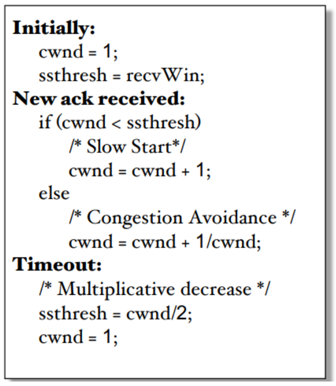
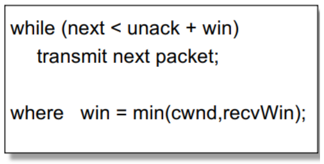
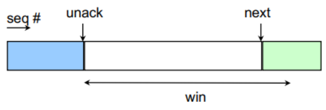
Multicast - how does it compare with unicast? Difference in terms of service model. In multicast, need protocol at IGMP (in unicast, no need). Need to know query reply og IGMP.
DVMRP - Need to know all group states: (S, G). Loop prevention via reverse path checking. No single POF since broadcasting, straightforward because you are sending to everyone. Issue is floods to everyone, including those who don't want it. They prune, but router has to remember who to not send it to. Alot of storage overhead.
MOSPF - Router keeps tracks of groups & asks for any related data every 30s. Membership driven because need to ask router. Doesn't need to worry about other groups that you don't need, also no pruned parts. Computation overhead to compute multicast trees.
CBT/PIM -
Need to know how they build you different types of trees.
Understand at a high level how they work and the pros + cons, may ask about comparing DVMRP, MOSPF, CBT/PIM or ask about source tree vs group tree.
source tree is better in terms of shorter delay and more spread out, but it has many more states in the router, so less scalable in terms of number of groups, for shared groups its the opposite.
All the sources sharing the same tree means higher chance of traffic congestion or longer delay because packet doesn't go through shortest path. PIM tries to have both by starting with a shared tree but if theres high data rate it can share to have a per source tree for that specific one.
TCP - TCP makes guesses about available network capacity + adapts (however, fear of oscillation). To deal with congestion, can use scientific approach with MIAD, AIAD, MIMD,AIMD (AIMD best cause converges to stable and fair cycle).

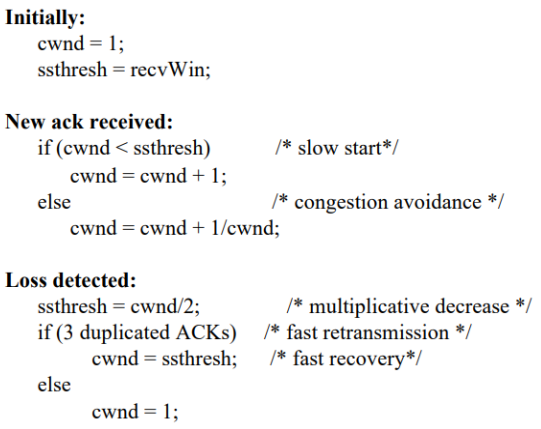


```
Initially:
    cwnd = 1;
    ssthresh = recvWin;
New ack received:
    if (cwnd < ssthresh)
        /* Slow Start*/
        cwnd = cwnd + 1;
    else
        /* Congestion Avoidance */
        cwnd = cwnd + 1/cwnd;
Timeout:
    /* Multiplicative decrease */
    ssthresh = cwnd/2;
    cwnd = 1;
```



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

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



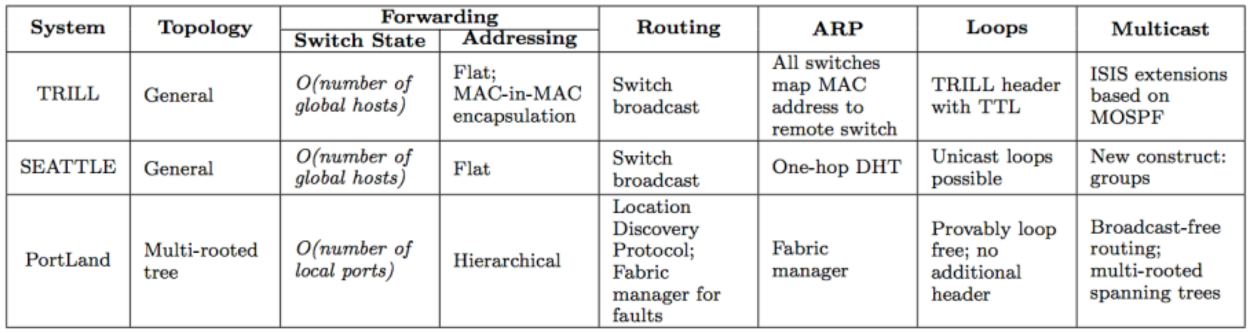


```
Initially:
    cwnd = 1;
    ssthresh = recvWin;

New ack received:
    if (cwnd < ssthresh)          /* slow start*/
        cwnd = cwnd + 1;
    else                          /* congestion avoidance */
        cwnd = cwnd + 1/cwnd;

Loss detected:
    ssthresh = cwnd/2;            /* multiplicative decrease */
    if (3 duplicated ACKs)      /* fast retransmission */
          cwnd = ssthresh;      /* fast recovery*/
    else
          cwnd = 1;
```

XCP - involving routers to help transport to have more accurate transmission control. Not required to know details, but know the idea of what they are doing to help.
DNS - Recursive query: keep going up if you don't know until you reach root, puts potentially alot of burden on root. Instead use interactive query: contacted server asks a bunch of other name servers to see who knows, avoids heavy load on root.
Replication: each zone (continuous sub-space, stored in same server & managed by same admin) should have 1 or more secondary servers at diverse locations to increase robustness.
Caching: Resolvers cache recent query results till TTL expire. Also cache intermediate results.
4 Types: A: hostname, IP address. CNAME: alias, actual domain/canonical name. NS: domain, authoritative name server. MX: hostname/domain, mail server hostname.
ERROR scenarios stemming from misconfiguration: Lame Delegation-NS Resource Record (RR) and A RR exists, but receive a non-authoritative answer or no response at all. Diminished Server Redundancy-Two different RR entries either belong to the same subnet, belong to the same AS, or the same city. Cyclic Zone Dependency (1)-The A glue RR for B.foo.com is missing, B.foo.com depends on A.foo.com, if A.foo.com is unavailable then B.foo.com is too (configuration error in the parent zone). Cyclic Zone Dependency (2)-Correctly configured zone, but error in

logic: say A.foo coupled with B.bar and A.bar coupled with B.foo in DNS. B servers depend on A servers, if A servers are unavailable then the B addresses are unresolvable.
RON - know the difference between overlay and RON, overlay does multicast so tries to build a tree, while RON tries to do unicast.
DHT - Chord, finger tables, O(log n) performance. Handles rapid arrival and failure of nodes.
ALM - Create a mesh via Narada Design (members have low degrees, shortest path delay between any pair is small). Then construct per-source trees using multicast routing algorithms. Add overlay link if overall utility is higher, drop if cost drops below threshold.

Bittorrent - understand problem they are trying to solve; if bandwidth/delay are large, why is TCP and why?
Ethernet - SEATTLE: hashing to determine switch that knows how to get to an end-host. Disadvantage: one switch fails every information in that switch goes away.
DCN - Fat tree topology for scalable layer-2 (transfer data between nodes) routing, forwarding, and addressing: if you have 2 links below, you need 2 links above. Uses PMAC to specify pod, position, etc. Centralized Fabric Manager (less configuration cost and management) responsible for ARP reso, fault tolerance & multicast.



System	Topology	Forwarding		Routing	ARP	Loops	Multicast
		Switch State	Addressing				
TRILL	General	*O(number of global hosts)*	Flat; MAC-in-MAC encapsulation	Switch broadcast	All switches map MAC address to remote switch	TRILL header with TTL	ISIS extensions based on MOSPF
SEATTLE	General	*O(number of global hosts)*	Flat	Switch broadcast	One-hop DHT	Unicast loops possible	New construct: groups
PortLand	Multi-rooted tree	*O(number of local ports)*	Hierarchical	Location Discovery Protocol; Fabric manager for faults	Fabric manager	Provably loop free; no additional header	Broadcast-free routing; multi-rooted spanning trees

NDN - named data networking; caching aka favor storage over bandwidth. Protect the content, not the channel.
SDN - Splits data plane (process+forward user traffic) & control plane (run routing protocols and other control protocols, compute routing table). Network devices are black boxes that are difficult to operate and enhance & network systems are distributed, hard to reason and control since have to contact vendors; instead, use SDN to split data + control plane, prolong life of hardware. In Google's case, implement control logics over a global network view.
OpenFlow; the open interface to hardware between Network OS & forwarding hardwares. Each device maintains flow tables, if no entry in flow table, contact Controller, who will send the result to the device. Google used SDN to create a master for centralized traffic engineering.