**HW4, five questions, 100 points**

**1.** **(20 points)** Assume a 3600 bytes packet (including 20 bytes header) arrives at a router, and the MTU for outgoing link is 700 bytes (including header). Fragmentation occurs (on the payload). Fill out the table below. Does the next router reconstruct the original packet from its fragments? Why? (2 reasons) (2 pts)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Fragment length (including header) | Flag | Offset |
| Fragment1 | **700** | **1** | **0** |
| Fragment2 | **700** | **1** | **85** |
| Fragment3 | **700** | **1** | **170** |
| Fragment4 | **700** | **1** | **255** |
| Fragment5 | **700** | **1** | **340** |
| Fragment6 | **200** | **0** | **425** |

**The next router does not reconstruct the original packet from its fragments because it would take the router longer to reconstruct the packet, then forward the packet. It would be simpler and shorter to forward the packet in its fragmented form because if the next hops require fragmentation then reconstruction becomes redundant. Also, the fragments can go to different paths to reach destination.**

**2.** **(20 points) IP protocol uses 1’s complement for IP header checksums. Suppose you have the following three 16-bit words: 0101011001100011, 1110001100000110, 0111001010101100. What is the 1’s complement of the sum of these 16-bit words? Show all work step by step in the worksheet given below. Note: first find sum of first two words, then find sum of that and the 3rd word, then find its 1’s complement. Why is it that IP takes the 1’s complement of the sum; that is, why not just use the sum? (2 points) With the 1’s complement scheme, how does the receiver detect errors? (4 points) Is it possible that a 1-bit error (one bit is flipped) will go undetected? If yes, give an example. (4 points) How about a 2-bit error? If yes, give an example. (4 points)**

**0101011001100011 (1st word)**

**ANS:**

The receiver can sum the values, include the 1’s complement then flip the sum. The nonzero result is easier than to compare two numbers bit by bit. The receiver will know if there is an error after summing the values of the word (taken from the header) and then taking the 1’s complement of that sum. If the result is not all 1, then there is an error. All 1-bit errors will be detected from this, it is not possible a 1-bit error will go undetected. It is possible for a 2-bit error to not be detected. As an example, if the sum of the header’s words is zero (which would be the 1’s complement: 1111 1111 1111 1111 1111). If during transmission, the last bit of word & last bit of checksum are corrupted then the sum is 0000 0000 0000 0001 + 1111 1111 1111 1110 = 1111 1111 1111 1111, so the 1’st complement is zero and the error would go unnoticed.

**+ 1110001100000110 (2nd word)**

**10011100101101001**

**0011100101101001 +1 (if required)**

**0011100101101010 (2 points)**

**+ 0111001010101100 (3rd word)**

**1010110000010110**

**N/A +1 (if required)**

**1010110000010110 (2 points)**

**(1’s complement)**

**0101001111101001 (2 points)**

**3.** **(20 points)** Assume you are in a private network connected to the Internet via a router. Your node’s IP address is 192.168.45.121/28.

Net ID in decimal? **192\_.\_168\_.\_45\_. 112\_\_\_**

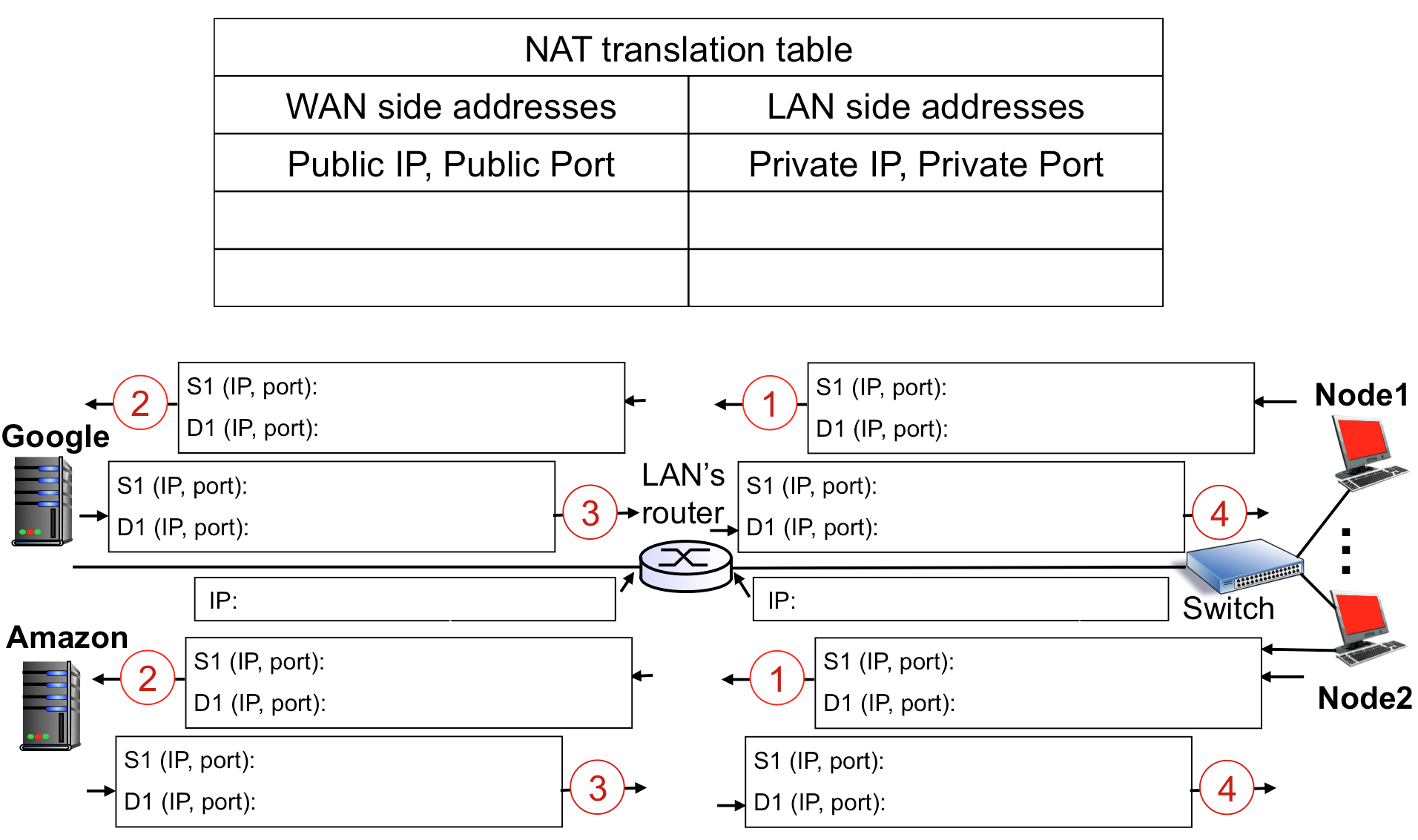
Range of valid IPs? **\_192\_.\_168\_.\_45\_.\_113** to **\_192\_.\_168.\_45\_.\_126\_**

Which IP address is used for the router interface? **\_192\_.\_168\_.\_45\_.\_113\_**

The router has another interface connected to the Internet. That public IP address is 144.37.12.45.

Node1 (192.168.45.121 port 3003) sends an HTTP request to Google server whose IP address is 54.23.130.30 (port 80), and Google server replies. Node2 (192.168.45.125 port 3003) sends an SMTP request to Amazon mail server whose IP address is 145.23.170.92 (port 25), and Amazon mail server replies. Create port numbers at router starting from 6200.

Fill in the blanks in the picture below. Enter the IP addresses and ports for packets’ sources and destinations at each step and also in NAT table. Enter the IP addresses asked in the picture (22 blanks).



**144.37.12.45:6200**

**54.23.130.30:80**

**54.23.130.30:80**

**54.23.130.30:80**

**192.168.45.121:3003**

**192.168.45.113**

**144.37.12.45**

**54.23.130.30:80**

**144.37.12.45:6200**

**192.168.45.125:3003**

**192.168.45.121:3003**

**144.37.12.45:6201**

**144.37.12.45:6200**

**192.168.45.125:3003**

**145.23.170.92:25**

**144.37.12.45:6201**

**145.23.170.92:25**

**145.23.170.92:25**

**192.168.45.125:3003**

**145.23.170.92:25**

**144.37.12.45:6201**

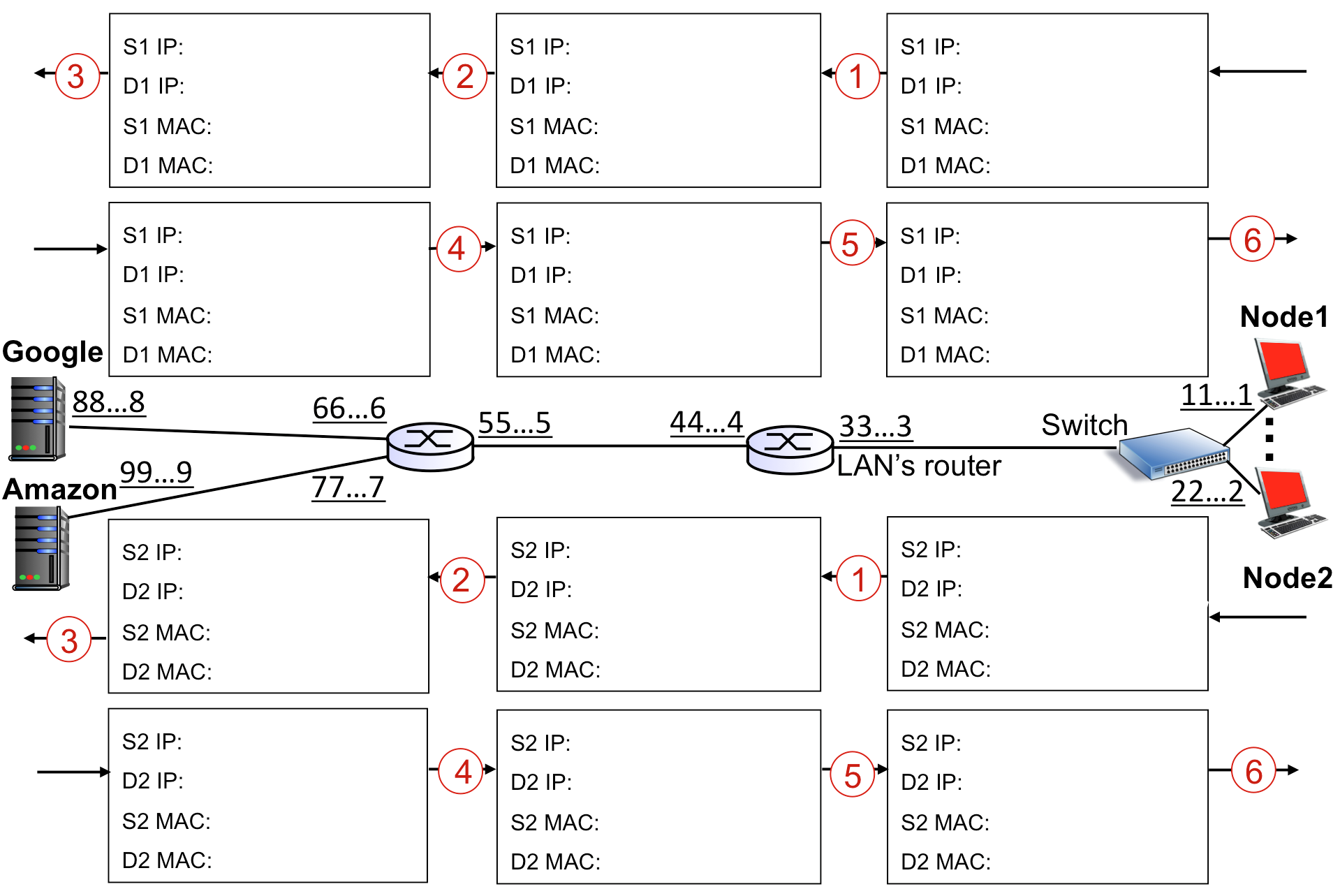
**192.168.45.121:3003**

**4.** **(20 points)** Using the configuration of last question, now assume packets pass two routers to reach destination. The picture below contains your private network, the servers, and the routers in between. Node1 sends a message to Google server. (Pay attention: who is the source, who is the destination?). A frame is created at node1 to send the packet to the router. Then the frame is processed and updated in each router to forward the packet to the next hop. Node2 also sends a message to Amazon mail server. Fill in the blanks in the picture. Specify the source and destination **IP addresses** and **MAC addresses** for frames and their enclosed IP packets. (48 items need to be filled). Assume the ARP tables contain all the required mappings.

The MAC addresses in this network are given in underlined numbers at each interface. For example 11…1 (meaning 11-11-11-11-11-11) is the MAC address of node1. You can use the same notation in your answer.

IP addresses of left router’s interfaces are given in the table below. Note that you may or may not need these IP addresses. The other IP addresses (like Google) are the same as the ones given in the last question.

|  |  |
| --- | --- |
| MAC address | IP address |
| 55…5 | 56.84.62.79 |
| 66…6 | 54.23.130.1 |
| 77…7 | 145.23.170.1 |

****

**22...2**

**33...3**

**192.168.45.125**

**145.23.170.92**

**44…4**

**55…5**

**144.37.12.45**

**145.23.170.92**

**77…7**

**99…9**

**144.37.12.45**

**145.23.170.92**

**33…3**

**22…2**

**145.23.170.92**

**192.168.45.125**

**55…5**

**44…4**

**145.23.170.92**

**144.37.12.45**

**99…9**

**77…7**

**145.23.170.92**

**144.37.12.45**

**11…1**

**33…3**

**192.168.45.121**

**54.23.130.30**

**44…4**

**55…5**

**144.37.12.45**

**54.23.130.30**

**66…6**

**88…8**

**144.37.12.45**

**54.23.130.30**

**33…3**

**11…1**

**54.23.130.30**

**192.168.45.121**

**55…5**

**44…4**

**54.23.130.30**

**144.37.12.45**

**88…8**

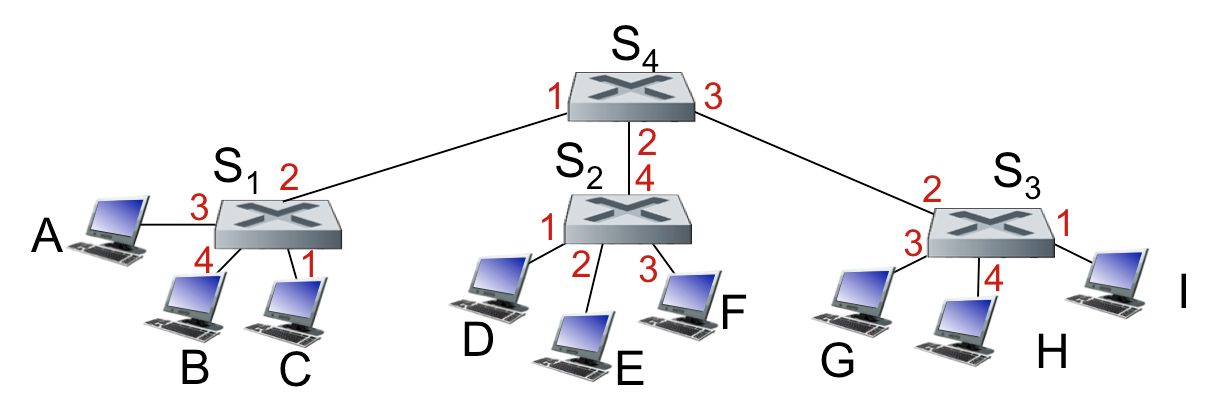
**66…6**

**54.23.130.30**

**144.37.12.45**

**5.** **(20 points)** In the picture below assume node B sends a message to node D, and then node D replies to B. Assume all ARP and Switch tables are empty at the beginning. Fill out the following table to show what will be stored at each switch at each step.

Note: at each step, a switch may learn and store one or no new entry.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Sender 🡪 Receivers | | S1 switch table | | S2 switch table | | S3 switch table | | S4 switch table | |
|  | Sender | Receivers | MAC | Interface | MAC | Interface | MAC | Interface | MAC | Interface |
| 1 | B | S1 | B | 4 |  |  |  |  |  |  |
| 2 | S1 | A, C, S4 |  |  |  |  |  |  | B | 1 |
| 3 | S4 | S2, S3 |  |  | B | 4 | B | 2 |  |  |
| 4 | S2  S3 | D.E.F  G,H,I |  |  |  |  |  |  |  |  |
| 5 | D | S2 |  |  | D | 1 |  |  |  |  |
| 6 | S2 | S4 |  |  |  |  |  |  | D | 2 |
| 7 | S4 | S1 | D | 2 |  |  |  |  |  |  |
| 8 | S1 | B |  |  |  |  |  |  |  |  |