

Assignment Report

*This report contains work from **Assignment 9** of course CSN 361.*

Name- Kaustubh Trivedi
Enrollment Number - 17114044
Class - CSE B.Tech. 3rd Year
Submission Files - [Repository Link](#)

QUESTION 1: Install Wireshark and explore its uses to capture network traffic. You have to capture normal internet traffic for 20-30 minutes from your system using Wireshark. You need to copy this data in CSV / TXT file.

SOLUTION SCREENSHOT:

Activities Wireshark Thu Oct 17, 00:45 Capturing from wlp2s0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
10523	851.379697372	10.42.0.56	10.42.0.56	TCP	66	443 → 39452 [ACK] Seq=5607 Ack=19568 Win=45952 Len=0 TSval=1636278840 TSecr=3983444376
10524	851.844984502	10.42.0.56	10.42.0.56	TCP	66	[TCP Keep-Alive] 43958 → 443 [ACK] Seq=1071 Ack=597 Win=64448 Len=0 TSval=1636278840 TSecr=3983444376
10525	851.844984524	10.42.0.56	10.42.0.56	TCP	66	[TCP Keep-Alive] 51374 → 443 [ACK] Seq=1687 Ack=6274 Win=64128 Len=0 TSval=1052784701 TSecr=1064772432
10526	851.848305172	13.224.17.78	10.42.0.56	TCP	66	[TCP Keep-Alive ACK] 443 → 43958 [ACK] Seq=566 Ack=1072 Win=64448 Len=0 TSval=1103687898 TSecr=1813150034
10527	851.849673636	52.4.56.49	10.42.0.56	TCP	66	[TCP Keep-Alive ACK] 443 → 51374 [ACK] Seq=6274 Ack=1688 Win=63808 Len=0 TSval=1064818552 TSecr=1052556664
10528	852.131318599	10.42.0.56	74.125.24.189	UDP	65	49585 → 443 Len=23
10529	852.239565957	74.125.24.189	10.42.0.56	UDP	62	443 → 49585 Len=20
10530	857.641757680	10.42.0.56	91.189.89.199	NTP	90	NTP Version 4, client
10531	857.918721007	91.189.89.199	10.42.0.56	NTP	90	NTP Version 4, server
10532	859.952553313	SamsungE_28:96:71	Broadcast	ARP	42	Who has 10.42.0.1? Tell 10.42.0.190
10533	863.009554046	74.125.24.189	10.42.0.56	UDP	81	443 → 49585 Len=39
10534	863.030467982	10.42.0.56	74.125.24.189	UDP	70	49585 → 443 Len=28
10535	863.731787947	13.224.17.78	10.42.0.56	TLSv1.2	97	Encrypted Alert
10536	863.731839794	10.42.0.56	13.224.17.78	TCP	66	43958 → 443 [ACK] Seq=1072 Ack=597 Win=64128 Len=0 TSval=1813390198 TSecr=1163699598
10537	863.734398006	13.224.17.78	10.42.0.56	TCP	66	443 → 43958 [FIN, ACK] Seq=597 Ack=1072 Win=64448 Len=0 TSval=1163699598 TSecr=1813150034
10538	863.776133675	10.42.0.56	13.224.17.78	TCP	66	43958 → 443 [ACK] Seq=1072 Ack=598 Win=64128 Len=0 TSval=1813390242 TSecr=1163699598
10539	864.270183881	10.42.0.56	74.125.209.188	TCP	66	[TCP Keep-Alive] 5228 → 5228 [ACK] Seq=1187 Ack=591 Win=64128 Len=0 TSval=3709424836 TSecr=3709379563
10540	864.461575761	74.125.209.188	10.42.0.56	TCP	66	[TCP Keep-Alive ACK] 5228 → 56488 [ACK] Seq=1187 Ack=2 Win=250 Len=0 TSval=3709424836 TSecr=2334781618
10541	876.419694579	10.42.0.56	185.199.109.153	TCP	66	[TCP Keep-Alive] 38952 → 443 [ACK] Seq=1969 Ack=317876 Win=156800 Len=0 TSval=4169973657 TSecr=389452721
10542	876.422583304	185.199.109.153	10.42.0.56	TCP	66	[TCP Keep-Alive ACK] 443 → 38952 [ACK] Seq=317876 Ack=1970 Win=63552 Len=0 TSval=389497781 TSecr=4169431451

▶ Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▶ Ethernet II, Src: IntelCor_1c:8d:fb (ac:ed:5c:1c:8d:fb), Dst: Cybertan_83:e6:bf (c8:3d:d4:83:e6:bf)
▶ Internet Protocol Version 4, Src: 74.125.24.189, Dst: 10.42.0.56
▶ User Datagram Protocol, Src Port: 443, Dst Port: 49585
▶ Data (20 bytes)

```
0000 c8 3d d4 83 e6 bf ac ed 5c 1c 8d fb 08 00 45 00  -c-@-.-.-.-E-
0010 00 30 00 00 40 00 38 11 d5 21 4a 7d 18 bd 0a 2a  -0-0-.-.-.-*
0020 00 38 01 bb c1 b1 00 1c 7b f5 40 2d 0c 12 02 6c  -8-.-.-.-{0-.-.-1
0030 d5 f9 f5 77 2f fb 5d 13 70 3e 8c f1 ae 5c      -w/-]-p>-.-.-\
```

wlp2s0: <live capture in progress> Packets: 10542 · Displayed: 10542 (100.0%) Profile: Default

QUESTION 2: Take the CSV / TXT, which is generated in Problem Statement 1 as an input. Write a code (in any programming language of your choice) to extract the following 11 features given below in the table:

Average Packet Size	Average Flow Duration
Average no of Packets Sent per Flow	Average no of Packets Received per Flow
Average amount of Bytes Sent per Flow	Average amount of Bytes Received per Flow
Average Ratio of Incoming to Outgoing Packets	Average Ratio of Incoming to Outgoing Bytes
Average Time Interval b/w Packets Sent	Average Time Interval b/w Packets Received
Average Ratio of Connections to Number of Destination IPs	

SOLUTION CODE:

```
import csv

reader = csv.DictReader(open("Q1_out.csv"))
tout = 0
tinc = 0
out = 0
inc = 0
inflow = dict()
outflow = dict()
distdstIP = dict()

for raw in reader:
    key = raw["Source"] + raw["Destination"] + raw["Protocol"] + \
    raw["Source Port"] + raw["Destination Port"]
    distdstIP[raw["Destination"]] = 0
    if key not in inflow:
        inflow[key] = 900
        outflow[key] = 0
    outflow[key] = max(outflow[key], float(raw["Time"]))
    inflow[key] = min(inflow[key], float(raw["Time"]))
    if raw["Source"] == "10.42.0.56":
        out += 1
        tout += int(raw["Length"])
    else:
        inc += 1
        tinc += int(raw["Length"])

toflow = len(outflow) + len(inflow)
todura = 0
topack = out + inc
```

```

for k in inflow:
    todura += (outflow[k] - inflow[k])

print(len(distdstIP))
print("Average Packet Size : {}".format((tinc + tout) / topack))
print("Average Flow Duration : {}".format((todura) / toflow))
print("Average Number of Packets Sent per Flow : {}".format((out) / toflow))
print("Average Number of Packets Received per Flow : {}".format((inc) / toflow))
print("Average Number of Bytes Sent per Flow : {}".format((tout) / toflow))
print("Average Number of Bytes Received per Flow : {}".format((tinc) / toflow))
print("Average Ratio of Incoming Packets to Outgoing Packets : {}".format((out) / inc))
print("Average Ratio of Incoming Bytes to Outgoing Bytes : {}".format((tout) / tinc))
print("Average Time Interval b/w Packets Sent : {}".format((1274.680214816) / tout))
print("Average Ratio of Connections to Number of Destination IPs : {}".format((50.0) / len(distdstIP)))

```

RESULT SCREENSHOTS:

```

In [5]: print(len(distdstIP))
print("Average Packet Size : {}".format((tinc + tout) / topack))
print("Average Flow Duration : {}".format((todura) / toflow))
print("Average Number of Packets Sent per Flow : {}".format((out) / toflow))
print("Average Number of Packets Received per Flow : {}".format((inc) / toflow))
print("Average Number of Bytes Sent per Flow : {}".format((tout) / toflow))
print("Average Number of Bytes Received per Flow : {}".format((tinc) / toflow))
print("Average Ratio of Incoming Packets to Outgoing Packets : {}".format((out) / inc))
print("Average Ratio of Incoming Bytes to Outgoing Bytes : {}".format((tout) / tinc))
print("Average Time Interval b/w Packets Sent : {}".format((1274.680214816) / tout))
print("Average Ratio of Connections to Number of Destination IPs : {}".format((50.0) / len(distdstIP)))

```

```

96
Average Packet Size : 781.5988486934929
Average Flow Duration : 70.38942772349914
Average Number of Packets Sent per Flow : 2.686613475177305
Average Number of Packets Received per Flow : 5.167553191489362
Average Number of Bytes Sent per Flow : 566.6068262411347
Average Number of Bytes Received per Flow : 5572.20079787234
Average Ratio of Incoming Packets to Outgoing Packets : 0.5199004975124378
Average Ratio of Incoming Bytes to Outgoing Bytes : 0.10168456715656853
Average Time Interval b/w Packets Sent : 0.000997195585278483
Average Ratio of Connections to Number of Destination IPs : 0.5208333333333334

```


QUESTION 3: In this problem, the behavior of TCP protocol will be studied using Wireshark. For this assignment download the Wireshark captured trace file named as **tcpethe-trace** from Piazza, which is a packet trace of TCP transfer of a file from a client system to a remote server (named as *ser1*), obtained by running Wireshark on the client machine. Open **tcpethe-trace** file in Wireshark and answer the following question:

- a. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to server (*ser1*)?
- b. What is the IP address of server (*ser1*)? On what port number it is sending and receiving the TCP segments for this connection?
- c. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and *ser1*? What is it in the segment that identifies the segment as a SYN segment?
- d. What is the sequence number of the SYNACK segment sent by *ser1* to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did *ser1* determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
- e. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command; you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.
- f. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the Round Trip Time (RTT) value for each of the six segments? What is the Estimated RTT value after the receipt of each ACK? Assume that the value of the Estimated RTT is equal to the measured RTT for the first segment, and then is computed using the following Estimated RTT equation for all subsequent segments.

$$\text{Estimated RTT} = (1 - \alpha) * \text{Estimated RTT} + \alpha * \text{SampleRTT}$$

where, the new value of Estimated RTT is a weighted combination of the previous value of Estimated RTT and the new value for SampleRTT. The recommended value of $\alpha = 0.125$.

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the *ser1* server. Then select: Statistics→TCP Stream Graph→Round Trip Time Graph.

- g. What is the length of each of the first six TCP segments?
- h. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
- i. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

SOLUTION:

A. 192.168.1.102:1161

B. 128.119.245.12:80

C. seq = 0, Flag being 0x002 signifies SYN segment bit is 1 and rest are 0. Refer below for screenshot:

```

▼ Flags: 0x002 (SYN)
  000. .... = Reserved: Not set
  ...0 .... = Nonce: Not set
  .... 0... = Congestion Window Reduced (CWR): Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...0 = Acknowledgment: Not set
  .... .... 0... = Push: Not set
  .... .... .0.. = Reset: Not set
  ▶ .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....S.]
    Window size value: 16384
    [Calculated window size: 16384]
    Checksum: 0xf6e9 [unverified]
  
```

D. 0, Ack field is set to 1, seq1 determined that value by inverting the ack bit received in the SYN segment sent previously to initiate TCP communication, SYN and ACK segments are set to 1 and the rest are 0. e. What is the sequence number of the TCP segm

E. seq = 1

```

▶ Frame 4: 619 bytes on wire (4952 bits), 619 bytes captured (4952 bits)
▶ Ethernet II, Src: Actionte_8a:70:1a (00:20:e0:8a:70:1a), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
▶ Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
▼ Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 1, Ack: 1, Len: 565
  Source Port: 1161
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 565]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 566 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ....0. .... = Nonce: Not set
    ....0. .... = Congestion Window Reduced (CWR): Not set
    ....0. .... = ECN-Echo: Not set
    ....0. .... = Urgent: Not set
    ....1. .... = Acknowledgment: Set
    ....1. .... = Push: Set
    ....0. .... = Reset: Not set
    ....0. .... = Syn: Not set
    ....0. .... = Fin: Not set

0020 f5 0c 04 89 00 50 0d d6 01 f5 34 a2 74 1a 50 18 .....P...4.t.P.
0030 44 70 1f bd 00 00 50 4f 53 54 20 2f 65 74 68 65 Dp...P0 ST/ethe
0040 72 65 61 6c 2d 6c 61 62 73 2f 6c 61 62 33 2d 31 real-lab s/lab3-1
0050 2d 72 65 70 6c 79 2e 68 74 6d 20 48 54 54 50 2f -reply.htm HTTP/
0060 31 2e 31 0d 0a 48 6f 73 74 3a 20 67 61 69 61 2e 1.1..Host: gaia.
0070 63 73 2e 75 6d 61 73 73 2e 65 64 75 0d 0a 55 73 cs.umass.edu..Us
0080 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c er-Agent : Mozill
0090 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 3b 20 a/5.0 (W indows;
00a0 55 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 35 2e U; Windo ws NT 5.
00b0 31 3b 20 65 6e 2d 55 53 3b 20 72 76 3a 31 2e 30 1; en-US ; rv:1.0
00c0 2e 32 29 20 47 65 63 6b 6f 2f 32 30 30 33 30 32 .2) Gecko o/200302
00d0 30 38 20 4e 65 74 73 63 61 70 65 2f 37 2e 30 32 08 Netsc ape/7.02
00e0 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 74 2f 78 ..Accept : text/x
00f0 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 ml,appli cation/x

```

F. Seq numbers of first 6 segments in TCP connection = 1,566, 2026, 3486, 4946, 6406

Time of sending for each of the first 6 segments = 0.026477, 0.041737, 0.054026, 0.05469, 0.077405, 0.078157

Length of each segment is 1460 bytes

Times can be seen from the graph for sending and ACK received.

Est. RTT1 = 0.02746

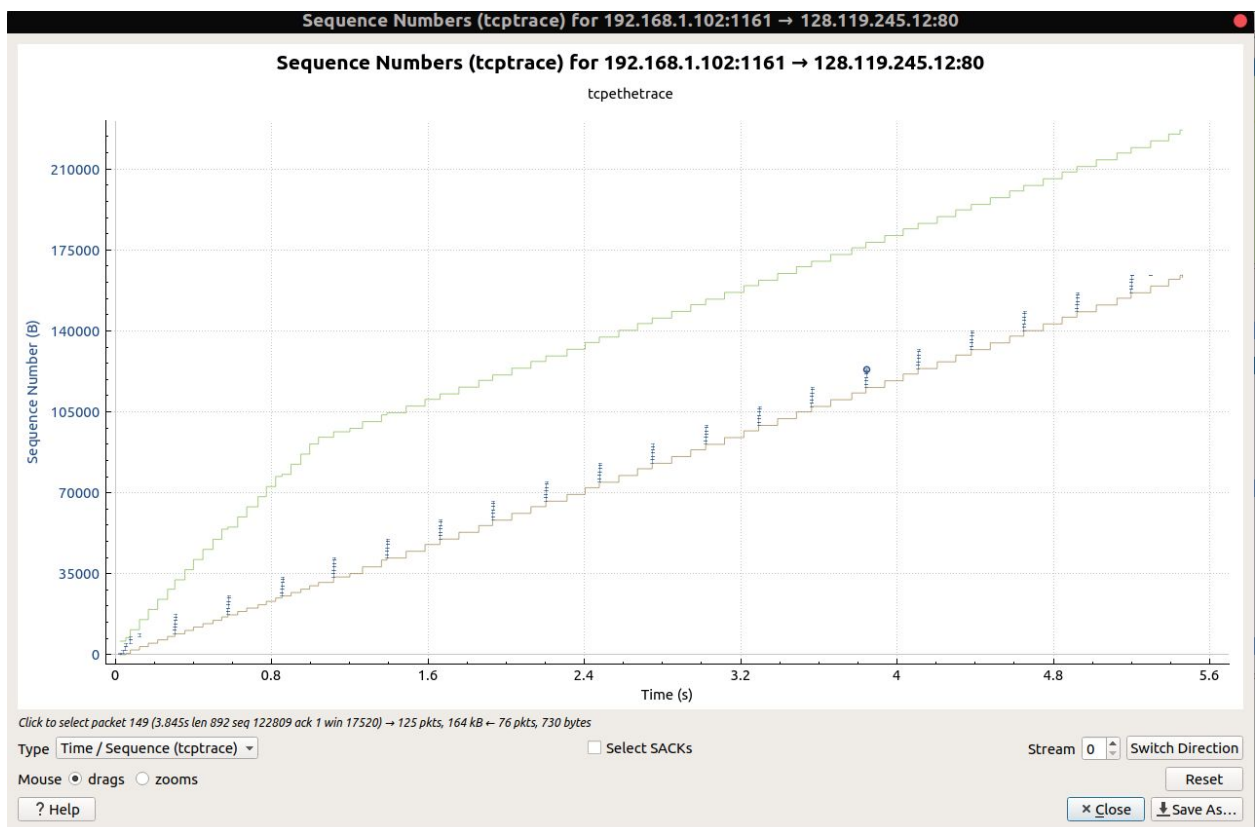
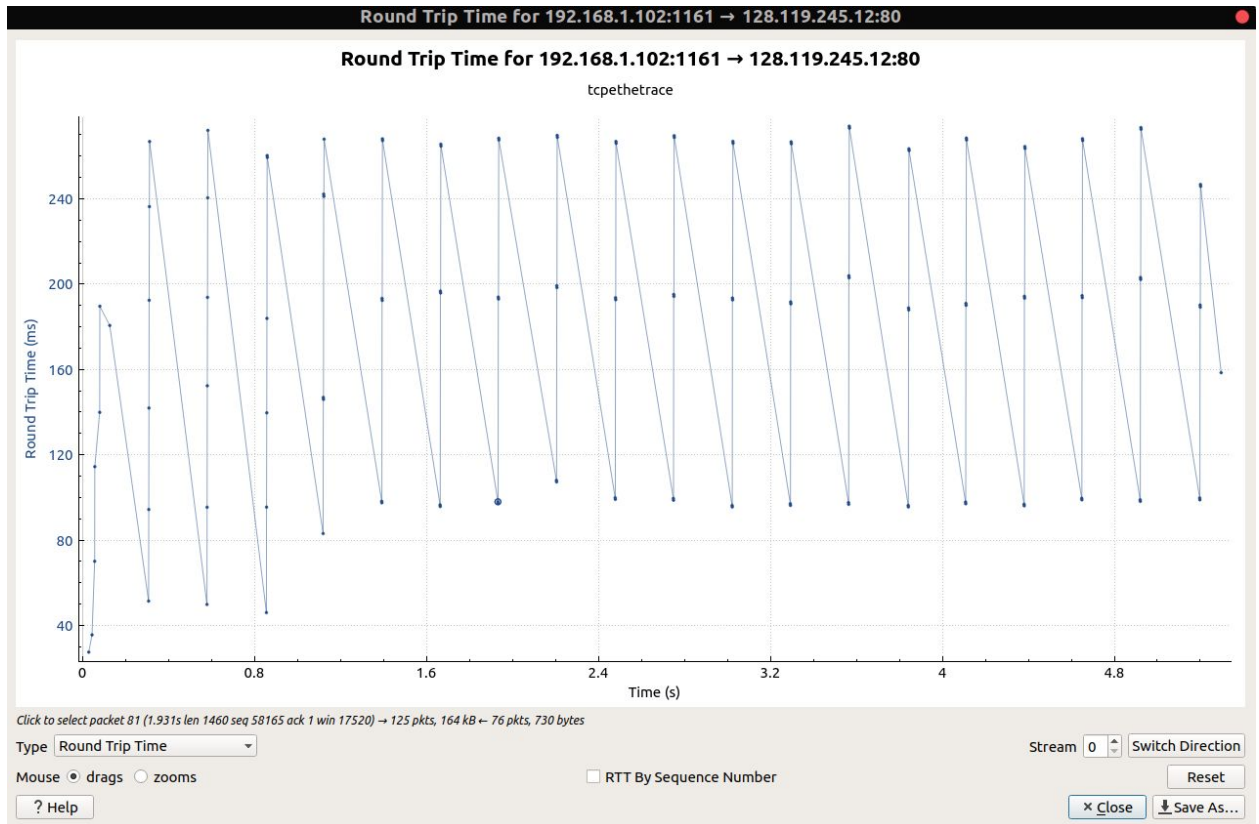
Est. RTT2 = $0.02746 * 0.75 + 0.25 * 0.035557 = 0.009621546415$

Est. RTT3 = $0.009621546415 * 0.75 + 0.25 * 0.070059 = 0.0180203069402164$

Est. RTT4 = $0.0180203069402164 * 0.75 + 0.25 * 0.114428 = 0.0301535207619163$

Est. RTT5 = $0.0301535207619163 * 0.75 + 0.25 * 0.139894 = 0.0381372224751006$

Est. RTT6 = $0.0381372224751006 * 0.75 + 0.25 * 0.189645 = 0.0528356501672178$



G. 1460 bytes

H. The minimum amount of buffer space (receiver window) advertised at ser1 for the entire trace is 5840 bytes from the first acknowledgement from the server.

This receiver window grows steadily until a maximum receiver buffer size of 62780 bytes.

The sender is never throttled due to lacking of receiver buffer space by inspecting this trace.

- I. The computation of TCP throughput largely depends on the selection of averaging time period.

We select the average time period as the whole connection time.

The average throughput for this TCP connection = ratio between the total amount data and the total transmission time.

The total amount data transmitted can be computed by the difference between the sequence number of the first TCP segment (i.e. 1 byte for No. 4 segment) and the acknowledged sequence number of the last ACK (164091 bytes for No. 202 segment).

Total data = $164091 - 1 = 164090$ bytes.

The whole transmission time is the difference of the time instant of the first TCP segment (i.e., 0.026477 second for No.4 segment) and the time instant of the last ACK (i.e., 5.455830 second for No. 202 segment).

Total transmission time = $5.455830 - 0.026477 = 5.4294$ seconds.

Hence, the throughput for the TCP connection is computed as $164090 / 5.4294 = 30.222$ KByte/sec.