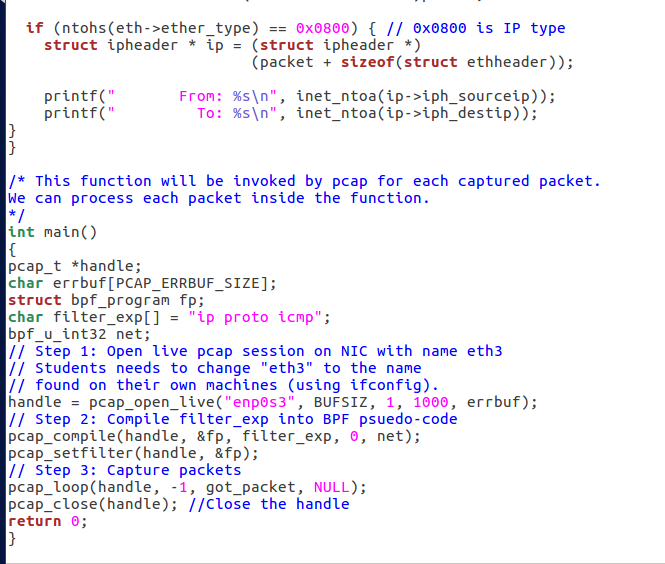
**Lab Task Set 1 – C**

**A2.1**

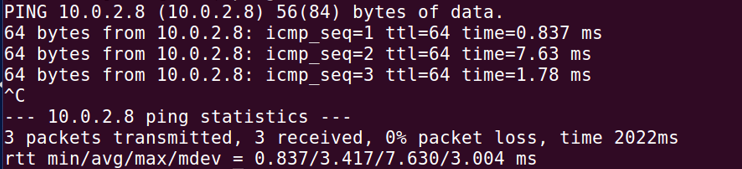
The code we built the sniffer by, and also print for whom the packet is intended+ who sends



Machine 1 sends a ping to the other machine (10.0.2.8)

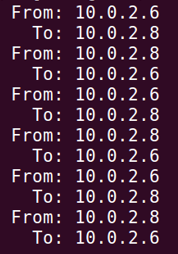
 key capture

more info



run two commands

  
The packets caught in machine 2 by the sniffer



**question 1:**

**Step one,** define with which interface you want to work and through which to capture.  
**Step two**, open a session and create the sniper by function - pcap\_open\_live.  
  
In this function we are determined: on which device we want to run the sniffer and other data such as the maximum size of bites that can be catch, times, catching errors and so on.  
  
**Step three**, if we want to capture certain packets by some filter we have defined then we need to use the function - pcap\_compile.  
  
This function sets the rules of the sniffing and gets the string of the desired filter and "compiles" them into a language that the PCAP library format can read.  
  
Then, run the pcap\_setfilter function because after the compiling it is possible to use the compiled data and therefore this function provides us the possibility to work with them after the compilation.

**Step four**, after the whole process we will want the sniffer to start working.  
The function - pcap\_loop means that it enters a loop where it captures packets until it reaches the size we defined for it and each time it receives a packet it calls the got\_packet function where we define what we want to do with this packet, such as: print the IP sender and IP receiver, Analyze the package and so on.

**Step five**, when we have finished getting all the information we wanted, the pcap\_close function closes the session.

**question 2:**

The program of the sniffer needs to access the information of the operating system (by certain processes) to get information about the network we work on, access it and send packages to it. so operations like these can not be performed without sudo permission (administrator).

The program *will not* perform what is required once we run it for these reasons as follows:  
  
we will run the two commands



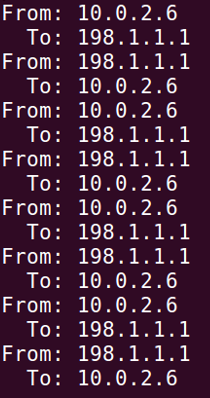
and we'll get



but when we will enter



the output will be as follows:

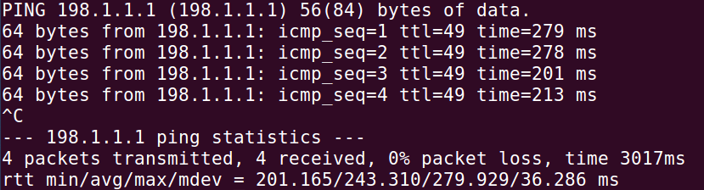


**question 3:**  
A promiscuous state is a state in which packets sent from a particular computer getting and received at another computer's sniffer even though they are *not intended* for that specific computer and a non-promiscuous state receives **only** packets intended for its computer, to its address.

Promiscuous state - sending ping from machine 1 to an arbitrary address  
run the command



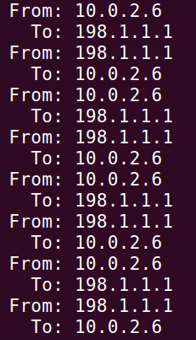
and get



Receiving the packets in machine 2 by the sniffer which is now in a promiscuous state  
  
run the command



and then

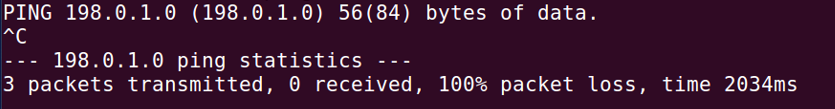


Not promiscuous situation  
  
Sending ping from machine 1 to an arbitrary address. we can see that the packets were lost,  
 - not received.

for example, we will run



and get



The packets in machine 2 were not accepted by the sniffer who is now in an promiscuous state

**B2.1   
First point**: capture ICMP packets between two specific computers.  
the code: (icmp and…)

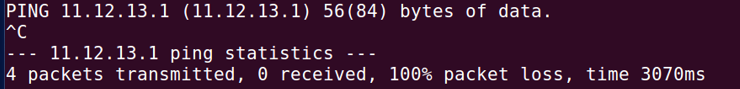


It can be seen that Ping was sent to an arbitrary address and it was not sniffed

Machine 1  
run command



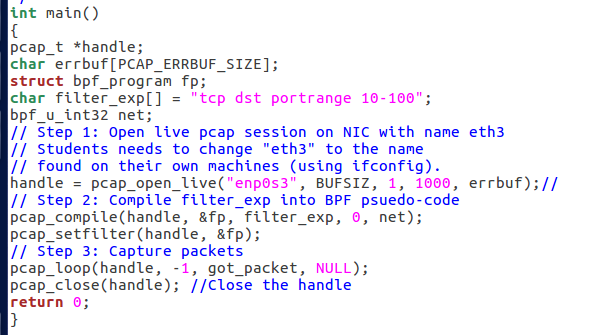
result will be



Machine 2 (with the sniffer)



**second point:** Capturing TCP packets with a target port number in the range of 10-100  
the code: (tcp dst portrange 10-100)



It can be seen that a tcp message was sent to an arbitrary address and to a port that is not in the range of 10 - 100 and therefore this message **will not be received.**

Machine 1  
 the port is not between 10-100

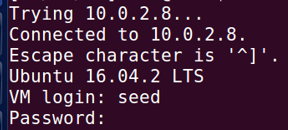
Machine 2 (with the sniffer) - we can see that the packet was not caught

**C2.1**

sniff.c code file  
  
Login for machine number 1 - enter login details, the password is seed.

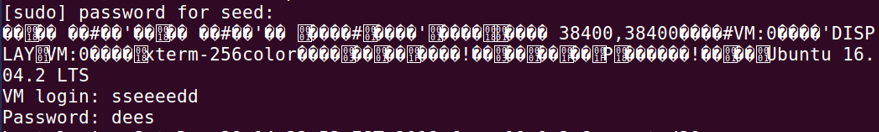
which 'server' address to connect through the particular port of the protocol





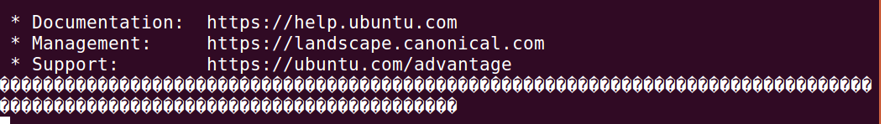
Machine 2 sniffs the details of machine 1  
  
run two commands





we will see machine number 1





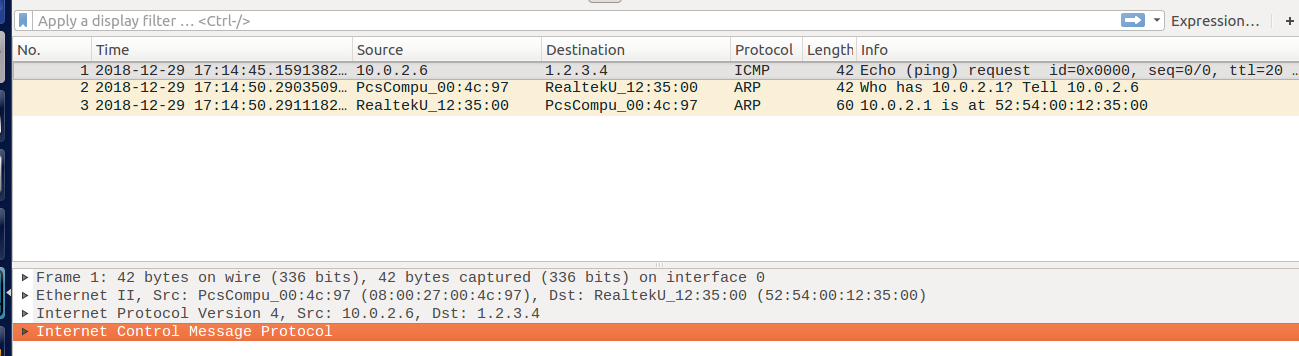
We can see that machine number 2 picks up all the login details and displays the password of machine 1.

The part of the information is at the end of the packet i.e. in the last part where we can see all the additional information that the packet contains.

**A2.2**

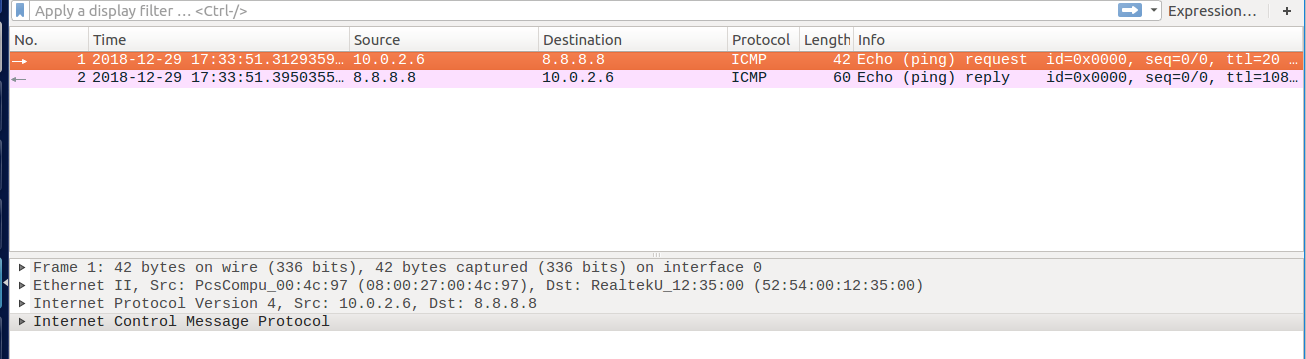
spoof1.c code file

In the picture we can see that the packet that was built, and goes to an arbitrary destination, was sent and captured.



**B2.2**

spoof2.c code file  
In the image we see that an ICMP packet was sent to a certain existing IP address and we get an echo back from that address



קיבלנו reply

**question 4:**

Yes, the IP header length can be set to arbitrary values.

**question 5:**

No, the computer fills it automatically.

**question 6:**

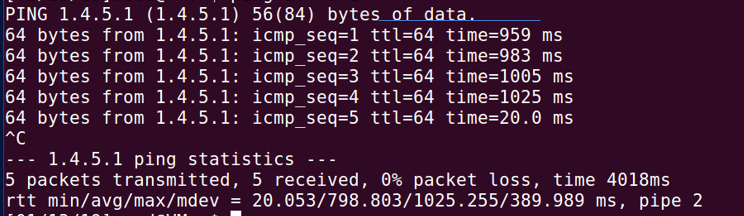
Because without root permission it would have been possible to build customized packets that could harm the computers, **therefore** - decided to give it root permission.

sniff-and-spoof as different code file**2.3**

Here we can see that machine number 1 is trying to send a ping to an arbitrary address  
  
run command



and then



It can be seen that the packets are received, i.e. there is a response to the ping we have sent and that is by machine number 2 which sniffs the ICMP packets coming from machine number 1 and create a fake ICMP response packet to machine number 1 with the answer machine number 1 expects to receive

run two commands:



and then we will see

