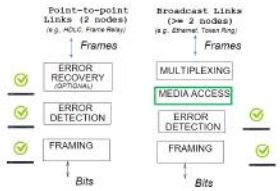
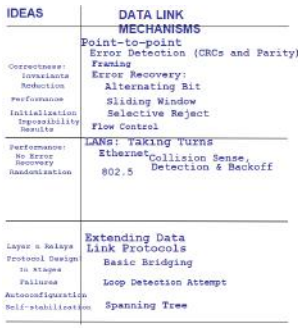
LANs



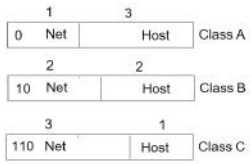
-
- Pros: cost/bandwidth/statistical multiplexing
- Aloha
    - Slotted vs. unslotted
    - Cannot perform collision detection/carrier sense (compare to 802.11)
    - Doesn't abort transmission until complete (vs. Ethernet)
    - No recovery from corruptions, retransmission for collisions
- Collisions
    - Min packet size = transmission speed x round-trip delay
        - Round-trip delay = 2 x max propagation delay
        - Needed for sender to detect collisions
        - Increase in speed or delay = decrease in the other to maintain same min packet size
    - Binary exponential backoff is used to randomly determine a resend order after collisions
        - Slots are 1 round-trip delay long and the number of slots doubles per attempt
    - Jamming: transmit small # of bits after collision detection so other stations can detect it too
    - Detection: done through increased avg. voltage level
- Ethernet => carrier sense/collision detection/exponential backoff
- 802.11
    - CSMA (Carrier Sense Multiple Access)
        - Listen before transmit and defer transmission if the channel is busy
        - Persistent CSMA retries the transmission immediately with prob. *p* when channel is idle
        - Non-persistent CSMA retries after some random interval
    - Hidden Terminal Problem
        - B can hear A and C, but A and C cannot hear each other => A and C may both transmit to B simultaneously and not know they're colliding => one needs to defer
        - RTS/CTS
            - Sender first sends RTS to station, which then sends a CTS
            - Other stations can overhear the CTS (which contains transfer duration) and idle
        - Backoff Interval
            - RTS packets may frequently collide => choose a backoff interval in [0 .. CW]
            - CW is contention window, needs to be adjusted as contention varies
            - Wait length of interval when medium is idle, suspend countdown when medium is busy
- Multicast
    - 3 types: solicitation (to any server), advertisement (to any client), free copies (to all clients)
    - MSB is 1, broadcast address is all 1s
- Bridging
    - Goal is transparent connection between LANs to create an extended LAN => must process fast
    - Learn station location based on source addr., forward based on dest. addr. =? filtering
        - If no dest. addr. info is known, send to other interface => flooding
        - Timeout entries to handle stale info
    - Spanning Tree
        - Root is the node with minimum ID, other bridges find port through which shortest path to root exists => min port
        - Each bridge also finds the ports for which this bridge is on the shortest path between root and corresponding LAN => designated ports
            - Smaller ID is tiebreaker
        - Min/Designated ports are turned on, designated are off => packets to/from off are dropped
        - Each LAN must have unique path to all other LANs => LAN gets to root thru designated bridge (not optimal route)
        - Distributed Algo
            - Each node assumes self as root and maintains estimate of root and dist to root (r,d)
            - When you hear a smaller root, adopt it and make distance d + 1
    - Cons
        - Address/max packet size/bandwidth incompatibility between DL protocols
        - Addresses are flat => all addresses in eLAN must be learned by bridges
        - Spanning tree is inefficient and flooding wastes throughput
    - Pros
        - Generality among varying routing protocols
        - Cost-performance ratio
        - Smaller amount of routing control traffic
    - Bridges connect a small # of compatible LANs to form an eLAn, routers connect eLANs to form a network



-

Routing

- IP routes packets to the right physical network based on network #
- Syntax
    - 192.* => prefix of 192 (8 bits)
    - 192.2/16 => prefix is 16 bits long
- Classful Addresses
    - Original IP addresses were Class A/B/C



    -
    - Inefficient address usage, routing table growth when Class Cs were assigned
    - Solution was to shift to longest matching prefix, still a problem so shift to NATs and IPv6
- Forwarding
    - Find longest matching prefix in forwarding table
        - Forward on default route if no match
        - Deliver packet if next hop is on local interface (ARP)
        - Send to next hop if not local
- 4 Problems
    - Routers need DL addresses of endnodes
        - ARP for MAC addr. of destination endnode

        - Endnodes need DL address of 1 router
            - DHCP provides the address of 1 router by autoconfiguration, can ARP from there
        - Endnodes on same LAN should be able to communicate w/out a router
            - Endnodes can compare masks to check if on same subnet, then ARP if they are
        - Endnodes should send through most efficient route
            - Send to bad router first, then bad router sends a redirect if the packet returns on same interface
- Internal Routing
    - 4 parts: set up addresses/topology (initialization), neighbor determination, compute routes, forward packets
    - Distance Vector
        - Routers gossip with neighbors to spread information => similar to spanning tree
        - Use port databases and central database computed on best port database
        - On link failure, delete stored distance vector for that port
            - No aging of info and send whenever info changes, unlike spanning tree in bridges
        - Performs badly on node failures due to count-to-infinity problem
    - Link State
        - Each router broadcasts its local neighborhood to all other routers within org.
            - Use default cost of outgoing links to neighbors
            - Called intelligent flooding
        - After convergence, all nodes have a complete and identical picture of the network graph
        - Then, each node can use Dijkstra's to compute the next node from S to all other nodes
        - On link failures, only adjacent nodes recompute LSP and rebroadcast
        - Modifications to use equal cost routes to promote parallelism
            - Use of ECMP (Equal Cost Multipath) to split traffic between equal cost next hops based on hash function => guarantees no TCP connection is reordered
        - More resilient to failure due to no count-to-infinity problem
    - General Principles
        - Best effort (TCP will fix)
        - Soft state => keep retransmitting route updates so wrong info will fix itself eventually
        - Decentralization => all nodes have same info
- External Routing
    - Link state/distance vector calculates routes within an AS (Autonomous System)
    - BGP (Border Gateway Protocol) calculates routes between ASes
    - Multiple ISPs means a need for collaboration among ISPs for interconnectedness
        - NAPs => network access points where all ISPs meet
        - Handle policies
    - BGP
        - Prefix-based path-vector protocol with policy-based routing based on AS paths
        - Establish session => exchange active routes => exchange incremental updates (continuously)
        - Node learns multiple paths to destination, stores the routes in a routing table, applies policy to select a single active route, and may advertise the route to neighbors
            - Store AS path and next hop => prevents loops and allows for policy management
            - Data packets flow in opposite direction of BGP updates
                - Have to ARP to get MAC of next hop
        - Incremental updates (contrast w/ distance vector) with announcements (add new route) and withdrawals (remove old routes)
    - BGP Attributes
        - Distance vector can only tune routes via link cost, but BGP can control them in more complex ways
        - AS path length, next-hop, origin (IGP vs EGP), local pref., Multi-exit discriminator, community
            - eBGP makes more sense than iBGP, all other things equal
        - MEDs help load balance by guiding routing of next AS
        - Community tags multiple routes under one label so they can be policed as a group
    - BGP Problems
        - Instability => no convergence guarantee (NP-hard)
        - Scalability is challenging => want to manage traffic to specific networks, but also want aggregation
        - Performance => doesn't load-balance between paths optimally
        - Only has access to local knowledge
        - AS path isn't a real distance measure
    - There exist alternatives like Google Espresso

TIDEAS	ROUTING MECHANISMS
Flexible; Flexibility; Scalability vs Efficiency	Addressing Partition address to reflect hierarchy Flexible Partition (IP masks) Unlimited levels (OSI)
Autoconfiguration Initialization Failures	Neighbor Greeting Send hellos to discover neighbor and detect failure Endnode Routing: must be simple Send to Router + Redirects
Post Office vs. Phone Model Failure Methods Quiet Recovery	Connectionless Routing Distance Vector: Similar to Spanning Tree except for Age Fields: Count to infinity. Solved by limit on cost Link State: Intelligent flooding
2 generals Handle Diversity!	Forwarding Stop Looping packets Hop County, Time to Live

-

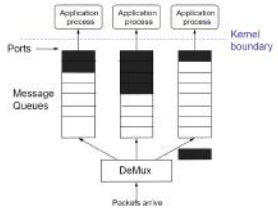
Naming

- Host name vs. IP addr. vs. MAC addr.
    - Host/IP are hierarchical, MAC isn't
    - DNS maps host ⇔ IP
    - ARP maps IP ⇔ MAC
- DHCP (Dynamic Host Configuration Protocol)
    - Broadcast-based LAN protocol algorithm
        - Host broadcasts "DHCP discover" on LAN (e.g. Ethernet broadcast)
        - DHCP server responds with "DHCP offer" message
        - Host requests IP address: "DHCP request" message
        - DHCP server sends address: "DHCP ack" message w/IP address
    -
    - Automates host boot-up process
    - Assign a unique IP addr. given a MAC addr.
    - Tell host other stuff about the LAN
    - Have fewer addr.s than hosts, and renumber the network
    - Addresses are leases, not grants and will timeout when the host leaves => diff. IPs at diff. times
- DNS (Domain Name Service)
    - Distributed administrative control, hierarchical namespace divided into zones and distributed over DNS servers
    - Hierarchy of servers => translation occurs in local DNS servers, which are usually near the endhost
        - Hosts are configured with local server or learn it thru DHCP
        - Redundant servers in case of crashes and allows for load balancing queries
        - Akamai fakes out DNS to find closest copy of service
    - Client app. gets server name from the URL and server app. gets it from the socket
    - UDP used for queries, which means reliability must be added on top
        - Alternate servers on timeout and exponential backoff when retrying same server
    - Responses are cached at endhosts and local servers
- NAT (Network Address Translation)
    - IP provides private address space that anyone can use => addresses that aren't routable and will be dropped
    - Gateway router can rewrite IP addr.s as packets leave/enter a given network
        - Replace private addr. with public ones => maintains/updates mapping
        - Router needs to see/update every packet
    - Entire LAN uses just one IP when viewed from the outside world => cannot be addressed/viewed externally
        - ISP can be changed without changing addr.s of local devices
    - Endhosts may not be aware of external IP addr.s, which may be a problem
    - NATed endhosts aren't reachable from the Internet => all connections must be initiated from within private net
    - Use of TCP port numbers to disambiguate stations => layer violation
        - Right solution is IPv6

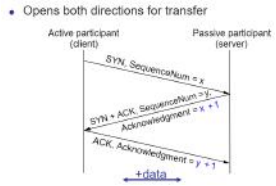
Transport

- Basic Questions
    - What function does a transport provide? Reliable/unreliable data delivery between processes
    - What is a connection? A shared state for each process pair that enables delivery
    - Why not have just one connection for all data? Fast processes would be hindered by slow ones

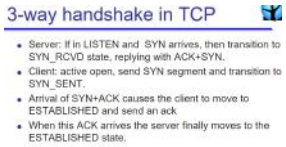
    - Why not keep connections up always? Too many possible process pairs
    - How do we address the receiving process in the receiver? => Use an OS independent mechanism: ports
        - PID/memory addr.s are OS-specific and transient => unique (IP, protocol, port) tuple
        - Servers typically bind to well-known ports while clients use OS-assigned, temporary ports
- UDP (User Datagram Protocol)
    - Provides unreliable message delivery between processes
    - Source port filled in by OS as message is sent, dest. port identifies UDP delivery queue at endpoint
    - Connectionless
    - Uses multiplexing to send packets to corresponding ports



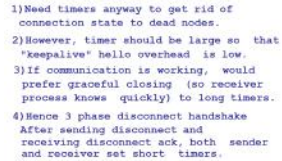
        -
    - Optional protection against errors with checksum (covers data, UDP header, and IP pseudoheader)
- TCP (Transmission Control Protocol)
    - Reliable, bidirectional bytestream between processes, using a sliding window protocol, for efficient transfer
    - Connection-oriented
    - Flow control prevents sender from overrunning receiver buffers
    - Congestion control prevents sender from overrunning network capacity
    - Connection Setup
        - Both sender and receiver must be ready before the start of data transfer => agree on params
        - Handshake protocols set up the state between 2 oblivious endpoints
            - Need to deal with delayed/reordered packets
        - Connections are 4-tuples: (srcIP, srcPort, destIP, destPort)
        - 3-Way Handshake



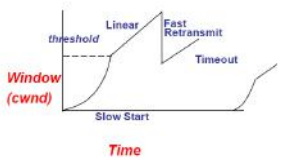
            -
            - Chosen to be robust, especially against delayed duplicates
            - Changing initial sequence numbers (ISNs) minimizes the chance of hosts that crash from previous connections
            - Choose ISNs by maximizing periods between reuse and minimizing guessability



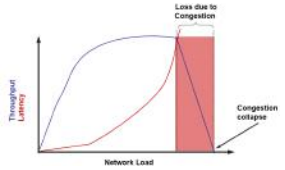
            -
        - Disconnect



            -
            - TIME_WAIT state
                - Wait 2 x MSL (Maximum Segment Lifetime) before completing the close process
                    - ACK might have been lost, resulting in a FIN resend, which could interfere w/ next connection
                - Realistically: abortive close, which just sends a RST packet instead of waiting
    - Reliable Delivery
        - Usual sequence numbers from Go-Back-N, but with larger range to handle out-of-order packets
            - Number the bytes, not the segments, so that TCP can change packet size in the middle of a connection
            - Sequence numbers no longer start at 0, but at the ISN
        - TCP reacts quicker to lost messages, uses a crude form of selective reject, and does flow control using a dynamic window size
        - Retransmission
            - Sender uses timers to decide when to retransmit
                - Too long => inefficient, too short => unnecessary extra traffic
                - Should be based on round-trip time, which needs to be measured, but not by the OS (large timer granularity)
            - TCP uses fast retransmit, where (3) duplicate ACKs indicates a loss, resulting in an immediate retransmission
                - Need to be careful if frames can be reordered
        - QUIC
            - Combines security/sequence handshake on first connection to server
            - If the server remembers client info, 0 handshakes are needed on later connections, still need 3-way handshakes
            - Stream multiplexing: multiple streams in a single QUIC connection
            - A single loss stalls all streams in TCP, not the case in QUIC
            - Congestion information is shared, removing TCP's long startup time
    - Congestion Control



        -
        - Sending shouldn't be faster than the sender's share of the bandwidth or faster than the network can process
        - Fair Bandwidth Allocation: each flow should receive as much bandwidth as their bottleneck, divided equally among all other flows
        - Buffer is intended to absorb bursts when the input rate is larger than output
            - If sending rate is persistently more than drain rate, a queue builds up
            - Dropped packets are wasted work => goodput vs. throughput



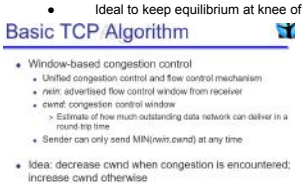
        -

- Congestion collapse is a result of livelock: all packets go to next hop and are dropped because of congestion
- Sender sending faster than bottleneck link speed, packets queue until dropped, sender retransmits, repeat
- Relief: reduce network load => send data slower (TCP uses a closed loop, host-based strategy)
  - Responds to feedback in network and adjusts bandwidth allocations, hosts are responsible
- Avoidance on the knee, control on the cliff

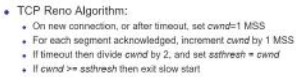
1) DETECT CONGESTION
2) FEEDBACK INFORMATION TO THE SOURCE
3) SOURCE ADJUSTS WINDOW: INCREASE POLICY
   DECREASE POLICY

TWO INTERESTING CASES:
a) HOW A SOURCE REACHES STEADY STATE.
b) HOW A SOURCE REACTS TO A NEW SOURCE TO PROVIDE A FAIR ALLOCATION.

- Detected through explicit signaling (receiver tells sender if queue is too full) or implicit signaling (packet loss/delay signals)
- Throttling
  - Window-based: constrain # of packets in network using window size (cheap but creates bigger bursts => larger buffers)
  - Rate-based: allow sending of x packets in period y (smooth traffic, but fine-grained connection timers)
- Send Rate
  - Ideal to keep equilibrium at knee of curve, in reality, adaptive approximation => probing network to increase/decrease rate

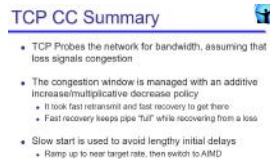


- Congestion Avoidance
  - Adapt to changes in available bandwidth
  - AIMD (Additive Increase Multiplicative Decrease)
    - Increase sending rate by constant MSS, decrease by a linear factor (div. by 2)
    - Converges to fair share of bottleneck link
    - Start with cwnd = 1 => slow start
      - Quickly increase sending rate until congestion begins

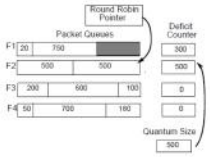


- Fast Retransmit and Recovery
  - Fast retransmit prevents slow timeouts => 3 duplicate ACKs to detect losses quickly
  - Fast recovery avoids stalling after loss => if ACKs are still coming, no need for slow start, divide cwnd by 2 after fast retransmit, increment cwnd by MSS for each dupe ACK

- Short Connections
  - Only contains a few packets
  - Interacts badly with slow start
  - Idea: the packet that gets dropped (SYN vs. slow-start vs. congestion avoidance) matters a lot
  - Most flows are short today



- Router Scheduling
  - Need scheduling at routers as well (think TCP vs. UDP) => give each flow its own queue (theoretically)
  - Use DRR (Deficit Round Robin), which schedules round-robins among queues in proportion to some weight parameter



  - RED (Random Early Detect)
    - Random chance (increases as queue fills) of dropping a good packet as an early form of congestion warning

IDEAS	TRANSPORT MECHANISMS
Just like reliable Data Link except	Error Control, Sequencing Multiple connections, larger sequence numbers
Fault-tolerance Delayed duplicates Crashes 2 generals	Connection Management 3-way handshakes, Timer-based Management Graceful disconnection
Adaptive Protocols Passing info in headers Deadlock vs. Livelock	Congestion Control Matching to network speed Control vs. Avoidance Congestion Collapse Bit or timeout based feedback

-