Networks and Security
Project part 1

Mads Petersen

19/11-2014

1 Distance Vector Algorithm

When the *main* method runs the *runReferenceProblemFromSlides* method:
First the hashmap of a router's neighbours with corrosponding weights is created. Then the router object is created with its ID, the total number of routers and the hashmap with its neighbours as parameters. This is done for 3 routers that are all related to each other. A new hashmap mapping the router's ID's to the hashmap of their neighbours is then created. Then their initial distance matrices is printed with the *printAllDistanceMatrices* method. This writes the following table to standard output:

———Distance matrix———

id:0	0	1	2
0	0	4	50
1	-	-	-
2	-	-	-

Outgoing distance vector: [0,4,50]

———Distance matrix———

id:1	0	1	2
0	-	-	-
1	4	0	1
2	-	-	-

Outgoing distance vector: [4,0,1]

———Distance matrix———

id:2	0	1	2
0	-	-	-
1	-	-	-
2	50	1	0

Outgoing distance vector: [50,1,0]

then the *runAlgorithm* method is run with the routers and the hashmap containing the mapping of ID -> neighbours as parameters. In the *runAlgorithm* method, first for each router in the list of routers supplied as parameter, that routers outgoing distance vector is obtained and shared with each of its neighbours. If *outgoingDistanceVector* does not return null then the boolean variable changes is set to true. When all the routers have be iterated through and their outgoing distance vector polled and shared with neighbours, the *calculateDistanceMatrix* method is called on each router in the list of routers. The if changes is true this loop runs again. The first iteration of this loop results in the following distance matrices:

———Distance matrix———

id:0	0	1	2
0	0	4	5
1	4	0	1
2	50	1	0

Outgoing distance vector: [0,4,5]

———Distance matrix———

id:1	0	1	2
0	0	4	50
1	4	0	1
2	50	1	0

Outgoing distance vector: [4,0,1]

———Distance matrix———

id:2	0	1	2
0	0	4	50
1	4	0	1
2	50	1	0

Outgoing distance vector: 5,1,0]

This represents that the vectors have been shared with neighbours, so that every router has a full table, and more optimal routes have been calculated for individual routers, but these new vectors have not yet been shared with neighbours. On iteration 2 the following tables are printed:

———Distance matrix———

id:0	0	1	2
0	0	4	5
1	4	0	1
2	5	1	0

Outgoing distance vector: [0,4,5]

———Distance matrix———

id:1	0	1	2
0	0	4	5
1	4	0	1
2	5	1	0

Outgoing distance vector: [4,0,1]

———Distance matrix———

id:2	0	1	2
0	0	4	5
1	4	0	1
2	5	1	0

Outgoing distance vector: 5,1,0]

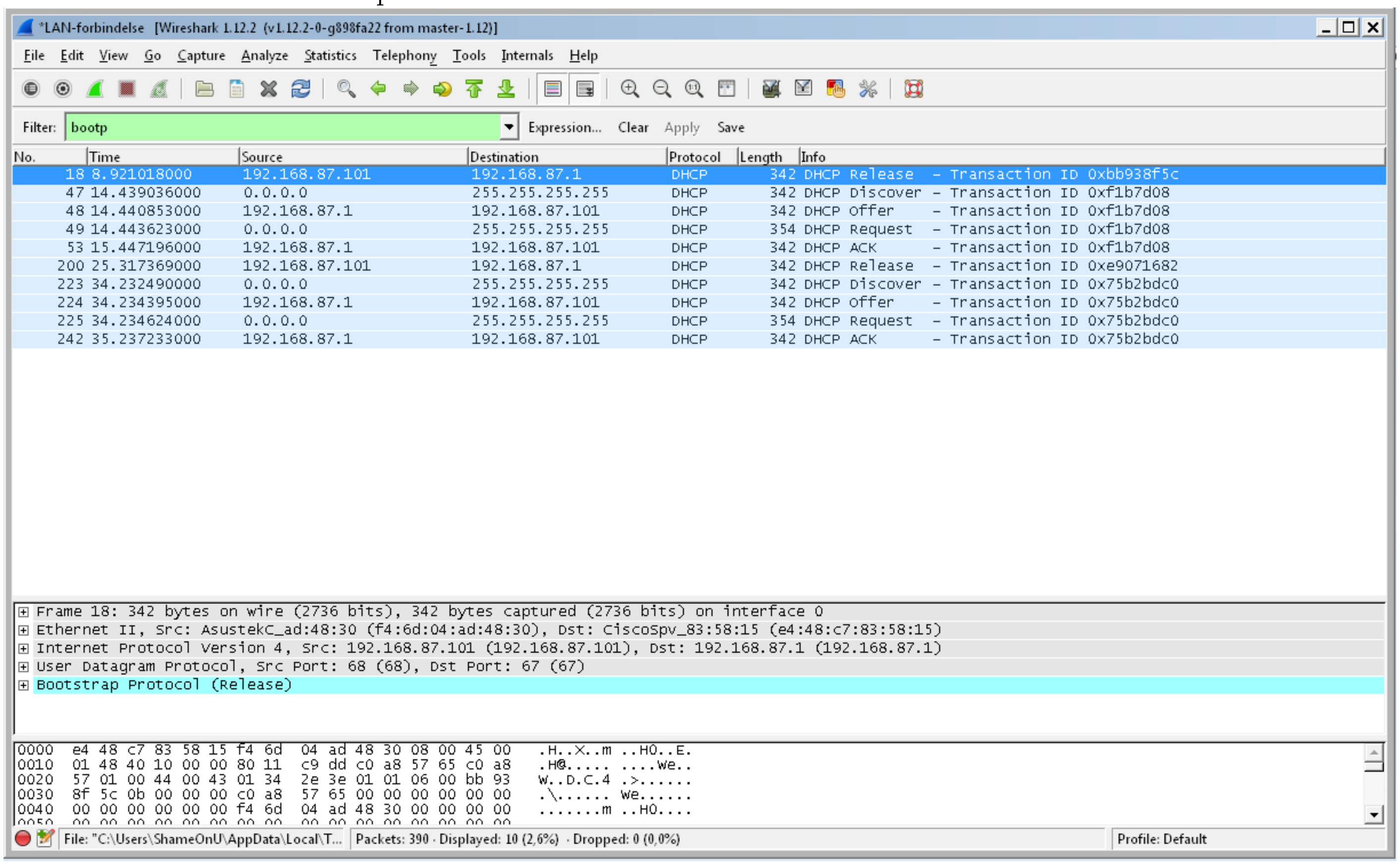
In the above tables it can be seen that the value 50 is changed to 5, this can be confirmed by observing that the only 2 routes that exsist to router 2 from router 0 is either directly since they are neighbours, or through router 1 and then from router 1 to router 2, since router 1 and 2 are neighbours. It can be seen that the direct route has a cost of 50, and the route that goes through router 1 has a cost of $4 + 1$, since it costs 4 to go from router 0 to router 1 and 1 more to go from router 1 to router 2. Now, since changes was set to true in the 2nd iteration when *outgoingDistanceVector* was polled, the loop runs one more time. But since this time the routing tables can't be optimized any further *outgoingDistanceVector* is going to return null and the exact same tabels are going to be printed one more time before the method terminates. The line that says "Outgoing distance vector:[x,y,z] was added to the code to check that they match with the distance matrix and left in because it is useful information.

When the weight to neighbours are changed to 1 for router 0 and 1 and *runAlgorithm* is called again the exact same happens. First the new distance vectors are shared and updated if necessary(i.e. faster routes exists through neighbours) and then those faster distance vectors are shared, and since no new routes can be found after this the exact same tables is printed once more and the method terminates in 3 iterations. The same is

true when the weights are updated to 60.
From these observations it should be true that the lowest number of iterations the algorithm can go through is 2, in the case where no optimization can be done, this can be confirmed by making routers that all have a cost of 1 to each other, and indeed it does go through 2 iterations. So the results does make sense. If, in *runAlgorithm*, a for-loop was used instead of the do-while loop. With a head looking something like: *for(Integer[] distanceVector = router.outgoingDistanceVector(); distanceVector = router.outgoingDistanceVector() ; distanceVector != null)* then the iterations could probably be reduced to a minimum of 1.

2 Wireshark DHCP

The screenshot of the DHCP packets:



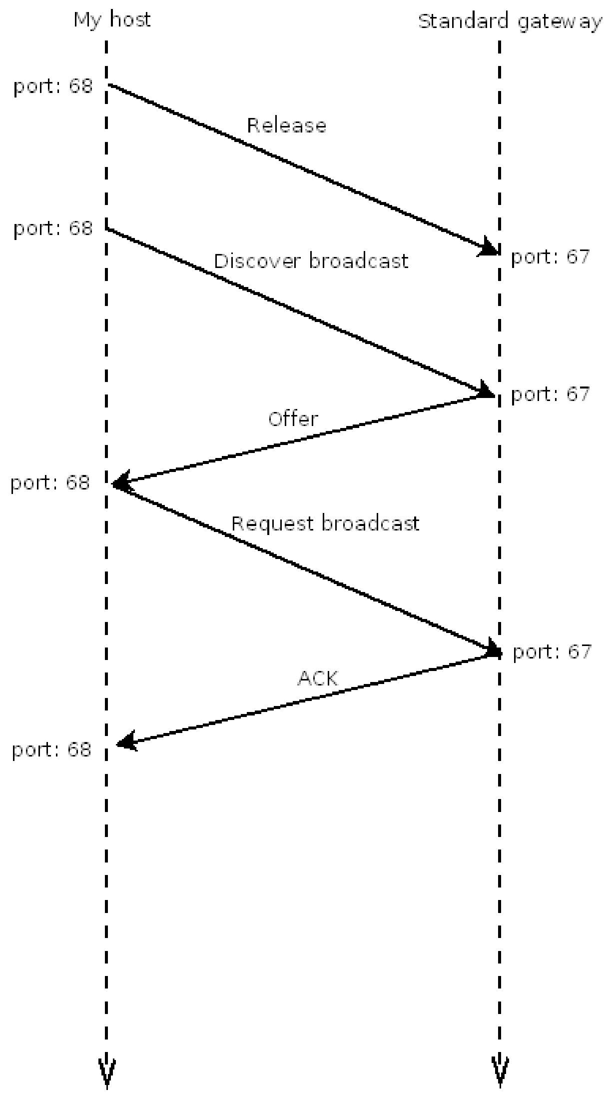
Question 1

No.	Time	Source	Destination	Protocol	Length	Info
18	8.921018	192.168.87.101	192.168.87.1	DHCP	342	DHCP Release - Transaction ID 0xbb938f5c

```
Frame 18: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Ethernet II, Src: AsustekC_ad:48:30 (f4:6d:04:ad:48:30), Dst: CiscoSpv_83:58:15 (e4:48:c7:83:58:15)
Internet Protocol Version 4, Src: 192.168.87.101 (192.168.87.101), Dst: 192.168.87.1 (192.168.87.1)
User Datagram Protocol, Src Port: 68 (68), Dst Port: 67 (67))
```

As can be seen from the above print of the first packet, User Datagram Protocol(UDP) is used.

Question 2



The source(src) and destination(dst) can be seen in the screenshot right after Userdatagram Protocol.

Question 3

Looking at the print of the first packet from above, it can be seen that the Ethernet adress can be found in the 2nd immediately following "Ethernet II, Src: AsustekC_ad:" and is f4:6d:04:ad:48:30

Question 4

There are several options in the Request message that aren't in the discover message.

No.	Time	Source	Destination	Protocol	Length	Info
49	14.443623	0.0.0.0	255.255.255.255	DHCP	354	DHCP Request - Transaction ID 0xf1b7d08

```
Frame 49: 354 bytes on wire (2832 bits), 354 bytes captured (2832 bits) on interface 0
Ethernet II, Src: AsustekC\_ad:48:30 (f4:6d:04:ad:48:30), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: 68 (68), Dst Port: 67 (67)
Bootstrap Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
```

```
    Transaction ID: 0x0f1b7d08
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: AsustekC_ad:48:30 (f4:6d:04:ad:48:30)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Request)
    Option: (61) Client identifier
    Option: (50) Requested IP Address
    Option: (54) DHCP Server Identifier
    Option: (12) Host Name
    Option: (81) Client Fully Qualified Domain Name
    Option: (60) Vendor class identifier
    Option: (55) Parameter Request List
    Option: (255) End
```

No.	Time	Source	Destination	Protocol	Length	Info
47	14.439036	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xf1b7d08

```
Frame 47: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Ethernet II, Src: AsustekC_ad:48:30 (f4:6d:04:ad:48:30), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: 68 (68), Dst Port: 67 (67)
Bootstrap Protocol (Discover)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x0f1b7d08
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: AsustekC_ad:48:30 (f4:6d:04:ad:48:30)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Discover)
    Option: (61) Client identifier
    Option: (50) Requested IP Address
        Length: 4
        Requested IP Address: 192.168.87.101 (192.168.87.101)
    Option: (12) Host Name
    Option: (60) Vendor class identifier
```

```
    Option: (55) Parameter Request List
    Option: (255) End
```

Options 54 and 81 are in the request but NOT in the discover.

Question 5

It can be seen from the screenshot under the info heading that the transaction ID from the first round of DCHP is: 0xf1b7d08, and from the second round it is : 0x75b2dbc0. The purpose of the transaction ID is to serve as identifier to both the host and the DHCP server since no IP has be estabished for the host yet.

Question 6

It can be seen from the screenshot the the source IP used in both discover and request is : 0.0.0.0 and the destination is : 255.255.255.255. The source in both Offer and ACK is : 192.168.87.1 the destination: 192.168.87.101

Question 7

The IP adress of my DHCP server is the source IP of the Offer and ACK messages i.e. 192.168.87.1.

Question 8

From the above print of the discover message, it looks like the IP 192.168.87.101 is requested by my host in the message and then repeated in the offer message, Offer message:

No.	Time	Source	Destination	Protocol	Length	Info
48	14.440853	192.168.87.1	192.168.87.101	DHCP	342	DHCP Offer - Transaction ID 0xf1b7d08

```
Frame 48: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Ethernet II, Src: CiscoSpv_83:58:15 (e4:48:c7:83:58:15), Dst: AsustekC_ad:48:30 (f4:6d:04:ad:48:30)
Internet Protocol Version 4, Src: 192.168.87.1 (192.168.87.1), Dst: 192.168.87.101 (192.168.87.101)
User Datagram Protocol, Src Port: 67 (67), Dst Port: 68 (68)
Bootstrap Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x0f1b7d08
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 192.168.87.101 (192.168.87.101)
    Next server IP address: 192.168.87.1 (192.168.87.1)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: AsustekC_ad:48:30 (f4:6d:04:ad:48:30)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Offer)
    Option: (1) Subnet Mask
    Option: (2) Time Offset
    Option: (3) Router
    Option: (23) Default IP Time-to-Live
    Option: (51) IP Address Lease Time
    Option: (54) DHCP Server Identifier
    Option: (6) Domain Name Server
    Option: (255) End
    Padding
```

Your (client) IP address: 192.168.87.101.

Question 9

The router address is the the address that the host should send packets to that has to go outside the network.

The subnet mask is used to determine when a message destination is located outside the network and thus has to go to the router. The bits in the subnet mask that are 1's indicate that this is part of the network ID and the bits that are 0's indicate that it is the host identifier, so if a message has a destination where the network ID is different from the network ID of the host's own IP address then it has to be sent to the router. This can either be done by counting the number of bit's that are 1 in the subnet mask and appending it to the IP i.e. 192.168.87.101/24, indicating that the first 24 bits are the network identifier and the last 8 bits are the host identifier, it can also be done by performing a bitwise AND with the subnet mask and the IP address.

Question 10

The release message tells the DHCP server that the client doesn't need it's IP address anymore, the ACK message is a reciept of the request message that confirms the settings requested. If the release message is lost then the IP address will be unavailable until the lease time runs out.