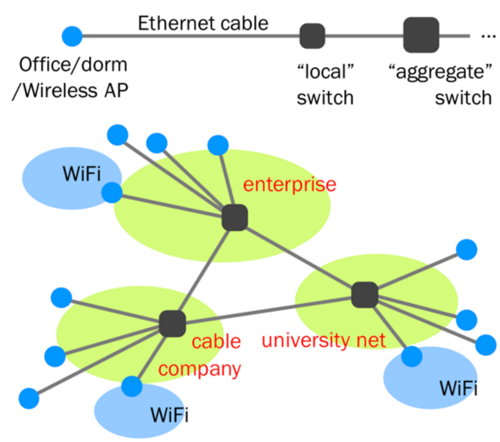
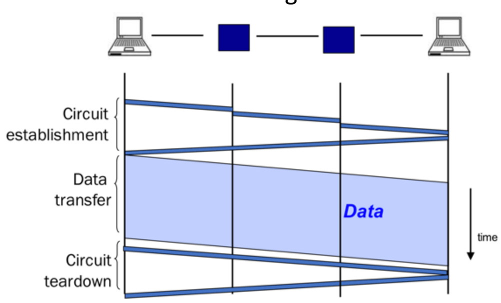
Lecture 2 (Networking concepts)



Switched networks enable efficient scaling
Sharing:
Reservations -> circuit switch
src sends reservations request to dst
Switches establish circuit
src starts sending data
src sends teardown message



Pros: predictable performance, fast switching
Cons: circuit setup/teardown, bad with bursty traffic, if switch fails circuit will fail
On demand -> packet switch
Each packet has dst and treated independently, buffers absorb transient overload
Pros: efficient use of network resources, simple to implement, robust against switch fails
Cons: unpredictable performance, buffer management, congestion control

Overview: Communication via Sockets

- Client
  1. Create a socket with the **socket()** system call
  2. Connect the socket to the address of the server using the **connect()** system call
  3. Send and receive data. The simplest is to use the **read()** and **write()** system calls.
- Server
  1. Create a socket with the **socket()** system call
  2. Bind the socket to an address using the **bind()** system call.
  3. Listen for connections with the **listen()** system call
  4. Accept a connection with the **accept()** system call (blocks until a client connects with the server)
  5. Send and receive data

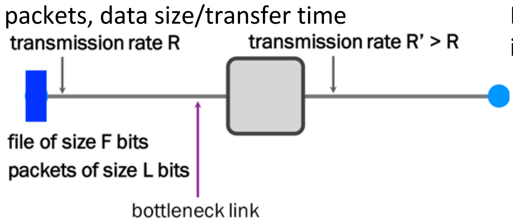
UDP does not need listen, accept, connect

Lecture 3 (Performance metrics)

Delay: How long does it take to send a packet from dst to src?

- Consists of four components
  - *transmission delay*
  - *propagation delay* } *due to link properties*
  - *queuing delay*
  - *processing delay* } *due to traffic mix and switch internals*

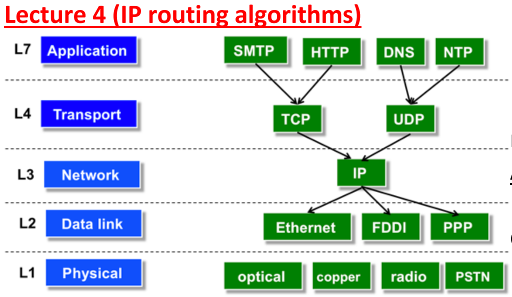
Link bandwidth: bits/sec
Propagation delay: secs for 1 bit to move through link
BDP=bandwidth x propagation delay
1mb = 1,000,000; 1ms = 1/10^3 s
Transmission delay: how long it takes to push all the bits of a packet into a link, packet size/link bandwidth
Queueing delay: how long a packet has to sit in buffer before being processed.
If arrival rate > departure rate, QD=inf.
A = average arrival rate
W = average packet wait time
L = average length of queue
L = A x W(hard to compute)
Processing delay: how long the switch takes to process a packet, negligible
Loss: fraction of packets dropped
Throughput: rate at which dst receiving packets, data size/transfer time

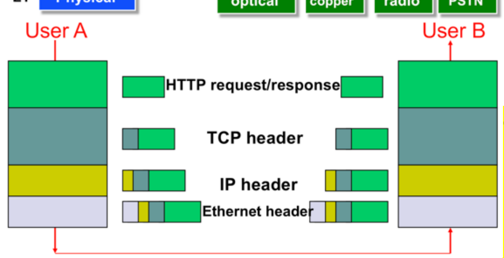


Transfer time = F/R + propagation delay + L/R'

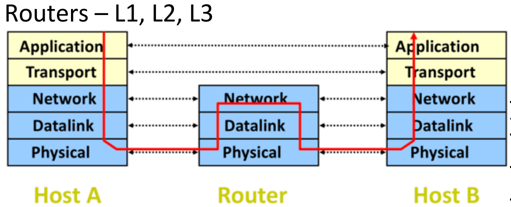
Average throughput = min { R, R' } = R

Lecture 4 (IP routing algorithms)





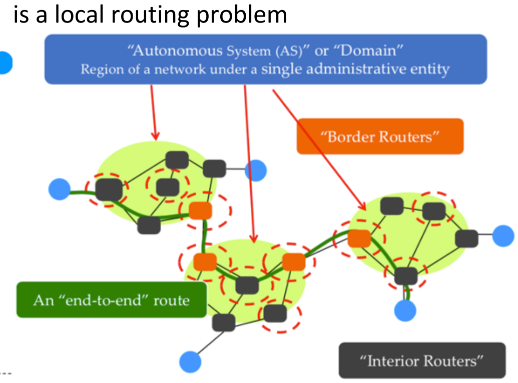
Host – implement all 5 layers
Switches – L1 and L2
Routers – L1, L2, L3



Forwarding: Local router determines output link, read address from packet's network layer header, search forwarding table
Routing: Network wide process to determine content of forwarding tables (end to end path for each destinations)
Local routing is the forwarding table in a single router, global state refers to collection of forwarding tables in each of the routers
Routing state is valid if and only if there are no dead ends and there are no loops
Distance vector: only know neighbors, when routing table changes, send to neighbors
Distance vector algorithm, broadcast and receive neighboring forwarding tables, can calculate next hop for a destination and total cost to get to that destination. Good news travels fast and bad news travels slowly.
Poisoned reverse: distance is inf. If routes through itself
Link state: node floods local information to every other node in the network. Dijkstra to compute shortest path to every other node

Lecture 5 (Addressing)

Distance vector (RIP, IGRP) vs link state (OSPF) is a local routing problem

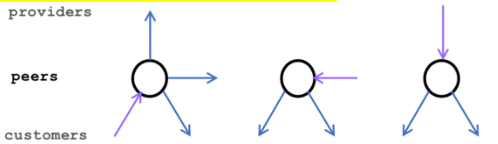


Forwarding table should have an entry for a range of addresses. Hose addressing is key
AS wants policy, autonomy, privacy
BGP (Border Gateway Protocol) extends distance vector to accommodate policy
IPv4 is 32 bit number split into prefix (network) and suffix (host)
Classful addressing only has 3 sizes, 1 byte, 2 byte, 3 byte network prefixes
CIDR (classless Interdomain routing) – flexible division between network and host addresses.
128.23.9/26 means 26 bits as network prefix.
26 represented by 32 bit mask
ICANN -> ARIN -> ISP -> individuals

Lecture 6 (BGP)

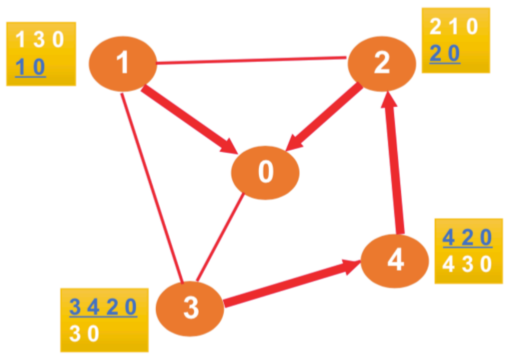
AS can be customer, provider, peer
Peers have similar traffic and route through each other to save money
BGP selects best route based on policy rather than shortest distance; sends entire path for each destination (rather than just distance metric, avoid loops); selective route

advertisement; aggregate routes for different prefixes
eBGP – BGP between border routers
iBGP – between routers within same AS
IGP – intradomain routing protocol
Security: AS can claim to serve a prefix they do not have access to, AS can forward along a route different than advertised



With Gao-Rexford, the AS policy graph is a DAG (directed acyclic graph) and routes are "valley free"

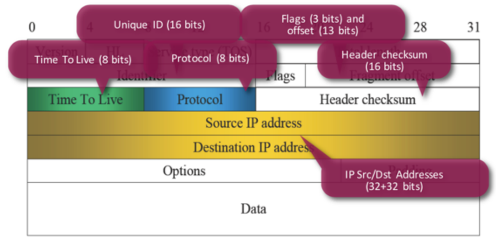
Gao-Rexford guaranteed to converge

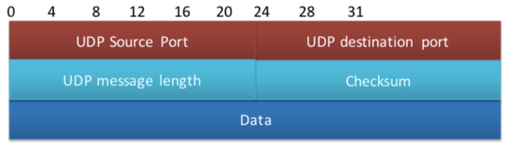


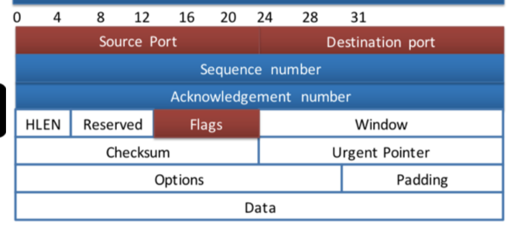
Persistent oscillation, AS path length mislead

Lecture 7, 8(IP, TCP)

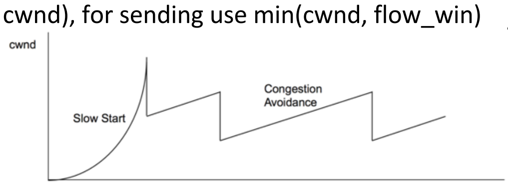
IP Datagram – 20 bytes; Address resolution protocol ARP maps IP address to link level address to be used in direct delivery
Internet control message protocol (ICMP) – exchange control/error messages about delivery of IP datagrams: address mask, timestamp, source quench (traffic overload), parameter problem (errors in IP datagram field), echo, time exceeded (report expired datagrams), redirect, destination unreachable, traceroute (send datagrams each with increasing TTL)
TCP: sequence # specifies position of the segment data, ack # specifies position of next byte expected from communication partner.
TCP flow control window: make sure receiving end can handle data (no regard for network).
Accept iff ack # < syn # < ack # + window
SYN/FIN – establishing/terminating connect
ACK – when ACK is valid; URG – urgent pointer says where non urgent starts; PUSH; ABORT – abort connection.
NACK – packet did not arrive; cum. ACK – ACKS for all k < n; SACK – selective ACK
Key func: flow control, data transfer, congestion control, connection setup
Throughout=window/RTT
Know packet dropped when time out, receive duplicate ACKs







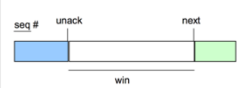
TCP Congestion Control: guessing about state of network, additive increase multiplicative decrease, KNEE – point at which throughput increases slow and delay increases fast, CLIFF – point after which throughput decreases to zero fast. Congestion control stay left of cliff, congestion avoidance stay left of knee.
AIMD – converges to efficiency and fairness
Maintain cwnd, flow_win, Ssthresh (update cwnd), for sending use min(cwnd, flow_win)



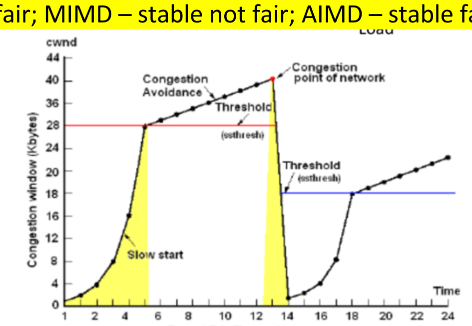

```
Initially:
   cwnd = 1;
   ssthresh = infinite;
New ack received:
   if (cwnd < ssthresh)
      /* Slow Start*/
      cwnd = cwnd + 1;
   else
      /* Congestion Avoidance */
      cwnd = cwnd + 1/cwnd;
Timeout:
   /* Multiplicative decrease */
   ssthresh = cwnd/2;
   cwnd = 1;
```

```
while (next < unack + win)
   transmit next packet;

where  win = min(cwnd,
                 flow_win);
```



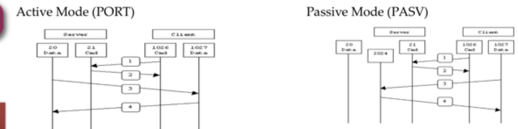
MIAD – not stable not fair; AIAD – stable not fair; MIMD – stable not fair; AIMD – stable fair

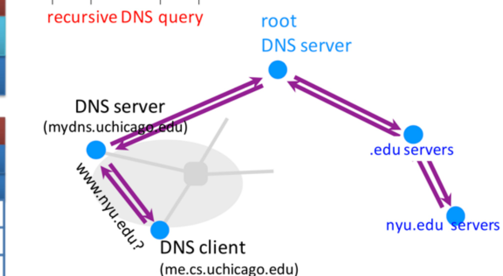


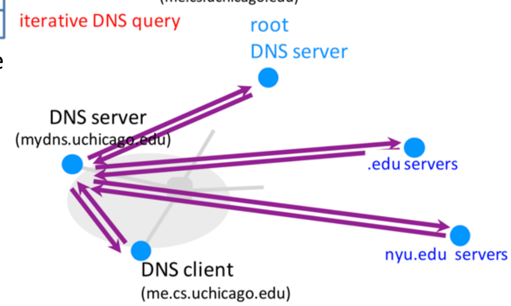
Lecture 9 (FTP, DNS)

FTP: transfer files between 2 computers; need to resolve OS, character, naming, directory
2 connections: Control (persistent connection, 21), Data (ephemeral connection, 20)

In active FTP the client tells the server which port the server should connect to (server initiates connection). Client firewall would usually block this so we use passive FTP
DNS – local servers and resolver software







Lecture 10 (DNS, RPC)

DNS caching: DNS servers cache responses to queries, delete cached entry after TTL expires, top level servers rarely change
Attacking DNS: impersonate local DNS server and give wrong IP address to client; denial of service the root to make them unavailable
DNS properties: easy unique naming, fate sharing for network failures, reasonable trust model, caching lends scalability, performance
Server designs: iterative vs concurrent; connections vs connectionless; stateful vs stateless
Remote procedure call: called procedure not in same address space, transport independence, hides network from programmer, BLOCKS, there is more delay, cannot pass pointers, idempotent can be repeated safely non idempotent cannot
RPC calls can be lost/duplicated, at least once/zero or more (reply/no reply)
Request – transaction id, procedure id, credentials, verifier, params(XDR); REPLY – trans id, status, verifier, status, resultsXDR
XDR – specify how data object encoded, client server need to agree on format

Concurrent, Connection-oriented Server

1. Create a TCP socket (s1)
2. Bind it to a port (INADDR_ANY)
3. Place the socket in passive mode (listen)
4. Accept connection (uses s1, returns new socket s2)
5. Fork slave process
   1. Close old socket (C-s1)
   2. Interact with the client using new socket (C-s2)
   3. Close new connection (C-s2)
   4. Exit
6. Close new socket (s2)
7. Go to (4) or...
8. Exit

One slave will not block another slave

But large overhead in opening and closing the slave process