Joakim Edlund
IK2218
2015-09-21

HW2

1. ARP

a. Firstly H3 will send an ARP query packet to all the hosts in its network since it needs the physical address to R1. Because of the ARP snooping H4 will save the MAC address of H3 in its cache but won't respon. R1 will do the same but will also respond to H3 and provide it with its MAC address. Now H3 can make send the ARP resolution to H2 via R1. However R1 doesn't know the physical address of H2 so it has to send ARP query packets to all in the left network since it knows that's the network it resides in. This means H1 will pick up the address of R1 and so will H2 but only H2 will answer R1 with an address. Now R1 can let the ARP resolution sent by H3 reach H2. The final caches will be:

H4:

IP-address	MAC-address
E	e

H3:

IP-address	MAC-address
B	b

R1:

IP-address	MAC-address
E	e
D	d

H1: A, a,

IP-address	MAC-address
A	a

H2:

IP-address	MAC-address
A	a

b. B1:

IP-address	MAC-address
S	d
E	a

Joakim Edlund
IK2218
2015-09-21

2. UDP and fragmentation

a. The IP-header takes 20 bytes and the UDP-header takes 8 bytes leaving 1472 bytes of application data for every fragment. 7400/1472 ≈ 5,03 which means that to fit all data a total of 6 fragments has to be transmitted.

b. The MF-bit is set for all fragments except the last one indicating that there are more fragments to follow.
The offset field will be in order 0, 185, 370, 555, 740 and 925.
The length field will be in order 1500, 1500, 1500, 1500, 1500, 68.

3. Routing

a. It will only contain the directly connected networks.

Network	Distance	Interface
208.218.2.0/24	1	-
208.218.4.0/24	1	-

b. E will send all its known routes to D. So there will be 4 new posts for D, however 2 of them will be to networks it already has in its routing table and they also have a shorter distance. This means according to one of the RIP operations it will mark the new longer routes distance to infinity (16).

Network	Distance	Interface
208.218.2.0/24	1	-
208.218.4.0/24	1	-
208.218.2.0/24	16	208.218.4.2
208.218.4.0/24	16	208.218.2.3
208.218.5.0/24	2	208.218.2.3
208.218.5.0/24	2	208.218.4.2

c. A sends three RIP responses, one to B, one to D and one to E.
The response message from A to D, interface 208.218.2.1 to 208.218.2.2, will be:
(network, distance, interface)
(208.218.1.0/24, 2, 208.218.2.1)
(208.218.4.0/24, 3, 208.218.2.1)
(208.218.5.0/24, 3, 208.218.2.1)
The response message from A to B, interface 208.218.1.1 to 208.218.1.2, will be:
(208.218.2.0/24, 2, 208.218.1.1)
(208.218.4.0/24, 3, 208.218.1.1)
(208.218.5.0/24, 3, 208.218.1.1)

Joakim Edlund
IK2218
2015-09-21

The response message from A to E, interface 208.218.1.1 to 208.218.1.2, will be:
(208.218.1.0/24, 2, 208.218.2.1)
(208.218.4.0/24, 16, 208.218.2.1)
(208.218.5.0/24, 16, 208.218.2.1)
The two last distances in the message from A to E will be infinite (16) because of the split-horizon with poison reverse. A remembers it got the routes from E so it will assume the routes will be longer if E will go through A so it sets the distance to infinity to avoid getting loops.

d. The routing state of D would be:

Network	Distance	Interface
208.218.2.0/24	1	-
208.218.4.0/24	1	-
208.218.2.0/24	16	208.218.4.2
208.218.4.0/24	16	208.218.2.3
208.218.5.0/24	2	208.218.2.3
208.218.5.0/24	2	208.218.4.2
208.218.1.0/24	2	208.218.2.1
208.218.4.0/24	16	208.218.2.1
208.218.5.0/24	16	208.218.2.1

The routing state of E would be:

Network	Distance	Interface
208.218.2.0/24	1	-
208.218.4.0/24	1	-
208.218.5.0/24	1	-
208.218.1.0/24	2	208.218.2.1
208.218.4.0/24	16	208.218.2.1
208.218.5.0/24	16	208.218.2.1

Joakim Edlund
IK2218
2015-09-21

4. ICMP

a. H1 will send an echo request message to H2 and H2 will answer with an echo reply message.

b. First H1 sends an echo request with TTL 1 to A. A then sends a Time exceeded message back to H1. Then H1 sends a new echo request but with TTL 2. A will then decrease TTL to 1 and send it to B. B then responds with Time exceeded. H1 tries again but with TTL 3 and this time C also responds with Time exceeded. Finally, with TTL 4, H3 will answer with an echo response back to H1. Sent packets:

Destination	Message type	TTL
90.59.5.2	echo request	1
90.59.5.2	echo request	2
90.59.5.2	echo request	3
90.59.5.2	echo request	4

Received packets:

Source	Message type	TTL
90.59.1.1	time exceeded	0
90.59.2.2	time exceeded	0
90.59.4.2	time exceeded	0
90.59.5.2	echo response	0

5. TCP

a. The MSS used by TCP in this case will be the MTU value minus the size if the IP header and the TCP header. 1040 - 20 - 20 = 1000 bytes.

b. The bandwidth delay product is about (0,2s * (4*10^6)bps) /8 = 100 KB and the receiver window size is 5 KB. It should be a bit larger than 15 KB since that is how much A wants to send at it fits without a problem on the line. So no the advertised window size of the receiver window B is not big enough and it should be a bit larger than 15 KB.

c. There will be 15 segments sent since it will send 15000 bytes of data and the MSS is 1000 bytes. These are the segments, time is relative to t0 and sequence number is ISN+1:

Time (ms)	Sequence no.	SRTT	RTTVAR	RTO
1	1	200	100	600
403	1041	225.125	125.25	726.125

404	2081	225.125	125.25	726.125
606	3121	222.11	99.97	722.11
607	4161			
608	5201			
609	6241			
809	7281			
810	8321			
811	9361			
812	10401			
813	11441			
814	12481			
815	13521			
816	14561			

d. The last segment is sent at time t0 + 816 ms and the propagations time is 100 ms to arrive at host B. Host B will then create an ACK for 1ms, since it now has 2 segments received, and send it back at time t0 + 917 ms. The propagation time is 100 ms so A receives the ACK for the last segment at time t0 + 1017 ms.