Northeastern University

CS 5700 Fundamentals of Networking

Project#2 Assigned: 5/20/20 Due: 6/06/20 [100 points]

NAME: PRANJAL NIMSE
NUID: 001081476

Overview: The goal of this assignment is to explore coding aspects of TCP using **Bonaventure book** TCP coding exercises; you can access text this book here – see Chapter 3 on TCP:
https://resources.saylor.org/wwwresources/archived/site/wp-content/uploads/2012/02/Computer-Networking-Principles-Bonaventure-1-30-31-OTC1.pdf

All traces and code resources are available here
https://github.com/obonaventure/cnp3/tree/master/book/transport/exercises

I. Implement a small transport protocol [50 points]

On page 119 of **Bonaventure book**, answer (implement) Question #4: *It is now time to implement a small transport protocol.*

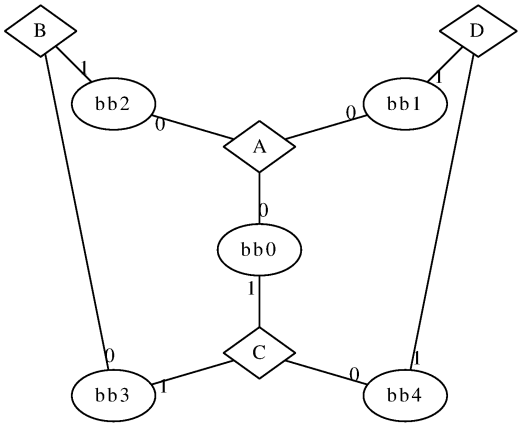
Solution: Find the attached files in folder "Question 1" for implementation of my UDP protocol. Attached screenshots for output too.

```
C:\Users\nimse\Desktop\CN\Project 2 code>python Sender.py localhost 0 4 in_file
Total segments formed:  1
Sending segment 0
Start timer
Waiting
Timeout
Sending segment 0
Start timer
Waiting
Received ACK number  0
Sliding the window
Start timer
Waiting
Timeout
Start timer
Waiting
Timeout
Start timer
Waiting
Timeout
Start timer
Waiting
```

```
C:\Users\nimse\Desktop\CN\Project 2 code>python Receiver.py 8080 4 out_file
Received segment:  0
Received expected segment
Sending ACK 0
```

II. Page 122 of **Bonaventure book - Emulating a network with netkit** [20 points]

Solution: The netkit labs are provided with necessary configurations to setup and run the network using lstart command.



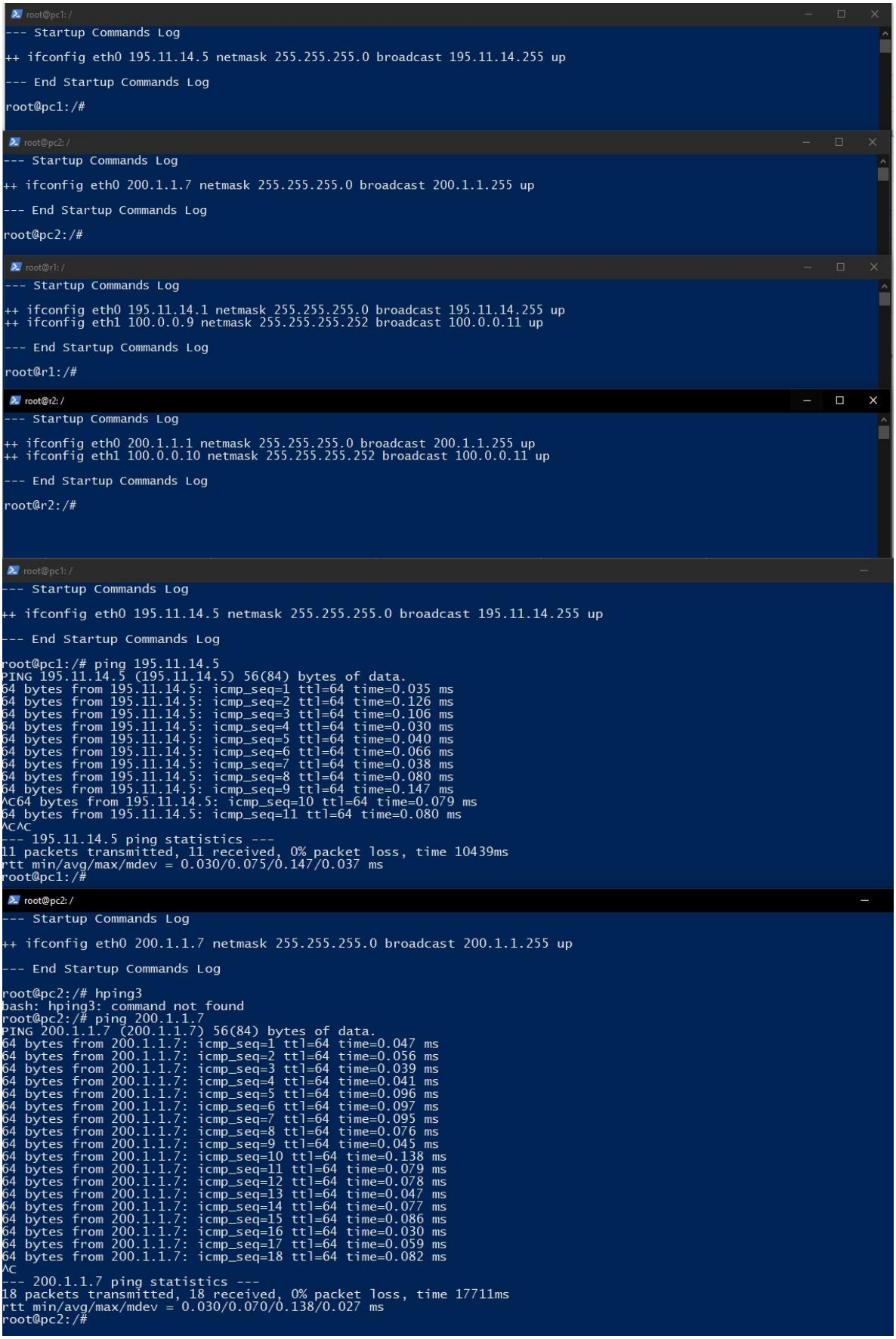
Lab notebook:

Faced difficulty in setting up the netkit since it is no longer available. Another alternative kathara wasn't running smoothly.

Still need better practice to implement this lab with excellence. Also, resources needed to complete the lab successfully were limited and not up to the mark.

Above image is a reference from online netkit resource. The lab would look so after running the lstart command.

Kathara output:


root@pc1: /
--- Startup Commands Log
++ ifconfig eth0 195.11.14.5 netmask 255.255.255.0 broadcast 195.11.14.255 up
--- End Startup Commands Log
root@pc1:/#
root@pc2: /
--- Startup Commands Log
++ ifconfig eth0 200.1.1.7 netmask 255.255.255.0 broadcast 200.1.1.255 up
--- End Startup Commands Log
root@pc2:/#
root@r1: /
--- Startup Commands Log
++ ifconfig eth0 195.11.14.1 netmask 255.255.255.0 broadcast 195.11.14.255 up
++ ifconfig eth1 100.0.0.9 netmask 255.255.255.252 broadcast 100.0.0.11 up
--- End Startup Commands Log
root@r1:/#
root@r2: /
--- Startup Commands Log
++ ifconfig eth0 200.1.1.1 netmask 255.255.255.0 broadcast 200.1.1.255 up
++ ifconfig eth1 100.0.0.10 netmask 255.255.255.252 broadcast 100.0.0.11 up
--- End Startup Commands Log
root@r2:/#
root@pc1: /
--- Startup Commands Log
++ ifconfig eth0 195.11.14.5 netmask 255.255.255.0 broadcast 195.11.14.255 up
--- End Startup Commands Log
root@pc1:/# ping 195.11.14.5
PING 195.11.14.5 (195.11.14.5) 56(84) bytes of data.
64 bytes from 195.11.14.5: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 195.11.14.5: icmp_seq=2 ttl=64 time=0.126 ms
64 bytes from 195.11.14.5: icmp_seq=3 ttl=64 time=0.106 ms
64 bytes from 195.11.14.5: icmp_seq=4 ttl=64 time=0.030 ms
64 bytes from 195.11.14.5: icmp_seq=5 ttl=64 time=0.040 ms
64 bytes from 195.11.14.5: icmp_seq=6 ttl=64 time=0.066 ms
64 bytes from 195.11.14.5: icmp_seq=7 ttl=64 time=0.038 ms
64 bytes from 195.11.14.5: icmp_seq=8 ttl=64 time=0.080 ms
64 bytes from 195.11.14.5: icmp_seq=9 ttl=64 time=0.147 ms
^C64 bytes from 195.11.14.5: icmp_seq=10 ttl=64 time=0.079 ms
64 bytes from 195.11.14.5: icmp_seq=11 ttl=64 time=0.080 ms
^C^C
--- 195.11.14.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10439ms
rtt min/avg/max/mdev = 0.030/0.075/0.147/0.037 ms
root@pc1:/#
root@pc2: /
--- Startup Commands Log
++ ifconfig eth0 200.1.1.7 netmask 255.255.255.0 broadcast 200.1.1.255 up
--- End Startup Commands Log
root@pc2:/# hping3
bash: hping3: command not found
root@pc2:/# ping 200.1.1.7
PING 200.1.1.7 (200.1.1.7) 56(84) bytes of data.
64 bytes from 200.1.1.7: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 200.1.1.7: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 200.1.1.7: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 200.1.1.7: icmp_seq=4 ttl=64 time=0.041 ms
64 bytes from 200.1.1.7: icmp_seq=5 ttl=64 time=0.096 ms
64 bytes from 200.1.1.7: icmp_seq=6 ttl=64 time=0.097 ms
64 bytes from 200.1.1.7: icmp_seq=7 ttl=64 time=0.095 ms
64 bytes from 200.1.1.7: icmp_seq=8 ttl=64 time=0.076 ms
64 bytes from 200.1.1.7: icmp_seq=9 ttl=64 time=0.045 ms
64 bytes from 200.1.1.7: icmp_seq=10 ttl=64 time=0.138 ms
64 bytes from 200.1.1.7: icmp_seq=11 ttl=64 time=0.079 ms
64 bytes from 200.1.1.7: icmp_seq=12 ttl=64 time=0.078 ms
64 bytes from 200.1.1.7: icmp_seq=13 ttl=64 time=0.047 ms
64 bytes from 200.1.1.7: icmp_seq=14 ttl=64 time=0.077 ms
64 bytes from 200.1.1.7: icmp_seq=15 ttl=64 time=0.086 ms
64 bytes from 200.1.1.7: icmp_seq=16 ttl=64 time=0.030 ms
64 bytes from 200.1.1.7: icmp_seq=17 ttl=64 time=0.059 ms
64 bytes from 200.1.1.7: icmp_seq=18 ttl=64 time=0.082 ms
^C
--- 200.1.1.7 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17711ms
rtt min/avg/max/mdev = 0.030/0.070/0.138/0.027 ms
root@pc2:/#

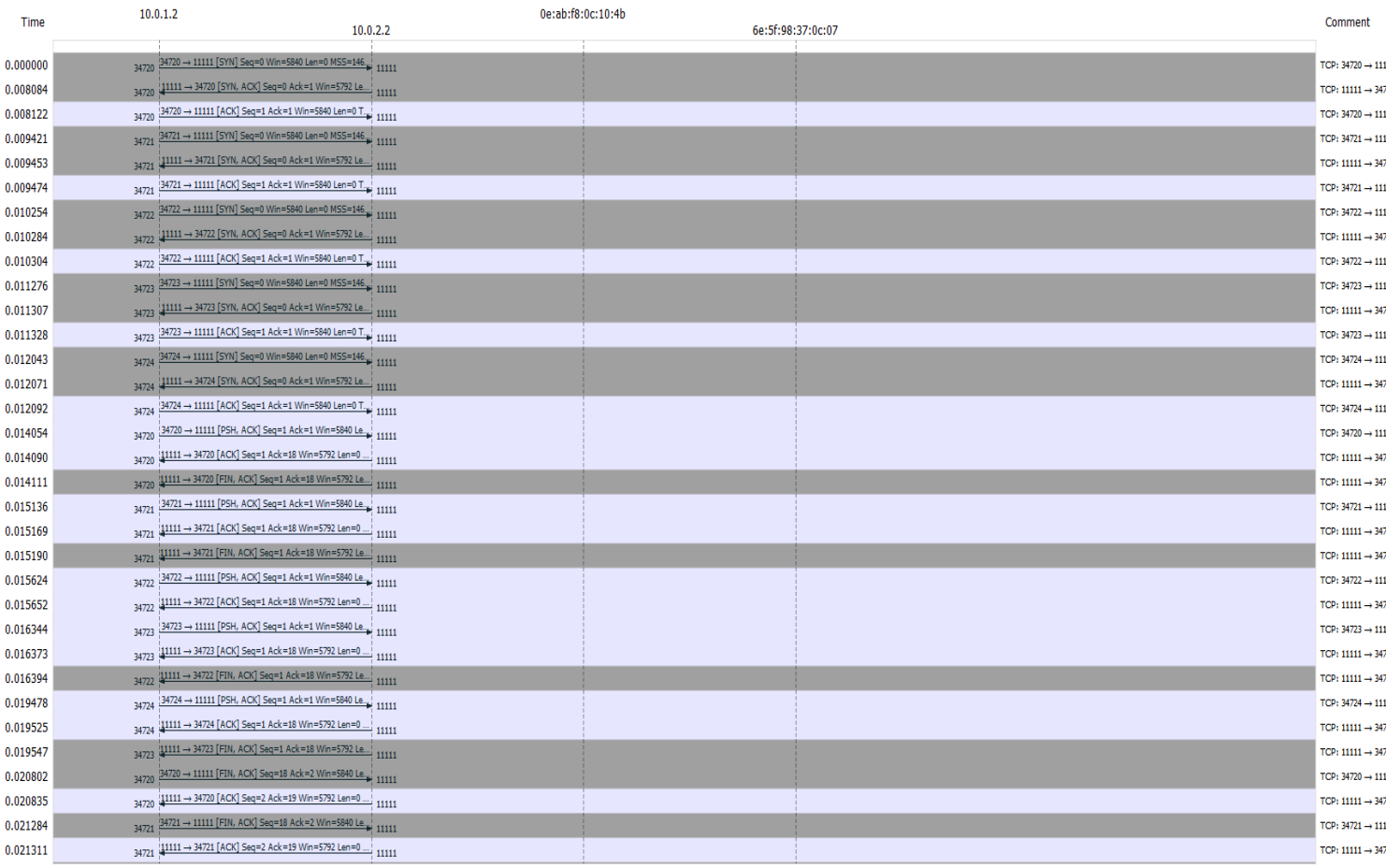
Hping3(8):

- It is a command line tool that allows to send IP packets and TCP segments.
- Handles fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols.

III. On page 123 of **Bonaventure book**, answer the below questions: [18 points]

Question 3

Solution: Flow graph of the trace.5connections_opening_closing:



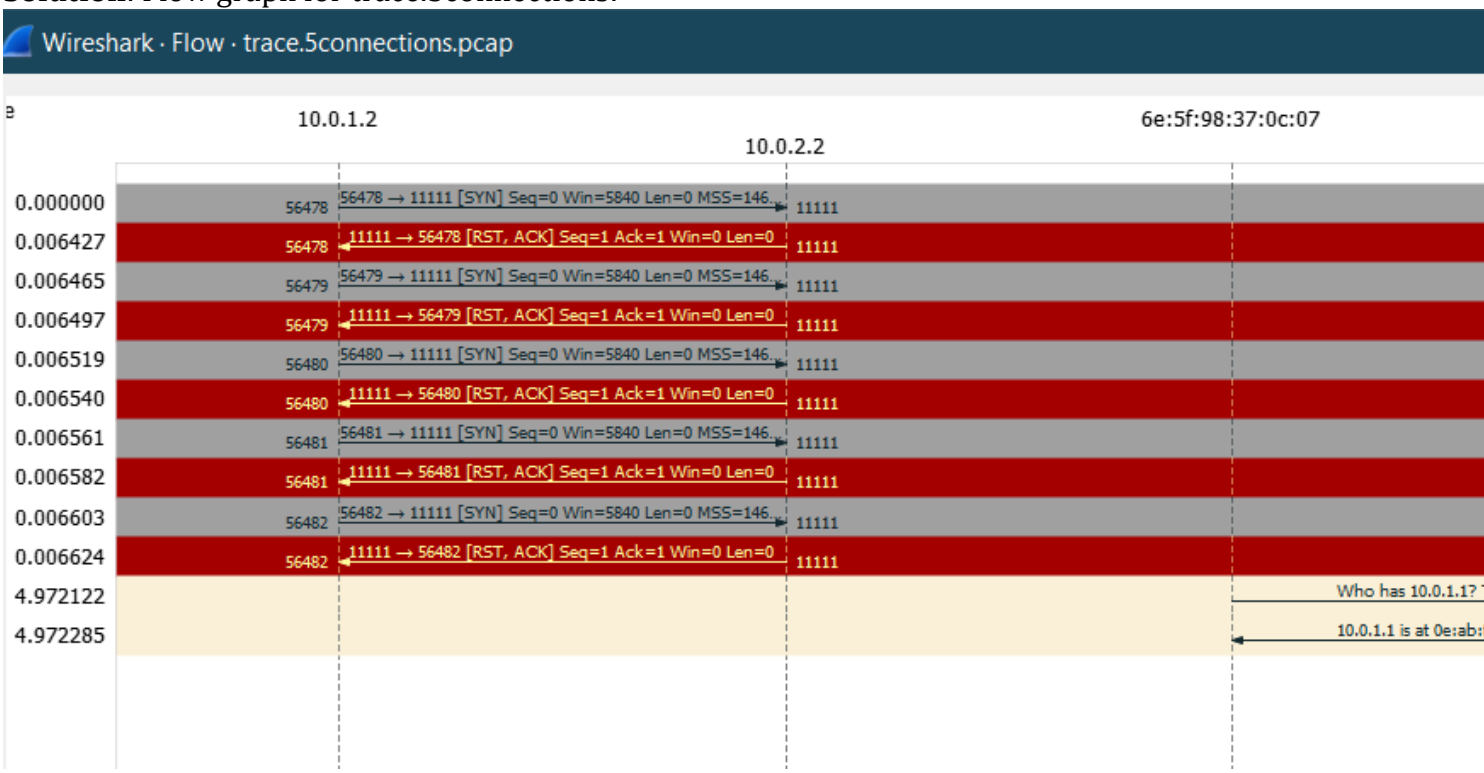
Time	10.0.1.2		10.0.2.2	0e:ab:f8:0c:10:4b / 6e:5f:98:37:0c:07	Comment
0.000000	34720	34720 → 11111 [SYN] Seq=0 Win=5840 Len=0 MSS=146…	11111		TCP: 34720 → 111
0.008084	34720	11111 → 34720 [SYN, ACK] Seq=0 Ack=1 Win=5792 Le…	11111		TCP: 11111 → 347
0.008122	34720	34720 → 11111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 T…	11111		TCP: 34720 → 111
0.009421	34721	34721 → 11111 [SYN] Seq=0 Win=5840 Len=0 MSS=146…	11111		TCP: 34721 → 111
0.009453	34721	11111 → 34721 [SYN, ACK] Seq=0 Ack=1 Win=5792 Le…	11111		TCP: 11111 → 347
0.009474	34721	34721 → 11111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 T…	11111		TCP: 34721 → 111
0.010254	34722	34722 → 11111 [SYN] Seq=0 Win=5840 Len=0 MSS=146…	11111		TCP: 34722 → 111
0.010284	34722	11111 → 34722 [SYN, ACK] Seq=0 Ack=1 Win=5792 Le…	11111		TCP: 11111 → 347
0.010304	34722	34722 → 11111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 T…	11111		TCP: 34722 → 111
0.011276	34723	34723 → 11111 [SYN] Seq=0 Win=5840 Len=0 MSS=146…	11111		TCP: 34723 → 111
0.011307	34723	11111 → 34723 [SYN, ACK] Seq=0 Ack=1 Win=5792 Le…	11111		TCP: 11111 → 347
0.011328	34723	34723 → 11111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 T…	11111		TCP: 34723 → 111
0.012043	34724	34724 → 11111 [SYN] Seq=0 Win=5840 Len=0 MSS=146…	11111		TCP: 34724 → 111
0.012071	34724	11111 → 34724 [SYN, ACK] Seq=0 Ack=1 Win=5792 Le…	11111		TCP: 11111 → 347
0.012092	34724	34724 → 11111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 T…	11111		TCP: 34724 → 111
0.014054	34720	34720 → 11111 [PSH, ACK] Seq=1 Ack=1 Win=5840 Le…	11111		TCP: 34720 → 111
0.014090	34720	11111 → 34720 [ACK] Seq=1 Ack=18 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.014111	34720	11111 → 34720 [FIN, ACK] Seq=1 Ack=18 Win=5792 Le…	11111		TCP: 11111 → 347
0.015136	34721	34721 → 11111 [PSH, ACK] Seq=1 Ack=1 Win=5840 Le…	11111		TCP: 34721 → 111
0.015169	34721	11111 → 34721 [ACK] Seq=1 Ack=18 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.015190	34721	11111 → 34721 [FIN, ACK] Seq=1 Ack=18 Win=5792 Le…	11111		TCP: 11111 → 347
0.015624	34722	34722 → 11111 [PSH, ACK] Seq=1 Ack=1 Win=5840 Le…	11111		TCP: 34722 → 111
0.015652	34722	11111 → 34722 [ACK] Seq=1 Ack=18 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.016344	34723	34723 → 11111 [PSH, ACK] Seq=1 Ack=1 Win=5840 Le…	11111		TCP: 34723 → 111
0.016373	34723	11111 → 34723 [ACK] Seq=1 Ack=18 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.016394	34722	11111 → 34722 [FIN, ACK] Seq=1 Ack=18 Win=5792 Le…	11111		TCP: 11111 → 347
0.019478	34724	34724 → 11111 [PSH, ACK] Seq=1 Ack=1 Win=5840 Le…	11111		TCP: 34724 → 111
0.019525	34724	11111 → 34724 [ACK] Seq=1 Ack=18 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.019547	34723	11111 → 34723 [FIN, ACK] Seq=1 Ack=18 Win=5792 Le…	11111		TCP: 11111 → 347
0.020802	34720	34720 → 11111 [FIN, ACK] Seq=18 Ack=2 Win=5840 Le…	11111		TCP: 34720 → 111
0.020835	34720	11111 → 34720 [ACK] Seq=2 Ack=19 Win=5792 Len=0 …	11111		TCP: 11111 → 347
0.021284	34721	34721 → 11111 [FIN, ACK] Seq=18 Ack=2 Win=5840 Le…	11111		TCP: 34721 → 111
0.021311	34721	11111 → 34721 [ACK] Seq=2 Ack=19 Win=5792 Len=0 …	11111		TCP: 11111 → 347

The 5 connections are:
34720 -> 11111
34721 -> 11111
34722 -> 11111
34723 -> 11111
34724 -> 11111

Each connection establishment uses a three-way handshake TCP. Once all pending packets are transmitted successfully, sender sends the TCP FIN to the receiver. The two packets TCP FIN and TCP FIN ACK are used for connection termination. Relative sequence numbers are used incremented on the receipt of ACKs for sent packet.

Question 4

Solution: Flow graph for trace.5connections:



RST and RST Ack closes the connection in both the directions immediately.
In the given pcap, the connections are being reset. Here, for RST/ACK, the device is acknowledging whatever data was sent in the previous packet(s) in the sequence with an ACK and then notifying the sender that the connection has closed with the RST.

Question 5

Solution: The TCP options supported by exercises/traces/trace.ipv6.google.com.pcap are :

- **No operation (NOP)** - If a TCP header has multiple options defined, then sometimes it makes sense to have them start on an even 32-bit boundary. When this is the case, multiple No Operation options can be chained together, filling out the space in between the other (real) options. The No Operation option does not have option-length or option-data fields. It is simply a one-byte option used to internally pad the TCP header.
- **Timestamps** – Has fields TSval and TSecr. The Timestamp option can be used to measure the round-trip time (RTT) of every packet that is acknowledged.
- **Window scale** - The TCP Window Scaling option provides the mechanism to increase that window scale value and push more data into the pipe at once.
- **Maximum segment size** - The default TCP Maximum Segment Size is 536. Where a host wishes to set the maximum segment size to a **value** other than the default, the maximum segment size is specified as a TCP option, initially in the TCP SYN packet during the TCP handshake.
- **SACK permitted** – Stands for Selective Acknowledgement. The SACK-permitted option is offered to the remote end during TCP setup as an option to an opening SYN packet.

- **End of option list** - The **TCP** End of **Option** List **option** is used to indicate the last **option** in the list has been reached.

IV. Page 125 of **Bonaventure book**, answer the below questions: [12 points]

Question 13

Solution:

- telnet: Usage: **telnet** host_name port_number - allows users to test connectivity to remote machines
- netstat:

```
C:\Users\nimse>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    10.0.0.205:50210       72.21.91.29:http       CLOSE_WAIT
  TCP    10.0.0.205:50620       52.230.222.68:https    ESTABLISHED
  TCP    10.0.0.205:50659       ec2-34-196-254-234:https  ESTABLISHED
  TCP    10.0.0.205:51423       64.88.171.12:http      ESTABLISHED
  TCP    10.0.0.205:51424       64.88.171.12:http      TIME_WAIT
  TCP    10.0.0.205:51425       64.88.171.12:https     ESTABLISHED
  TCP    10.0.0.205:51426       104.18.13.245:https    ESTABLISHED
  TCP    10.0.0.205:51427       104.18.13.245:https    ESTABLISHED
  TCP    10.0.0.205:51429       237:https              ESTABLISHED
  TCP    10.0.0.205:51430       237:https              TIME_WAIT
  TCP    10.0.0.205:51433       104.17.70.206:https    ESTABLISHED
  TCP    10.0.0.205:51435       a104-100-106-53:https  ESTABLISHED
  TCP    10.0.0.205:51439       server-99-84-238-82:https  ESTABLISHED
  TCP    10.0.0.205:51443       server-13-35-121-85:https  ESTABLISHED
  TCP    10.0.0.205:51444       a23-5-254-170:https    ESTABLISHED
```

How nmap operates ?

- It is a Network Mapper used for vulnerability scanning and network discovery. It is used mostly by the network administrators to identify what devices are running on the system, discovering available hosts, services they offer, finding open ports and detecting security risks.
  It can be used to monitor single hosts as well as vast networks that consist of thousands of devices and multitudes of subnets.

```
C:\Users\nimse>nmap 172.217.6.68
Starting Nmap 7.91 ( https://nmap.org ) at 2020-10-27 18:58 Pacific Daylight Time
Nmap scan report for sfo07s17-in-f4.1e100.net (172.217.6.68)
Host is up (0.061s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 10.36 seconds

C:\Users\nimse>
```

Question 15

Solution:

- TCP options : NOP and Timestamps are used

- RTT evolution:
  Trace 0: no particular trend in the RTT graph
  Trace 1: seems consistent except for 3 packets that need retransmission
  Trace 2: higher RTTs compared to trace 1. Multiple retransmissions
  Trace 3:
- Fast restransmit:
  Trace 0: frame 187
  Trace 1: frame 194
  Trace 2: frame 218
  Trace 3: frame 182
- Restransmit due to TCP transmit timeout:
  Trace 0: frame 42
  Trace 1: frame 113
  Trace 2: frame 64
  Trace 3: frame 241
- Implement heuristics for retransmission: if duplicate packets acknowledgement is received, we can set heuristics to retransmit the packet
- File exchanged during transfer:
  Trace 0:

  .ÌÔçúÞÜ0WÝ`´?þÀþÀ®0:zÖÏòWºÁi
  @i=
  mh6ves that fills the user's buffer before a
  PUSH is seen, the data is passed to the user in buffer size units.

  TCP also provides a means to communicate to the receiver of data that
  at some point further along in the data stream than the receiver is

  [Page 12]

  September 1981
  Transmission Control Protocol
  Philosophy

  currently reading there is urgent data. TCP does not attempt to
  define what the user specifically does upon being notified of pending
  urgent data, but the general notion is that the receiving process will
  take action to process the urgent data quickly.

  2.9. Precedence and Security

  The TCP makes use of the internet protocol type of service field and
  security option to provide precedence and security on a per connection
  basis to TCP users. Not all TCP modules will necessarily function in
  a multilevel secure environment; some may be limited to unclassified

use only, and others may operate at only one security level and compartment. Consequently, some TCP implementations and services to users may be limited to a subset of the multilevel secure case.

TCP modules which operate in a multilevel secure environment must properly mark outgoing s

Trace 1:

.ÌÔçúÞÜ0WÝ`´?þÀþÀÆ>0:Û"9¹`=Þ
@¶
+&>ol
Functional Specification

user at site B to send any data on it will result in the site B TCP receiving a reset control message. Such a message indicates to the site B TCP that something is wrong, and it is expected to abort the connection.

Assume that two user processes A and B are communicating with one another when a crash occurs causing loss of memory to A's TCP. Depending on the operating system supporting A's TCP, it is likely that some error recovery mechanism exists. When the TCP is up again, A is likely to start again from the beginning or from a recovery point. As a result, A will probably try to OPEN the connection again or try to SEND on the connection it believes open. In the latter case, it receives the error message "connection not open" from the local (A's) TCP. In an attempt to establish the connection, A's TCP will send a segment containing SYN. This scenario leads to the example shown in figure 10. After TCP A crashes, the user attempts to re-open the connection. TCP B, in the meantime, thinks the connection is open.

TCP A	TCP B
1. (CRASH)	(send 300,receive 100)
2. CLOSED	ESTABLISHED
3. SYN-SENT --> <SEQ=400><CTL=SYN>	--> (??)
4. (!!) <-- <SEQ=300><ACK=100><CTL=ACK>	<-- ESTABLISHED

5

Trace 2:
.ÌÔçúÞÜ0WÝ`´?þÀþÀÆ=0: mRp□ä
@F
ëlæelete the
TCB, and return.

[Page 70]

SEGMENT ARRIVES

third check security and precedence

SYN-RECEIVED

If the security/compartment and precedence in the segment do not exactly match the security/compartment and precedence in the TCB then send a reset, and return.

ESTABLISHED STATE

If the security/compartment and precedence in the segment do not exactly match the security/compartment and precedence in the TCB then send a reset, any outstanding RECEIVEs and SEND should receive "reset" responses. All segment queues should be flushed. Users should also receive an unsolicited general "connection reset" signal. Enter the CLOSED state, delete the TCB, and return.

Note this check is placed following the sequence check to prevent a segment from an old connection between these ports with a different security or precedence from causing an abort of the current connection.

fourth, check the SYN bit,

SYN-RECEIVED
ESTABLISHED STATE
FIN-WAIT STATE-1
FIN-WAIT STATE-2
CLOSE-W

Trace 3:

.ÌÔçúÞÜ0WÝ`´?þÀþÀÆã0:Üh¹â
y
@û
²<¬r Note that data following the
urgent pointer (non-urgent data) cannot be delivered to the user
in the same buffer with preceeding urgent data unless the
boundary is clearly marked for the user.

To distinguish among several outstanding RECEIVEs and to take
care of the case that a buffer is not completely filled, the
return code is accompanied by both a buffer pointer and a byte
count indicating the actual length of the data received.

Alternative implementations of RECEIVE might have the TCP

[Page 48]

allocate buffer storage, or the TCP might share a ring buffer
with the user.

Close

Format: CLOSE (local connection name)

This command causes the connection specified to be closed. If
the connection is not open or the calling process is not
authorized to use this connection, an error is returned.
Closing connections is intended to be a graceful operation in
the sense that outstanding SENDs will be transmitted (and
retransmitted), as flow control permits, until all have been
se