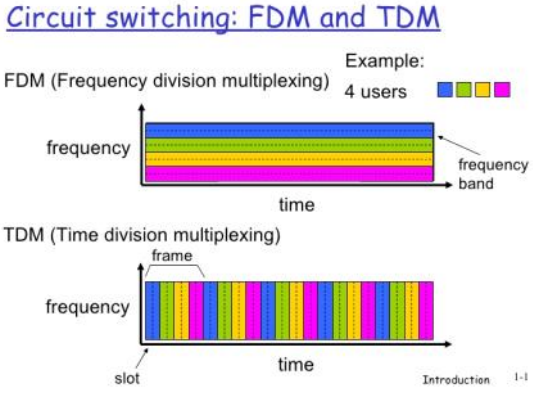
Week 1

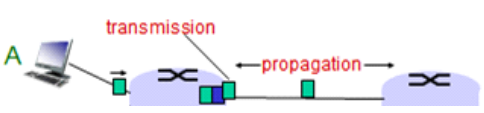
Saturday, 27 August 2016 3:05 PM

- **Network edges**
- DSL: voice/data transmitted over different frequencies over a dedicated line to that household goes to a central office: data goes to the ISP and voice goes to the telephone net
- Cable: Data/TV transmitted at different frequencies over a shared cable - unlike DSL which has dedicated access to the central office
- FTTH (fiber to the home): full optical fibre path to households
- Ethernet: enterprise access networks: multiple devices connect to an ethernet switch, data goes to an institutional router and institutional mail/web servers, router goes to ISP
- WAN
  - Wireless LAN: within building (100ft), 802.11b/g (WiFi), 11-54Mbps rate
  - Wide-area wireless access: Provided by cellular operator, 10's km range, between 1-10Mbps: 3G, 4G, etc.
- Host: sends data packets
  - Takes app message -> breaks into smaller chunks (packets length L bits) -> transmits packet into access network at rate R (bandwidth)
  - Packet transmission delay = time needed to transfer L-bit packet into link = L/R
  - PTD IS HOW LONG IT TAKES TO GET ALL THE DATA INTO THE WIRE (not across it).
- Physical media
  - Bit: propagates between transmitter/receiver pairs
  - Physical link: What lies between transmitter/receiver
  - Guided media: signals propagate in solid media: copper, fiber, coax
  - Unguided media: signals propagate magically, e.g. radio
- Physical media:
  - Twisted pair (TP): two insulated copper wires. Either 100Mbps or 10Gbps
  - Coaxial cable: Two concentric (one inside the other) copper conductors: multiple channels on the cable
  - Fiber optic: Glass fiber carrying light pulses: high speed, 10-100'sGbps - low error rate, repeaters are far apart and immune to electromagnetic noise
- Magical media: radio
  - Signal carried in electromagical spectrum
  - Propagation affected by: reflection, obstruction by objects, interference, wizards
  - Link types:
    - Terrestrial microwave: up to 45Mbps
    - LAN (e.g. Wifi): 11Mbps, 54Mbps
    - Wide-area: e.g. 3G, few Mbps
    - Satellite: Kbps up to 45Mbps channels, 270msec end-end delay, geosynchronous vs low-earth orbiting

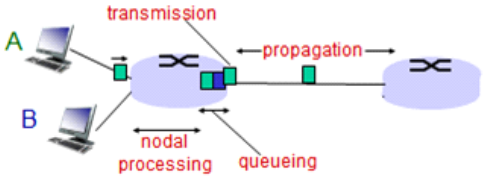


- **Network core**
  - Mesh of interconnected routers/switches
  - Two forms of switching: circuit switching (used in legacy telephone networks), packet switching (used in internet)
- Circuit switching: Reserves capacity along a path. Send reservation request, establish a circuit for its connection, send data, then sends a 'teardown circuit' message
  - FDM and TDM ->
  - Pros: Highly reliable- guaranteed bandwidth and service quality
  - Cons: Limited resources, inefficient, need to set up the circuit, can become unstable during a crisis
- Packet switching
  - Data is sent as chunks of formatted bits (packets) -> contain a header + payload
  - Packets a "forward" switched based on their header, each packet travels independently
  - PIPES
    - If arrival rate exceeds transmission rate, packets will queue at the switch and wait to be transmitted, packets can be dropped if memory fills up
  - Pros: Great for lots of data: resource sharing, no call setup
  - Cons: Excessive congestion possible: packet delay and loss
- Internet structure: network of networks
  - End systems connect to internet via ISPs -> ISPs are connected to global ISP groups -> global ISP groups connect via Internet Exchange Points. Content provider networks like Google can run their own networks

- **Delays and loss**
- Links
  - Bandwidth: capacity (bit/sec or bps)



- **Delays and loss**
- Links
  - Bandwidth: capacity (bit/sec or bps)
  - Propagation delay: time for data to travel length of link (seconds)
  - Bandwidth-delay-product (BDP): volume of link: amount of data that can be "in flight" at once -> propagation delay x bandwidth
- Packet delay sources
  - Nodal processing: check bit errors + determine output link: typically < msec
  - Queuing delay: Time waiting to be transmitted, depends on router congestion
  - Transmission delay:
    - L = packet length (bits)
    - R = link bandwidth (bps)
    - Transmission delay = L/R
  - Propagation delay
    - D: length of physical link
    - s: propagation speed in medium (~$2x10^8$ m/sec)
    - Propagation delay = d/s

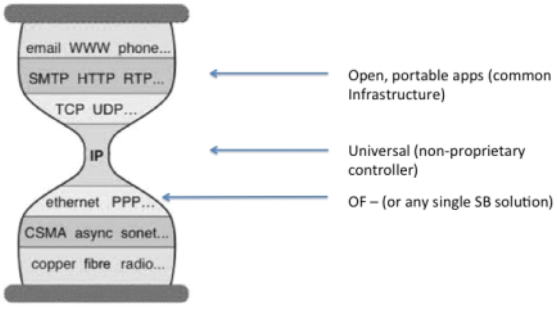


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Week 2

Saturday, 27 August 2016 5:45 PM

- **Protocol layers, service models**
- Modules in networking
  - Bits on wire (physical)
  - Packets on wire (physical)
  - Deliver packets within local network (datalink)
  - Deliver packets across global network (network)
  - Ensure packets get to the destination (transport)
  - Do something with the data (application)
- Protocol stack
  - Application: FTP, HTTP, SMTP, Skype…
  - Transport: TCP, UDP
  - Network: routing of datagrams from source to destination: IP, routing protocols
  - Link: data transfer between neighbouring network elements: Ethernet, 802.11 (Wifi)
  - Physical: bits on the wire
- Each layer depends on that below, supports layer above, independent of others
- Multiple versions in layer: interfaces differ somewhat
- Only one IP layer: unifying protocol
- Benefits: abstraction to hide detail, compatibility with different link types, easy to implement/debug
- Possible cons: layers may duplicate lower level functionality (e.g., error recovery to retransmit lost data), information hiding can hurt performance (e.g. packet loss due to corruption vs congestion), headers start to get big (e.g. TCP + IP + ethernet headers)
- What layers exist where?
  - Host: implements all layers (bits arrive on a wire and need to be translated into the application)
  - Router: Bits arrive (physical), packets must be delivered to next hop (datalink), routers participate in global delivery (network), but routers don't support reliable delivery (transport layer and above not supported)



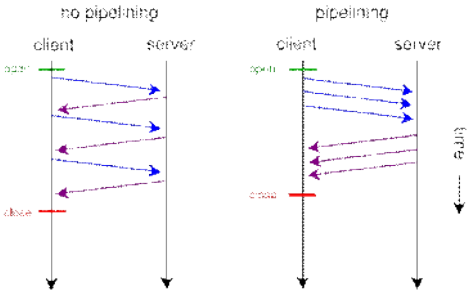
- **Network security**
- Malware: can get to hosts in the form of a virus (self-replicating infection by executing object) or worm (self-replicating infection by passively receiving object that gets itself executed)
- Spyware can record keystrokes, web history, and upload info to a collection site
- Botnets -> used for spam or DDoS attacks
- Packet sniffing: read and record data on shared ethernet/wireless
- IP spoofing: sending packets with false source address

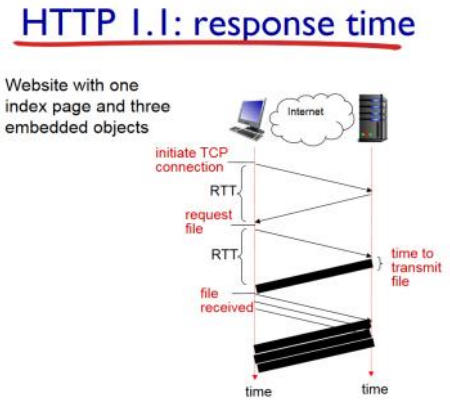
- **Application layer**
- Processes communicate via Inter-process communication (IPC)
  - On a machine: shared memory
  - Across machines: we need other abstractions (message passing)
- Sockets
  - Process sends/receives messages to/from its socket: analogous to a door
- Addressing processes
  - In order to receive messages a process has an identifier: this is the IP address of the host and the port number the process is associated with on that host
- Server:
  - exports a well-defined request/response interface
  - Long lived, always waits for requests
  - Carries a request out upon receiving it
  - Permanent IP address, static port conventions (http: 80, email: 25, ssh: 22)
  - May communicate with other servers to respond
- Client (user side):
  - Short-lived process that makes the request
  - May have dynamic IP
  - Does not communicate directly to other clients
- P2P
  - No always-on server
  - Arbitrary end systems (peers) communicate directly
  - Pros: self-scalable: new peers bring new service capacity, fast speed due to less contention, reliable, geographic distribution
  - Cons: decentralised- no shared memory or clock, possible mutually conflicting decisions
- App layer protocol defines:
  - The types of messages: request, response
  - Messages syntax: what fields are in the messages
  - Message semantics: the meaning of information
  - Rules for when and how processes send and respond to messages
  - Open protocols: HTTP, SMTP
  - Proprietary protocols: Skype

- **Transport layer**
- TCP:
  - Reliable transport

    - Flow control: sender won't overwhelm receiver
    - Congestion control: throttle sender when network overloaded
    - Does not provide timing, minimum throughput guarantee, or security
    - Connection oriented: setup is required between client and server processes
- UDP: unreliable data transfer, no reliability, flow control, congestion control timing, throughput guarantee, or security: nothing at all
- Security
    - TCP/UDP: no encryption, cleartext passwords
    - SSL: encrypted TCP connection, data integrity, end-point authentication
    - SSL is at the app layer: apps use SSL libraries which "talk" to TCP

- **Web**
- URLs: Uniform Record Locator
    - Protocol://host-name[:port]/directory-path/resource
- HTTP
    - Web's application layer protocol
    - Client: browser that requests/receives using HTTP protocol
    - Server: Web server sends objects in response to requests
    - Uses TCP: client initiates TCP connections (creating a socket) to server on port 80 -> server accepts TCP connection from client -> HTTP messages are exchanged between browser and web server -> TCP connection closed
    - Stateless- server maintains no info about past client requests
    - All text- relatively human readable, no issues with encoding, but not very efficient -> sending "12345678" as a string is 8 bytes instead of a 4 byte integer
    - GET: The request is appended to the URL (e.g. animalsearch?monkeys&banana)
        - Used when an operation can be applied multiple times without changing the result (e.g. getting a web page)
    - POST:
        - Input uploaded to server
        - Used when a request changes the state of the session or server database, when sending a request twice would be harmful (browsers warn against sending multiple POSTs), when users input non-ascii chars, when input is very large
- User-sever state: cookies
    - Used for authorization, shopping carts, recommendations, user session state (web email)
    - Permit sites to learn a lot about you, 3rd party cookies can follow you across sites
- Most web pages have multiple objects (eg HTML file + embedded images) - need to get these one at a time, new TCP connection for each small object
    - RTT: time for a small packet to travel from client to server and back
    - Non-persistent HTTP response time: one RTT to initiate TCP connection + one RTT for HTTP request and return + file transmission time -> non-persistent HTTP response time = 2RTT + file transmission time
    - Persistent HTTP
        - Server leaves connection open after sending response, subsequent messages are sent over the same connection
        - Increases congestion window
        - Without pipelining: client issues new request only after previous response is received, one RTT for each referenced object
        - W/ pipelining: the default in HTTP/1.1, client sends request as soon as it encounters a referenced object, as little at one RTT for all objects
- Web Caching
    - Objects are stored in a proxy server (cache) -> browser sends HTTP request to cache -> cache returns object is it contains it, or else cache requests object from origin server that then returns object to client
    - Cache is both a client and a server
    - Pros: reduces response time for client request, reduced traffic on an institutions access link





- **FTP**
- FTP client contacts FTP server using TCP, client is authorised over a controlled connection
- When the server receives a command from the client, the server opens a second TCP data connection is opened TO the client to transfer the file: "out of bound" connection
- Maintained state: current directory, authentication

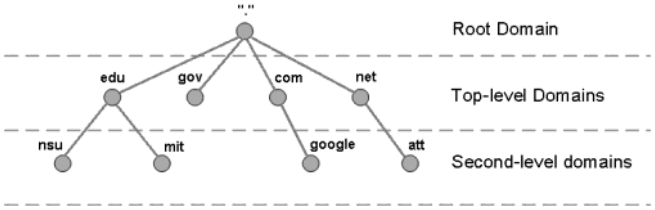
Week 3

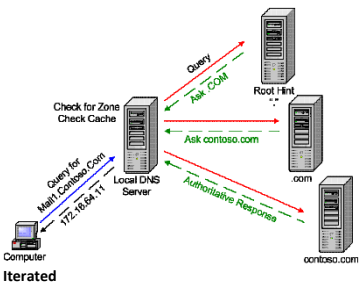
Sunday, 28 August 2016 2:51 PM

- **Mail**
- Three major components: user agents (e.g. Outlook, Thunderbird), mail servers, and SMTP (simple mail transfer protocol)
- Servers contain: mailbox (contains incoming messages for user), message queue (outgoing messages to be sent), and the SMTP protocol (between mail severs to send email)
- SMTP
  - Uses TCP to send email from client to server: port 25
  - Direct transfer: sending server to receiving server
  - Three transfer phases: handshake, transfer, closure
  - Command/response interaction: commands are ASCII text, response is a status code + phrase
  - Message must be in 7-bit ASCII - attachments are encoded into 7-bit ASCII
  - Persistent connections
  - Multiple objects sent in a multipart message: unlike HTTP where each object is encapsulated in its own response message
- Phishing
  - Spear phishing: attempts directed at specific individuals or companies, attackers gather personal inform (social engineering) to increase success, most popular type of phishing (over 90% of all attacks)
  - Clone phishing: Taking a legitimate email that has an attachment and cloning it, but replacing the attachment with a malicious object, and then sent from an email address spoofed to look like the original sender
- Format
  - Header: To, From, and Subject header lines
  - Body: the message: ASCII characters only
- Receiver accessing mail from their mail server: use POP (Post Office Protocol), IMAP (Internet Mail Access Protocol), or HTTPS (like Gmail or Outlook)
- POP3
  - Stateless across sessions
  - "Download and delete" mode: user cannot re-read emails if they change client
  - "Download and keep" mode: there are copies of messages across different clients
- IMAP
  - Keeps all messages in one server
  - User can organise messages in folders
  - Keeps user state: like names of folders

- **Socket programming**
- With UDP:
  - No connection between client and server -> no handshaking, sender attaches IP destination and port number to each packet
  - Data may be lost or out of order
- With TCP:
  - Client must contact server to establish connection, server must have a socket that welcomes the client
  - Client makes TCP socket and specifies IP address + port of the server process -> client TCP establishes connection to server TCP -> server TCP creates new socket to communicate with that particular client

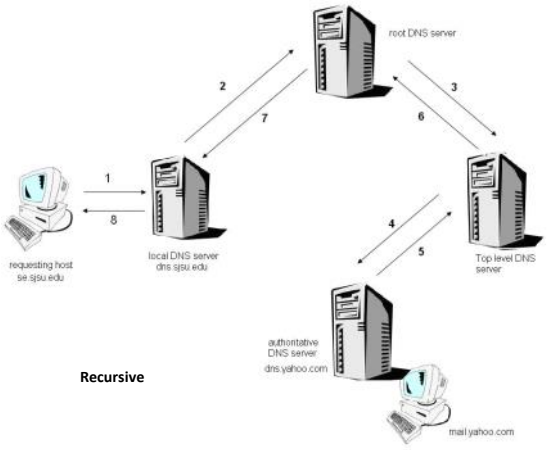
- **DNS (Domain Name System)**
- Distributed database: a hierarchy of many name servers
- How do we get the IP address?
  - Browser passes hostname to the DNS application on your machine -> DNS client sends query to map the hostname to an IP into the DNS hierarchy over UDP -> DNS client receives a reply -> browser can now initiate TCP connection with the server process at this IP address
- Services:
  - Hostname -> IP translation
  - Host aliasing (creating an alias name for IP addresses)
  - Load distribution (replicated web servers: many IPs to one name)
- Administrative authorities responsible for portions of the hierarchy
- Levels
  - Top: Root servers (location hardwired into other servers)
  - Top level: .com, .edu, etc - managed professionally
  - Bottom level- Authoritative DNS servers: actually store the name-to-address mapping, maintained by corresponding authority
- Each server needs to know about other servers responsible for other portions of the hierarchy: Every server knows root, root server knows about all top-level domains
- 13 root servers around the world
- Anycasting
  - Routing finds shortest paths to destination
  - If several locations have the same address, the network delivers the packet to the closest location
- TLD (top-level domain servers)





Iterated

    - Responsible for com, org, net, edu, etc. and top-level country domains, e.g. uk, fr, ca, jp
- Local DNS name server
    - Not strictly in hierarchy
    - Each ISP has one (called the default name server)
    - Host DNS queries are sent to the host's local DNS server -> local server has a cache of recent name-to-address mappings
- Once any name server learns a mapping, it caches it -> disappears after some time (TTL)
    - TLD servers normally cached in local name servers
- DNS record types - RR (resource record) format: (name, value, type, ttl)
    - A: Name is a hostname, value is an IP
    - NS: name is a domain (e.g. foo.com), value is the hostname of the authoritative name server or this domain
    - CNAME: Name is an alias for the real name (e.g. ww.ibm.com is actually servereast.backup2.ibm.com), value is the canonical (real) name
    - MX: Value is the name of the mailserver associated with name
- DNS query and reply messages have the same format: identification + flags, questions, answers, authority and additional info
- Inserting records
    - New startup -> register name at a DNS registrar -> create an authoritative server of type A and MX
- UDP usually used for queries
- Authoritative DNS server monitors the load for its multiple web servers and replies with the IP of the least loaded one -> also considers client location
- Reverse DNS used in troubleshooting tools, tracking users, etc.
- Trust: censorship can happen

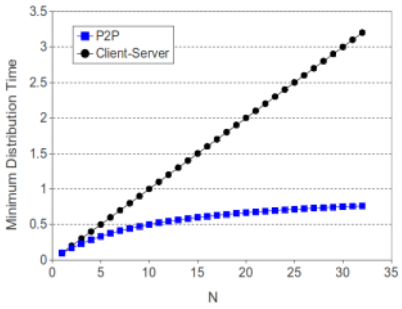


Recursive

- **Content Distribution Networks**
- Storing copies of videos/content at multiple geographically distributed sites

Week 4

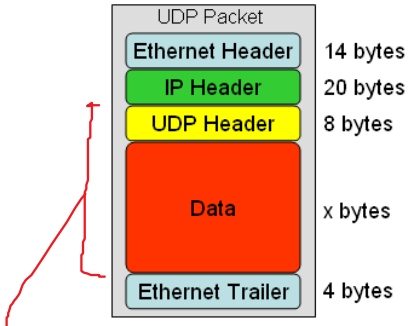
Sunday, 28 August 2016 4:47 PM

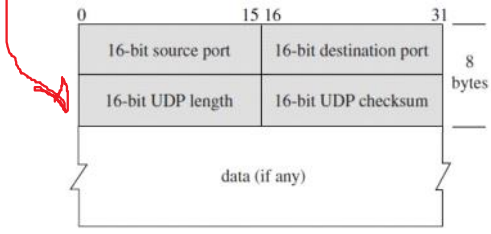


a

- **Peer-to-peer**
- No always on server, arbitrary end systems directly communicate, peers connected
- Tracker: tracks the peers participating, peers obtain connections from the tracker
- .torrent file contains the address of tracks and a list of the file chunks and their cryptographic hashes -> ensures the chunks aren't modified
- Chunk requests
  - Different peers have different subsets of file chunks -> a user periodically asks other peers for a list of the chunks they have -> user requests missing chunks from those peers, rarest chunks first
  - Users send chunks to those four peers sending them chunks at the highest rate -> re-evaluated every 10 seconds
  - Every 30 seconds, randomly select another p eer to start sending chunks to
- Distributed Hash Tables (DHTs) distribute tracker info
  - e.g.: key is the file name, value is the BT tracker peers
  - Distribute (key, value) pairs over millions of peers -> peer queries a DHT with that key to get the value
  - Assign integer id to each peer (e.g. node ID is a hash of the user's IP address)

- **Transport Layer**
- Network layer: finds paths through network, routing from one host to another
  - Does not guarantee paths or reliability
  - Logical communication between hosts
- Transport services: provide logical communication between app processes on different hosts, transport protocols run in the end systems
  - Logical communication between processes
  - Relies on and enhances network layer services
- Network layer: Multiplexing/demultiplexing
  - Multiplexing at sender: handle data from multiple sockets + add transport header
  - Demultiplexing at receiver: use header info to deliver received segments to correct socket
    - Host receives IP datagrams (datagram has source IP + destination IP), each datagram has one transport-layer segment, each segment has a source + destination port number
    - Host uses IP addresses and port numbers to direct segment to the right socket
- Connectionless demultiplexing: UDP segments must specify the destination IP and destination port #. When receiving a UDP segment, the host checks the port # in the segment and directs it to the port with that #
- Connection demux:
  - TCP socket identifies by source IP, source port #, destination IP, destination port #
  - Receiver uses all four values to direct the segment to the right socket



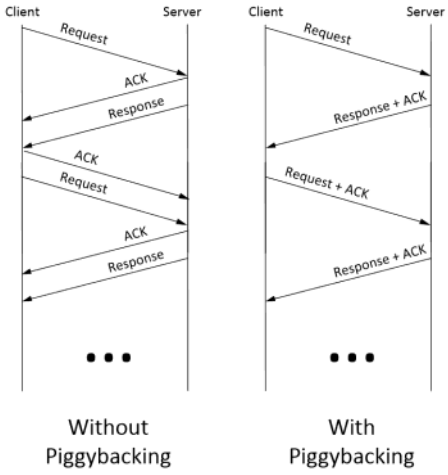


- **Transport layer: UDP**
- "Bare-bones" transport protocol, packets may be lost or delivered unordered
- No connection
- Pros: no connection establishment (adds delay), simple (no connection state at sender/receiver), small headers, no congestion control (UDP can blast away as fast as desired)
- Checksum: addition of segment contents (as a one's complement sum): sender puts checksum in UDP header, receiver adds segments together and verifies that the result is 1111 1111 1111 1111

Week 5

Sunday, 28 August 2016 10:00 PM

- **Transport layer**
- Pipelining: sender allows multiple "in-flight" packets
  - Go-back-N: sender can have up to N unacknowledged packets in pipeline, receiver sends a cumulative ack - sender only has timer for the oldest unacked packet
    - Waits for a timeout on the last unacked packet - will resend all packets including and after the packet it timed out for
  - Selective repeat: Sender has up to N unack'd packets in timeline, receiver sends an individual ack for each packet, there's a timer for each unacked packet
    - Only resends packets for which it got a timeout
- **Piggybacking**
  - Usually both sides of the connection are sending data
- **TCP**
  - RTT and timeout
    - EstimatedRTT = (1-a)*EstimatedRTT + a*sampleRTT Typically a = 0.125
    - Timeout interval = EstimatedRTT plus a "safety margin"
    - DevRTT = (1-b)*DevRTT + b*(SampleRTT - EstimatedRTT) Typically b = 0.25
  - Fast retransmit
    - Duplicate acks are a sign of isolated loss - 3 duplicated acks will cause the sender to resend the packet corresponding to that ack
  - Flow control
    - Receiver "advertises" free buffer space by including a "rwnd" value in its TCP header
    - Sender limits amount of unacked (in flight) data to receiver's rwnd value
    - Guarantees receive buffer won't overflow
  - Changing ISN (initial sequence number)
    - Makes it harder to hijack a TCP connection
    - Prevents the potential (rare chance) interference of still in-flight data sent from the port and IP from an earlier connection
  - Handshake: SYN, SYN+ACK, ACK
    - What if the initial SYN packet is lost? Sender will typically resend it after 3 seconds
  - Closing: FIN, ACK, FIN, ACK
  - SYN attack
    - Attacker creates a fake SYN packet with IP address of the victim (usually a server), source is spoofed
    - Victim allows buffer and sends a SYN ACK to the spoofed address
    - ACK never comes back, after a timeout the connection is freed
    - Doing this on a large enough scope can overwhelm the server
    - Solutions:
      - Increase size of connection queue
      - Decrease timeout wait for the 3-way handshake
      - Firewalls: list of known bad source IPs
      - TCP SYN cookies
  - TCP SYN cookie
    - Server gets syn
    - Instead of immediately creating a connection, it creates an ISN hashed from the source IP, dest IP, and port number of the SYN packet
    - Replies back with a SYNACK containing the ISN
    - If the original SYN is genuine, an ACK will come back -> receiver runs the hash on the sequence number of the ACK packet and checks that the ACK is equal to ISN + 1
    - Only creates a connection if the above is true



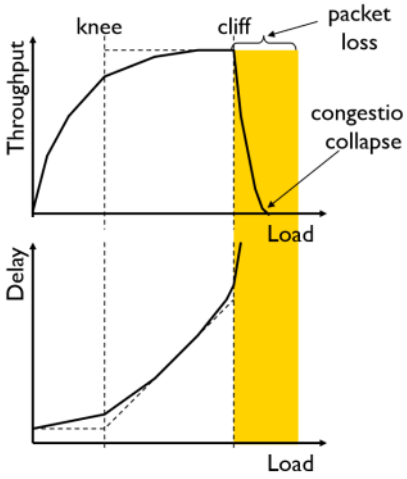
Without Piggybacking | With Piggybacking

Week 7

Tuesday, 27 September 2016 6:21 PM

- **Congestion**
  - Too much stuff):<
  - Congestion control is about bandwidth allocation and traffic regulation
  - Different from flow control (an agreement between the sender and receiver)
  - Results in lost packets and long delays

- Knee – point after which
  - Throughput increases slowly
  - Delay increases fast
- Cliff – point after which
  - Throughput starts to drop to zero (congestion collapse)
  - Delay approaches infinity

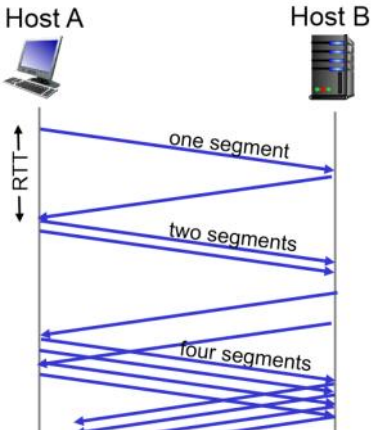


- **Two approaches to congestion control:**
  - End-end congestion control
    - No explicit feedback from network, congestion inferred from end-system observed loss and delay -> Used by TCP
  - Network-assisted congestion control
    - Routers provide feedback to end systems- single bit indicating congestion
    - Explicit rate for sender to send at
- **TCP's approach**
  - TCP connection has a window (cwnd) controlling the max packets in flight
  - TCP sending rate: send cwnd bytes, wait RTT for acks, then send more bytes

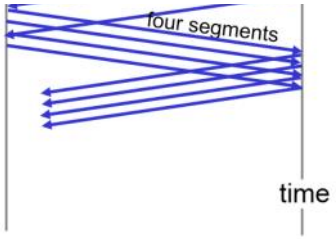
$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

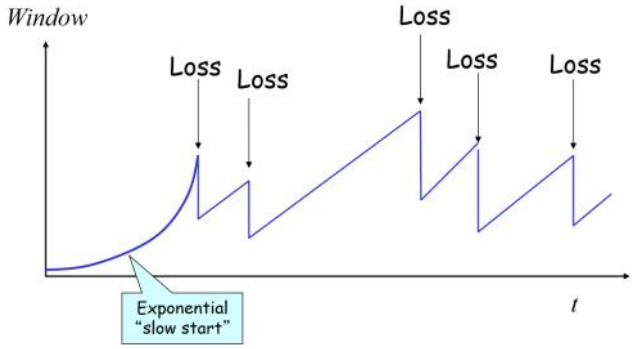
- **Windows**
  - Congestion window: CWND
  - Flow control window: RWND
  - Sender-side window = min(cwnd, rwnd)
- **Detecting congestion**
  - Packet delays
    - Tricky: delay varies
  - Routers tell end-hosts they're congested- required explicit feedback
  - Packet loss
    - Fail-safe signal that TCP detects
    - Complication: non-congestive loss, like checksum errors
    - Duplicate acks: isolated loss
    - Timeout: more serious, not enough dup acks -> must have suffered several losses
- **Slow start**
  - Starts slowly and ramps up exponentially
  - Initially cwnd = 1 MSS
  - Double CWND every RTT
- **Adjusting to varying bandwidth**
  - Available bandwidth will vary -> sender will probe (increase rate) and backoff (rate decrease)
  - Known as Congestion Avoidance (CA)
- **AIMD**
  - Sender increase transmission rate until loss occurs
    - Additive increase: increase cwnd by 1 MSS every RTT until loss detected
    - Multiplicative decrease: cut cwnd in half after loss

Slow start

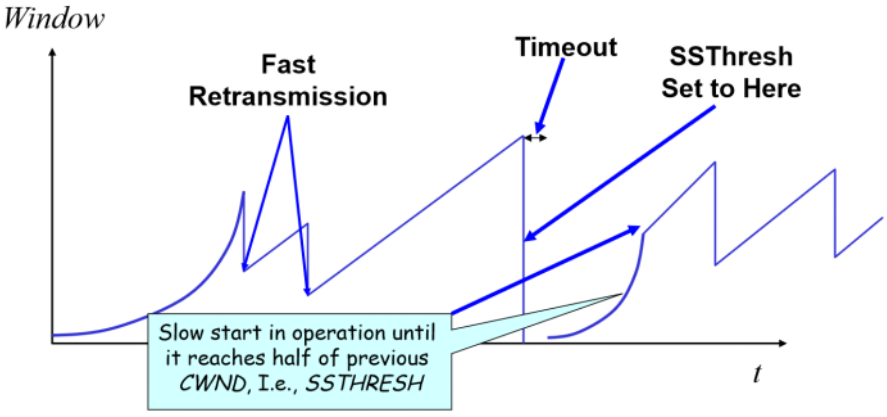


- Sender increase transmission rate until loss occurs
  - Additive increase: increase cwnd by 1 MSS every RTT until loss detected
  - Multiplicative decrease: cut cwnd in half after loss





- **Slow start vs AIMD**
  - When to use slow start versus additive increase?
  - "Slow start threshold" - ssthresh
    - Initialized to a large value, on timeout, ssthresh = cwnd/2
  - When cwnd = sssthresh, sender switches from slow start to AIMD
- **Timeout**
  - On timeout, cwnd = 1
- **Duplicate acks**
  - If 3 duplicates received, ssthread = cwnd/2, cwnd = cwnd/2

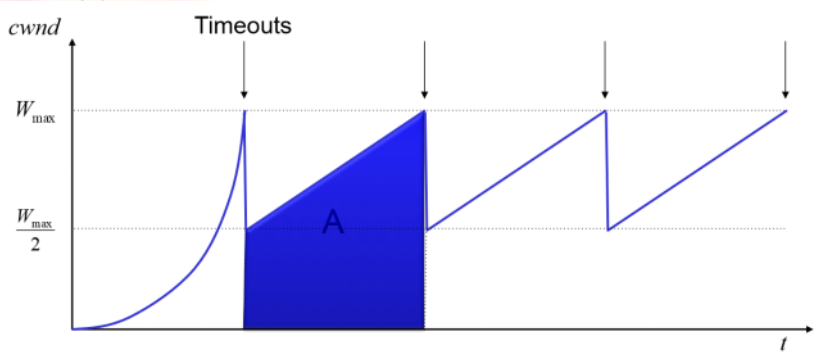


- **Fast recovery**
  - Problem: say with a window of 10, sender will send 10 packets. If packet 1 is dropped, the sender will get 10 acks for packet 1 back, even though it'll fast retransmit on the 4th ack. We won't be able to continue increasing the cwnd until we finally go through the next 6 duplicates and get a new ack -> this stalls the process
  - Solution: grants the sender temporary "credit" for each dupack received in order to keep packets in flight
  - If dupACKcount = 3,
    - ssthresh = cwnd/2
    - cwnd = ssthresh + 3
  - While still in fast recovery, cwnd++ for each duplicate ack
  - Exit fast recovery after getting a new ack, then send cwnd to ssthresh
- **TCP flavours**
  - Tahoe: cwnd = 1 on triple duck ACK and timeout
  - Reno: cwnd = 1 on timeout, cwnd = cwnd/2 on triple dup ack
  - NewReno- Reno with improved fast recovery

Week 7 part 2

Tuesday, 27 September 2016 7:41 PM

- **Fairness**
  - If K TCP sessions share the same bottleneck link of bandwidth R, each should get an average rate of R/K
- **TCP Throughput**



Packet drop rate, $p = 1/A$, where $A = \frac{3}{8}W_{max}^2$

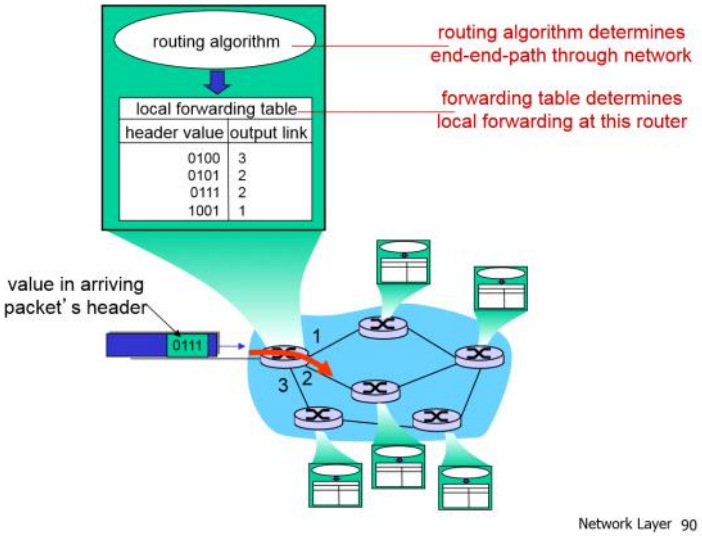
Throughput, $B = \frac{A}{\left(\frac{W_{max}}{2}\right)RTT} = \sqrt{\frac{3}{2}}\frac{1}{RTT\sqrt{p}}$

Congestion Control 67

  - Flow is inversely proportional to RTT -> TCP is not fair with heterogeneous RTTs!

- **Adapting TCP to high speeds**
  - Once past a threshold speed, increase cwnd faster
  - Other approaches: multiple simultaneous connections, router-assisted approaches
- **Equation-based Congestion Control**
  - TCP throughput is chopping -> swings between w/2 and w
  - Some apps prefer steady rates, like for streaming
  - Equation-based congestion control: ditch TCP's increase/decrease rules and just follow the throughput equation above, measure drop percentage p and set rate accordingly
- **Problems**
  - TCP can confuse corruption with congestion
  - Short flows never leave slow start
  - With short flows there can be too few packets to actually cause dupacks -> isolated loss can lead to timeouts
  - Cheating is possible (increasing cwnd faster than +1 MSS every RTT, or opening too many connections, or a large initial cwnd)
- Two key functions of the network layer:
  - Forwarding: move packets from router's input to router output
  - Routing: determine route taken by packets from source to destination

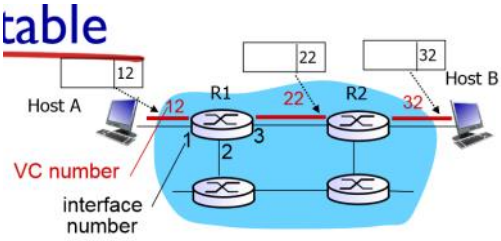
Interplay between routing and forwarding

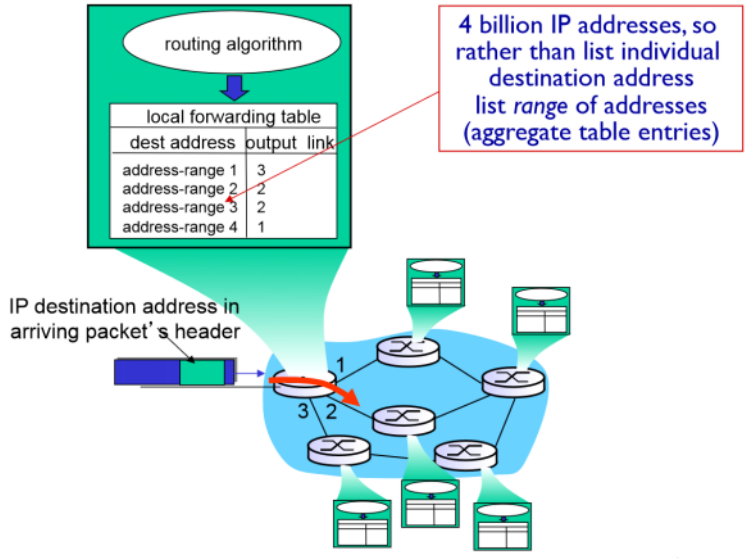


Network Layer 90

- **Connection setup**
  - Before datagrams flow, two end hosts AND intervening routers establish a virtual connection
  - Network vs transport layer connection service:
    - Network: between two hosts
    - Transport: between two processes
- **Service models for networks**
  - Example of service model for individual datagrams:
    - Guaranteed delivery
    - Guaranteed delivery with less than 40msec delay
  - Example service for a flow of datagrams:
    - In-order datagram delivery
    - Guaranteed minimum bandwidth flow
    - Restrictions on changes in inter-packet spaces
- **Connection, connection-less service**
  - Datagram network providers network-layer connectionless service
  - Virtual-circuit network provides network-layer connection service
  - Analogues to TCP/UDP services
- **Virtual circuits**
  - Source-to-dest path that behaves like a telephone circuit
  - Call setup and teardown for each call before data can flow
  - Each packet carries a VC identifier
  - Every router on source-dest path maintains "state" for each passing connection
  - Link, router resources (bandwidth and buffers) may be allocated to the VC
  - Has a forwarding table with the columns: Incoming interface, incoming VC#, outgoing interface, outgoing VC#
  - VC routers maintain connection state information (i.e. which VC# to expect next, where they'll be sent)
- **Datagram networks**
  - No call setup at network layer
  - Routers have no state about end-to-end connections
  - Packets are forwarded using the destination host address

Virtual circuits



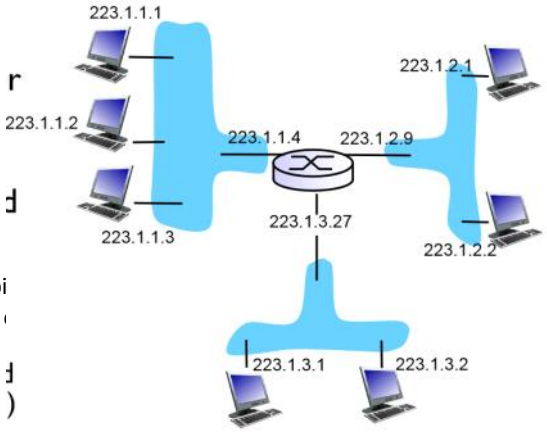


Network Layer 102

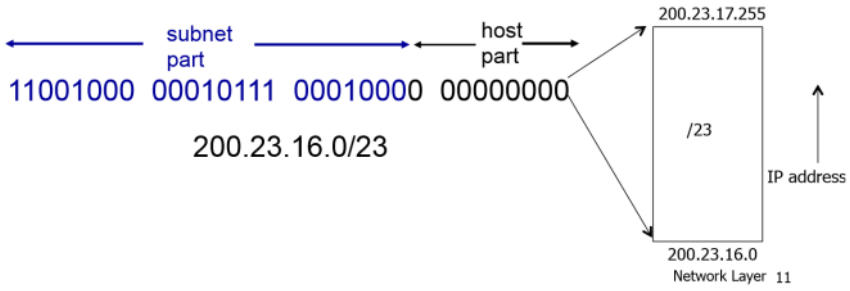
- **Datagram vs VC**
  - Datagram:
    - Elastic service, no strict timing required
    - Many link types
    - "Smart" end systems - can adapt, perform control, error recovery
  - VC
    - Evolved from telephony
    - Like human conversation: strict timing, reliability requirements, need for guaranteed service
    - Dumb end systems- complexity is inside the network

Week 8

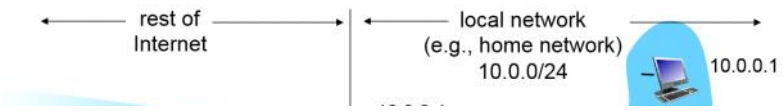
Wednesday, 28 September 2016 4:44 PM

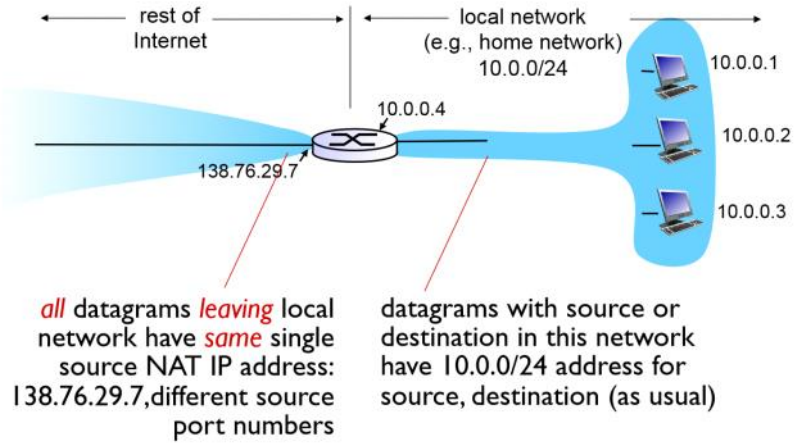


- **IP addressing**
  - IP address: a 32 bit identifier: router interface
  - Interface: connection between host/router and physical link
  - IP addresses associated with each interface
- **Subnet**
  - IP address: subnet part is the high order bits, host part is the low order bi
  - Subnet: device interfaces with same subnet part of the IP that can reach router (on the right, 223.1.3.X, 223.1.1X, and 223.1.2.X are subnets)
- **Today's addressing: CIDR**
  - Classless InterDomain Routing
    - Subnet portion of address is arbitrary length
    - Address format a.b.c.d/x, where x is # of subnet bits



- **Subnet Address**
  - Subnet Mask
    - Used in conjunction with the network address to indicate how many higher order bits are used for the network part of the address (check slide 12)
  - Broadcast Address
    - Host part is all 1111's, e.g. 223.1.1.255
  - Subnet address
    - Host part all 0000's, e.g. 223.1.1.0
- **IP addresses**
  - How does a host get one? -> hard-coded by system admin in a file
  - DHCP: Dynamic Host Configuration Protocol
    - Allows host to dynamically obtain an IP address from a network server when it joins the network
      - Can renew its lease on the address
      - Allows reuse of address
      - Support for mobile users who join network
    - Overview:
      - Host broadcasts "DHCP discover" msg
      - DHCP server responds with "DHCP offer"
      - Host requests IP address with "DHCP request"
      - DHCP server sends address with "DHCP ask"
    - Also provides:
      - The address of first-hope router for client
      - Name and IP of DNS server
      - A network mask (that indicates the network vs host portion of the address)
  - Network gets the subnet part of its IP address from the ISP
  - How does an ISP get a block of addresses?
    - ICANN: Internet Corporation for Assigned Names and Numbers
      - Allocates addresses, manages DNS, assigns domain names, resolves disputes
- **NAT: network address translation**



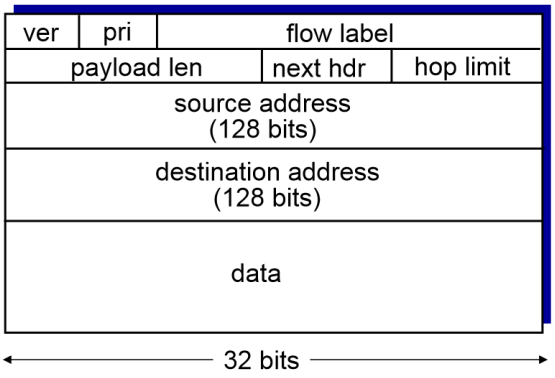


- **NAT router must:**
  - Outgoing datagrams: replace source IP address and port # to a NAT IP address and new port #
  - Remember (in a NAT translation table) every (source IP address, port #) -> (NAT IP address, new port #) translation pair
  - Incoming datagrams: replace (NAT IP address, new port #) with (source IP address, port #)
- **NAT advantages**
  - Range of IP address not needed from ISP: just one IP address for all devices
  - Can change addresses of devices in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
- **NAT issues**
  - Controversial: routers should only process up to layer 3, violates end-to-end argument (e.g., NAT possibility must be taken intoac count by P2P app designers)
  - NAT modifies port # and IP address: requires recalculation of TCP and IP checksum
  - Some apps embed IP address or port numbers in their message payloads
  - If applications change port numbers periodically, NAT must be aware of this
  - NAT traversal problems: how to setup a server behind a NAT router, how to talk to a Skype user behind a NAT router?
- **ICMP: Internet Control Message Protocol**
  - Used by hosts and routers to communicate network-level info (error reporting, echo requests/replies)
  - ICMP messages are carried in IP datagrams
  - ICMP message: type, code plus first 8 bytes of IP datagram causing error

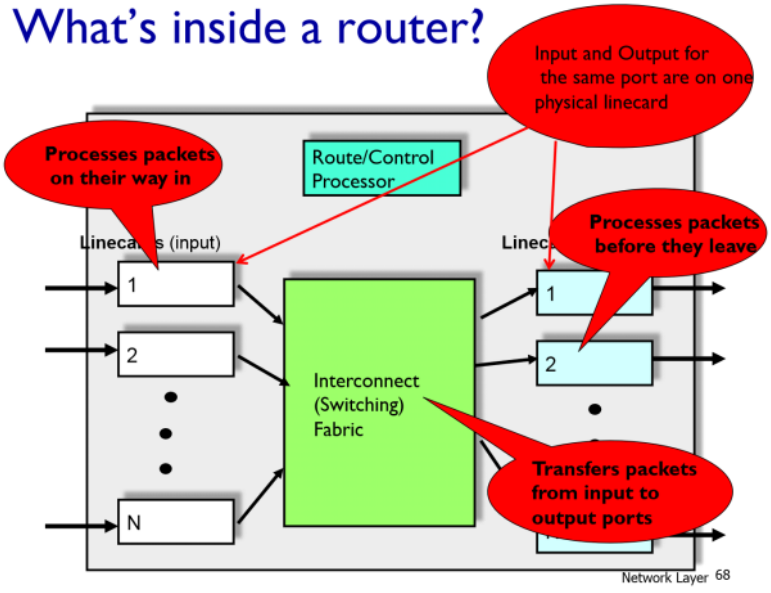
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

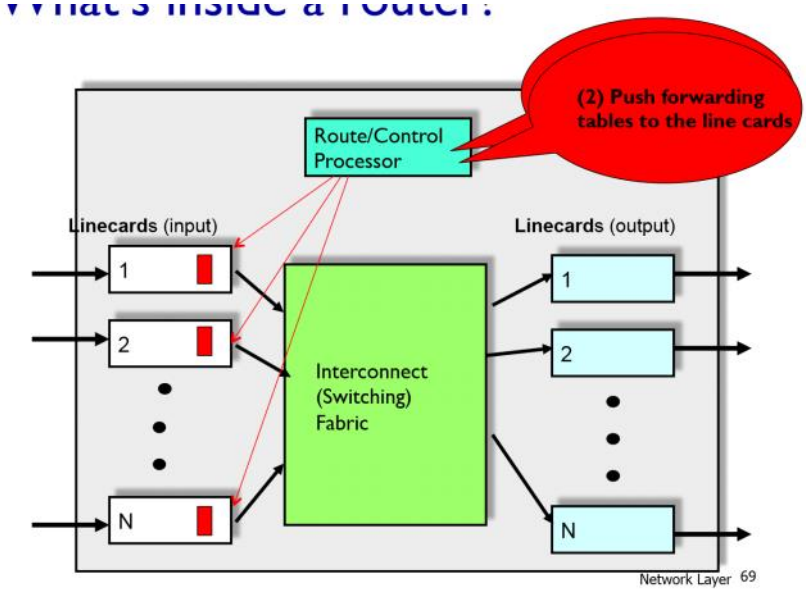
- **IPv6 motivation**
  - Initial motivation: 32-bit address space soon to be completely allocated
  - Header format helps speed processing/forwarding

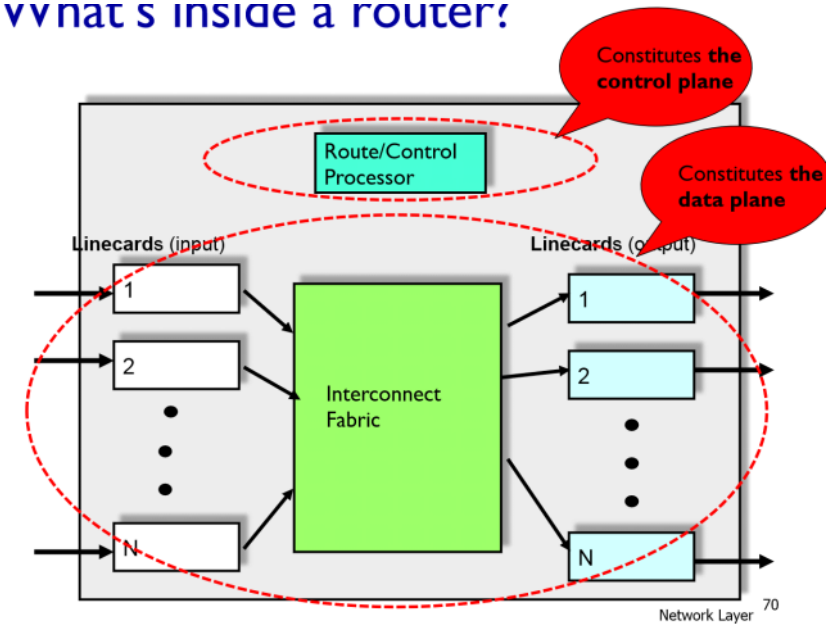
- **IPv6 datagram format**
  - Priority: identify priority amongst datagrams in flow
  - Flow label: identify datagrams in the same "flow"
  - Next header: identify upper layer protocol for data



- **IPv6: other changes**
  - Checksum removed entirely to reduce processing time
  - Options: allowing, but outside header, are indicted by "Next Header" field
  - ICMPv6: new version of ICMP
    - Additional messages, e.g. "Packet too big"
    - Multicast group management functions
- **Transition from IPv4**
  - Not all routers can be upgraded simultaneously
  - Tunnelling: IPv6 datagram carried as a payload in IPv4 datagram amount IPv4 routers
- **Routers**







- **Input Linecards**
  - Tasks
    - Receive incoming packets
    - Update the IP header
      - TTL, Checksum, Options (maybe), Fragment (maybe)
    - Lookup the output port for the destination IP address in the Forwarding Table using Longest Prefix Matching
    - Queue the packet at the switch fabric
- **Output linecard**
  - Packet classification: map each packet to a "flow"
  - Buffer management: decide when and which packets to drop
  - Scheduler: decide when and which packets to transmit
  - Used to implement policies, e.g. denying all email traffic from ISP-X to Y
- **Simple: FIFO Router**
  - No classification
  - Drop-tail buffer management: when buffer is full, drop the incoming packet
  - FIFO scheduling: schedule packets in the same order they arrive
- **Packet classification**
  - Classify an IP packet based on a number of fields in the packet header
- **Scheduler**
  - One queue per "flow"

    - Scheduler decides when and from which queue to send a packet
    - Goals of scheduling algorithm: speed, but mostly depends on the policy being implemented (fairness, priority, etc)
- **Bursty loss from drop-tail queuing**
    - TCP depends on packet loss
    - Drop-tail queueing leads to a loss in bursts
        - If link is congested, many packets encounter a full queue
        - Thus, loss synchronisation: many flows lose one or more packets, in response, many flows divide sending rate in half
- **Slow feedback from drop tail**
    - Feedback comes when buffer is completley full
    - As the buffer fills, RTT is increased, making detection even slower
    - Might be better to give early feedback about congestion and to get some connections to slow down before it's too late
- **Random early detection (RED)**
    - Basic idea of RED:
        - Router notices that queue is getting backlogged and randomly drops packets to signal congestion
    - Packet drop probability: drop probability increases as queue length increases - if queue length is stable, drop probability is a function of average queue length and time since the last drop

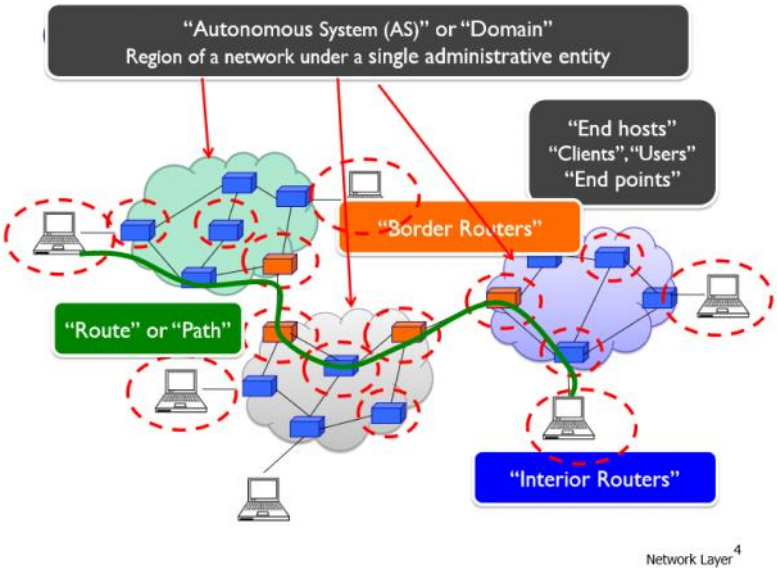
    - Drops packets before the queue is full
    - Drops in proportion to each flow's state (high rate flows have more packets -> more chance of being selected)
    - Drops are spaced in time, desynchronises the TCP senders
    - Tolerant of burst in traffic by basing decisions on average queue length

    - Problems:
        - Hard to get the parameters just right
        - Has mixed adoption in practice

Week 9

Thursday, 20 October 2016 11:54 AM

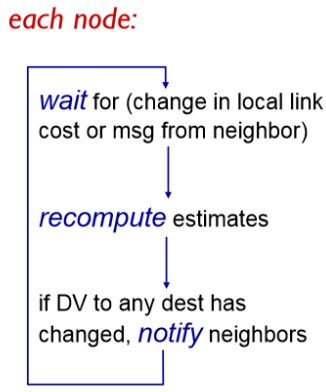
- **Routing**
  - Routing algorithms (in the router) determine end-end paths through networks, forwarding table determines the local forwarding at the router



  - Internet routing works at two levels:
    - Intra-domain: Each Autonomous System runs an intra-domain routing protocol that establishes routes within the domain
    - Ass participate in the inter-domain routing protocol that establishes routes between domains
- **Links**
  - Link cost is typically simple: all are considered equal, and the least cost path is the path with the least hops. But network operators add policy exceptions: lower operational costs, peering agreement, security concerns, etc.
  - Two routing algorithm classes:
    - Link State (Global)
      - Routers maintain cost of each link in network
      - Connectivity/cost changes flooded to all routers
      - Converges quickly (less inconsistency, looping)
      - Limited network sizes
    - Distance Vector
      - Routers maintain next hop & cost of each destination
      - Connectivity/cost changes iteratively propagate from neighbour to neighbour
      - Require multiple rounds to converge
      - Scales to large networks
  - Link State
    - Each node has a local "link state", a list of direct neighbour links and their costs
    - When receiving a new link state message, each router forwards the message to all its neighbours (apart from the one it got the message from)
    - Challenges are packet loss or OOA, solutions are acks/sequence numbers/TTLs
    - Use Dijkstra's algorithm to find shortest path
    - **Lots of technical stuff in week 9, look at lecture slides**
  - Distance Vector
    - Each router knows the links to its neighbours
    - Each router has a provisional "shortest path" to every other router (the distance vector)
    - Routers exchange this dv with their neighbours

each node:

Each router knows the links to its neighbours
        - Each router has a provisional "shortest path" to every other router (the distance vector)
        - Routers exchange this dv with their neighbours
        - Routers look over the set of options offered by their neighbours and select the best one
        - Converges to set of shortest paths
        - Each local iteration is caused by a:
            - Local link cost change
            - DV update message from neighbour
        - Distributed:
            - Each node notifies neighbours only when its dv changes, neighbours notify their neighbours if necessary

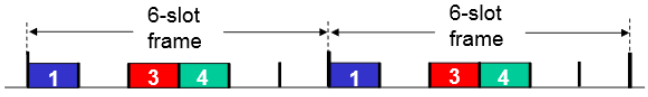


    - LS vs DV
        - Message complexity:
            - LS: with n nodes and E links, O(nE) msgs sent
            - DV: Exchange between neighbours, convergence time varies
        - Speed of convergence
            - LS: $O(n^2)$ algorithm requires O(nE) msgs, may have oscillations
            - DV: convergence time varies: may be routing loops, count to infinity problem
        - Robustness: What happens if router malfunctions?
            - LS: Node can advertise incorrect *link* cost, each node computes only its own table
            - DV: DV node can advertise incorrect *path* cost
            - Each node's table used by others: errors propagate through network
    - Hierarchical routing
        - Aggregate routers into regions, "autonomous systems"
        - Routers in same AS run same routing protocol, routers in difference AS can run different intra-AS routing protocols
        - Gateway router is at the "edge" of its own AS, has link to router in another AS
    - Autonomous Systems (AS)
        - AS is a network under a single administrative control (currently over 30,000 ASes - think AT&T, UNSW, IBM, etc.)
        - Sometimes called domains
        - Each has a unique identifier- 16bit AS number
        - Inter-AS routing is responsible for determining the gateway routers
        - Inter-AS routing also determines which neighbouring AS to send a packet to
            - Hot potato routing: send packet towards closest of two ASs
- Link layer
    - Hosts and routers are nodes, communication channels connecting nodes are links
    - Paths include wired links, wireless links and LANs
    - Link layer services:
        - Framing, link access:
            - Encapsulate datagram into a frame, adding a header and trailer
            - Grant channel access if through a shared medium
            - Use a "MAC" address in the frame to identify source/dest
        - Reliable delivery between adjacent nodes
        - Flow control, error detection, error correction, half-duplex/full-duplex
    - Implemented in each and every host
    - Simple parity:
        - For every block of d bits (e.g. d = 7), count the number of 1's. If it's odd, add a 1 parity, if it's even, add a 0 parity.
        - Eg. 0010110 110110 0110010 -> 1 + 0 + 1
        - If an odd number of bits is flipped we'll detect it -> no detection if even number
        - With 2D parity checking (going down columns too) we can detect which bit is flipped

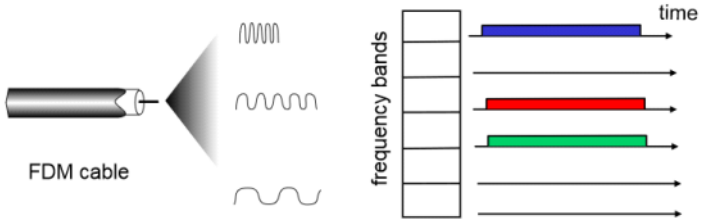
Week 10

Tuesday, 15 November 2016 11:56 AM

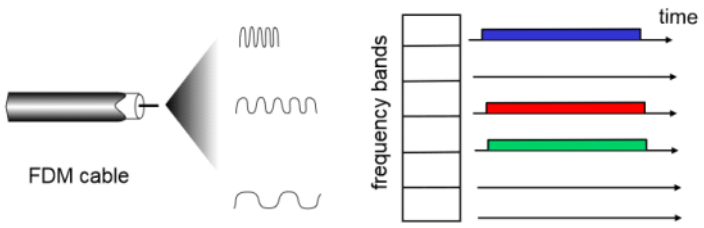
- Multiple access protocols
  - A single shared broadcast channel- two or more simultaneous transmissions will cause interference/collision
  - Multiple access protocols must have a distributed algorithm that determines how nodes share the channel
  - Communication about the channel uses channel itself: no out-of-band communication
- MAC protocols: three broad classes
  - Channel partitioning: divide the channel into smaller pieces (like time slots of frequency), allocate pieces to a node for exclusive use
  - Random access- allow collisions and recover from them
  - "Taking turns"- nodes take turns, nodes with more data to send can take longer turns
- MAC partitioning: TDMA (time division multiple access)
  - Access to channel in "rounds" - each station gets a fixed length slot - in each round, unused slots go idle
    - example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



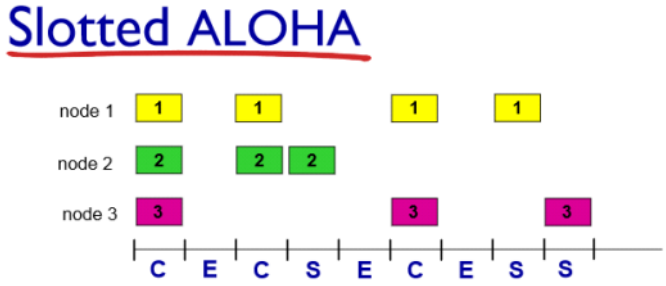
- MAC partitioning: FDMA- frequency division multiple access
  - Channel spectrum divided into frequency bands, each station assign a freq. band, unused transmission lines go idle
    - example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



- Random access protocols
  - When a node has a packet to send, it just transmits at full channel data rate R
  - Two or more transmitting nodes collide
  - RA MAC protocol specified how to detect and recover from collision
  - Started with AlohaNet
  - Aloha assumptions:
    - All frames are the same size
    - Time is divided into equal size slots
    - Nodes start to transmit only at the slot beginning
    - Nodes are synchronised
    - If multiple nodes transmit in a slot, all nodes detect collision
  - Operation:
    - When nodes obtain fresh frame, transmit in next slot: if NO collision, node can send new frame in next slot
    - IF collision, node retransmits frame in each subsequent slot with probability p until success



Pros:
- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:
- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

- CSMA (carrier sense multiple access)
  - Listen before transmitting
  - Does not remove collision due to propagation delay
  - Biggest problem is that when collisions happen, they waste the whole

slot
- CSMA/CD (collision detection)
  - Collisions detected in a short time and colliding transmissions are aborted
  - Easy in wired LAN's, difficult in wireless
  - Exponential back-off: after an m^{th} collision, the node chooses a K at random out of the span {0, 1, 2, …, 2^m-1}. The node will wait K*512 time
- Minimum packet size
  - Gives the host enough time to detect collisions. Minimum packet size in Ethernet is 64 bytes
- "Taking turns" protocols try to look for the best of both worlds
  - Polling
    - Master node invites slave nodes to transmit in turn
    - Concerns: single point of failure, latency, polling overhead
  - Other type: token passing
    - Control token passed from one node to the next sequentially
    - Concerns: token overhead, latency, single point of failure (the token)
- Cable access network
  - DOCSIS: data over cable service interface spec
    - FDM over upstream and downstream freq channels
    - TDM upstream: some slots assigned, some have contention

Week 11

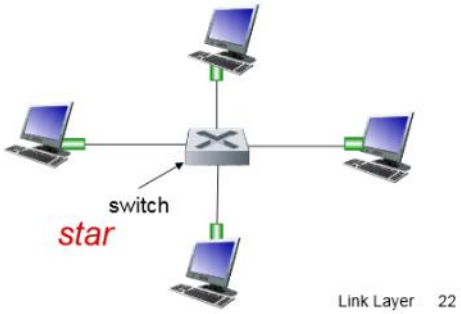
Tuesday, 15 November 2016 1:09 PM

- MAC/LAN/Physical/Ethernet address
  - 48 bit address (12 hex digits)
  - MAC allocation administered by IEEE

MAC	IP
Hard-coded in ROM when adaptor is built	Configured, or learned dynamically
Like a social security number	Like a mailing address
Flat name space of 48 bits	Hierarchical name space of 32 bits
Portable, can stay the same as the host moves	Not portable, depends on where host is attached
Used to get a packet between interfaces on the same network	Used to get packet to destination IP subnet

- ARP: Address resolution protocol
  - How do we determine an interface's MAC address knowing its IP?
  - ARP table: each IP node on a LAN has a table
    - IP/MAC mappings for some LAN nodes
    - TTL time after which mapping is forgotten (typically 20 minutes)
    - <Ip address; MAC address; TTL>
  - Procedure
    - A wants to send a datagram to B. B's MAC address is **not** in A's ARP table.
    - A *broadcasts* ARP query packet, containing B's IP address, with dest MAC address FF-FF-FF-FF-FF-FF.
    - All nodes in the LAN receive the ARP query.
    - B received the ARP packet, and replies to A with its MAC address.
    - A saves IP-to-MAC address pair in its table until the TTL.
  - ARP is "plug and play" -> nodes create their ARP tables without intervention from a netadmin
- ARP Security issues
  - Denial of service: Hacker replies back to an ARP query for a router with a fake MAC address

    - Man-in-the-middle: Hacker can insert his/her machine alone the path between the victim machine and gateway router
    - Hacker needs physical access to the network
- Ethernet today
    - Widely used
    - Star: active switch in the centre and each node off it runs a separate Ethernet protocol- no sharing, no collision

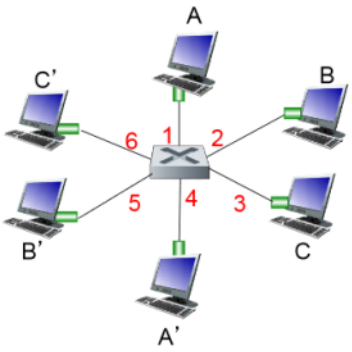


- Ethernet frame

    - Preamble: 7 bytes with pattern 10101010 followed by one byte with pattern 10101011, used to synchronize receiver and sender clocks
    - Addresses: 6 (12 hex numbers) byte MAC addresses
    - Type: indicates higher layer protocol (mostly IP)
    - CRC: cyclic redundancy check at receiver
        - Error detected, frame is dropped
- Ethernet: unreliable and connectionless
    - Connectionless: no handshaking between sending and receiving NICs (network interface controller)
    - Unreliable: data dropped is only recovered in a higher layer like TCP
    - Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff
- Ethernet switches
    - Link layer device that takes an active role: stores and forwards ethernet frames, examines incoming frame's MAC address and selectively forwards frame to one or more outgoing links.
    - Transparent: hosts are unaware of switches
    - Plug-and-play, self-learning - no need to be configured

Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions



switch with six interfaces (1,2,3,4,5,6)

- How does A' know there's a reachable interface is via 4?
  - Each state has a switch table, each entry is: MAC address of host, interface to reach host, time stamp
  - Looks like a routing table
- Switches are self-learning
  - Switch learns which hosts can be reached through which interfaces.
  - When a switch received a packet from a sender, it nodes down the sender's MAC addr and interface:

MAC addr	interface	TTL
A	*1*	*60*

- Switch procedure
  - Record an incoming packet, mac addr of sender, interface num.
  - Look for the destination MAC addr in the table
  - If entry found, send the packet to the destination -> if not found, flood all interfaces except the arriving one
- Switches vs routers

Routers	Switches
Network-layer devices (examine network headers)	Link-layer devices (examine line layer headers)
Compute tables using routing algorithms, IP addresses	Learn forwarding table using flooding, learning, MAC addresses

- Security
  - In a switched LAN, once the switch table entries are established, frames are not broadcast- sniffing frames is harder than pure broadcast LAN
  - Switch poisoning: attacker fills up switch table with bogus entries-

since switch table is full, genuine packets must be broadcast and

- VLAN- virtual local area network
    - Switches supporting VLAN can be configured to define multiple virtual LANs over single physical LAN infrastructure
- Link layer summary
    - Error detection, ocrrection
    - Sharing a broadcast channel:multiple access
    - Link layer addressing
    - Ethernet
    - Switched LANS, VLANs

Week 11 part 2

Tuesday, 15 November 2016 3:50 PM

- Elements of a wireless network
  - Infrastructure mode
    - Base station connects mobiles into wired network
  - Ad hoc mode
    - No base station, nodes can only transmit to other nodes within link, nodes organise themselves into a network
- Differences from wired link
  - Decreased signal strength
  - Interference from other sources
  - Multipath propagation
- Code division multiple access (CDMA)
  - Unique "code" assigned to each user - all users share the same frequency but each has their own "chipping" sequence to encode data
  - If the codes are "orthogonal", multiple users can coexist and transmit simultaneously
  - Encoded signal = original data X chipping sequence
  - Decoding: The inner-product of the encoded signal and chipping sequence
- Wireless hosts are grouped into a "Basic Service Set" containing
  - Wireless hosts
  - An access point / base station
- Host must associate with an AP
  - Host scans channels, listening for beacon frames containing the AP's name (SSID) and MAC
  - Selects AP to associate with
  - May do authentication
  - Will typically run DHCP to get an IP address in the AP's subnet
- Passive/active scanning
  - Passive scanning
    - Beacon frames sent from Aps
    - Association request frame sent from host to AP
    - Association response frame sent from AP to host
  - Active scanning
    - Host sends probing broadcasts
    - Probe response frames are sent from APs
    - Association request sent from host to selected AP

        - Association response frame sent from AP to host
- IEEE: 802.11 MAC Protocol: CSMA/CA
    - Sender
        - If sense channel idle, transmit entire frame
        - If channel busy, start a random backoff time then transmit when timer expires - if no ACK, increase random backoff interval and repeat
    - Receiver
        - If frame received OK, return ACK
- Collision avoidance - another way
    - Allow senders to 'reserve' channel
    - Sender first transmits a small request-to-send packet using CSMA
    - Base station broadcasts a clear-to-send (CTS) in response to RTS
    - CTS is heard by all nodes: sender transmits data frame, other stations defer transmissions
- 802.11 - Mobility within same subnet
    - Host remains the same: IP address can remain same
    - Switch: which AP is associated with the host?
        - Self-learning switch will see a frame from the host and "remember" which switch port can be used to reach them
- 802.11 - Advanced capabilities
    - Rate adaption: when BER (bit error rate) is too high, which to a lower transmission rate but with lower BER
    - Power management: node can tell AP it will go to sleep until the next beacon frame, AP knows not to transmit frames to this node

Week 12

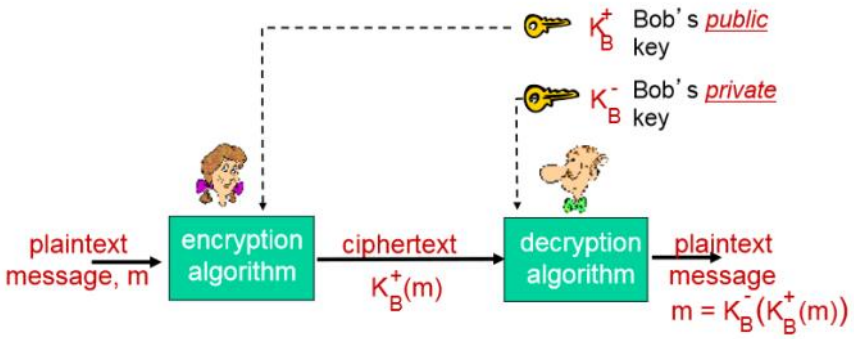
Tuesday, 15 November 2016 6:50 PM

- Security is:
  - Confidentiality - sender encrypts, receiver decrypts
  - Authentication
  - Message integrity- receiver and sender want to ensure the message is not altered
  - Access and availability- services must be accessible and available to users
- Attackers can eavesdrop, impersonate (spoof), hijack and DOS
- Symmetric key cryptography
  - Receiver and sender share the same key
- Breaking an encryption scheme
  - Brute force- search through all keys
  - Statistical analysis
- Simplistic approach: map letters to other letters. e.g. a->b, b->c, etc.
- Slightly more sophisticated approach: have n different substitution mappings, and cycle through them.
  - e.g. n = 4, ciphers: M1, M2, M3, M1
  - Dog -> d from M1, o from M2, g from M3
- Two types of symmetric ciphers
  - Stream ciphers: encrypt one bit at a time
  - Block ciphers: break plaintext message into equally sized block, encrypt each block as a unit
- Block cipher
  - Ciphertext processed as k bit blocks -> 1 to 1 mapping between blocks, e.g. with k = 3, "000" might go to "011".
  - Have a table of input-output mapping, like:

Input	Output
000	110
111	001
001	111
010	101
011	100
100	011
101	010
110	000

  - For large k's, this table is randomly permuted.
- DES: Data Encryption Standard
  - 56-bit symmetric key, 64-bit plaintext input
- 3DES -> encrypt 3 times with 3 different keys

- CBC: Cipher Block Chaining
  - Have encryption of current block depend on result of previous block
  - Initialization vector for the first block
- But how do the sender and receiver know the key? - Instead of symmetric key crypto we use public key crypto
  - The receiver will give the sender a public key, and keep a private key



- Creating a public/private key pair
  1) Choose two large prime numbers p, q
  2) Compute n = pq, z = (p-1)(q-1)
  3) Choose an e < n that has no common factors with z
  4) Choose d s.t. ed-1 is exactly divisible by z (ed mod z = 1)
  5) Public key is (n,e) , private key is (n, d)
- To encrypt a message, computer
  - $C = m^e \bmod n$
- To decrypt
  - $M = c^d \bmod n$
- RSA in practice
  - DES is at least 100 times faster
  - Use public key crypto to establish a secure connection, then establish a second key- symmetric session key- for encrypting data
    - e.g. Bob and Alice use RSA to exchange a symmetric key, then once they both have the key, they use symmetric key cryptography
- Authentication
  - Ap4: To prove that Alice is Alice, Bob sends her a nonce, R. Alice must reply to Bob with R encrypted with their shared secret key.
  - Ap5: Alice sends her public and private key
  - Security hole: man in the middle attacks
- Digital signatures
  - Bob "signs" a message by adding that message encrypted onto it
    - m goes t m, K(m)
  - Alice verifies it by applying Bob's public key to the signed part
- Hash functions / message digests
  - It's expensive to public-key-encrypt long messages.
  - Hash functions give a fixed-length, easy to compute digital

"fingerprint"
    - Apply hash function H to m, getting a fixed size message H(m)
    - MD5 is widely used - computes a 128-bit message in a 4-step process
- Process: sending out a digitally signed, hashed message
    - Bob has message m. He will send m + K^-(H(m)).
    - Alice receives this. She will apply H to the first part of the message, and K^+ to the second part. Now both parts should be H(m).
- Certification authority
    - Bind public key to a particular entity
- Certificate contains serial number (unique to the issuer), info about the cert owner (name, address, domain name, etc.), owner's public key, digital signature (signed with issuer's private key)
- Secure email
    - Alice sends confidential email to Bob:
        - Alice generates a random symmetric session key, K_s
        - Encrypt her message with K_s
        - Also encrypts K_s with Bob's public key
        - Sends both K_s(m) and K^+_B(K_s) to Bob
    - Bob recovers email:
        - Bob uses his private key to decrypt and recover K_s
        - Uses K_s to decrypt K_s(m) and recover m
    - Alice wants to provide her authentication message integrity
        - Alice sends m + K_A^-(H(m))
    - Alice wants to provide secrecy, sender authentication, and message integrity
        - Uses all three keys: her private key, Bob's public key, and newly created symmetric key
- Firewalls
    - Isolates organization's internal net from larger net- allows some packets and blocks others
    - Prevent DOS attacks, such as: SYN flooding (many bogus TCP connections)
    - Prevents illegal modification/access of internal data
    - Allow only authorized access to the inside network
    - Three types:
        - Stateless packet filters
        - Stateful packet filters
        - Application gateways
- Stateless packet filtering
    - Internal network connected to internet via router firewall
    - Router filters packet-by-packet. Decision to forward/drop based purely on:
        - Source/dest IP
        - TCP/UDP port nums

        - ICMP message type
        - TCP SYN/ACK bits
    - Access control list: a table of rules
- Stateful packet filtering
    - Track status of TCP connections