CS 252: Lab 2

Abhinav Gupta, Gurnoor Singh Khurana, Sambit Behera
190050003, 190050045, 190050104

May 11, 2021

Network Diagnostic Tools

Key

Measurement 1 (**M1**): On a Mac device by Gurnoor Singh Khurana, located in Amritsar, Punjab.
Measurement 2 (**M2**): On a Windows device by Abhinav Gupta, located in Ghaziabad, UP.
Measurement 3 (**M3**): On a Mac device by Sambit Behera, located in Bhubaneswar, Orissa.

Solutions/Measurements

(i) `ifconfig`

Number of bits used in IPv4, IPv6 and MAC(hardware) address are 32, 128 and 48 respectively. Some retailers are also rolling out 64-bit MAC addresses. MTU stands for Maximum Transmission Unit. It is measured in bytes (1 byte = 8 bits). It is the largest *protocol data unit* that can be communicated in a single network layer transaction.
Transmit Queue Length is measured in the number of packets per kernel. It is the maximum number of packets allowed in the transmit queue of a Network Interface Device(NIC).

M1

- IPv4 address: 192.168.29.183
- IPv6 address: fe80::49e:1ec9:1f97:1ffe
- Hardware Address (MAC): a4:83:e7:34:ba:df
- Transmit Queue Length : **Receiving**: 128 packets **Sending**: 256 packets
- MTU: 1500

M2

- IPv4 address- **Ethernet**: 192.168.1.9, **WiFi**: 192.168.1.4
- IPv6 address- **Ethernet**: fe80:0:0:0:b361:bb1c:5323:a8d6 (or fe80::b361:bb1c:5323:a8d6 in condensed format), **WiFi**: fe80:0:0:0:3533:11f7:130b:fcf (or fe80::3533:11f7:130b:fcf in condensed format)
- Hardware address- **Ethernet**: 80:e8:2c:c6:b5:cc, **WiFi**: d0:ab:d5:c8:ac:30
- Transmit Queue Length- 1000 packets for both interfaces
- MTU: 1464 (found pinging packets of different sizes until they fragment, basically binary search)

M3
For Wifi :-

- IPv4 address: 192.168.1.3
- IPv6 address: fe80::420:b225:94c2:80c4
- Hardware address (MAC)- 38:f9:d3:ec:56:bd
- Transmit Queue length- **Receiving**: 128 packets **Sending**: 256 packets
- MTU: 1500

(ii) `traceroute`

Here is number of hops along with average RTTs from our devices M1, M2, M3 and from 5 different geographical locations G1,. . . , G5 ({hops, RTT(in ms)} format).
Locations are, namely (a) **G1:** Madrid, (b) **G2:** Brisbane, (c) **G3:** Johannesburg, (d) **G4:** Montreal; and (e) **G5:** Hong Kong

	Google	CNN	IITD
M1	11, 14.3	14, 37.7	23, 21.3
M2	8, 4.3	6, 3.7	13, 6.7
M3	11, 30.6	—	—
G1	7, 0	4, 0	14, 135
G2	12, 11.5	6, 0	26, 174
G3	8, 15	4, 0	24, 227.75
G4	9, 1	13, 8	25, 227.25
G5	22, 12	10, 2.25	17, 223

Figure 1

Here are the destination IP addresses of the servers (rather websites) when accessed from these locations.

	Google	CNN	IITD
M1	172.217.24.228	151.101.153.67	103.27.9.24
M2	216.58.196.196	199.232.21.67	103.27.9.24
M3	142.250.71.4	151.101.153.67	103.27.9.24
G1	216.58.209.68	151.101.133.67	103.27.9.24
G2	172.217.167.100	151.101.97.67	103.27.9.24
G3	172.217.170.68	151.101.225.67	103.27.9.24
G4	172.217.13.164	151.101.125.67	103.27.9.24
G5	108.177.97.99	151.101.77.67	103.27.9.24

Figure 2

Note: '—' is given when we reach a firewall somewhere in the path to server but destination is anyways shown so IP table is filled.

We see 0 readings in the RTT table. We are not sure how (because earlier hops had non-zero timings), but it takes server very less time to reach the web-address specified.
We can observe that the IP addresses are different for the same domain names(or websites). This is because websites tend to have multiple IP addresses hosted at multiple locations, mostly for load balancing and redundancy. The DNS server decides which IP address to return when we ping the domain name. When provided with multiple options, the DNS server can return any of the available IP addresses. The command `dig <domain name>` on Linux/Mac will return information about the DNS server and available IP addresses for that location.
Also, for the IITD website, destination IP address is same no matter what the source. This is most likely because they have only one server hosting their website at a single location.

(iii) `ping`

Continent	Website	RTT(in ms)			
		M1	M2	M3	ping.eu
Australia	sydney.edu.au	179	429	357	293
Asia	iitd.ac.in	23	7	120	155
North America	mit.edu	42	41	284	159
South America	fearp.usp.br	485	372	453	240
Africa	gov.za	364	302	485	180
Europe	tu-braunschweig.de	169	150	190	17

Figure 3: Average RTTs for different geographical locations

Observations:

(a) RTT increases as the geographical distance between client and server increases.For measurements from our machines, RTT values to iitd.ac.in are the least while for https://ping.eu/ , RTT values to tu-braunschweig.de are the least. This clearly follows from the fact that we are located in Asia and servers of https://ping.eu/ are located in Europe.

(b) For different packets, even from the same location, RTT times are different. This is because the network traffic changes continuously and the traffic at a given point determines the RTT.

(iv) `iperf`

Server		Clients		
		M1	M2	M3
iperf.scottlinux.com	TCP	2.42	2.08	31.4
	UDP	90.8	96.3	82.4
ping.online.net	TCP	8.59	9.32	42.9
	UDP	95.8	96.3	85.8

Figure 4: Protocol-wise bandwidths(in Mbps) on different PCs (on two servers)

Exact UDP bandwidths are measured by setting parameter `-b` to `0M` (this was discovered from the man page of iperf). Final command being

```
iperf3 -c <server> -u -b 0M
```

As expected, the bandwidths for UDP are 2-10 times the TCP bandwidths in each measurement. This is because there is no packet loss check provided by the UDP protocol, leading to a faster but inaccurate data transmission. All three measurements give the same expected results.

Bonus:

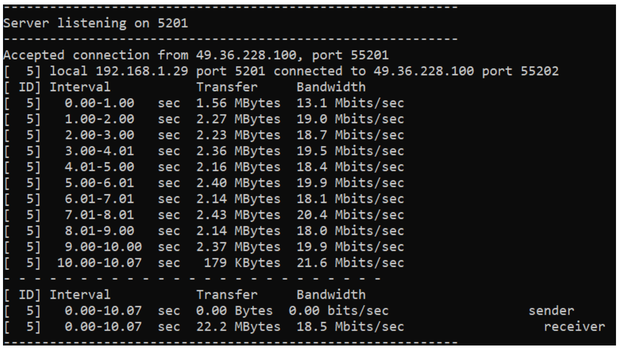
To setup the iperf3 server in the Windows PC, following steps were followed:

- Made IP address static in Network Settings, instead of original dynamic IP. This was done by overriding the automatic IP allocation system of the LAN.
- Forwarded port 5201 from router to my PC by going on router setup page, so that all requests coming to that port on my router(public IP) gets redirected to my PC. There was nothing specific with port 5201 but it was the default port captured by `iperf3` application.
- On the PC, enabled port 5201 for outside communication by setting up firewall rules for TCP and UDP protocols seperately.

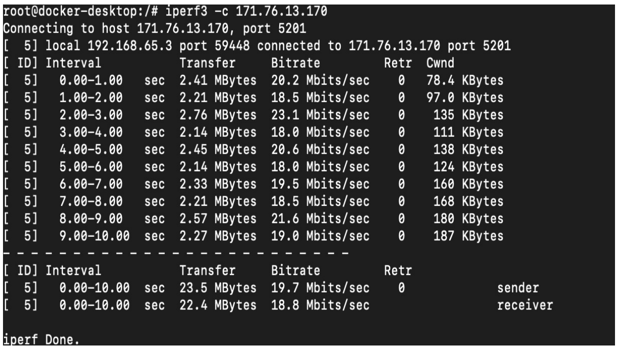
Server		Clients	
		M1	M3
M2	TCP	22.4	31.2
	UDP	92.9	86.3

Figure 5: Protocol-wise bandwidths (in Mbps) (on M2 as server)

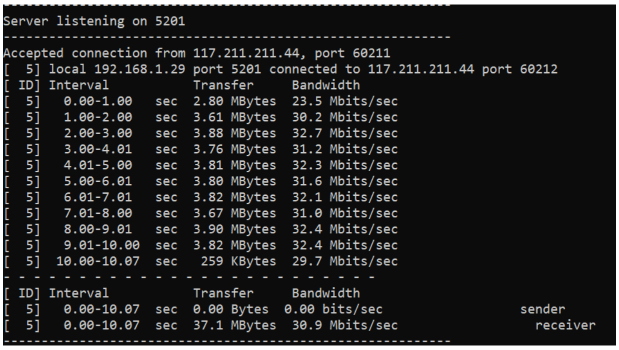
Screenshots of server as well as client side for both TCP and UDP protocols, results are same i.e. higher speeds but also high packet loss in case of UDP as compared to TCP. Throughout the experimentation, **M2** was the server.



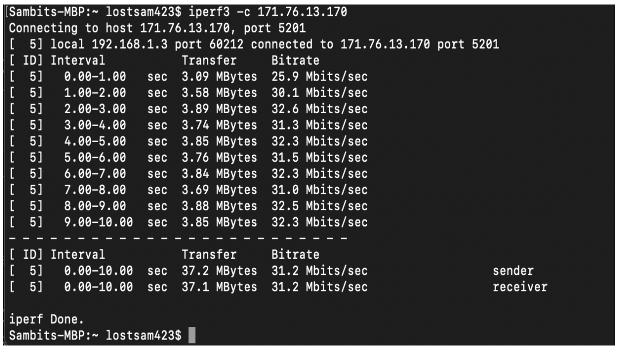
(a) Server: M2-M1 connection



(b) Client: M2-M1 connection

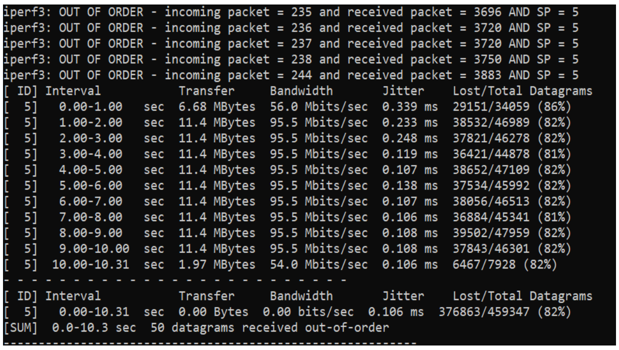


(c) Server: M2-M3 connection

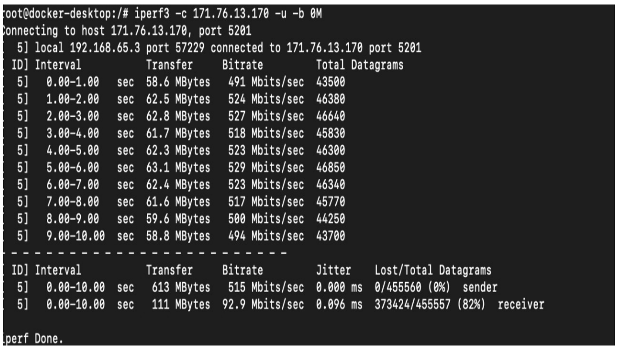


(d) Client: M2-M3 connection

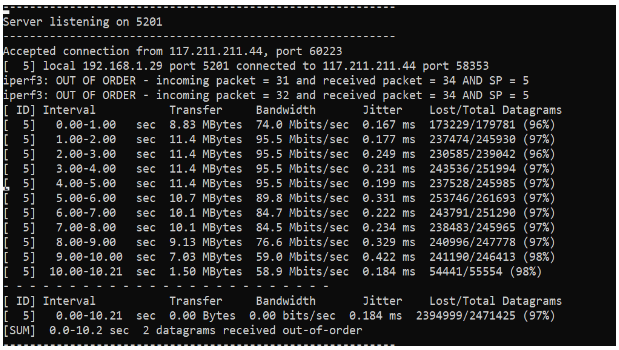
Figure 6: Screenshots for TCP connection



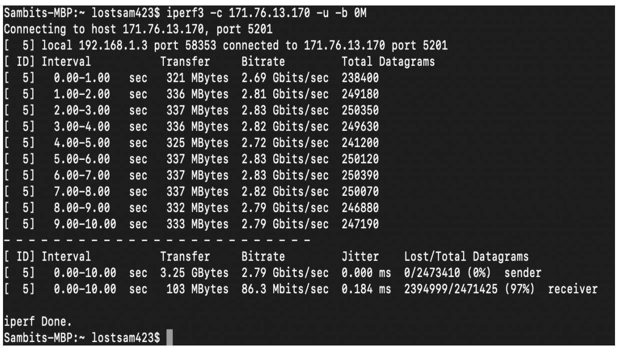
(a) Server: M2-M1 connection



(b) Client: M2-M1 connection



(c) Server: M2-M3 connection



(d) Client: M2-M3 connection

Figure 7: Screenshots for UDP connection

Note:

(a) The website given for traceroute from different cities in the problem statement was not working. So we used https://www.dotcom-tools.com/network-trace-test.

(b) Also, for Windows `traceroute` wasn't available, so an analogous service `tracert` is used to collect data.

(c) The measurements by M1 for CNN and IITD site were taken using an ubuntu container in Docker as it was not working on Mac terminal. Also, -I flag was used.

(d) iperf measurements by M1 were also taken using ubuntu container.