**CS 6043: Computer Networking**

**SPRING 2016**

**PROJECT 2**

**Given: Feb. 22, 2016**

**Due: March 11 (Friday), 2016 (NO LATER THAN 11:59PM)**

**Sneha Jinturkar M08698220**

**Submission Instructions:**

1. Submit only on-line files on Blackboard before midnight. No hard copy will be accepted.

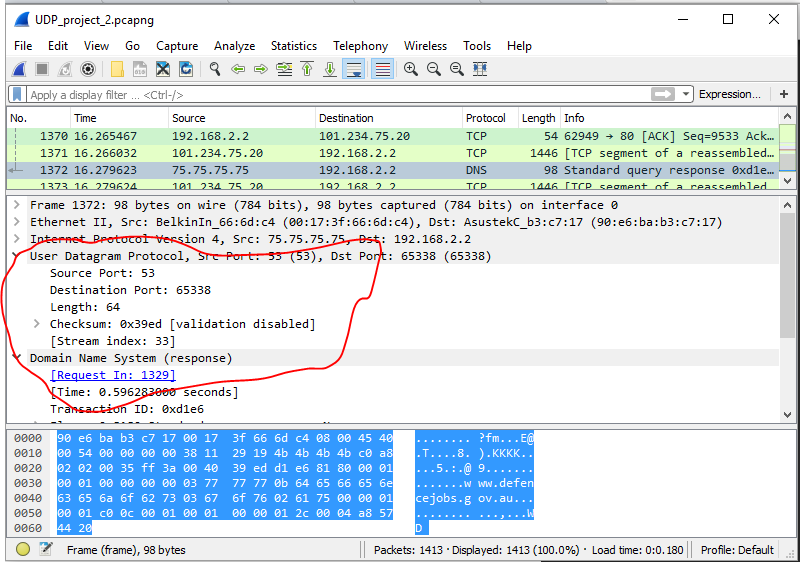
2. For students who are working in a team, *one* submission for the team is sufficient.

3. Wireshark files for this project can be found in the zip file “Project\_2\_Wireshark\_Traces.zip”.

**Part I: UDP**

Load the file ‘UDP\_project\_2.pcapng’ in Wireshark and answer the following questions.

1. Find a UDP packet in the trace file and determine the length (in bytes) of each of the UDP header fields. Provide screenshot with necessary annotations.



The UDP Header has four fields: source port, destination port, length and checksum.

Length: 64.

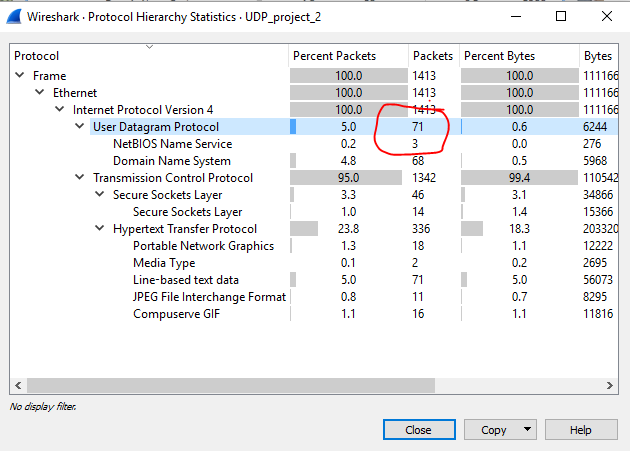
Data Length: 33.

UDP Header length = Packet Length - Data Length = 64 - 33 = 31 Bytes.

Length of Each UDP Header = 31 /4 = 8 Bytes.

1. Using statistics feature of Wireshark, determine the percentage of UDP packets in the capture.

Answer:



Number of packets = 71

Percentage of UDP packets among the others is 5%

1. What are the source port and length of the first UDP packet in the trace file? What is the maximum possible length (excluding headers) of a UDP segment? Explain.

Answer: source port: 54922 and length: 42.

since each udp segment can have the length = 0xffff (16 bits). = 65535 bytes.

max possible length = 0xffff - (sizeof(ip header) + sizeof(udp header))

length of ip header = 20 bytes.

length of udp header = 8 bytes.

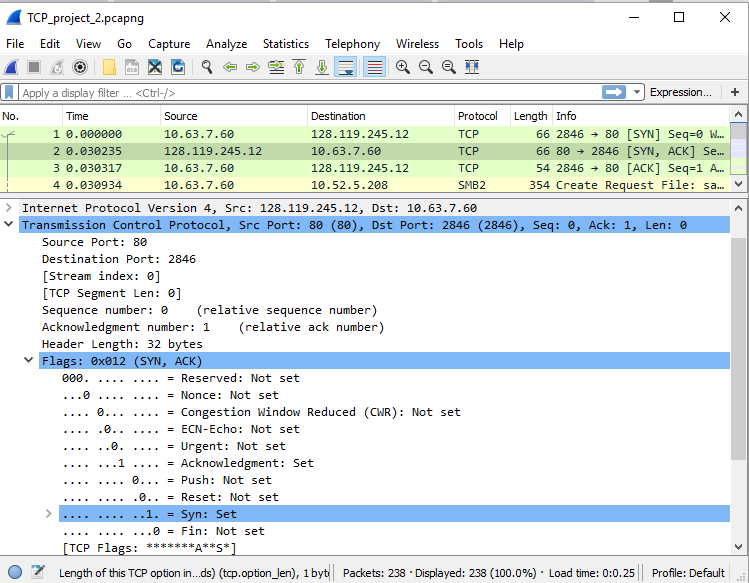
therefore,

max possible length of udp message = 65535 - (20 + 8) = 65507.

**Part II: TCP**

Load the file ‘TCP\_project\_2.pcapng’ in Wireshark and answer the following questions. The trace file was captured while transferring ‘alice.txt’ file from a computer under UC network to gaia.cs.umass.edu web server using the HTTP POST method.

1. What is the sequence number and the value of the acknowledgement field of the first ACK segment sent from client computer in reply to the SYNACK from server? How were the values determined?

Answer: 

Acknowledgement field is set to 1

1. What are the minimum and maximum amount of available buffer spaces advertised at the receiver for the entire trace?

Answer: The buffer size values advertised by the receiver for the entire trace:

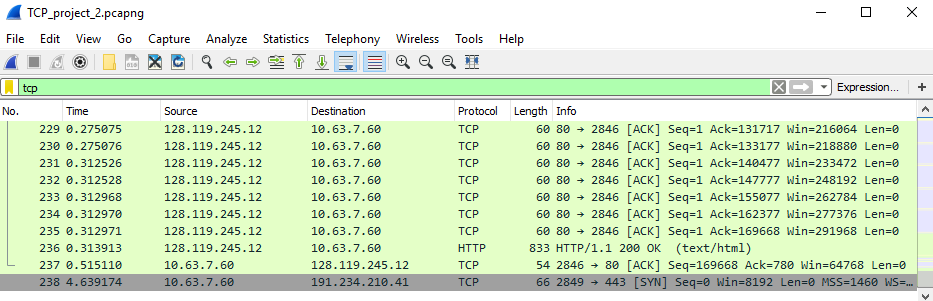
Minimum = 29200 (Packet 2)

Maximum = 291968 (Packet 235)

1. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

Answer: The average throughput can be calculated as the ratio between the total amount data and the total transmission time.

Total Data Transferred = difference between the sequence number of the first TCP segment (node 3) and the acknowledged sequence number of the last ACK (node 237) = 169668 - 1 = 169885 bytes.



Time = Time between the first packet and last packet = 0.5151 - 0.030317 = 0.484783s.

Therefore, the average throughput = 169667 / 0.484783 = 349985.457411 bytes/sec = 349.985 KByte/sec.

**Part III: Socket Programming**

1. You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the server’s file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP “404 Not Found” message back to the client.

Below you will find the skeleton code for the Web server. You are to complete the skeleton code. The places where you need to fill in code are marked with #Fill in start and #Fill in end. Each place may require one or more lines of code.

Put an HTML file (e.g., HelloWorld.html) in the same directory that the server is in. Run the server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26). From another host, open a browser and provide the corresponding URL. For example:

http://128.238.251.26:6789/HelloWorld.html

‘HelloWorld.html’ is the name of the file you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number 6789. The browser should then display the contents of HelloWorld.html. If you omit ":6789", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80.

Then try to get a file that is not present at the server. You should get a “404 Not Found” message.

*You will hand in the complete server code along with the screen shots of your client browser, verifying that you actually receive the contents of the HTML file from the server.*

Skeleton Python Code for the Web Server:

# Import everything from socket

from socket import \*

# Define the server port to desired port

serverPort=9876

# Start main function

def main():

# Alert user that server is up

print 'the web server is up on port:',serverPort

while True:

#Establish the connection

serverSocket = socket(AF\_INET, SOCK\_STREAM)

#Prepare a sever socket

serverSocket.bind(('',serverPort))

serverSocket.listen(1)

print 'Ready to serve'

connectionSocket, addr = serverSocket.accept()

# Attempt to do following

try:

#Recive message on the set up socket

message = connectionSocket.recv(1024)

#Output the message

print message,'::', message.split()[0],':',message.split()[1]

# Set desired file as requested file

filename = message.split()[1]

print filename,'||',filename[1:]

# Open the file

f = open(filename[1:])

# Read the file

outputdata = f.read()

# Print the file

print outputdata

#Send one HTTP header line into socket

connectionSocket.send('\nHTTP/1.1 200 OK\n\n')

connectionSocket.send(outputdata)

# Close connection

connectionSocket.close()

# Do this if an error occurs

except IOError:

connectionSocket.send('Error 404: File not found')

# Close connection

connectionSocket.close()

pass

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

