Homework 2

Md Ali A20439433 & Samantha Berg A20456450
CS 542: Computer Network Fundamentals I

December 6, 2020

Exercise 1. Consider the following details of three hosts and a router with proxy ARP for them.

Node	**IP Address**	**Physical Address**
Host A	130.15.40.15	AA223344BB12
Host B	130.15.40.16	AA223344BB34
Host C	130.15.40.17	AA223344BB56
Router	115.8.9.18	AA223344BB78

Router received an ARP request for host B from a host (IP address: 115.8.9.56, Physical address: 346738ABCDE0). Give the ARP request and reply packet format filled with all necessary fields. (Consider the Ethernet as hardware type and IPv4 as protocol type)

Proof. Since the router has proxy ARP this means there is an ARP acting on behalf of the the set of hosts provided. These hosts are A, B, and C. The question is regarding ARP packets, so the ARP packet has a format as shown below.

<table>
<tr><td colspan="2">Hardware Type: Ethernet is type 1</td><td colspan="2">Protocol Type: IPv4 is 0x0800</td></tr>
<tr><td colspan="2">Hardware Length: Ethernet is 16</td><td>Protocol Length: IPv4 is 4</td><td>Operation: 1 or 2</td></tr>
<tr><td colspan="4">Sender Hardware Address</td></tr>
<tr><td colspan="4">Sender Protocol Address</td></tr>
<tr><td colspan="4">Target Hardware Address</td></tr>
<tr><td colspan="4">Target Protocol Address</td></tr>
</table>

Now we need to convert some IP addresses into hexadecimal format in order to complete our ARP request and reply packets. Below is the needed IP address that we converted into hexadecimal

$$\text{Request: } 115.8.9.56 \longrightarrow 0x73080938$$
$$\text{Router: } 115.8.9.18 \longrightarrow 0x73080912$$
$$\text{Host B: } 130.15.40.16 \longrightarrow 0x820F2810$$

Using this format we can now answer our question about the ARP request and reply packet format. Notice for an ARP request, this should be the value 1 in *Operation*, while it would be 2 for ARP reply.

In an ARP request, the target hardware address will always be all 0s. Below is the ARP request.

0x0001	0x0800	
0x06	0x04	0x0001
0x346738ABCDE0		
0x73080938		
0x000000000000		
0x820F2810		

Below is the ARP reply.

0x0001	0x0800	
0x06	0x04	0x0002
0xAA223344BB78		
0x820F2810		
0x346738ABCDE0		
0x73080938		

□

Exercise 2. The sender A (IP address: 141.23.56.21, Physical address: A46EF45983AB) is a host and wants to send a packet to another host B (IP address: 114.5.7.89, Physical address: 457342ACAE32) on another network. The next-hop (router) for this destination in the sender's routing table is Router R1 (IP address: 141.23.56.24, Physical address: A46EF45983DE). Give me the ARP request packet format filled with all necessary fields from sender A. (Consider the Ethernet as hardware type and IPv4 as protocol type)

Proof. We will need the ARP request and we will use the ARP format from question 1, but this is again shown below.

Hardware Type: Ethernet is type 1	Protocol Type: IPv4 is 0x0800	
Hardware Length: Ethernet is 16	Protocol Length: IPv4 is 4	Operation: 1 or 2
Sender Hardware Address		
Sender Protocol Address		
Target Hardware Address		
Target Protocol Address		

We are asked to provide the ARP request from sender A. We will first convert all our IP address to hexadecimal format. This is shown below.

$$\text{Sender A: } 141.23.56.21 \longrightarrow 0x8D173815$$
$$\text{Host B: } 114.5.7.89 \longrightarrow 0x72050759$$
$$\text{Router R1: } 141.23.56.24 \longrightarrow 0x8D173818$$

The ARP request is shown below for sender A, and as always the target hardware will always be 0s.

<table>
<tr><td colspan="2">0x0001</td><td colspan="2">0x0800</td></tr>
<tr><td>0x06</td><td>0x04</td><td colspan="2">0x0001</td></tr>
<tr><td colspan="4">0xA46EF45983DE</td></tr>
<tr><td colspan="4">0x8D173815</td></tr>
<tr><td colspan="4">0x000000000000</td></tr>
<tr><td colspan="4">0x72050759</td></tr>
</table>

□

Exercise 3. Consider a network where the hardware address length is 6 bytes, and the protocol address length is 4 bytes. A node in this network received an ARP packet where all bits from 145th to 192nd are 0s. Is that ARP packet a request or a reply? Explain your answer.

Proof. When the node receives the ARP packet we take note from the problem that the hardware length is 6 bytes, protocol address length is 4 bytes, and that all bits from 145th to 192nd are all 0s. The ARP packet is indeed a request due to the operation

The ARP packet is a request due to the opcode field not being 2. The significant difference between the Request and the Response is included in the Opcode field. In an ARP response, in this specific field it contains the value of 2.

□

Exercise 4. Consider the ARP package. When is a queue created and an IP packet added to it?

Proof. Considering an ARP package, the ARP package will maintain a set of queues to hold the IP packets while the ARP attempts to resolve the hardware address. An output module sends and unresolved packets into a queue. The queue is created when there is unresolved packets that needed to be stored inside a queue. Hence the queue is created. An IP packet is added to the queue and will wait until and checks the cache table if we're receiving a IP packet. The output module checks the cache table to find an entry corresponding to the destination IP address of the packet. The destination IP address of the IP packet must watch the protocol address of the entry. If the entry is found and the state is *resolved*, the packet along with the destination hardware address is passed to the data link layer for transmission.

Now if the entry is found and the state of the entry is *pending*, the packet waits until the destination hardware address is found. Since the state is already *pending* there is already a queue created, so the packet gets sent to the queue. If there is no entry found, the module creates a queue and proceeds to enqueues the packet. Then a new entry with the state of *pending* is created and then a separte variable *attempts* is set to 1. Then an ARP request packet is then broadcast. □

Exercise 5. Consider the updated cache table. The maximum number of attempts is 5, and the time-out value is 900 seconds. After 110 seconds, the input module receives two ARP packets. These are the only two packets host received in the last 110 seconds. What is the updated cache table after these packets have been received? Support your answer with an explanation. (Consider cache table is updated every 10 seconds).

Packets received:

- An ARP reply from the host with IP address 19.1.7.82 and Physical address 4573E3242ACA
- An ARP reply from the host with IP address 220.55.5.7 and Physical address A46EF45983BC

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Address*	*Hardware Address*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
F					
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

Proof. For every state that is a *free* entry in the cache table, the Cache Control Module does nothing and continues. For every state that has a *pending* entry, the *Cache Control Module* increments the *attempts* filed by 1. If the *attempts* reaches the maximum number of attempts, in our case 5, the state will change to *free* and the queue is discarded. If the state is *resolved*, then the *Cache Control Module* decrements the timeout value by the time elapsed. In the problem, we are told that the total time elapsed is 110 seconds. This means that for every *resolved* entry, the timeout will decreased by 110 seconds. For the entry with the protocol address 19.1.7.82 , the state will change from *resolved* to free at the 60 second mark of the 110 elapsed time. During this time, the *pending* entry, the *attempts* field values is increased by 1 every 10 seconds. At the elapsed time of 110 seconds, the following protocol values would be destroyed since there would be an increased of 11 attempts, assuming that they were not resolved. The following list is of pending state with their corresponding protocol address that will be destroyed in the 110 seconds elapsed.

- 129.34.4.8 - Attempts 2 to 13 after elapsed time. This will be destroyed since it is greater than the max number of attempts.

- 201.11.56.7 - Attempts 5 to 16 after elapsed time. This will be destroyed since it is greater than the max number of attempts.
- 220.55.5.7 - Attempts 1 to 12 after elapsed time. This will be destroyed since it is greater than the max number of attempts.
- 188.11.8.71 - Attempts 1 to 14 after elapsed time. This will be destroyed since it is greater than the max number of attempts.

At the 110 second mark, the input module receives two ARP reply from the host with IP address 19.1.7.82 and 220.55.5.7. For the protocol 19.1.7.82, the state became *free*, at the 60 second mark and the corresponding queue was destroyed. A new entry for this IP address will not be created because this is an ARP Reply Packet. Since, the packet is an ARP reply it already contains its hardware address and an entry in the cache table will not be created. For the protocol 220.55.5.7, this was destroyed after reaching the maximum number of attempts at the 40 second mark, which also concluded for the corresponding queue to be destroyed. A new entry for this IP address will not be created because this is an ARP Reply Packet. Since, the packet is an ARP reply it already contains its hardware address and an entry in the cache table will not be created. The table after the 110 second time elapse is created below.

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Address*	*Hardware Address*
R	5		790	180.3.6.1	ACAE32457342
F					
F					
R	8		340	114.5.7.89	457342ACAE32
F					
F					
F					
F					

□

Exercise 6. In an IP packet, the value of HLEN is 1110 in binary, and the value of the total length is 0000000110000000. How many bytes of data is this packet carrying?

Proof. First we will convert both of these values from their binary forms into decimals. This is showcased below.

HLEN bits = 1110 $\longrightarrow 14 \cdot 4 = 56$ bytes
Total Length bits = 0000000110000000 $\longrightarrow$ 384 bytes

From there we can calculate the number of bytes in the data field length with the following equation below.

Packet length = Data Field Length + Header Length

Plugging in our values we get

384 bytes = Data Field Length + 56 bytes $\longrightarrow$ Data Field Length = 328 bytes

Hence from above, the packet is carrying 328 bytes of data. □

Exercise 7. The total IP datagram length is 200 bytes, out of which the HLEN value is 1111, and the size of options in the header is 30 bytes. Is this example a valid datagram or not? Give your supporting reasons.

Proof. This is not a valid datagram because the HLEN value is equal to 60 bits meaning that this option is at it's max. Hence the size of option in the header can't be 30 bytes. □

Exercise 8. An IP fragment has arrived with the first few hexadecimal digits, as shown below:

4500282400012367.....

This is the second fragment. How many bytes of data does this fragment contain? What is the third fragment offset?

Proof. From the second byte, we can see that our header length will be $5 \cdot 4 = 20$ bytes. To get he payload or data we must look at the byte values 0x2824 and that will be how many bytes of data that this specific fragment contain.

0x2824 $\longrightarrow$ 10276 bytes is the amount of data this fragment contains, without the header this number will be 10256 bytes.

To locate the offset of the thrid fragment, we look at the last four digits 2367. Converting this into binary we get

0x2367 $\longrightarrow$ 10001101100111

We know that the flag has to be 3 bits long and the fragmentation needs to be 13 bits, so we add two zeros to the front of this binary making the number, 0010001101100111. From there we take the first three bits away to find the third fragment offset, which is 0001101100111. Converting this to decimal below we get

0001101100111 $\longrightarrow$ 871

Hence the third fragment offset is 871. □

Exercise 9. A packet has arrived with a D bit value of 0 and an M bit value of 0, and fragmentation offset value to zero. Is this packet the first fragment, last fragment, middle fragment, or the only fragment?

Proof. We can conclude that D bit is 0, M bit is 0, and the fragmentation offset is 0. Since M bit is 0 and fragmentation offset is 0, this makes the fragment the only fragment. □

Exercise 10. An IP packet has arrived with the first few hexadecimal digits, as shown below:

460000400001000023.....

The initial 'Time to Live' value in the hexadecimal format is AB. How many hops has this packet already traveled? How many hops can this packet travel before being dropped?

Proof. We are given that the initial "Time to Live" is given in hexadecimal as AB. This translates to 171 hops, so initially the IP packet could have traveled 171 hops. In our hexadecimal fragment to find the current hops that are left we will skip the first 16 hexadecimal digits and then look at the the digits. Here we see that our 23 in hexadecimal left which translates to in decimal to 35 hops that this IP packet can travel to.

As a result this IP packet has traveled 136 hops, since 171-35=136, and the IP packet has 35 hops left before being dropped. □

Exercise 11. When does the Fragmentation Module of the IP Package send an 'ICMP' error message?

Proof. The Fragmentation Module of the IP Package sent an ICMP error message when a packet is too big for a physical link, an intermediate router might divide it into various smaller datagrams in order to make it suitable. This process is considered "forward" IP fragmentation and the smaller datagrams are called IP fragments. A solution to these problems was included in the IPv4 protocol which states A sender can set the DF (Don't Fragment) flag in the IP header, requesting intermediate routers to never perform fragmentation of a packet. Instead a better option would be for a router with a link containing a smaller MTU will send and communicate an ICMP message "backward" and inform the sender to reduce the MTU for this connection. □

Exercise 12. The IP datagram was fragmented. The first fragment contains bytes 0 to 399. The total amount of data(not including the header) in the datagram is 2500 bytes. Each fragment has only a base header. How many fragments are needed to send this datagram?

Proof. We know that the total amount of data in the first fragment contains 400 bytes. This includes a 20 bytes header space, meaning that the maximum payload in each fragment is 380 bytes. So taking the total size of the datagram and dividing that by the maximum payload size while using the ceiling function we will have the following.

$$\text{Number of fragments} = \lceil \frac{2500}{380} \rceil \longrightarrow \lceil 6.58 \rceil = 7$$

Hence, from above we can see that we need 7 fragments in total to send this datagram. □

Exercise 13. A client has received a UDP datagram. The corresponding port number has been found in the control block table, but there is no queue number. Is it possible? Why?

Proof. A client has received a UDP datagram. The corresponding port number has been found in the control block table, but there is no queue number. It is possible, because if there is no queue number then we must allocate a queue. If the corresponding entry in the control block table is not found then we must ask the ICMP module to send an "unreachable port" message and discard the user datagram. However, if it was found then we must check the queue field to see if the queue has been allocated. Then we enqueue the data in the corresponding queue. □

Exercise 14. The TCP sliding window values of rwnd and cwnd are 16 and 10, respectively. The last acknowledgment number was 203. A segment with the acknowledgment number 207 and the rwnd of 8 has just been received. Draw a diagram showing the window before and after. Assume that the cwnd hasn't changed.

Proof. Starting with the before diagram. We take the minimum value between rwnd and cwnd to find the windows size.

Size of sliding window before = Minimum(rwnd, cwnd) = Minimum(16,10) = 10. Hence our sliding window has size of 10. We also know for previous acknowledgement number was 203. From there we have the representation below.

...	203	204	205	206	207	208	209	210	211	212	...

Hence the window size is 10
Closing window will be 203
Opening window will be 212

Now we will take a look at the window after. The next acknowledgement number is 207, with a new rwnd of 8.

Size of the sliding window after = Minimum(rwnd, cwnd) = Minimum(8,10) = 8. Hence our sliding window has a size of 8. From there we have the representation below.

...	207	208	209	210	211	212	213	214	...

Hence the window size is 8
Closing window will be 207
Opening window will be 214 □

Exercise 15. Given the following TCP header dump in the hexadecimal format, give your answers in the decimal format

CA950071 00000257 00000000 500209AB 00000000

(i) what is the source port number?

(ii) What is the sequence number?

(iii) What is the header length?

(iv) What bits are set in the control field, and what does it mean?

(v) What is the window size?

Proof. Below is the answers to all the parts listed above

(i) The source port number in decimal format is 51861

(ii) The sequence number in decimal format is 599

(iii) The header length is 20 bytes long due the digit 5 and having to multiple that by 4.

(iv) The bits 002570 are the control field, with the none empty bits we have the PSH, RST, and SYN. Meaning that this specifically we want to push, reset connection, and synchronize sequence numbers

(v) The window size in decimal format is 2475 □

Exercise 16. The UDP header in the hexadecimal format is 0913 000D 00AE E247.

(i) What is the source port number?

(ii) What is the destination port number?

(iii) What is the total length of the user datagram?

Proof. Below is all the parts of the problem

(i) Here we know that the source port number is the first 4 hexadecimal digits is 0913_{16} to calculate the decimal value, we consider the following.

$$0913_{16} = 0 \cdot 16^3 + 9 \cdot 16^2 + 1 \cdot 16^1 + 3 \cdot 16^0$$

Hence the source port number is 2323.

(ii) The destination port number of the second four hexadecimal digits is $000D_{16}$ to calculate the decimal value, we consider the following.

$$000D_{16} = 0 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + D \cdot 16^0 = 0 + 0 + 0 + 13 \cdot 16^0 = 13$$

Hence the destination port number is 13.

(iii) In order to find the total length of the user datagram we must consider the four hexadecimal digits $00AE_{16}$ to calculate the decimal value, we consider the following.

$00AE_{16} = 0 \cdot 16^3 + 0 \cdot 16^2 + 10 \cdot 16^1 + 14 \cdot 16^0 = 174$

Hence the total length of the UDP is 174 bytes. □