Exercise 1 (3 points)

NOTE: This is a theory exercise. You are supposed to answer the questions without actually executing those commands.  
Instead of the commands mentioned in the book, use the following three commands. You don't have to run these commands. Answer only the LAB REPORT questions stated here for this exercise:

* iperf3 -s -p 5555
* iperf3 -c 192.168.1.17 -p 8911 -u
* iperf3 -c 192.168.1.17 -p 8911 -n 10K -l 200

LAB REPORT: Explain the operation of each command.

1. Start Server mode on the port 5555

2. Start Client mode trying to UDP connect 192.168.1.17 at server port 8911

3. Start Client mode trying to connect 192.168.1.17 at server port 8911, the number of client sending to server is 10K and the length of buffer is 200

LAB REPORT: How long is a packet at the application layer for the second command? How many packets will be transmitted or how long will the transmission take, after the execution of the second command?

solution:

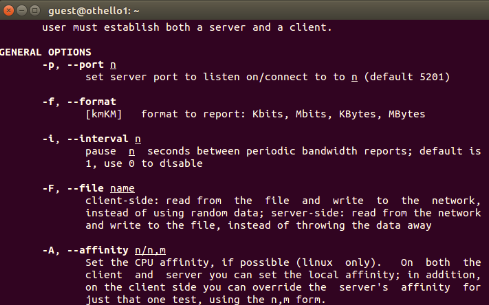
LAB REPORT: For the second and/or the third command, does it make a difference if there is a server running?

solution:

Exercise 2

Study various options associated with the iperf3 program. A brief list of options can be displayed by typing iperf3. More details on iperf3 can be found in the manual.

solution: type man iperf3



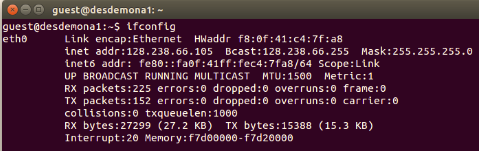
iperf3 can test bandwidth by using different applications. for example, -i + second indicates the interval of testing.  -n + number indicates the length of data send to destination. -u means using UDP.

Exercise 3 (4 points)

Use ifconfig to find out the MTU of the ethernet interface.   
While running   
sudo tcpdump -enx -vv src host <your\_host> and not tcp   
where <your\_host> is the IP address of the iperf3 client, execute the following command with different values for <size>  
iperf3 -c <remote\_host\_running\_server> -u -l <size> -n <size>  
while your <remote\_host> is executing an iperf3 server  
iperf3 -s  
Note that in both appearances of <size> you have to use the same number.  
Increase <size> until fragmentation occurs.   
Note: Be careful that iperf3 sends 2 UDP packets, one from each side, for application layer usages at the beginning of the transmission. Ignore those 2 packets.   
  
LAB REPORT: What is the maximum value of <size> for which the UDP datagram can be sent without IP fragmentation? Justify your answer with the ifconfig output.

solution:

From the ifconfig: The MTU of IP datagram is 1500 bytes.



the maximum value of <size> is 1472 using which there is no IP fragmentation.

Justify: from the ifconfig, the MTU of IP datagram is 1500 including header of IP datagram which is 20 bytes and 1480 bytes IP data. Then in IP data, which encapsulate UDP, and UDP header is 8 bytes.

The value of <size> indicates the number of bytes in the UDP data part. so its maximum value should be 1480-8=1472 bytes.

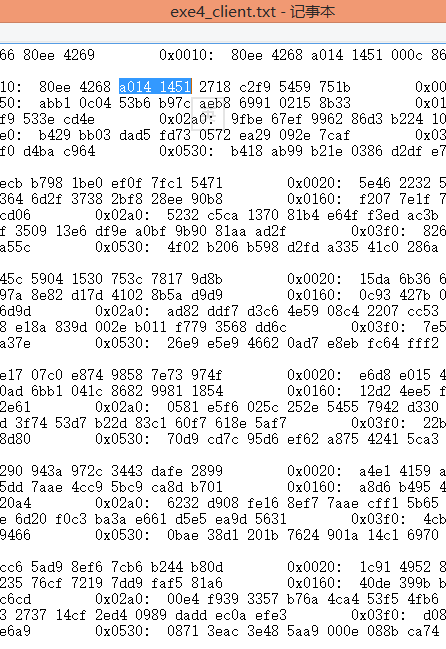
And when <size> equals to 1473, then fragmentation happens.

Exercise 4 (4 points)

NOTE: Please be careful that in this exercise we will look at the tcpdump output directly. Do not use wireshark to answer this question.wireshark has as a goal to make the data more presentable, therefore sometimes what you see there may not be 100% accurate. If you want to save the output of your tcpdump output, you can use the redirect (>) operator.   
  
Capture the data packets generated by the following command   
iperf3 -c <remote\_host\_running\_server> -u -l 10000 -n 10000   
using   
sudo tcpdump -enx -vv src host <your\_host> and not tcp > exe4.txt   
while your <remote\_host> is executing an iperf3 server   
iperf3 -s   
Save the tcpdump output for the lab report.   
  
LAB REPORT: Explain the tcpdump output in terms of the IP header fields that are used in fragmentation. When IP fragmentation occurs, only the first fragment has the UDP header. How do you verify this fact from the tcpdump output?

solution:

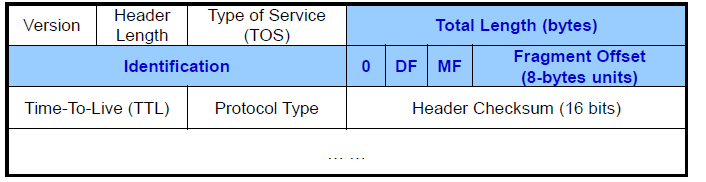
From the txt file, I find that the source port number is 40980(16#a014) and the destination port number is 5201(16#1451), so I search in the txt file to find corresponding bytes, find that only in the first IP datagram can show the bytes. So the UDP header could only shows in the first IP datagram when IP cut the whole UDP datagram into pieces.



Exercise 5

While running   
sudo tcpdump -enx -vv src host <your\_host> and not tcp   
execute the following command with different values for <size>   
iperf3 -c <remote\_host\_running\_server> -u -l <size> -n <size>   
while your <remote\_host> is executing an iperf3 server   
iperf3 -s   
in order to find out the maximum size of a UDP datagram that the system can send or receive, even when fragmentation is allowed.   
  
LAB REPORT: What is the maximum size of user data in a UDP datagram that the system can send or receive, even when fragmentation is allowed? Why?

solution: The limitation is on the total length field in  IP header, that the length of that field is 2 bytes. which means in this field, the maximum value is ffff(65535). However the actual value is smaller than 65535, considering the occupation of headers of IP and UDP, the eventual number could be 65535-20-8=65507 bytes. Because our connection frequently disconnected with each other, we didn't get the output window.



Exercise 7

In order to compare the transfer rates of FTP and TFTP, we will retrieve a large file from a remote server using FTP and TFTP, respectively. First run the following tcpdump command: tcpdump host your\_host and remote\_host -w exe7\_ftp.out. Then, get the /home/LAB/large.dum file from remote\_host using ftp. Also, from the ftp window, record the transfer rate displayed. Restart the above tcpdump command, with the last argument changed to exe7\_tftp.out. Now use tftp to get the /tftpboot/large.dum file.

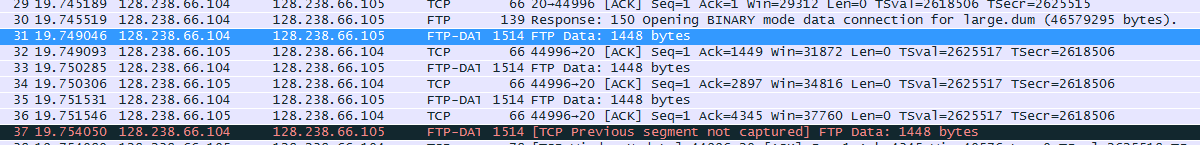
Lab Report: Examning the saved tcpdump output file. Identify the starting and ending time of actual data transfer. Don't include the time spent establishing the TCP connection. Calculate the time spent for data transfer. Compare the time with the value displayed in ftp window. Are they consistent? If there exists any significant difference, what might be the reason? Now, from the saved tftp file, carefully determine the starting and ending time of data transfer for the tftp program. Compare the time with the value displayed in tftp window. Are they consistent? If there exists any significant difference， what might be the reason? By comparing the actual data transfer times of ftp and tftp, which of these two is faster, and why?

solution:

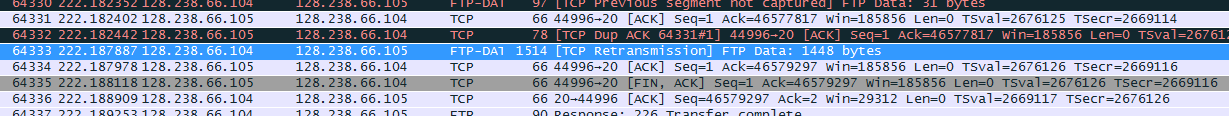
FTP

the FTP uses TCP as transport protocol. the starting and ending time are shown below:

starting time



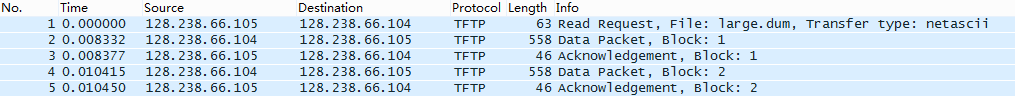
ending time



so, the time spending on data transferring is 203, which is not consistent with the ftp window. the reason might be the control messages also take times. Because of TCP connection by FTP, the establishment and finishing of connection via control connection as well as the both ends coordinate parameters for TCP connection may cost time.

TFTP

starting time:



ending time:

C:\Users\ZHANGJ~1\AppData\Local\Temp\enhtmlclip\ScreenClip(176).png

However, for TFTP, the time that shown on the tftp window is consistent with the time calculated in the wireshark. the reason is that TFTP uses UDP which is connectionless and unreliable. So when the TFTP server port 69 receives the TFTP file request, if file exists on server, then server will send data to the client directly by UDP, no other steps like coordinating windows size or sequence number. the control message is near 0 in UDP, So the data transferring time is equal to the total time.

Compare the time costing between FTP and TFTP, FTP is faster. the reason might be the different tranfer protocol they use. For TFTP, it uses wait for acknowledge strategy, which means that after every data transfer from server to client, the server wait for acknowledgement datagram from client, besides the length of TFTP data is given by TFTP application the number of which is from 0 to 512 bytes. While for FTP, it uses TCP protocol, which uses sliding window to decide how much data sent each time rather than given by application, FTP only has role putting bytes data into buffer, but as for sending data length which is decided by TCP. Besides, FTP client does not have to wait till the correct datagram receiving if corrupt or missing of datagram happens.

Exercese 8

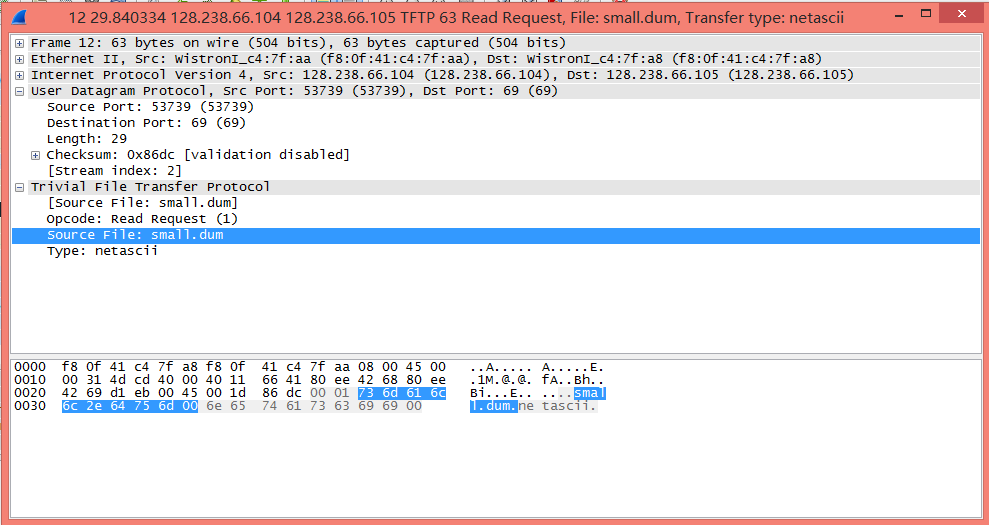
Capture the packets that are exchanged during a tftp session for the /tftpboot/small.dum file between your host and a remote\_host. Oberserve the protocol in action. Analyze various types of TFTP messages used by examining the content of .out file.

Lab Report: List all the different types of packets exchanged during the tftp session. Compare them with TFTP message format. Why does the server's port number change?

solution:

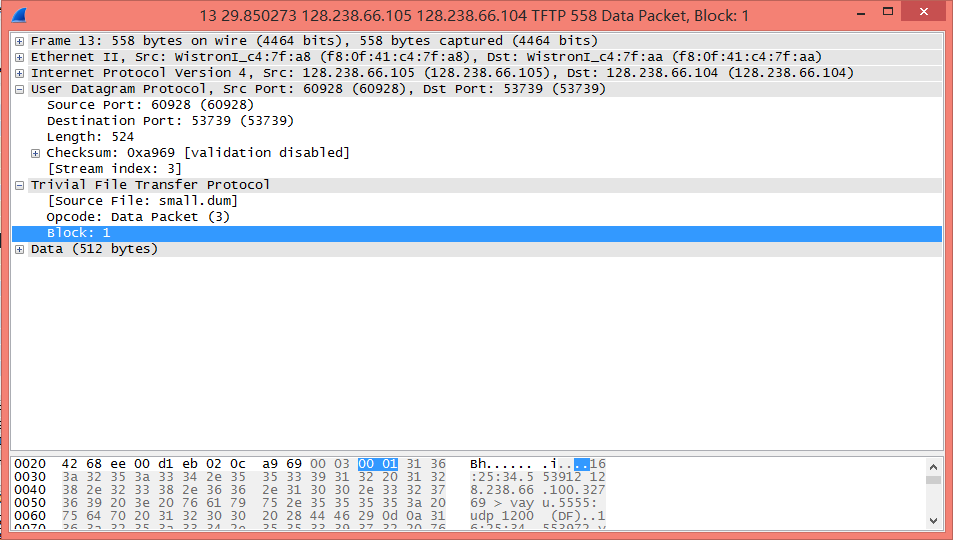
there are three different types in TFTP

TFTP read request



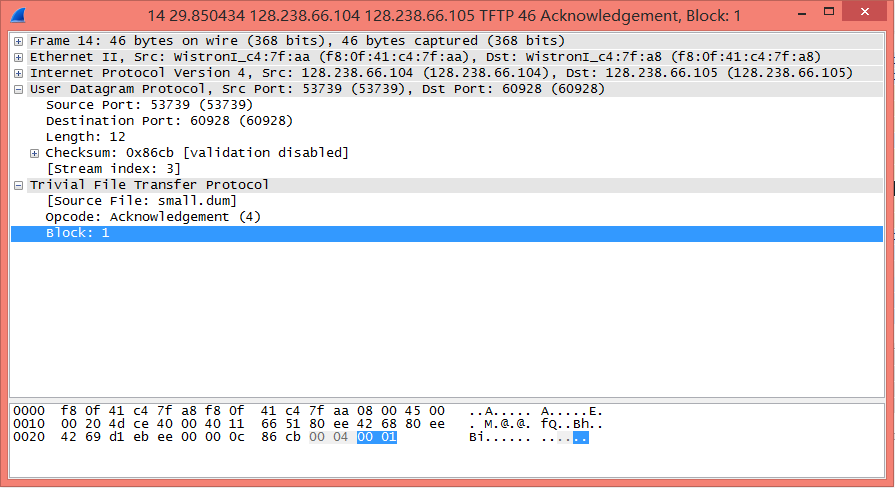
opcode = 0x1

TFTP data



opcode = 0x3

TFTP data Acknowledgement



opcode = 0x4

the port 69 on server functions as a listening port, the role of which is receiving data transfer request. If the port 69 take action both listening and transferring data, so no other port can listen request on the server, which is inefficient.

In terms of TFTP connection,

Lab Report: In most cases, tftp service is restricted. Why is tftp service not generally available to users?

solution: checking the TFTP connection establishing process, the one who connect to server port 69 could request for any file in the server. So this is very unsafe for server that has no way to protect data.

Lab Report: In Exercise 5, we found the maximum size of a UDP datagram in your machine. With tftp, which uses UDP, we transferred a file larger than the maximum UDP datagram size. How do you explain this?

solution:

in TFTP, the UDP datagram has two fields, one is data field which is 512 bytes, the other is control field which has opcode, block number, file number(different types of the TFTP has different control field). So the TFTP is actually has its sequence number field. Although one file in total, it actually was transferred by many UDP datagram. While in Exercise5, the maximum length of UDP is basically one UDP datagram and was cut in the network layer by IP protocol.

Exercise 9

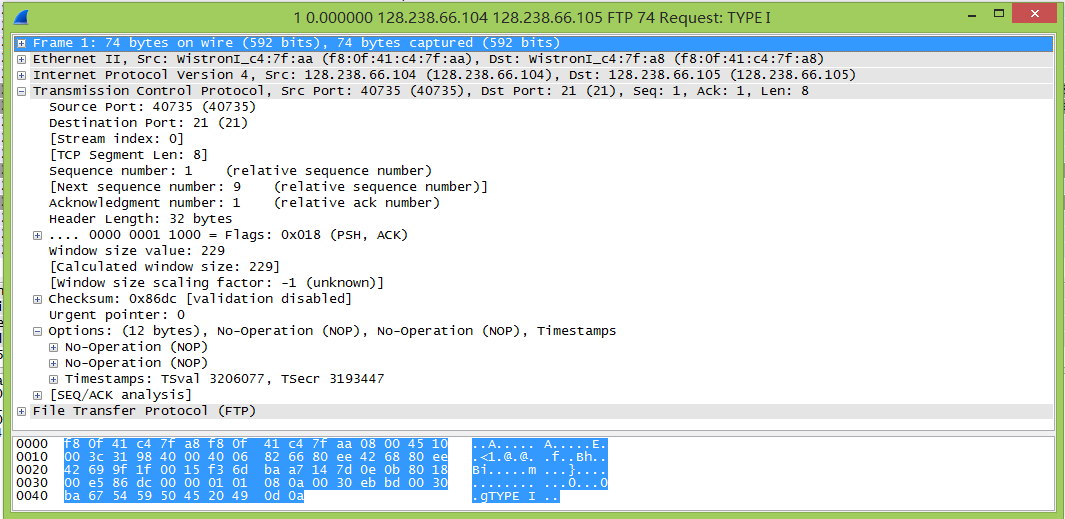
Repeat the above experiment, but use ftp and change the output file name. Capture a trace of the packets exchanged when downloading the /home/LAB/small.dum file using ftp.

Lab Report: How many well-known port numbers were used? which machine used the well-known port numbers? What were the other machine's port numbers?

solution:

2 well-knownnumbers were used which is 21 for control connection and 20 for data connection, both of which are on server.

the port number on client is 40735 and 36219



Lab Report: As can be seen from the tcpdump output, FTP involves two different connections, ftp-control and ftp-data. Why are two different connections used. instead of one connection?

solution:

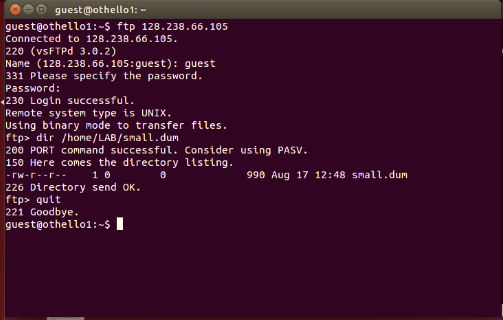
there are some benificial for two different connections in FTP. the control connection can listen to the channel if other clients want to transfer files when server already has connection to one client. If only one port used,  the distinguishment between control message and data message might be much more difficult and complicated, that the data message cannot used the same field as some special control message values. Besides it is a kind of protection, that if client suddently ceases the transport windows, the control connection between client and server might normally close TCP connection via control connection.

Run ftp in the debug mode using: ftp -d remote\_host.

After logging into the remote host, type dir /home/LAB/small.dum in the ftp window. Then type quit to terminate the ftp session, and save the ftp window output.

Lab Report: Submit what you saved in this exercise, explaining each line of the output. Explain how the PORT command works. Which connection, the control connection or the data connection, did the server send the reponse(the LIST output) on?

solution:



ftp debug mode can display all ftp commands pass between client and server.

Thus,