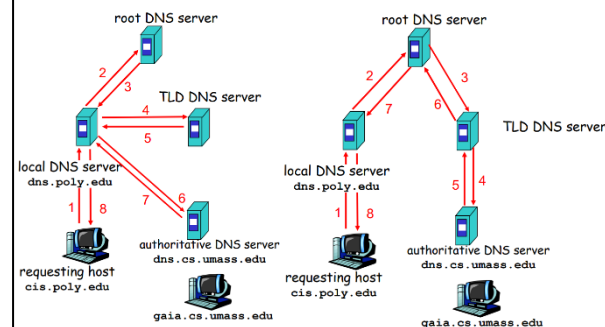
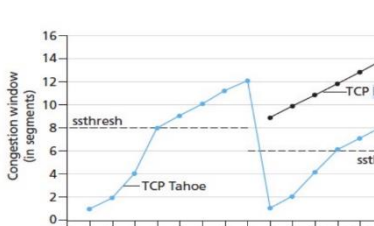


<p>Protocol: format & order of messages exchanged between two or more communicating entities; actions taken on the transmission and/or receipt of a message or other event. OSI 7-layer: application, presentation, session, transport, network, data link, physical. 5-layer: w/o presentation, session layer.</p>	<p>Service model: connection(less)? Reliable? VoIP: C, NR; RUDP: R, CL</p> <p>Network Edge: hosts run apps; C-S model: web c-s, email c-s; p2p: skype, BitTorrent Data rate= Bandwidth*log2(1+S/N). signal2noise ratio</p>	<p>Circuit Switching: FDM: Each user uses one freq band. TDM: each user uses timeslot for full bw. Packet Switching: use as needed, full bw used. Statistical MUXing: shared on demand. $Pr(\text{users} \geq \text{bound}) = \sum_{i=\text{bound}}^N C_N^i p^i (1-p)^{N-i}$Packet Delay: $d_{\text{trans}} = L(\text{packet len}) / R(\text{BW})$ $d_{\text{prop}} = \text{physical link len} / \text{prop speed}$ $d_{\text{queue}} = La/R$ (a: avg packet arr. Rate) $\rightarrow 1$, $d \rightarrow \text{inf}$ d_{proc}: checksum, forwarding, etc. Throughput: bottleneck</p>
<p>Process: program running within a host. Use socket to send/rcv messages Port num to distinguish processes on same host.</p>	<p>Web: objects (HTML file, JPEG, js, etc.) has a URL. HTTP: "stateless". Nonpersistent: one TCP connection per object. Time= 2RTT+ file transmission time. Persistent: w/o pipelining: 1RTT for TCP, then 1 RTT for each obj(incl.HTML) w/: 1RTT for TCP, 1 RTT for HTML, then request all objects on page(~1RTT).</p>	<p>HTTP Methods: GET, POST, HEAD (ask server to leave requested object out of response), PUT(upload file), DELETE HTTP Response: 200 OK, 301 Moved Permanently, 400 Bad Request, 404 Not Found, 500 HTTP Version Not Supported. Cookies: User saves cookie file, server saves cookie ID. Keeps state of user (auth, shop cart, session state, recommendations, etc.)</p>
<p>Web Cache: Typically installed in LAN, client sends request to cache server, if hit then OK, otherwise request origin server. Pros: ↓ response time, traffic on access link, effectively deliver content. Cache server conditional GET: if object not modified, don't send.</p>	<p>DNS: Distributed, hierarchical. Root DNS servers: contacted by local name servers, contacts authoritative name server if mapping not known. Top-level domain (TLD) servers: org, net, edu, uk, fr, edu, etc. Authoritative DNS servers: organization servers, e.g. Web. Email. Iterative: burden on local DNS server Recursive: burden on contacted name server. Caching: server learns mapping. TLD servers cached in local NS. RR format (name, value, type, ttl) A: (name, value)=(hostname, address) NS: =(domain, hostname of authoritative name server of domain) CNAME: =(alias, canonical (real) name) MX: (name, name of mail server) Inserting records to TLD(register at registrar): (networkutopia.com, dns1.networkutopia.com, NS) (dns1.networkutopia.com, 212.212.212.1, A)</p>	<p>FTP: Out of band control (21: TCP control connection, 20: TCP data connection). Client sends control command, server opens TCP data connection and transfer file. Server maintains state: current dir, auth.</p>
<p>MUX: gather data from multiple sockets, envelope with header (src port, dest port) DeMUX: deliver recvd segments to correct socket (using port num). UDP socket: use 2-tuple to direct to socket. (dest IP, dest port num). (Doesn't care about src IP or port). TCP socket: (src IP, src port, dest IP, dest port). All 4 values to direct to correct socket, support many TCP sockets on one host (port num differs).</p>	<p>TCP Flow Control: Sender won't overflow receiver. Receiver "advertises" free buffer space (rwnd) in TCP header. Sender limits unACKed (in-flight) data to rwnd.</p>	
<p>UDP: Datagram, connectionless; simple, no congestion control=as fast as desired. Used by: streaming, DNS, SNMP, DHCP. Format: src port#, dest port#, length (incl. header), checksum, actual data (message). Checksum: Sender: treat segment as 16-bit integers, add them together (wrap around if overflow), and take 1's complement (反码) and put into checksum field. Receiver: sum of all 16-bit integers should be 16'b1, otherwise error has occurred.</p>	<p>TCP Close connection: 1. Client: decides to close, no longer send, can recv. FINbit=1, seq=x. → 2. Server: ACKbit=1, ACKnum=x+1. Can still send some data. ← 3. Server done sending, can no longer send data. FINbit=1, seq=y. ← 4. Client ACKbit=1, ACKnum=y+1. → Client waits for 2*max segment life before closing. (in case ACK is lost). Server closes when receiving this ACK.</p>	<p>TCP: seq num: number of first Byte in segment data. ACK num: seq # of next Byte expected from other side. Cumulative ACK: ACK = n means all bytes < n has been received. SampleRTT: measured RTT (send~ ACK) EstimatedRTT = (1-α)*EstimatedRTT + α*SampleRTT DevRTT = (1-β)*DevRTT + β*SampleRTT-EstimatedRTT TimeoutInterval = EstimatedRTT + 4*DevRTT (safety margin) TCP delayed ACK: recv in-order segment, wait 500ms b4 ACK. If another in-order segment arrives, send immediately. Out of order: higher than expected seq#, gap detected: send DUP ACK. Fast retransmit: sender sends many segments at once. If 3 DUP ACKs are received (not incl. the 1st correct one), one of the segments is lost, so immediately retransmit.</p>
<p>TCP 3-way Handshake: 1. Client: choose init seq# x. send SYN → SYNbit=1, seq=x. 2. Server: choose init seq# y. send SYNACK. ← SYNbit=1, ACKbit=1, Seq=y, ACKnum=x+1. 3. Client knows server is alive. Sends ACK for SYNACK: ACKbit=1, ACKnum=y+1. + may contain some data. 4. Server knows client is alive.</p>		<p>Causes of congestion: lost packets (buffer overflow), timeout caused by queuing. Costs: reduced goodput (useful throughput), unneeded retransmissions, upstream capacity wasted.</p>
<p>Switching: decentralized, destination based: Longest prefix matching, forward to interface with longest address match. Switching Fabrics: Memory, bus, crossbar. Scheduling (choose next packet to send on link): FIFO; priority (hi and low priority queue, hi prio does not interrupt if low prio is being processed); Round Robin (Scan from different class queues); WFQ (each queue has a weight)</p>	<p>IP datagram: ver, len, TTL, upperlayer, header checksum, src IP, dest IP, fragment offset. MTU limits largest link-level frame; fragments within net, reassembled at destination. ClasslessInterDomainRouting: subnet arbitrary x bits, e.g. 200.23.16.0/23, first 23 bits is subnet IP.</p>	<p>TCP congestion control: sending rate ≈ cwnd / RTT initially slow start (exponential, double cwnd every RTT). When over ssthresh, grow linearly (+1 per RTT). When loss detected (Timeout or 3 DUP ACK), ssthresh = 1/2*cwnd_before_loss. TCP Tahoe: Timeout or 3 DUP ACK: set cwnd to 1. TCP Reno: Timeout: set cwnd to 1; 3 DUP ACK: cwnd_new = 1/2*cwnd + 3. TCP Throughput. Ignore slow start, let window size when loss occurs = W. avg. window size is 3W/4; thrupt = 3/4*W/RTT.</p>
<p>Switching: decentralized, destination based: Longest prefix matching, forward to interface with longest address match. Switching Fabrics: Memory, bus, crossbar. Scheduling (choose next packet to send on link): FIFO; priority (hi and low priority queue, hi prio does not interrupt if low prio is being processed); Round Robin (Scan from different class queues); WFQ (each queue has a weight)</p>		<p>DHCP: Client: DHCP discover (broadcast), server DHCP offer, client DHCP request, server DHCP ACK. Server returns IP address, address of first hop router, name and IP of DNS server, network mask. NAT: maintains NAT translation table (src IP, port#)→(NAT IP, new port#)</p>

Routing Algos: Link state: Each node knows whole net topology. Use Dijkstra's Algo. Oscillations possible.

Distance Vector: node x updates DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\}$$

Each node: **maintains its and its neighbors' distance vectors.** Wait for change in local link cost or neighbor, recompute DV, if DV changed, notify neighbors.

Count to infinity problem: Y routes thru Z to get to X. (6). Z routes thru Y to get to X. (7).

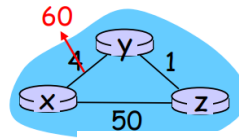
BGP. eBGP: obtain subnet reachability information from neighboring ASes. **iBGP:** propagate reachability information to all AS internal routers.

BGP route selection:

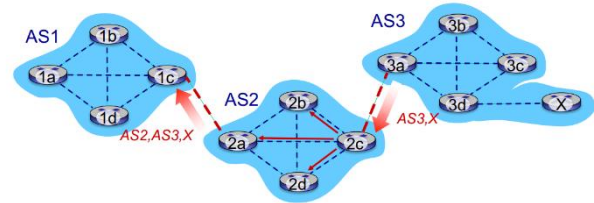
1. local preference value attribute: policy decision
 2. shortest AS-PATH
 3. closest NEXT-HOP router: hot potato routing
 4. additional criteria
- hot potato routing: don't worry about inter-domain cost, choose gateway with least intra-domain cost.

ICMP: carried in IP datagrams, used for error reporting, echo reply and request (ping).

Multiple ACess protocols: 1. channel partitioning (FDMA, TDMA). 2. Random Access. **Slotted ALOHA:** synchronized, one node can try to send new frame in next slot, if no collision then OK, if collision, retransmit in each subsequent slot with prob. P until success. With N nodes, the prob of any node successful is $Np(1-p)^{N-1}$. Max efficiency as $N \rightarrow \infty$ is $1/e = 0.37$. **Pure ALOHA:** No synchronization, send immediately. Efficiency: $p(1-p)^{N-1}(1-p)^{N-1} = 0.18$ for optimum p and $N \rightarrow \infty$.



OSPF: uses link state, floods OSPF LS advertisements to all other routers in entire AS. Hierarchical OSPF: areas + backbone, node only knows detailed area topology, area border routers summarize distances to own area, advertise to other border routers. They are connected by backbone routers, and a outgoing boundary router.



2d parity: n*n matrix, correct 1 bit error.

CRC: data bits D, r+1 bit generator, G; r bit CRC R. $\langle D, R \rangle$ exactly divisible by G (modulo 2). can detect all burst errors less than r+1 bits. ($D \ll r \text{ XOR } R$). $R = \text{remainder}[(D \ll r) / G]$

CSMA: Listen before transmit, if channel busy, defer transmission. Collision: entire packet transmission time wasted.

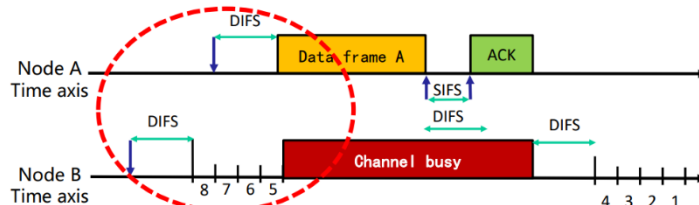
CSMA/CD: Abort when collision detected. After aborting, enter binary (exponential) backoff: randomly chooses K from $\{0, 1, 2, \dots, 2^m - 1\}$, where m is # of collisions, wait $K \cdot 512$ bit times, and then start sensing. Efficiency = $1/(1 + 5t_{prop}/t_{trans})$. Used in Ethernet.

CSMA/CA: no collision detection because hidden terminal, fading in wireless networks. ACK: hidden terminal. Backoff timer: Binary exponential backoff, avoid continuous collisions. DIFS > SIFS: higher priority for ACK, avoid unnecessary collision. Used in 802.11.

Taking turns: polling. Master node polls from "dumb" slave nodes. Cons: polling overhead, latency, single PoF. **Token passing:** in a ring, pass token, one with token can speak. Same cons as polling. Used in Bluetooth.

ARP: To get MAC address of B, A broadcasts ARP query. B receives ARP packet, replies to A with B's MAC address. A caches B in ARP table until timeout.

Ethernet frame: preamble: 7Bytes of 10101010 and 1Byte 10101011, used to sync receiver, sender clock rates. Dest addr, src addr, type (IP), payload, CRC. Is **connectionless, unreliable**. **Switch: self-learning.** Record sender/MAC address pair in switch table when receiving a packet.



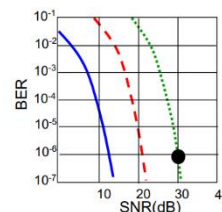
VLAN: Dynamically assign ports on switch into multiple VLANs; multiple switches, same network, using "trunk port". Frames have "VLAN ID" info header. Forwarding via routing.

Wireless links: 802.11, LTE, CDMA, GSM... Infrastructure mode: base stations, connected to infrastructure; ad hoc mode nodes talk to other nodes in range. Characteristics: decreased signal strength, interference, multipath propagation.

802.11: host associate with AP. Passive scanning: APs send beacon frames, host replies to selected AP. Active scanning: host device probe request frame, APs reply, host selects

- scans channels, listening for beacon frames containing AP's name (SSID) and MAC address
 - selects AP to associate with
 - may perform authentication
 - will typically run DHCP to get IP address in AP's subnet
- Mobility: H1's IP address can remain same within same subnet when moving among APs, switch remembers switch port to H1

802.15: adhoc, master/slaves, <10m, Bluetooth devices.



base station, mobile dynamically change transmission rate (physical layer modulation technique) as mobile moves, SNR varies

1. SNR decreases, BER increase as node moves away from base station
2. When BER becomes too high, switch to lower transmission rate but with lower BER

