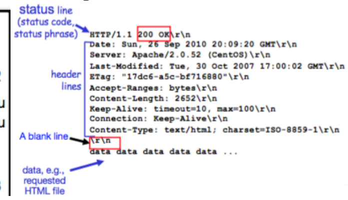
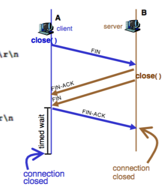
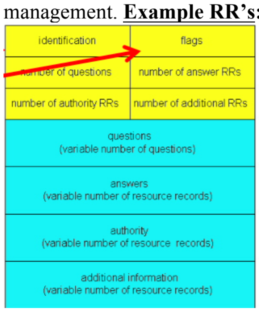
Protocols: Define format, order of messages sent and received among network entities, and actions taken when messages are transmitted or received.
Application layer protocols: Design issues. Data exchange between app processes. HTTP, SMTP, DNS.
Transport layer protocols: Cares about delivering to the right process, reliability, and congestion control. Multiplexing within host. TCP, UDP.
Network layer protocols: Forward data from source to destination. IP. **Link layer protocols:** Send packet across a specific medium. Ethernet.
Physical layer protocols: Send physical bits across a medium. **Delay measurements:** Delay (sec, msec), Throughput: (bits/sec), Loss rate (% of packets lost).
Nodal processing delay: Check bit errors. Determine output link. **Transmission delay:** L/R (L = packet length (bits), R = link bandwidth (bits/sec)).
Queuing delay: # packets in the queue. Each has a transmission time. **Propagation delay:** d/s (d = length of physical link, s = propagation speed in medium 2×10^8 m/s).
Store and forward: Entire packet must arrive at router before transmission. **Max data in pipe:** Bandwidth * propagation delay.
Servers: Reachable by IP address. Always on. Wait for requests. **Client:** Initiates communication with server. **Process:** A program running on a host.
Socket(): Create socket. **Bind():** Bind socket to local IP and port. **Connect():** Initiate connection to another socket. **Listen():** Wait for connections.
Accept(): Accept a new connection. **Write():** Write data to a socket. **Read():** Read data from a socket. **URL:** Identifies an object on a server.
HTTP: Uses TCP. Stateless. 1.0 (non-persistent). 1.1 (persistent). **Standard ports:** HTTP = 80, SSH = 22

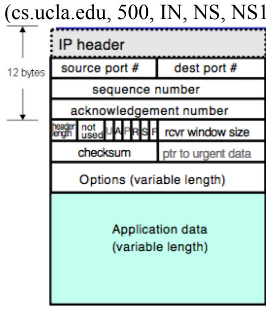
edu	NS	a.edu
edu	NS	b.edu
a.edu	A	1.1.1.1
b.edu	A	1.1.1.2
ucla.edu	NS	a.ucla.edu
ucla.edu	NS	b.ucla.edu
a.ucla.edu	A	2.2.2.1
b.ucla.edu	A	2.2.2.2
www.ucla.edu	A	2.2.2.3

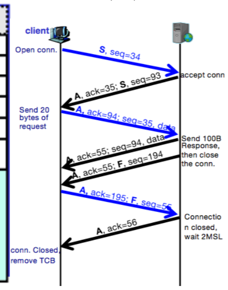


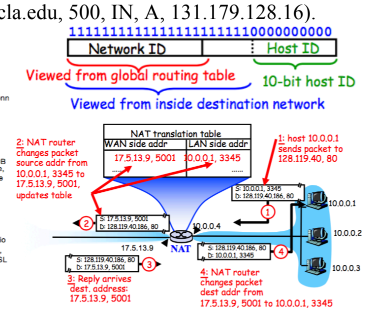


200 OK: Request succeeded. Object in message. **301 Moved Permanently:** Object moved. New location in message. **400 Bad Request:** Request not understood.
404 Not Found: Requested document not found. **505 HTTP Version Not Supported**: HTTP version not supported. **RTT:** Time between sending and receiving.
Non-persistent HTTP: 1 RTT for setting up TCP. 1 RTT for request and 1st byte of response. Total = 2 RTT. Entire process repeated for each object.
Persistent HTTP: 1 RTT for setting up TCP. 1 RTT for request and 1st byte of response. Total = 2 RTT. Does NOT have to repeat TCP creation for new objects.
Persistent HTTP with pipelining: 1 RTT for setting up TCP. 1 RTT for many requests and many responses.
Cookies: If request doesn't have cookie, server creates one and sends back "Set-cookie: <data>" in response. If client has cookie, send "Cookie: <data>" in request.
Ads: Ads can set cookies for clients. If a client visits another website with an ad from the same company, the ad company can learn more about your browsing history.
Cache: Browser can be set to use cache. If the object is in the cache, it returns it. Otherwise, the cache first requests it from the server.
Without cache: Long queues. Bandwidth is a bottleneck. **With cache:** %access link * (internet delay + access delay) + %cache * (LAN + cache delay).
If-modified-since: 304 Not Modified if server copy is older than cached copy. **HTTP 1.1 issues:** Pipelining not supported by default. Head-of-line blocking (long requests queued before short requests. Workaround? Open multiple TCP connections). Big HTTP header. **HTTP 2.0:** Client and server keep header tables until connection closes. (Reduces redundant information sent in every request). Frames (basic communication units). Message (request or response). Steam (a virtual channel with priority). Big messages are broken down into multiple frames. Frames can be interleaved. 1 TCP connection can have streams of different priorities.
User agent: Mail app. Outlook, Apple Mail, browser. **Mail server:** Contains incoming messages, and a queue of outgoing messages.
Mail transfer protocol: SMTP (port 25) **Mail retrieval protocols:** Many. POP3, IMAP. **Mail process**: User goes to user agent, which sends mail to user's mail server using SMTP. Mail server sends mail to receiver's mail server with SMTP. Receiver agent retrieves mail with POP3 or IMAP protocol, which is seen by the receiver.
Example: 1) telnet zimbra.cs.ucla.edu 25 2) helo cs.ucla.edu 3) mail from: lixia@cs.ucla.edu 4) rcpt to: lixia@cs.ucla.edu 5) data 6) <data here /r/n . /r/n> 7) quit.
Codes: 1st digit = good/bad/incomplete. 2nd digit = specific category. 220 service ready. 221 I'm closing too. 250 requested action OK. 500 command not recognized.
TCP stages in SMTP: Handshake. Message transfer. Closing (not TCP close). **HTTP vs. SMTP:** HTTP = Pull. Each object in its own response. SMTP = Push. Multiple objects encoded in multipart message. **MIME**: Multipurpose Internet Mail Extensions. Non-text attachments. Non-ASCII characters. Multi-part messages.
POP: Post Office Protocol. Stateless. Cannot re-read messages on other clients. **IMAP:** Internet Mail Access Protocol. More features (multiple folders). Manipulation of stored messages. Can keep state. **HTTP:** Gmail, Hotmail, etc. **P2P architecture:** No dedicated server. End systems communicate directly.
BitTorrent: Tracker (server keeps track of participating nodes). Seeds (nodes with entire content). Chunk (Each file is divided into 256KB chunks). A client first learns about what chunks each peer has. Then it downloads chunks by rarest piece first. Each piece is verified with hashes. Received chunks are advertised to peers.
Client-server: Time to send F to N clients = max { NF/μ_s, $F/\min(d_i)$ }. **P2P**: Time to send F to N clients = max { F/μ_s, $F/\min(d_i)$, $NF/(\mu_s + \mathrm{sum}(\mu_i))$ }.
DNS: Name -> IP translation. Name owners set up authoritative servers. Each ISP offers caching resolvers. Resolvers and servers speak DNS query protocol.
DNS name space: A variable-depth tree. Non-nodes are domains that belong to administrative authorities. 13 root name servers worldwide.
TLDs: Top-level domains. Controlled by ICANN. Com, edu, net, etc. **Zone:** A domain that contains name servers.
Recursive: Server finds final answer before responding. (caching resolver). **Iterative:** Server immediately returns answer, or a reference. (authority server).
Robust DNS: Redundancy = replicate authoritative servers for safety. Caching = save copies of all query replies. Use a TTL value to know when to delete entries.
RR format: (name, TTL, class? = IN, type (A, NS, CNAME, MX), data). **DNS major parts:** 1) Defines a hierarchical name space. 2) Creates a distributed database, implemented through a hierarchy of authoritative servers (provided by domain owners). 3) Local caching resolvers (provided by ISPs) look up the database. 4) DNS query protocol. **Why not centralize DNS?:** Single point of failure, traffic volume, RR maintenance, long distance between clients and servers, need for distributed management. **Example RR's:** (cs.ucla.edu, 500, IN, NS, NS1.cs.ucla.edu). (NS1.cs.ucla.edu, 500, IN, A, 131.179.128.16).





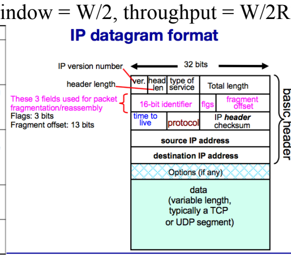


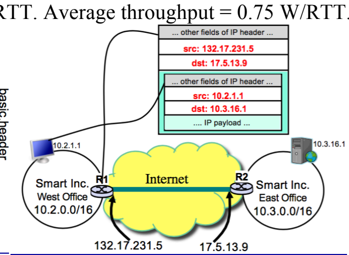


Transport layer: Logical communication between processes. Sender breaks data into segments. Receiver pieces segments back together and passes to app layer.
Transport vs. Network layer: Transport = between processes. Network = between hosts. **TCP:** Congestion control, flow control, connection setup. **UDP:** Unreliable.
Multiplexing: Handle multiple messages from different ports while sending. **Demultiplexing:** Direct received message to appropriate sockets.
UDP: Best effort. Connectionless (no handshake). Add reliability in application layer. Simple. Small header. No congestion control. Length = # bytes including header.
UDP checksum: Sender (1's compliment sum of segment contents). Receiver (add all segments including checksum). **RDT:** Reliable data transfer. 3 types of errors. 1) Corrupted bits (checksum). 2) Packet loss (timeout). 3) Out-of-order (sequence numbers). **3 components of RDT:** 1) Sequence number. 2) ACK. 3) Retransmission timer. **Connection demultiplexing:** TCP socket identified by source IP, dest IP, source port, and dest port. **BDP:** Bandwidth * round trip propagation delay
Simplest reliable protocol: Use sequence number. Send ACK with last correctly received sequence number. Inefficient. Only sending 1 packet at a time.
Pipelining: Have a window size. Send multiple packets. Receiver says how many un-ACKed packets can be sent. Send ACK only if all packets up until *n* have been received. **GBN:** Go back N. If a timeout occurs, resend the entire window. ACK last in-order packet. **Selective repeat:** Set a timer for every packet. Only resend packets that timed out. Problem = receiver needs to keep track of out-of-order packets. **Window size vs. sequence #:** Window size <= $2^n / 2$ (n = bits in sequence #).

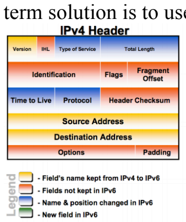
Effective throughput: Throughput * utilization. Throughput = max { packet size * window size / RTT, bandwidth }. Utilization = window size / max window size. **TCP handshake:** 1) Client sends TCP SYN with initial sequence #. 2) Server receives SYN. Replies with SYN_ACK and SYN. 3) Client replies with SYN_ACK, and may send data. **TCP close:** !) One end sends TCP FIN. 2) Other end receives FIN. Replies with FIN_ACK. Sends FIN when also ready to close. 3) Other end receives FIN. Replies with FIN_ACK. 4) Other end receives FIN_ACK and closes the connection. 5) Other end waits a reasonable time before closing (2 * max segment lifetime). "Two army problem". **TCP flow control:** Prevent overrunning the receiver by sending too much data. Receiver informs sender of amount of free buffer space. Sender keeps amount of transmitted, unACKed data <= the received window size. SRTT = (1-a) * SRTT + a * x. difference = SampleRTT – SRTT. SRTT' = (1-a) * SRTT + a * SampleRTT = SRTT + a * difference. DevRTT' = (1-b) * DevRTT + b * |difference| = DevRTT + b * (|difference| - DevRTT). Retransmission timer (RTO) = SRTT + 4 * DevRTT. (a is usually 1/8. B is uaually 1/4). **Karn's algorithm:** Double the retransmission timer value after each timeout. Take RTT measurement again after next successful data transmission. **Initial SRTT and DevRTT:** Initial SRTT = DevRTT = RTO = 3 sec. After first sample, SRTT = SampleSRTT. DevRTT = SRTT/2. **TCP fast retransmission:** Assume a packet was lost if the sender receives 3 duplicate ACKs. **Network-assisted congestion control:** Routers provide feedback to end hosts by using a single-bit congestion indication or setting a rate that the sender should send at. **End-based congestion control:** No feedback from the network. Hosts infer congestion based on losses and delays. **Congestion solution:** Add a "congestion control window" on top of the flow control window. Two phases. Slow start and congestion avoidance. **Slow start:** Start with cwnd = 1 MSS. If all packets ACKed, cwnd = 2 * cwnd. If timeout, cwnd = 1 MSS. **Slow start with ACK cloning:** Start with cwnd = 1 MSS. If receive an ACK, cwnd = cwnd + 1 MSS. If timeout, cwnd = 1 MSS.
Congestion avoidance: Loss is detected by 3 duplicate ACKs. If all sent packets ACKed, cwnd = cwnd + 1 MSS. If 3 duplicate ACKs, cwnd = cwnd / 2. Additive Increase Multiple Decrease (AIMD). If loss detected, ssthreshold = cwnd / 2. **TCP throughput:** Let W = window-size when loss occurs. When window is W, throughput = W / RTT. Just after loss, window = W/2, throughput = W/2RTT. Average throughput = 0.75 W/RTT. **TCP Flags:** URG, ACK, PSH, RST, SYN, FIN

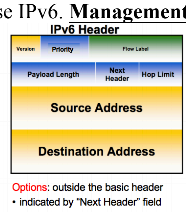
State	Event	TCP Sender Action	Commentary
Slow Start (SS)	Received ACK for previously unacked data	cwnd = cwnd + MSS If (cwnd > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of cwnd every RTT
Congestion Avoidance (CA)	Received ACK for previously unacked data	cwnd = cwnd+MSS * (MSS/cwnd)	Additive increase, resulting in increase of cwnd by 1 MSS every RTT
SS or CA	Loss event detected by 3 duplicate ACK	Threshold = cwnd/2, cwnd = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. cwnd will not drop below 1 MSS.
SS or CA	Timeout	Threshold = cwnd/2, cwnd = 1 MSS, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	cwnd and Threshold not changed

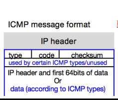


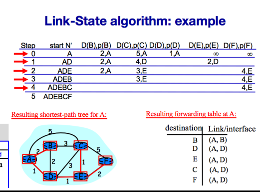


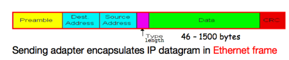
Network layer: In every host and router. Routing = calculate best path to each destination. Forwarding = move packets from router's input to appropriate output.
Data plane: Local, per-router forwarding function. Determines how data arriving as router input is forwarded to router output.
Control plane: Network-wide logic. Determines how data moves along a path between two hosts. Two approaches (routing algorithms, software-defined networking).
Datagram network: Hosts are connected to subnets. Subnets are connected by routers. All hosts and routers speak IP.
MTU: Different subnets have different Maximum Transmission Unit sizes. **Fragmentation:** IP packets can be fragmented and sent as smaller pieces.
Local forwarding table: Each router has a table of ranges to see where to forward packets based on IP addresses.
IPv4 structure: 32 bits. CIDR (Classless Inter-Domain Routing). Network ID takes arbitrary number of bits. **Subnetting:** The host part is split into the subnet and host parts. **CIDR format:** a.b.c.d/x (x = # bits in network ID). Network mask = $2^{32} - 2^{(32-x)}$. **Longest prefix matching:** Use the longest address prefix that matches the destination address. Order items by length. Linear lookup too slow. **From source to dest:** If [A's addr & subnet mastk] == [B's addr & subnet mask], then A and B are in the same net. Use the link layer protocol. Else, send the packet to the default router. **Patricia tree:** Store IP prefixes as a tree. Even faster, Use a k-ary tree, which is faster but may take up more space. **Subnetting:** An organization gets one address block, and splits the host part in two (the subnet and the host). **CIDR vs. subnet:** CIDR used over global routing. Each prefix has its own subnet mask. Subnet is used inside a destination network, with a common, configured subnet mask.
Info needed to be connection to the internet: 1) IP address for host. 2) IP address for default router. 3) Subnet mask. 4) IP address for caching resolver. These fields can be manually configured, or set with DHCP (Dynamic Host Configuration Protocol). It basically broadcasts itself to the network so it can get set up. **RIRs:** Regional Internet Registries. 1 for each major continent. All RIR's have exhausted IPv4 addresses. **How to get an IP address:** 1) Buy an IPv4 address. 2) Use a private IP address within a subnet. 3) Use an IPv6 address. **Tunneling:** An organization can connect two private subnets by using tunneling to route requests to the default router in the other subnet. **NAT:** Network Address Translation. A host in a private network can send a request to an outside host. The router keeps a table that holds an IP address and port specific to the private host so that the response can be routed back correctly. Problems: Single point of failure. Private hosts can't be reached from outside. Solution 1: manually configure NAT to route requests to a certain port to a private host. Solution 2: UPnP (Universal Plug and Play). Allow a private server to add and remove NAT port mappings dynamically. Solution 3: Use a relay. The private server connects to an outer relay, which is connected to clients (care must be taken to keep the relay open). **Private addresses:** Can be used as a security measure since hosts can only be accessed from within the subnet. Short term solution. Long term solution is to use IPv6. **Management:** IETF -> IANA -> RIR -> ISPs -> User

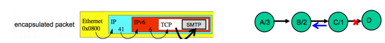


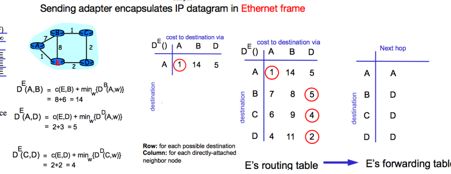


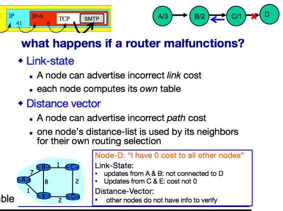






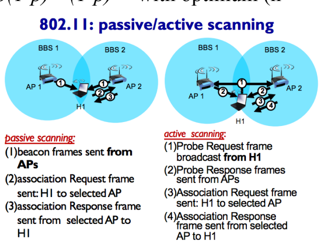


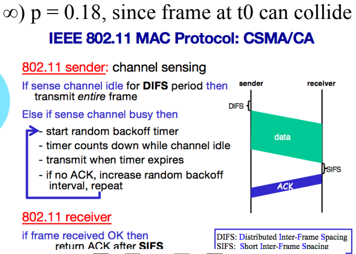


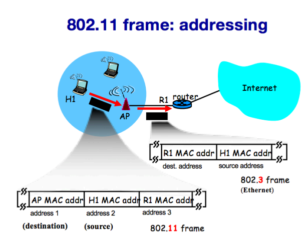


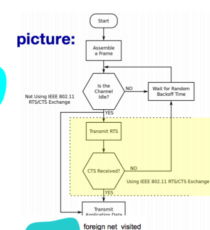
Dual stack: Support IPv4 and IPv6. **Tunneling**: Send an IPv6 packet through IPv4 routers by encapsulating IPv6 packet in IPv4 packet. Make IPv4 IP address translation. **ICMP**: Internet Control Message Protocol. Used by hosts and routers for feedback, status, and error reporting. **Traceroute**: Send UDP segments to dest. Each has increasing TTL. When a router gets a packet, sends back ICMP message. Use this to calculate RTT. Done 3 times per hop. At last router, return "port unreachable" to stop traceroute. **Flow table**: Each router contains one computed and distributed by routing controller. **OpenFlow data plane**: Generalize forwarding. Pattern match. Drop, forward, or modify matched packet to controller. Disambiguate patterns. Count #bytes, #packets. **Link-state**: Dijkstra. Each router computes shortest path to all destinations. **Distance-vector**: Bellman-Ford. Each router computes path based on shortest paths of neighbors. Dx(y) = min {c(x,v) + Dv(y). Iterative, asynchronous: Each local iteration caused by local link cost change or DV update message from neighbor. Distributed: Each node notifies neighbors only when its DV changes. Asynchronous: Nodes need not exchange info/iterate in lock step. **Count-to-infinity problem**: If a link breaks, routers may get stuck sending to each other. Cost increases to infinity. Causes routing loops. **RIP (Routing Information Protocol)**: Neighbor routers exchange data every 30 seconds. If no update within 180 seconds, the router is declared dead. **OSPF**: Every node broadcasts a piece of the topology graph to the entire network. Piece them all together to get the complete graph. Neighbor nodes send each other HELLO messages. If no message in 30 min, declared dead. **Global Internet**: Interconnection of a large number of autonomous systems (AS). **Autonomous Systems (AS)**: Stub AS: End user networks. Transit AS: Internet Service Provider. Intra-AS: Within a campus or ISP. Inter-AS: Between ISPs or between stub and transit AS. Routers keep forwarding tables of intra-AS and inter-AS entries. **Border Gateway Protocol (BGP)**: Obtain subnet reachability info from neighbor ASes. Propagate reachability info to internal routers. Determine "good" routes base on reachability and policy. Advertise own prefix to the internet. **BGP path distribution**: BGP routers establish semi-permanent TCP connections. When you advertise a prefix, you promise to forward packets to that prefix. eBGP between ASes. iBGP between routers in the same AS. **BGP routing policy**: 1) A provider passes all prefixes to its customer ASes. 2) A customer does not pass prefixes between providers. 3) A provider does not pass prefixes that are not its own to another provider. **No valley policy**: An AS does not provide transit between any two providers or peers. **Why different intra- and inter-AS routing?**: Policy: Inter-AS admin wants to control how traffic is routed. Intra-AS has a single admin so policy decisions are not needed. Scale: Hierarchical routing saves table size. Performance: Intra-AS focuses on performance. Inter-AS focuses on policy. **MAC / LAN / Ethernet Address**: 48 bits. Broadcast address: FF-FF-FF-FF-FF-FF. Each adapter on LAN has unique address. Allocated by IEEE. LAN card can move between

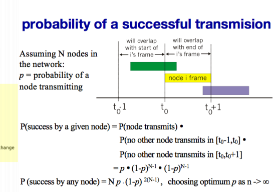
LANs. If MAC = SSN, IP = postal address. Link-layer frame is added to data with source and destination address. **Address Resolution Protocol (ARP)**: A device knows the destination IP but wants the MAC address. Broadcast ARP query with B's IP address and broadcast MAC address. **ARP plug and play**: Every IP node on LAN keeps ARP table. < IP; MAC; TTL >. **Send to different LAN**: Use subnet mask to see if destination is on different LAN. Use ARP to get router MAC address. Send frame to router. Router uses ARP to get destination MAC in new LAN. Router sends to destination with MAC address. **Ethernet**: Bus topology was popular until mid 90s. Switch-based star topology prevails now for speed. **Ethernet**: Unreliable data delivery. Connectionless. Unreliable. Uses CSMA/CD concerned with collision detection. **Ethernet switch**: Link layer device. No configuration needed. Store and forward. Hosts don't know about them. Needs forwarding table indexed by destination MAC address. Sends broadcast signals on all links (except where it came from) using flooding. Interconnected switches just work. Self-learning. **Switches vs. routers**: Both store and forward. Both build forwarding tables. Routers: Network layer. Examine IP headers. Run routing protocols. Switches: Link layer. Examine Ethernet headers. Self-learning algorithms. **Switch advantages**: Transparent. Isolates collision domains. Can connect Ethernet of different speeds. **Router advantages**: Arbitrary topologies. Efficient multicast routing. Work well in large networks. **Wireless Network**: Mobile devices connect to base stations (Infrastructure mode and Ad hoc – P2P). 3 things – Wireless link, Wireless host, Base Station (AP) – connected to wired network, forwards to mobile devices. **Ad hoc**: No base station, node to node transmitting – both distance vector and link state have been tried. **Infrastructure mode**: Base stations connect mobile to wired, mobiles switch base statins after move for continual internet access (*handoff*). **Wireless Link characteristics**: 1.) Decreased signal strength – radio signal attenuates while propagating through matter. 2.) Interference signals from other sources. 3.) Multipath propagation – radio signals reflect around arriving at different times. **Hidden terminal**: Two terminals sending to another terminal can't hear each other -> collision. **Signal attenuation**: Two terminal can't hear each other because signals are out of phase at terminal point -> destructive interference. **802.11 LAN architecture**: wireless host connects to AP. Base Service Set (BSS) – contains wireless hosts and AP. Spectrum divided into 11 channels in 802.11b and admin chooses frequency for AP – if neighbor uses same then interference. AP sends *beacon frame* periodically containing Service Set ID and MAC. Host must associate with AP before transmitting – scan channels for beacon frames, select AP using *association protocol*, use DHCP to get IP and other info about AP's subnet. Passive -> AP sends beacon frame. Active -> Host requests frame by broadcast. **Multiple access**: Like Ethernet uses CSMA to sense channel before transmitting. *No collision detection* because signals fading, difficult to receive when transmitting, can't sense all (hidden terminal). Receiver sends ACK. Uses **CSMA/CA**. **Collision Avoidance**: Allow sender to "reserve" channel -> avoid collisions of long frames. 1.) sender transmits **small** Request to send (RTS) to AP using CSMA (RTS may collide but short). 2.) AP broadcasts clear to send (CTS) in response. 3.) CTS heard by all nodes in range, sender transmits data frame and other stations defer transmissions. **Mobility within same subnet**: You can move between BSS's, and find other AP to attach to while IP address remains the same. Switch will perform self-learning to find interface to communicate to you by. **Link Layer Overview**: Transfer packets to and fro physical nodes (router, hosts) . Ethernet, WLAN. IP encapsulated in layer 2 frame (Link layer frame). **Link Layer function**: Link type: Simplex – 1 way at a time, half duplex – 1 way at the same time, full duplex – two way at same time. MAC address. *Data framing* – receive sequence of bits from physical, demarcate start and end. *Error detections* – CRC. *Channel Access Protocol*. **Adaptors communicating**: Sending – encapsulate IP in frame, add error checking bits, send frame out using access control protocol. Receiving – look for errors, if OK, extract datagram and pass to upper layer. **Data framing**: Frame has a header, and optionally a trailer. *Byte-Oriented Framing Protocol* demarcate frame start and end with sequence **01111110**. *Byte stuffing* – if data contains sequence, then add another sequence or ever occurrence of sequence (so receiver can remove 1 each time to get actual data). **Error detection**: EDC – Error detection and correction bits (D – data protected by error checking) – not 100% reliable. Larger EDC field -> better detection and correction. Can be like Internet checksum (1's complement sum of 16-bit segments). Usually uses CRC instead. **Multiple Access Control**: Determines which node can transmit when, communication about channel sharing must use channel itself, 3 classes of solution – 1.) Channel partitioning (TDMA, FDMA) 2.) Coordinated Access (polling, token passing) 3.) Random access (ALOHA, Slotted ALOGA, CSMA/{CD | CA}. *Channel partitioning* split by time or frequency. *Coordinated access*: 1.) Polling, master asks slave nodes to transmit in turn – overhead, latency, single point of failure. 2.) Token passing – 1 token message passed from 1 node to next sequentially, whoever gets token can send 1 data frame then pass to next – latency, single point of failure as master generates token and monitors it. *Random Access* Transmit full channel data (no pervious coordination), how to detect collision and recover from collision. **ALOHA**: If node has data to send, send whole frame immediately (if collision retransmit frame again with probability p – assume all frames of same size, overlapping frames collide, success rate = $p(1-p)^{N-1}(1-p)^{N-1}$ with optimum (n -> ∞) p = 0.18, since frame at t0 can collide with a frame sent from t0-t1 to t0 + 1. **Slotted ALOHA**: Divide time into equal size slots.

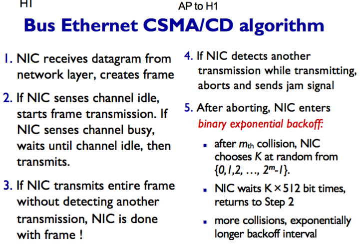


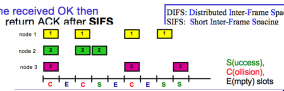


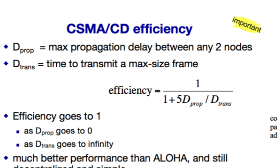


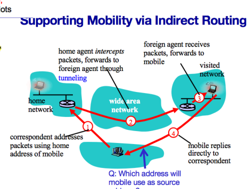


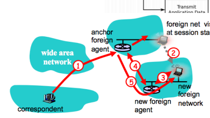


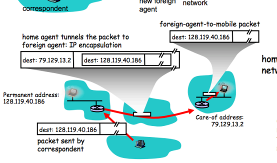


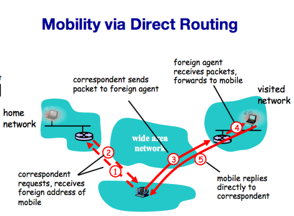












Slotted Aloha: Divide time into equal slots. Clocks in all slots are synchronized. If 2 or more nodes collide in a slot, all nodes detect collision. Each node begins transmitting at beginning of next slot. If no collision, send another frame in next slot. If collision, retransmit in next slots with probability p until success. **Slotted Aloha efficiency**: Probability of successful transmission for a node = $p(1-p)^{(N-1)}$. Success = N * $p(1-p)^{(N-1)}$. As n -> ∞, success = 1/e = 0.37. **CSMA**: Listen before transmit. If idle, transmit. Else, wait until idle. Chance of collision increases with distance between nodes. To cut loss early, use CSMA/CD. **CSMA/CD**: Compare transmitted with received signals. Abort collided transmissions. **NIC**: Network Interface Card. **Jam signal**: Notify all devices that the channel is busy. **Ethernet frame limits**: Upper limit for fair sharing of the link. Lower limit to reliably detect collisions. 48 bits. **Home network**: Permanent home of the mobile device. **Home agent**: Entity that performs mobility functions on behalf of a mobile device. **Permanent address**: The mobile's address in its home network can always be used to reach the mobile. **Correspondent**: A computer that wants to communicate with a mobile device. **Visited network**: Network in which the mobile currently resides. **Care-of-address**: Mobile's address obtained from the visited network. **Foreign agent**: An entity in a visited network that provides mobility function on behalf of a mobile device. **Mobility approaches**: 1) Let routing handle it. No change to end systems. Not widely scalable. Routing table indicates where mobiles are. 2) Let end-systems handle it. Indirect routing: Correspondent sends packets to mobile's home agent, which forwards to mobile. Direct routing: Correspondent gets mobile's foreign address and sends packets directly. **Mobility registration**: Mobile contacts foreign agent, which contacts home agent to inform of changes. **Summary of indirect routing**: Mobility is transparent to the correspondent and any transport protocols like TCP. Mobile can perform foreign agent function itself. Mobility is transparent to correspondent. May result in triangle routing which is inefficient: correspondent -> home network -> mobile. **Direct routing**: Good: Eliminates triangle routing inefficiency. Bad: Correspondent must be aware of mobility support. What if the mobile moves again? **Accommodating mobility with direct routing**: Anchor the foreign agent in the first visited network. Data is always routed to the first anchor FA. When mobile moves, new FA notifies the old FA to have data forwarded (chaining). **Mobility via indirect routing**: Correspondent -> home agent: Source = CD, Destination = P (permanent address). Home agent -> mobile: Outer IP Source = P, Destination = CA, Inner IP Source = CD, Destination = P. Mobile -> correspondent: Outer header Source = CA, Destination = CD, Inner header Source = P, Destination = CD.