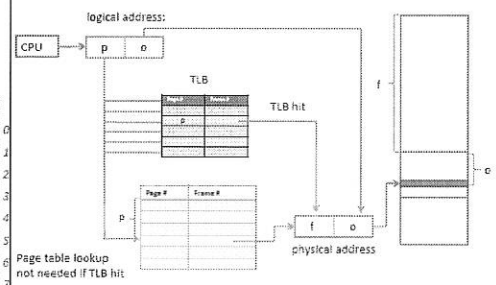
Hashed Paging



Inverted Paging



Address Translation with TLB (Illustration)



From Review

PCB

– Process ID,
– Process state (e.g. running, ready, waiting, …),
– Program counter (address of next instruction),
– CPU registers,
– CPU scheduling information,
– File management (list of open files, working directory, …),
– I/O status information,
– Memory management information (pointers to text, data, stack).

FORK

• Local variables survive
• Global variables survive
• File descriptor table survives
• PCB
– Registers survive
– Stack pointer doesn't survive
– PC doesn't survive
– PID doesn't survive

Round Robin

Preemptive version of FIFO
Ready list is queue
Quantum used to generate interrupts
+ Relatively simple
+ Fair: each job gets quantum
- Quantum size is tricky choice
- More overhead than FIFO

Shortest Job First (SJF)

Non-Preemptive
Schedule job with shortest CPU expected
+ Provably optimal (conditions)
+ Little overhead
- Poor response time
- Long jobs could starve

Shortest Remaining Time SRTCF

Shortest Remaining Time to Completion First
Preemptive version of SJF
Schedule job with shortest CPU expected
Preempt and reschedule when
– Job terminates
– Job arrives
– Internal event (yield)
Evaluation
+ Provably optimal AVG resp (conditions)
+ Short jobs preempt long jobs (Why?)
- Unfair
- Long jobs could starve

Critical Section

• Mutual Exclusion: prohibit other processes from entering their critical section while a process executes its critical section
• Progress: if no process is in their critical section, there must be progress on any process that wants to enter their critical section.
• Bounded waiting: there must be a bound on the number of times other processes are allowed to enter their critical section after a process has made a request to enter its critical section.

Semaphore

Semaphore
– Has a value
– Implies a history
– wait() may sleep or may proceed depending on value
– signal() will increment a value and wake a sleeper
– signal() before wait() will effect result of wait

Contition Variable (monitor)

Condition Variable
– Doesn't have a value
– Implies no memory
– wait() always sleeps
– signal() will wake a sleeper
– signal() before wait() has no effect

Read First Block of a File

Assuming root directory is in memory
N times (N is directory depth):
Search directory for subdirectory inode
Read inode
Find next directory block
Read next directory block
2(number of dir) + 2(for file) = number of reads

From Deadlocks

Deadlock Situations

– Mutual exclusion: only one process at a time can utilize the resource.
– Hold and wait: a process holds a resource and does not release it until before acquiring another resource held by other processes.
– No-preemption: a resource can be released only voluntarily by the process holding it, not by another process waiting for it, until the task is completed.
– Circular wait: a set of processes holding resources and requiring resources in such a way that no process can finish their task because it is waiting for a resource held by another process.

Bankers Algorithm

• Handles resources with multiple instances
• process must claim – not request – resources a priori
• process may have to wait when a process requests a resource
• process must return all its allocated resources in a finite amount of time

1. Let Work and Finish be vectors of length m and n, respectively. Initialize
Work= Available and Finish[i] =false for i = 0, 1, ... , n - 1.
2. Find an index i such that both
a. Finish[i] ==false
b. Need; ::; Work
If no such i exists, go to step 4.
3. Work = Work + Allocation;
Finish[i] = true
Go to step 2.
4. If Finish[i] ==true for all i, then the system is in a safe state.

Deadlock Recover

• After deadlock has been detected several methods may be applied to handle deadlock situation:
a) Terminate all processes involved in the deadlock process.
b) Terminate one process at a time until deadlock cycle is eliminated.
c) Take away a resource from a process and assign it to another process.
• Method to recover from deadlock should cause least amount of harm to system or user processes.
– a rollback method may return process to previous safe state

From Filesystems

Disk Access – FIFO

arrival order, Completely fair, No starvation, Not very efficient

Shortest seek time first (SSTF)

Really shortest traversal first, Can cause starvation

Scan – Elevator

Really shortest traversal first, Can cause starvation

File Access

• Sequential
– Data is accessed in order
– Common in general-purpose systems
• Random
– Data is accessed in any order
– Common in general-purpose systems
• Content-Based
– Data is accessed based on content instead of position in file
– Common in database systems and TLB

Directory Implementation

• Linear list of file names with pointers to the data blocks.
– easy to program,
– requires more time to execute specific operations such as the search of a directory
• A hash table that stores file names and pointers to data blocks.
– more efficient for searching files and directories
– must handle collision where file names are mapped to the same location in the hash table

Linked Allocation

• Large files are now possible
• File growth is no longer limited
• Fragments are usable
• All files are slower to access
• Random access not possible
• Linked list can be broken by failure

Contiguous Allocation

• Large files are not handled well
• File growth is limited
• Fragments are unusable

Indexed Allocation

• Large files are no longer viable
• File growth is limited by user-defined max
• Fragments are usable
• All files are slower to access
• Random access as good as it gets
• Failure no longer a problem

MultiLevel Indexed Allocation

maximumBlocks=headerPointers*pointersPerBlock ^indirection level+...
maximumFileSize=maximumBlocks*bytesPerBlock

From Memory Management

Memory Allocation

– Best fit: Choose the smallest hole in memory that fits the process. Requires that the entire list of available memory holes must be searched.
– First fit: Choose the first memory hole in the list that is big enough for the new process.
– Worst fit: Choose the largest hole in memory. Also requires the entire list of available memory holes to be searched.

Logical Address -> Physical Address

• Given logical address (LA) and page size (PS)
– Find page number (P) P = LA / PS
– Compute offset in that page (Offset) Offset = LA % PS
– Lookup page frame number (F) in page table (PT) F = PT[P]
– Find physical address (PA) of offset in that page frame
PA = F * PS + Offset
EXAMPLE:
• LA = 12345 and PS = 4096
– Find page number (P) P = 12345 / 4096 = 3
– Compute offset in that page (Offset) Offset = 12345 % 4096 = 57
– Lookup page frame number (F) in page table (PT) F = PT[3] = 5
– Find physical address (PA) of offset in that page frame
PA = 5 * 4096 + 57 = 20480 + 57 = 20537

Page Tables

I. Hierarchical paging.
– Primary page table points to a secondary page table, which points to memory location.
– Virtual address contains two page numbers, one for outer page table, one for inner page table.
II. Hashed paging.
– Page table is a hash table with page number as the hash key.
– A linked list stores the page table entries mapped to the same hash table location.
– Linked list is searched for matching page number, to obtain frame number.
III. Inverted paging.
– Page table is searched for matching page number and process id.
– Location of matching page number and process id is the frame number.

Page Fault Handling

The page fault causes an OS trap. The OS locates free space in memory or evict from page table, loads missing page from disk into memory , then updates the page table and restart s the process (or puts it on waiting queue) .

FIFO Removal

• Toss oldest page
• Pro
– Fair - all pages live same amount of time
– Implementation – order pages
• Con
– Inefficient - tosses heavily used pages

Optimal Removal

• Toss page that won't be used for longest amount of time into the future
• Pro
– lowest number of page faults
• Con
– Implementation – Can't see into future

Least Recently Used (LRU)

• Least Recently Used
• Toss page that hasn't been use for longest time
• Assumes ``past performance indicates future behavior''
• Pro
– Good approximation of MIN
• Con

```
int sem_wait( sem ) {
  Disable_Interrupts();
sem.count--;
  if (sem.count < 0) {
    waitlist.enqueue(self);
    Enable_Interrupts();
    block();
Disable_Interrupts();
  }
Enable_Interrupts();
return 0;
}
```

```
int sem_post( sem ) {
  Disable_Interrupts();
sem.count++;
if (sem.count <= 0) {
    proc =
waitlist.dequeue();
    readyList.insert(proc);
  }
Enable_Interrupts();
return 0;
}
```

LOCK
- Kernel implementation of MUTEX
- Has two functions
  - Acquire() requests lock, only returns when lock is given to process
  - Release() gives lock back to kernel to reallocate to another process
- Presented as "Class/Object" for clarity
- Really implemented in C

```
class LOCK {
  int status = FREE;
}
LOCK::Acquire() {
Disable_Interrupts();
  while (status == BUSY) {
Enable_Interrupts();
Disable_Interrupts();
  }
status = BUSY;
Enable_Interrupts();
}
```

```
class LOCK {
  int status = FREE;
}
LOCK::Acquire() {
  ...
}
LOCK::Release() {
  status = FREE;
}
Disable_Interrupts();
Enable_Interrupts();
```

- Busy wait loop still present
  - Eats CPU time while waiting
- High priority thread waiting?
  - Holder (low priority) never runs
- Waiting thread could be "infinitely unlucky"

Corrections
- Add "Waiting List for Lock" (queue)
- Need enqueue() and dequeue() for list
- Must be able to put "self" on list
- Idea
  - If lock is BUSY,
    - put self on waiting list
    - go to sleep
  - When releasing lock,
    - wake a sleeping thread

FIFO
Non-preemptive
1st on CPU, when done, 2nd on CPU
+ Simple to implement (no switching)
+ Little overhead (no switching)
- Short jobs stuck behind long jobs

Round Robin
Preemptive version of FIFO
Ready list is queue
Quantum used to generate interrupts
+ Relatively simple
+ Fair: each job gets quantum
- Quantum size is tricky choice
- More overhead than FIFO

Shortest Job First (SJF)
Non-Preemptive
Schedule job with shortest CPU expected
+ Provably optimal (conditions)
+ Little overhead
- Poor response time
- Long jobs could starve

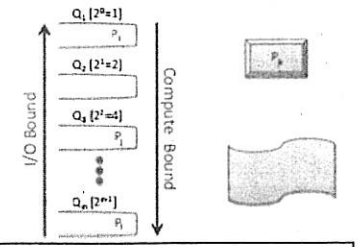
Shortest Remaining Time to Completion First
Preemptive version of SJF
Schedule job with shortest CPU expected
Preempt and reschedule when
- Job terminates
- Job arrives
- Internal event (yield)

Evaluation
+ Provably optimal AVG resp (conditions)
+ Short jobs preempt long jobs (Why?)
- Unfair
- Long jobs could starve

Multilevel Feedback Queuing
N queues/"levels" numbers 1 – N
Queue has priority and quantum
Quantum is "exponentially increasing"
Qi has priority i and quantum 2^(i-1)
Lower number is higher priority
All new processes start at priority 1
Priority altered based on process behavior

MLFQ Algorithm
Scheduler picks process from highest priority nonempty queue
– If Qi
is empty, try Qi+1
Process goes to CPU from Qi
If quantum expires
– Add process to Qi+1
If I/O initiated
– Move process to blocked list
– Return process to either Qi-1 or Q
Problem: Enough I/O & compute jobs starve
Unix "fix" (process aging)
- Set time when process added to Qi
- If time expires before service provided
  – Move process to Qi-1
  – Reset timer

MLFQ – Effect



SRTCF

Job	t_a	CPU	t_c	td	Resp	T/A
P_0	0	8	0	26	0	26
P_1	2	4	3	13	1	11
P_2	4	3	5	9	1	5
P_3	6	5	14	19	8	13
			AVERAGE		2.50	13.75
			OVERHEAD		6	

- Monitors are a high-level synchronization construct.
- Monitors allow safe sharing of abstract data types among concurrent processes.
- Monitors provide synchronized methods that can only be entered by one process or thread at a time.
- Monitors provide condition variables with two operations
  – wait: puts process or thread asleep
  – signal: triggers a waiting process or thread to resume operation exactly at the position where process or thread called wait

Comparison
Semaphore
– Has a value
– Implies a history
– wait() may sleep or may proceed depending on value
– signal() will increment a value and wake a sleeper
– signal() before wait() will effect result of wait

Condition Variable
– Doesn't have a value
– Implies no memory
– wait() always sleeps
– signal() will wake a sleeper
– signal() before wait() has no effect

```
sem_t fullB;
sem_t emptyB;
pthread_mutex_t mutex;
void * consumer(void *id) {
  int myID = (int)id;
  while(1) {
    sem_wait( &fullB );
    pthread_mutex_lock( &mutex );
    eatWidgetFromBuffer();
pthread_mutex_unlock( &mutex );
    sem_post( &emptyB );
  }
}
```

A: Initially, the value emptyB must be the size of the buffer and fullB must be 0.

```
Barber() {
while (1) {
  barber.Wait();
mutex.Acquire();
numWaiting--;
  mutex.Release();
done.Signal();
}
}
Customer() {
mutex.Acquire();
if (numWaiting < CHAIRS){
  numWaiting++;
mutex.Release();
barber.Signal();
done.Wait();
} else
    mutex.Release();
leave();
}
```

Assumptions:
- We have 4 queues
- 3 jobs arrive at time 0
- P_0: (2 CPU + 4 I/O) x 3
- P_1: (4 CPU + 2 I/O) x 3
- P_2: (20 CPU + 0 I/O) x 1

Lots of data to keep
Use an array w/ 4+Q columns

MLFQ

t	Q_1	Q_2	Q_3	Q_4	CPU	Wait	
0	P_1 P_2				P_0		All t_4
1	P_2	P_0			P_1		
2		P_0 P_1			P_2		
3		P_1 P_2			P_0		
4		P_2			P_1	$P_0$8	
6			P_1		P_2	$P_0$8	
8	P_0		P_1 P_2		P_0		
9		P_0	P_1 P_2		P_2		
10			P_2		P_1	$P_0$14	

t	Q_1	Q_2	Q_3	Q_4	CPU	Wait	
10			P_2		P_1	$P_0$14	
11					P_2	$P_0$14 $P_1$13	
13		P_1			P_2	$P_0$14	
14	P_0	P_1			P_2		
15		P_1		P_2	P_0		
16		P_0		P_2	P_1		
18			P_1	P_2	P_0		
19				P_2	P_1	$P_0$23	

t	Q_1	Q_2	Q_3	Q_4	CPU	Wait	
19				P_2	P_1	$P_0$23	
21					P_2	$P_0$23 $P_1$23	
23		P_1			P_2		$P_0$24
29				P_2	P_1		
31			P_1	P_2	P_1		
33					P_2	$P_1$35	
35					P_2		P_1t4
38							P_2t4

Background on Process Synchronization
- Concurrent access to shared data may result in data becoming inconsistency.
- Maintaining data consistency while supporting concurrent processing requires additional OS mechanisms
  – shared data may be accessed by multiple processes simultaneously
  – outcome of operation on shared data may depend on the order by which processes execute instructions leading to inconsistent results

```
int g;
void *tFunc(void *param){
  char *str = (char *)param;
  int loc = 0;
  g += 10;
  loc = g + 5;
  printf("%s: g %d: loc %d\n', str, g, loc);
  pthread_exit(0);
}
int main(int argc, char **argv){
  pthread_t tid1, tid2;
  g = 10;
  pthread_create(&tid1, NULL, tFunc, (void *)"ONE");
  pthread_create(&tid2, NULL, tFunc, (void *)"TWO");
  pthread_join(tid1, NULL);
pthread_join(tid2, NULL);
return 0;
}
```

Synchronization Concepts
Race Condition: situation in which the final outcome of a result is determined by the order of process execution.
– Different executions produce different results
– Almost always bad
– Scheduling allows order to be imposed

Critical Section: a code section in a process that accesses shared data.
– need to be protected using synchronization mechanisms to ensure data integrity
– while one process enters its critical section, no other process may enter their critical section to manipulate shared data

Solution to Critical Section Problem
- Mutual Exclusion: prohibit other processes from entering their critical section while a process executes its critical section
- Progress: if no process is in their critical section, there must be progress on any process that wants to enter their critical section.
- Bounded waiting: there must be a bound on the number of times other processes are allowed to enter their critical section after a process has made a request to enter its critical section.

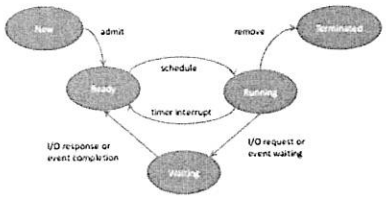
Critical Sections and Threads
- Threads must proceed through an entry and exit sections.
- Threads must not die or quit inside a critical section.
- Threads may be context switched inside a critical section.
- A context switch is different from an exit because another thread may not be allowed to enter their critical section.

Atomic Actions
- Once started, must complete
- Can't be interrupted in the middle
- Completes or doesn't start
- Statements (generally) NOT atomic
- Instructions are atomic

F Global variables are stored in the PCB, hardware
F Interrupts must be disabled immediately after a interrupt occurs.
T the mode bit changes from user to kernel mode when a system call is executed.
F Using the basic OS of a thread, stack is shared between 2 threads in a process.
T User-level threads require less overhead in the exec. at a context switch than kernel-level threads.
F Using the basic OS of a process data is shared between 2 processes.
T when an interrupt occurs, reg. values of a running process must be stored in its PCB.
F User-level threads are less susceptible to priority inversion than kernel-level threads.
T A soln. of the bounded buffer reqs in addition to a lock two semaphores to control the producers consumer
T The File descriptor table survives an exec() call.
A process shares data, code, & heap with its threads.

OS maintains a PCB (Process Control Block) for each process to keep track of a process in memory:
– Process ID,
– Process state (e.g. running, ready, waiting, ...),
– Program counter (address of next instruction),
– CPU registers,
– CPU scheduling information,
– File management (list of open files, working directory, ...),
– I/O status information,
– Memory management information (pointers to text, data, stack).

Process assume different states during execution life cycle:
New: the process is being created.
Running: the process is executed by the CPU.
Waiting: the process is waiting for some event to occur.
(e.g. an IO device returning some results)
Ready: the process is waiting to be executed by the CPU.
Terminated: the process has finished execution.



1. Kernel is aware of another process requesting to be started. No memory has been allocated.
2. PCB is allocated and initialized. Memory for the new process is allocated. Process is ready for execution.
3. Process is waiting, ready for access to CPU. OS CPU scheduler may select process for execution.
4. Process is selected for execution by scheduler. Alarm interrupt is set for now + quantum. Process is "placed" on CPU; state changes to running.
5. CPU fetches and executes process instructions. Process may loose CPU access in three different ways.
6. Quantum expires – Interrupt triggers kernel to resume control. Process is moved to ready list and next process is scheduled
7. Process exits – return from main(), calls exit(), exception. Still a process – resources remain allocated until deleted by OS..
8. Kernel reclaims resources and return value is given to parent. Could become a zombie process.
9. Process initiates input or output or could wait() for other processes. Process started something that will take a while.
10. Process waits for input or output to complete or for another process to wake it up. Process is not ready to continue execution.
11. Input/Output or wait() is completed; process is ready to be selected for execution.

• SP has address of top of runtime stack
• Runtime stack contains "activation records"
• Stack grows down
• Call to function:
– creates activation record for function
– pushes AR onto stack
Contains
• parameters
• local variables
• return value
• return address
• other items
– jumps to code for function

Function Calls:
1. Push arguments onto stack
2. Push local vars onto stack
3. Push return address onto stack
4. Push return value onto stack (created activation record)
5. Jump to user-defined code

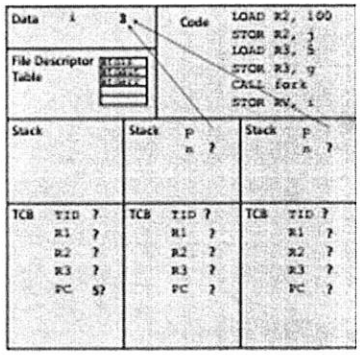
Limited in what it can do
Only access to user-mode instructions

Transitions to Kernel Mode
1. Software
• System call
• Called a "trap" into the kernel
• Jump to well-known function
• Hardware switches to kernel mode
• Example: call to printf() or read()
2. Exception
• Error state or debugging
• Similar to system call without return (error)
• Jump to well-known function
• Hardware switches to kernel mode
• Example: division by zero or segmentation • R
3. Hardware
• Called an "interrupt"
• Communication between kernel & devices
• Can occur between any two instructions
• Similar to system call without call
• Jump to well-known function
• Hardware switches to kernel mode
• Example: clock tick or I/O complete

Context Switch
When process Pi removed from CPU,
• Register values stored in PCBi
• PC stored in PCBi
• PCBi
state changed to ready
• Kernel schedules next process
– May be same process is only one is currently executing
– Still follows same rules
When process Pj returned to CPU,
• Register values restored from PCBj
• PCBj
state changed to running
• CPU mode changed to User-mode
• PC copied from PCBj to CPU
– Jumps back to process
– Last thing to happen

Threads Concept
• Definition: A thread (= job) is a lightweight process, representing a single unit of program execution within a process.
– shares code, data & file descriptor table
– has individual stack, registers & PCB (TCB)
• Two types of threads
– User-level threads: managed by library in user space;
library performs context switch
– Kernel threads: managed by kernel; kernel performs context switch
• Thread creation similar to process creation except
– Processes begins execution at main()
– Thread begins execution at specified function
– Creation call identifies initial function

Benefits of Multithreaded Programs
• More responsive: even if parts of the program are blocked, other parts may still be running.
• Resource sharing: threads within a process share the same address space and global variables.
• Economy: because threads share resources within a process, creating new threads is less costly in terms of time and memory consumption than creating new processes.



Process – Threads Differences
Threads share process resources, processes share computer resources.

☑ OS maintains per thread
program counter,
stack
state
registers

☑ OS maintains per processes
an address space,
list of open files,
child process,
signals and signal handlers
accounting

CPU Scheduling Overview
• Allocate CPU to the next process ready to execute.
• CPU scheduling is needed when:
– 1. Process switches from running to waiting state.
– 2. Process switches from running to ready state.
– 3. Process switches from waiting to ready state.
– 4. Process terminates.
• Switching a process from running to ready state or from waiting to ready state is carried out preemptively.
– Switch is not triggered by process but by the OS instead.

Optimization Criteria
Goal: Keep CPU as busy as possible.
Minimize waiting time
- Wait = sum of times process is in ready queue
Minimize response time
– Response = te – ta
Maximize throughput
– Throughput - # jobs completed per unit of time
– Throughput = 1 / Turnaround
– Minimize turnaround →Maximize throughput
Fairness
– Usually a trade-off

Preemptive
– Can take resource at any time
– Control passes back to kernel
– Internal and external events can cause
• Internal: yield(), exit(), etc.
• External: Quantum expiration

Non-Preemptive
– Can't take resource from process
– Process voluntarily gives up resource
– External events not allowed
• No quantum expirations
– Only internal events can cause
• System calls: yield(), exit()
– CPU: "Run 'till done"

```
Calling threadfunc()
L05 - Threads COP 4634 Fall 2012 31
int main(int argc, char ** argv){
  pthread_t tidA, tidB;
  g = 1;
printf("prethreads [%d]\n", g);
  pthread_create(&tidA, NULL, tFunc,
          (void *)"threadA");
  pthread_create(&tidB, NULL, tFunc,
          (void *)"threadB");
  printf("postthreads [%d]\n", g);
  pthread_join(tidA, NULL);
  pthread_join(tidB, NULL);
  printf("main done [%d]\n", g);
  return 0;
}
```

Process Creation, 1
• During boot, one process is created
– PID is 1
– Usually named "init"
• All other processes are descendents of init
• Processes form hierarchy
• Parents & children
• All children have one parent
• Parents can have any number of children
• Child process may share address space with parent process or create a new address space.
• System call to create new process

fork()
• Takes no parameters
• Creates copy of current proc
• New PCB, new memory, same content
• Memory copy: old to new
• PCB copy: old to new
• New PID

The fork()
• Local variables survive
• Global variables survive
• File descriptor table survives
• PCB
– Registers survive
– Stack pointer doesn't survive
– PC doesn't survive
– PID doesn't survive

```
int main(int argc, char ** argv)
{
  pid_t pid;
  printf("Output line 1\n");
  pid = fork();
  printf("Output line 2 with pid = %d\n", pid);
  return 0;
}
```

What is the output?
Assume parent is PID 94 and child is PID 223.

```
Output line 1
Output line 2 with pid = 0
Output line 2 with pid = 223
```

Child printed first

```
Normal Usage
L03 Processes COP4634 Fall 2012 64
int main(int argc, char ** argv)
{
  pid_t pid;
doCommonEntryCode();
pid = fork();
if (pid) {
   doParentCode();
  } else {
   doChildCode();
  }
doDuplicatedExitCode();
return 0;
}
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *path, const char *arg , ..., char *
const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
The exec family of functions replaces the current
process image with a new process image.
The initial argument for these functions is the
pathname of a file which is to be executed.
The const char *arg and subsequent ellipses in
the execl(), execlp(), and execle() functions
can be thought of as arg0, arg1, ..., argn. They
describe a list of one or more pointers to nullterminated strings that
represent the argument list
available to the executed program. The first argument
should point to the file name associated with the file
being executed. The list of arguments must be
terminated by a NULL pointer, and, since these are
variadic functions, this pointer must be cast (char *)
NULL.
What happens after exec()?
• Local variables don't survive
• Global variables don't survive
• File descriptor table survives
• PCB
– Registers don't survive
– Stack pointer doesn't survive
– PC doesn't survive
– PID survives
```