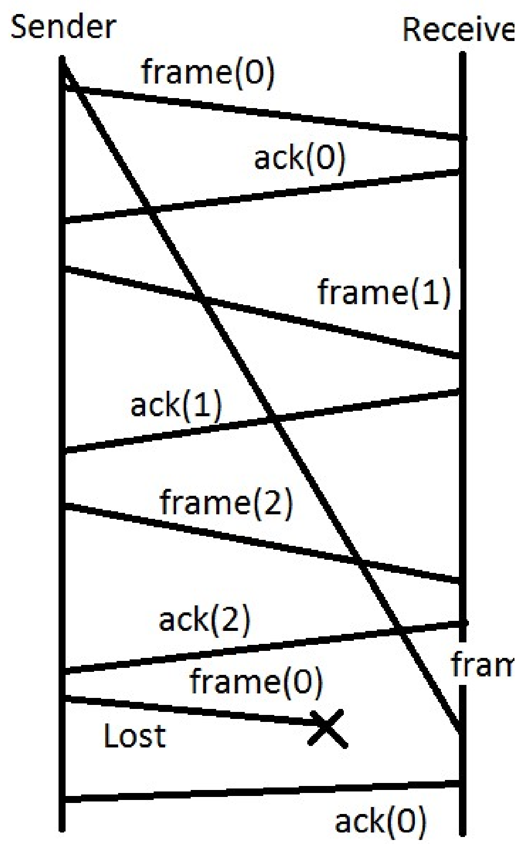
Homework 2 - 4390 Summer 2018

Alex Lundin

Question 1 (Individual Acknowledgement Protocol)

a) Give an example (i.e. a scenario) in which the channel can duplicate, reorder, and lose messages that results in incorrect behavior of the individual acknowledgement protocol even though timeouts are accurate.

ReliableDelivery-II Slide 31



This diagram shows a scenario where the Individual Acknowledgement Protocol, describes in question 1a will incorrectly acknowledge a duplicated message.

Steps:

Two copies of frame(0) show up in the channel.

Acknowledgments happen until the sequence numbers wrap around. Upon restarting at 0, frame(0) is put back in the channel. However this is lost, but frame(0) from before arrives within the timeout. Now the channel acknowledges the wrong frame(0), which was a duplicate from the very first frame sent.

b) (Part (b) is independent of part (a)). Assume that channels can reorder and lose messages, but not duplicate them. Assume timeouts are not accurate. What restrictions would you impose to make sure that the protocol works correctly? (Assume that there is an upper bound on message lifetime, perhaps a very large bound but a real bound nonetheless).

ReliableDelivery-II Slide 24
r = sequence number bound, which is large

We should limit the speed of the sender. This would control how fast we reach the upper bound of the sequence number range,

Question 2 (Switching)
a) Why can we have the packet header in virtual circuit switching to be much smaller than the typical header in datagram switching?

Because the in virtual circuit switching, the connection request contains the full destination address. The header only contains a few characters. In datagram switching, each packet must carry the entire destination address.

b) Assume that we are using datagram switching (i.e. we are not using virtual circuits). Is it possible to guarantee bandwidth to each source-destination pair? Are there disadvantages of doing this using datagram switching?

It is difficult to guarantee quality of service in datagram routing. There is no mechanism to determine what packet belongs to what node in datagram routing.

c) What is the big-O complexity of processing the header of a packet if we are using virtual circuits? Briefly explain why.

Hash table. O(1).

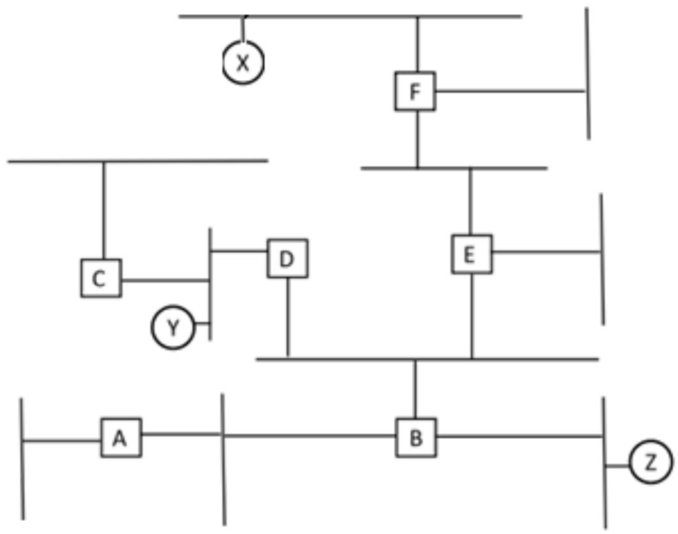
d) Why in virtual circuits do we need a datagram routing table (in addition to the virtual circuit table)

Because initially, the virtual circuit tables are empty, so virtual circuit routing needs a routing table during the setup phase. Once the setup phase is complete, then the virtual circuit tables are filled out.

e) In the Internet, we have the option of doing source routing (IP does have an option for this). What do you think could be the uses for it? Can you come up with two of them?

Source routing gives the ability to specify the IP path for network communications. This could be used for security purposes, to avoid a certain network. It could also be used to directing traffic to a lower cost ISP than a higher cost ISP.

Question 3 (Ethernet Bridges, i.e. switches)



Consider the above extended LAN, where squares are bridges (also known as switches) and circles are hosts.
Assume we start in a state where all bridges have no knowledge of any hosts.
ARP request:
Last slide notes

a) Assume first that X sends an ARP REQUEST message looking for Y. After this completes, what has each bridge learned?
Each bridge will now know the mac address of X because the ARP REQUEST is a broadcast message that floods the network.

b) Assume then that Y sends an ARP REPLY message to X. After this completes, what has each bridge learned?
The ARP reply will go directly from Y to X. So, the bridges D, B, E and F will now know the MAC address of Y.

c) Assume that after (a) and (b) above complete, Z sends an ARP REQUEST looking for Y. After this completes, what has each bridge learned?

Each bridge will now know the MAC address of Z because an ARP REQUEST is a broadcast message that floods the network.

Question 4 (Distance Vector Routing)

Modify the code for the distance vector routing protocol such that every node p has an input array linkCost: array [G] of integer
where linkCost[g] is the cost of the link from the node to the neighbor g (assumed simply to be one in the current code). In addition to this, also add split-horizon with poisoned-reverse to the code.

```
node p

in class p

  G   : set {g | g is a neighbor of p}

  up  : array [G] of boolean,        {is my neighbor up}

  linkCost: array [G] of integer


var

  rtb  : array [NID] of G,            {routing table}

  cost : array [NID] of integer,      {cost to reach each node}

  d    : NID,                         {temporary variable}

  c    : array [NID] of integer       {temporary variable}

param

          g:          element of G                    {g can be any neighbor}


event: receive update(c) from g  then

resp:   {compare costs to all destinations}

                for every d do    {for every destination, compare c[] and cost[] }

                        if d = p then cost[p] := 0

                else if d ≠ p ∧

                                (rtb[d] = g ∨ cost[d] > c[d] + LinkCost[g])

                        then

                                rtb[d]  := g;

                                cost[d] := c[d]+ LinkCost[g]

                        end if

                end for

end {code}
```

Question 5 (Broadcast for link state routing)
CH4-BcastForLinkState Slide 7
Consider a modification of the broadcast for link-state routing protocol as follows. Each node p has an additional array variable as follows:

parent: array [NID] of G

Initially, all nodes have all values of array parent set to nil. Then, when any node p receives a bc(q, t, v) message from neighbor g, it does the following:

- If parent[q] = nil then
  - ts[q] := t; d[q] := v; parent[q] := g;
  - forward bc(q,t,v) to all neighbors except g.
- If parent[q] != nil and g != parent[q] then throw the message away
- If parent[q] != nil and g = parent[q] and ts[q] < t then
  - ts[q] := t; d[q] := v; parent[q] := g;
  - forward bc(q,t,v) to all neighbors except g.
- If parent[q] != nil and g = parent[q] and ts[q] >= t then throw the message away

Assuming that there is no message loss in any channel, and that no node ever dies, does this protocol work? (I.e. each node always learns the latest value of d[q] for all other nodes q). Argue why yes or why no.

Yes, I think that all nodes will receive all the most current values with this algorithm because the method is driven by the timestamp value. This method covers the initial setup when a node has no parent, the method also uses the fact that some nodes already have parents, so they can disregard messages from anyone that is not their parent. Finally, there is a condition to update a value with a newer timestamp, from the current parent.

Question 6 (link-state routing)
Assume every node sends link-state advertisements (LSA) every 30 seconds, and that the flooding of a LSA takes about 2 seconds. Assume there is a routing loop (nodes pointing to each other) at time t. What is the maximum time that the loop could remain? (assuming no topology changes and no link cost changes). Briefly explain why.

In Link State Routing the link state packet sent from each node gives a snapshot of the topology. So at time t=32 seconds, the entire network will be known, so the loops are guaranteed to be removed a t>=32 seconds.

Question 7 (Ethernet)
Assume that a system administrator makes a mistake, and he/she uses too many repeaters (i.e. hubs) in the network, yielding an end-to-end propagation delay that is larger than the allowed by the standard. Show me a scenario (in detail) of how this can cause an improper behavior of the network.

Lan and Mac Slide 12

More repeaters increase the time it takes from the hosts that are farthest away from each other to send messages. If the repeaters increase the propagation delay to longer than the channel timeouts, then the message will reach the receiving host AFTER the message timeout declares the message as dead. So, the receiving host will automatically disregard the message since it is past the maximum timeout.

Question 8 (MACA)

Assume that we have two values, r and R (r < R). We call r the "receiving range" and R the "interference range"

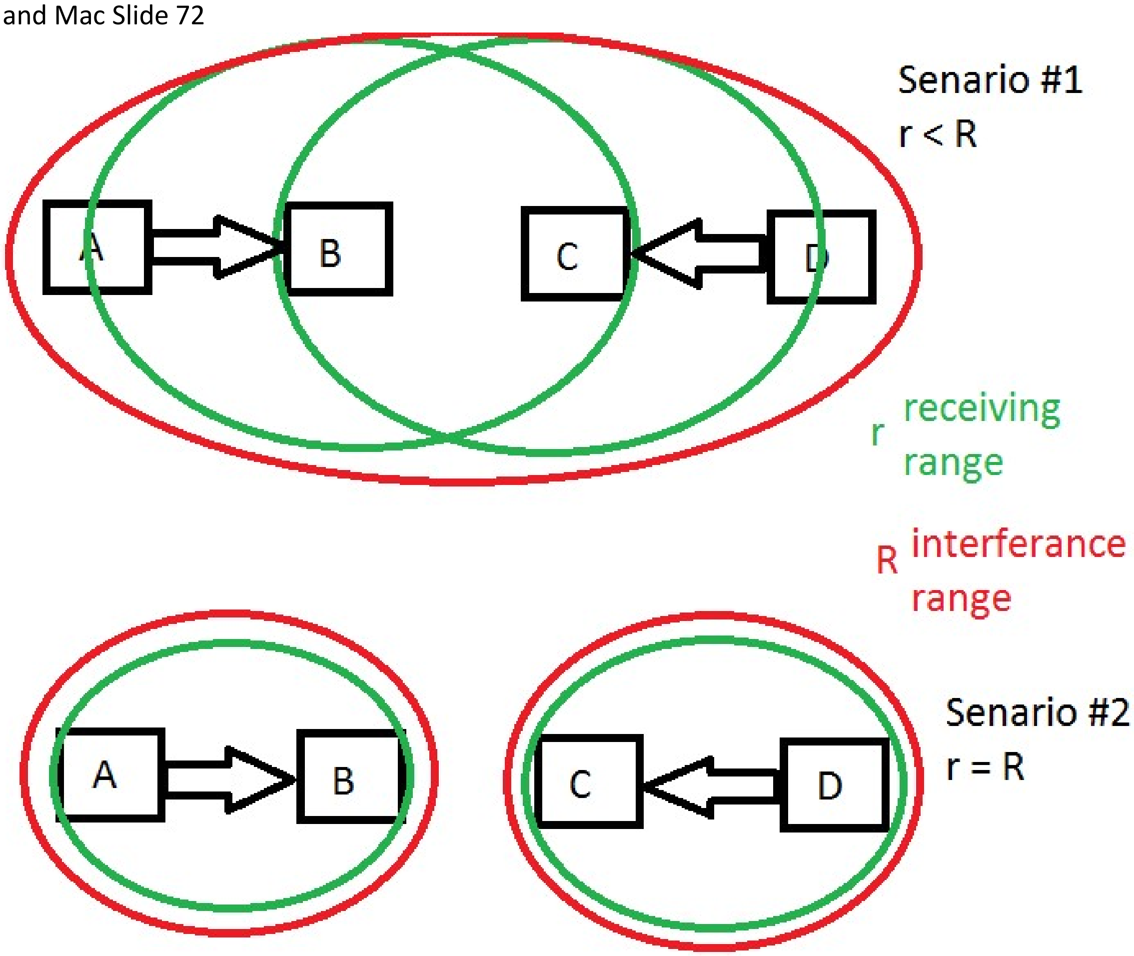
When two nodes, A and B, are r meters apart, if A transmits, then B can correctly receive the information transmitted by A.

However, if the distance from A to B is greater than r but at most R, then: a) B will be able to sense that someone is transmitting, but it cannot correctly read the information, and b) if B is receiving a message from another node C, and A transmits, B's reception of C's message is corrupted.

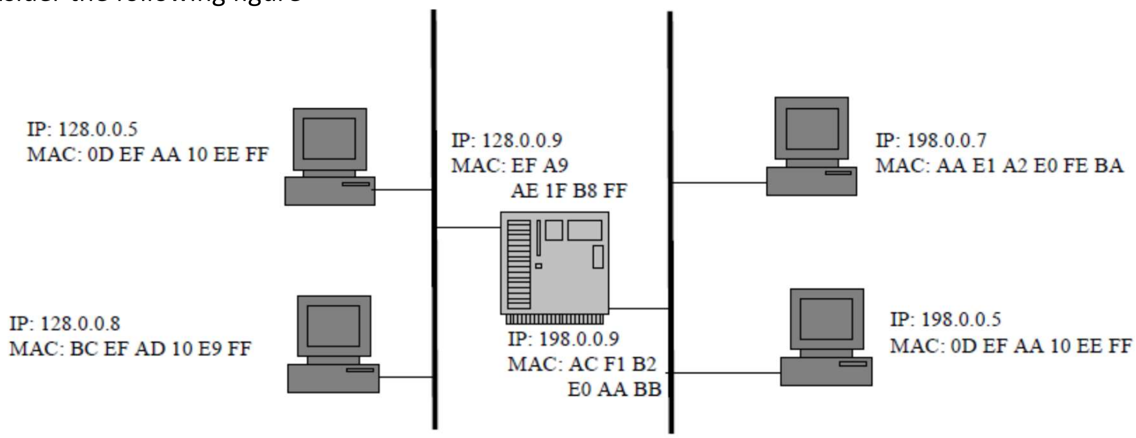
Under these conditions, show me a scenario where a collision occurs when r < R, and the same scenario would not result in a collision if r = R.

Note that in the slides we have assumed that r = R.

Lan and Mac Slide 72



Question 9 (IP Addressing)
Consider the following figure



IP: 128.0.0.5 MAC: 0D EF AA 10 EE FF IP: 198.0.0.5 MAC: 0D EF AA 10 EE FF IP: 198.0.0.7 MAC: AA E1 A2 E0 FE BA IP: 128.0.0.8 MAC: BC EF AD 10 E9 FF IP: 128.0.0.9 MAC: EF A9 AE 1F B8 FF IP: 198.0.0.9 MAC: AC F1 B2 E0 AA BB

a) Note that hosts 198.0.0.5 and 128.0.0.5 have the same physical (MAC) address. Will this cause any problems? Argue why yes or why no.

Yes, it will cause problems. The switch learns a MAC address from the host that sent the most recent packet. So, the switch will update its table every time it gets a new packet from host 198.0.0.5 and it will also update its table when it receives a packet from 128.0.0.5. Whoever is the most recent communicator with the switch, will overwrite the other hosts data. The switch will only store the most recent MAC address of any hosts that share the same MAC address.

b) Assume that host 198.0.0.7 sends a TCP message to host 128.0.0.8 using the broadcast physical (MAC) address. Will the TCP process in host 198.0.0.5 receive (i.e. see) a copy of this message? Will the IP process in host 198.0.0.5 receive (i.e. see) a copy of this message? Argue why yes or why no

The TCP process of 198.0.0.5 will not receive a copy of this message because the IP process of 198.0.0.5 will realize the message is not addressed to I, therefore the IP processes will not send the message up to the TCP process.

The IP process of 198.0.0.5 will receive a copy of this message because the sender is using broadcast communication from the switch, meaning every host on the network will receive a copy of the message at the IP level. The IP process of each host will determine if it needs to pass the message up to the TCP layer. This only happens when the message is addresses to the specific host in question, in this case, 128.0.0.8 will be the only host whose IP layer passes the message up to the TCP layer.

Question 10 (IP configuration)

Assume you have a PC attached to an Ethernet, and the PC has been turned off for days. Then, you turn the PC on, and after it boots, you try to fetch the web page www.cs.utdallas.edu. List all the packets that the PC will transmit over the Ethernet before the first message from www.cs.utdallas.edu is received by the PC.

Since the PC is off, the ARP cache will be empty. So, the first message will be a ARP Request to www.cs.utdallas.edu, this message will travel to the default gateway because the ARP cache is empty. Once the ARP request makes its way to www.cs.utdallas.edu's sever, then www.cs.utdallas.edu's sever will send a ARP reply to our PC. The MAC address of www.cs.utdallas.edu's sever will be stored in the cache. From there we can receive packets by directly making requests from the MAC address that is stored in the ARP cache now.