Universiteit Antwerpen

Title: CAN WE MEASURE UDP LATENCY OVER WI-FI BETWEEN TWO DEVICES?

Wasefi Mohammad Asif

Table of contents:

CONTENTS

Intro

For this project we need to measure the time of transmission of a UDP packet between the server and the client. There are two possibilities: one-way transmission or two-way transmission. Each method has its pros and cons. Further in this document we will discuss which method is helpful and why it was chosen.

Materials

- Ubuntu Server
- External network card to create a direct wireless access point
- nodeMCU or alternative receiving device
- Python programming language

Methods

To conduct this experiment we have two choices for measuring UDP latency between two devices: one-way transmission time or two-way transmission time.

One-way transmission time

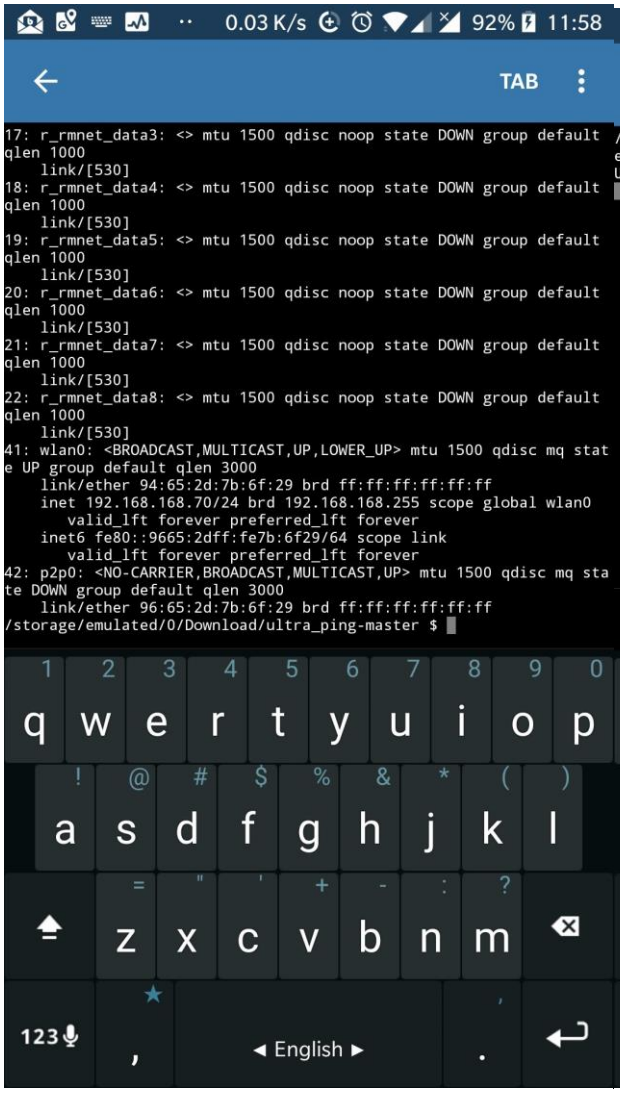
In order to measure the time of travel in one-way only from server-to-client or vice versa we need to make sure both devices are synchronized to a highly price precise external clock source. This might be an atomic clock, high-frequency radio clock etc. Keep in mind that these clocks are used for very highly time-critical applications like space programs, military communications etc. In the context of our project it would require these additional clock sources and significant amount of time to further research on how to integrate the packet sender (Ubuntu server) and receiver (nodeMCU).
However there is also an alternative clock source to synchronize the server and client with: NTP server. An NTP (Network Time Protocol) server needs both devices to send a time synchronization request in order to set the clock. Both devices need to synchronize with the same NTP server preferably not far enough and via a stable internet connection. This technology of time synchronization is used in every smartphone and computer with an internet connection. Here are a few examples of stable NTP servers:
2.be.pool.ntp.org
time.google.com
…

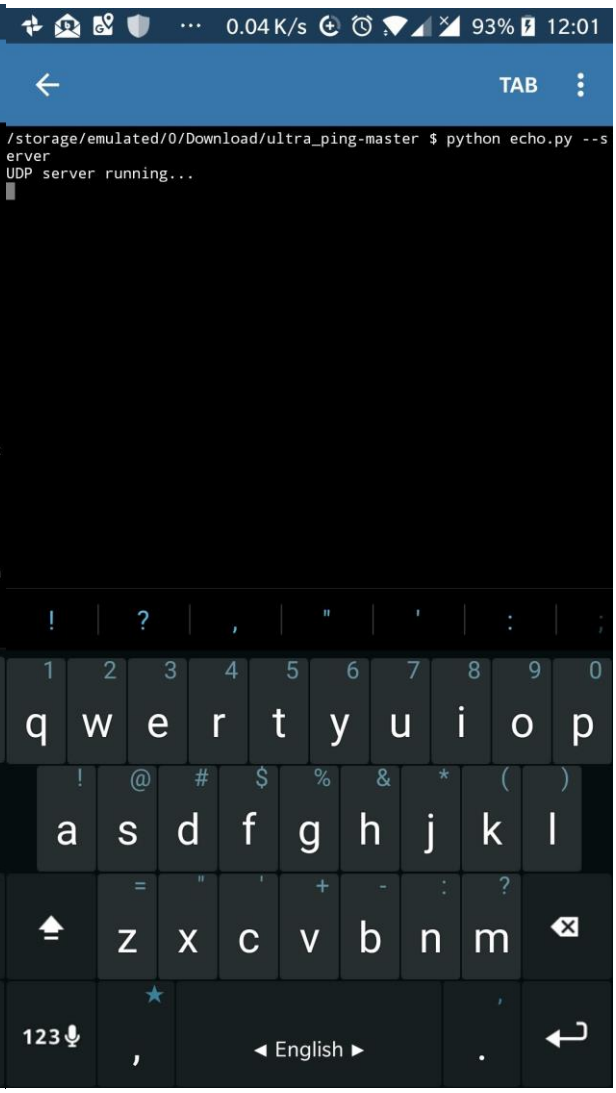
Two-way transmission time

The two way transmission time does not need great resources to accomplish the task. A server sends a UDP packet on time t_0. Upon receiving this packet back from the client it records this time as t_1. The travel time would be t_1-t_0.
For our project we have chosen this method because the communication needs to take place in both directions: server-to-client and client-to-server. A second reason would be its ease of use. We only need to generate a script to conduct this experiment. We have found a great source of code to exactly perform this operation in Python environment. Python is another programming language used in modern day devices.
Our Ubuntu server features Python by default. However our nodeMCU needs to install this program. Since our nodeMCUs are already flashed with Arduino firmware it would be inconvenient to reinstall it with a Python firmware only to run this piece of code. Thus we might also use another client device such as an Android smartphone on which installing a Python environment is pretty straight-forward. I have installed an app on my smartphone named Pydroid.

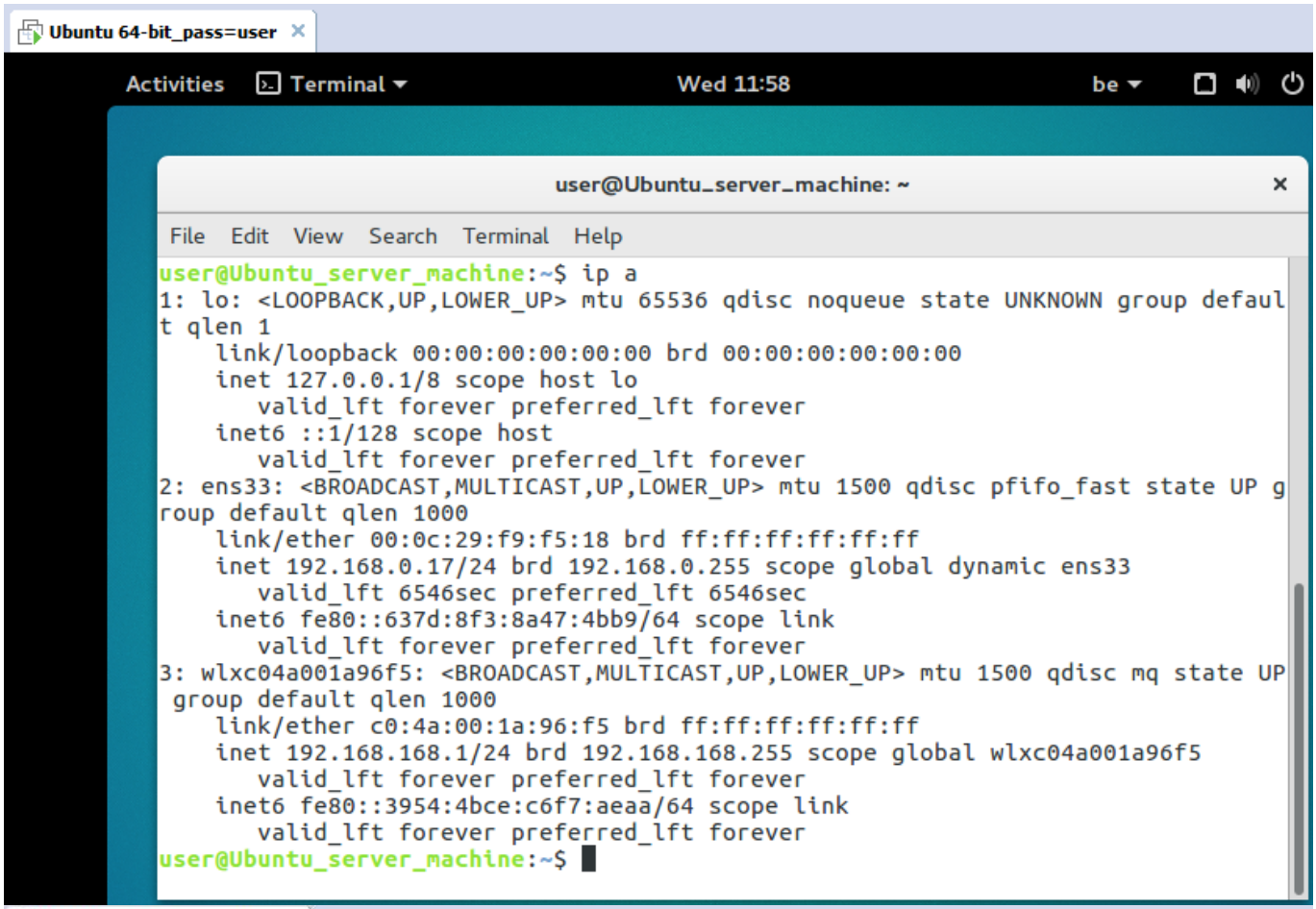
Here is how it is done:
We install the app on our smartphone and download the code to run from the Github repository. We also put this code on our Ubuntu server. Assuming that both devices are on the same network we run the code on the both devices as described in the Github repository.

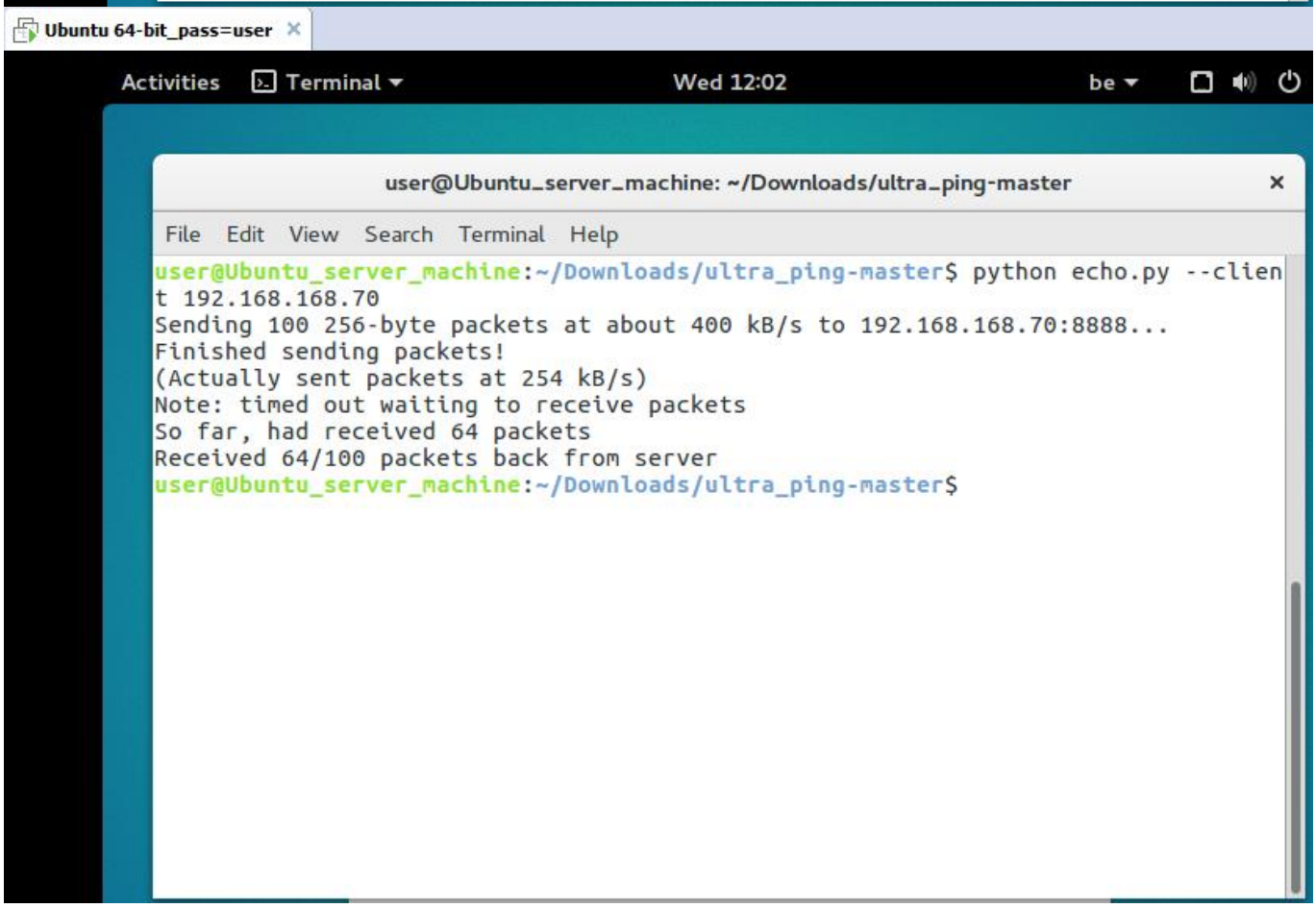
Client


0.03 K/s 92% 11:58
TAB
17: r_rmnet_data3: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
18: r_rmnet_data4: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
19: r_rmnet_data5: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
20: r_rmnet_data6: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
21: r_rmnet_data7: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
22: r_rmnet_data8: <> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/[530]
41: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 3000
link/ether 94:65:2d:7b:6f:29 brd ff:ff:ff:ff:ff:ff
inet 192.168.168.70/24 brd 192.168.168.255 scope global wlan0
valid_lft forever preferred_lft forever
inet6 fe80::9665:2dff:fe7b:6f29/64 scope link
valid_lft forever preferred_lft forever
42: p2p0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 3000
link/ether 96:65:2d:7b:6f:29 brd ff:ff:ff:ff:ff:ff
/storage/emulated/0/Download/ultra_ping-master $
English

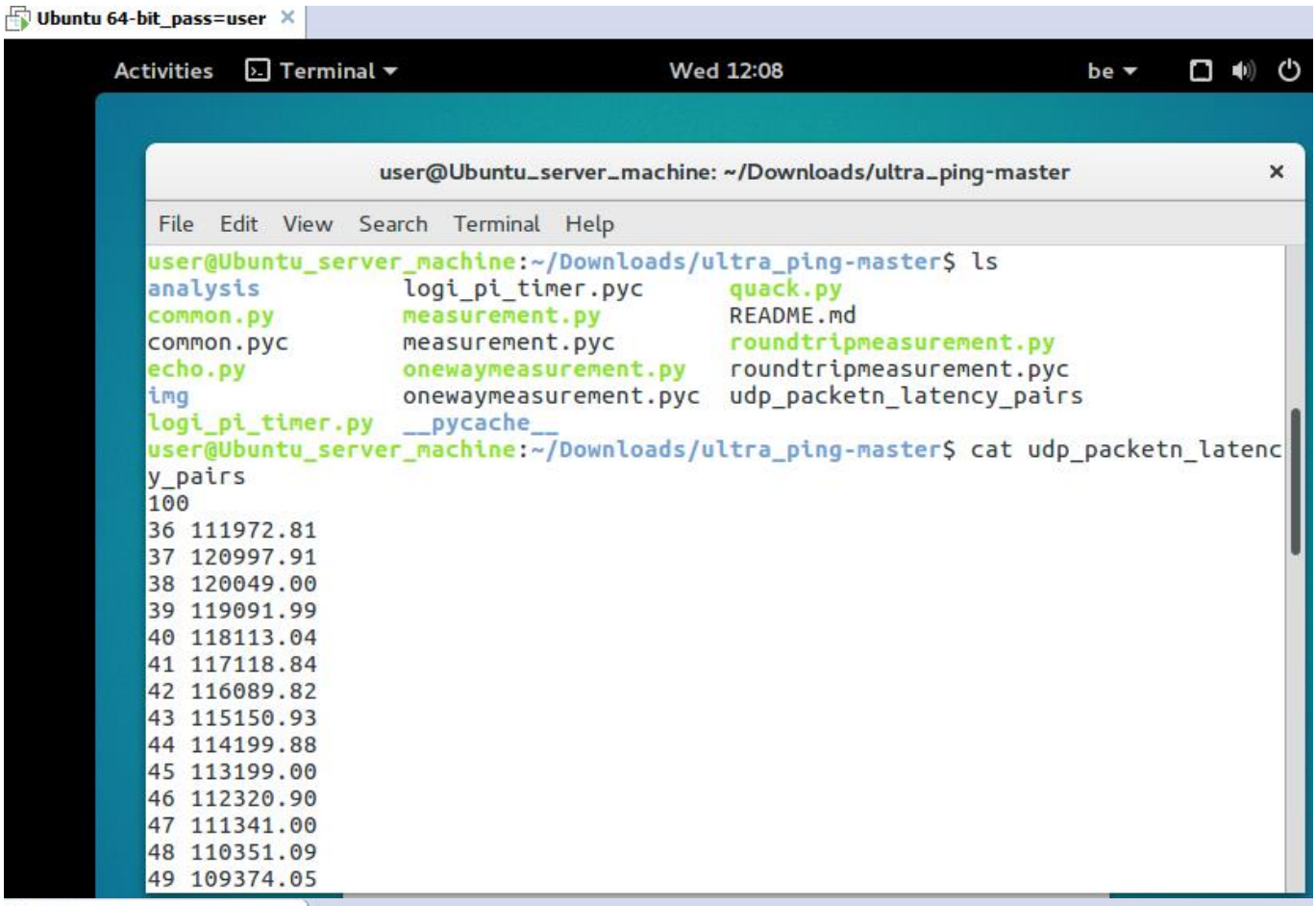

0.04 K/s 93% 12:01
TAB
/storage/emulated/0/Download/ultra_ping-master $ python echo.py --server
UDP server running...
English

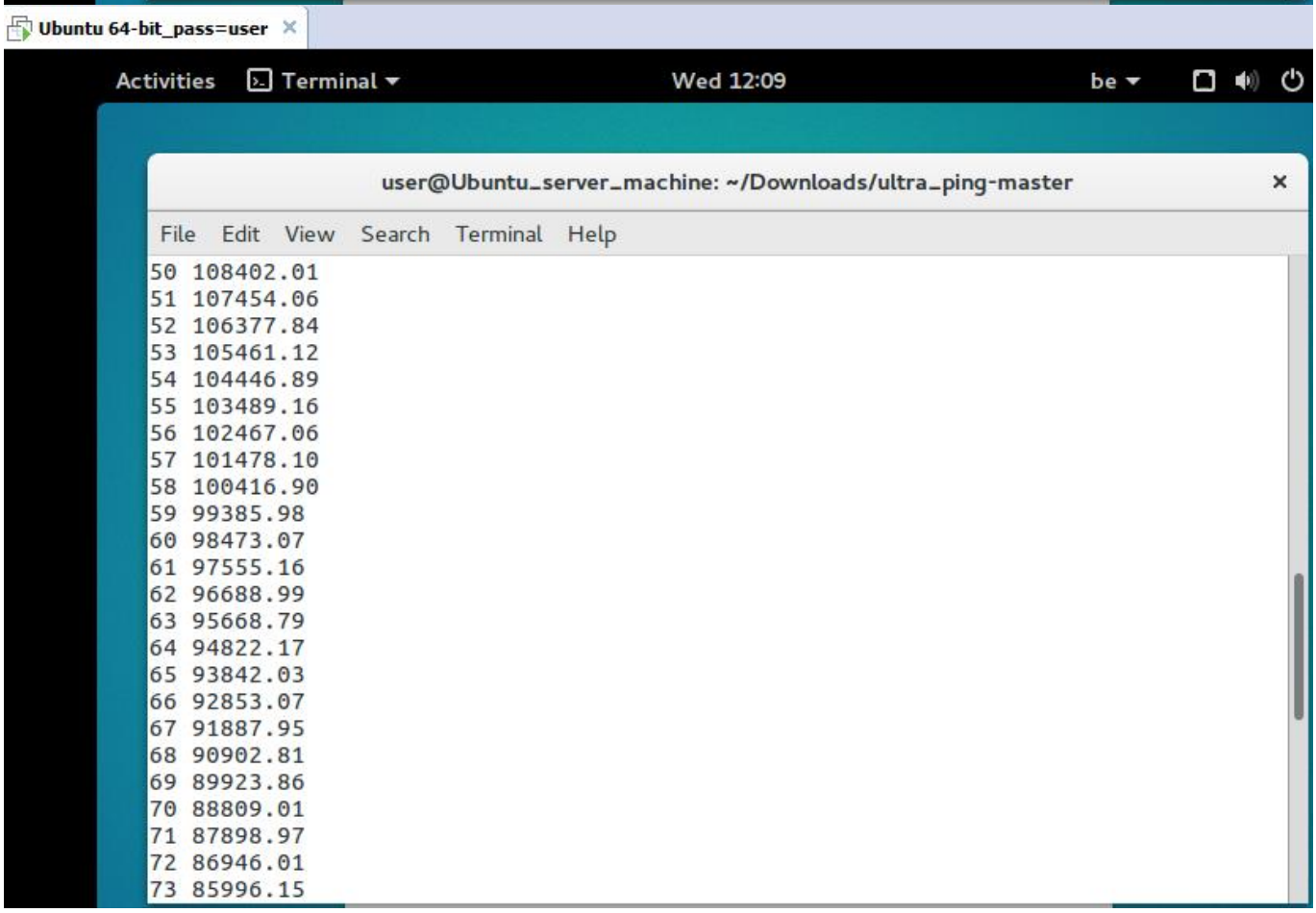
Server

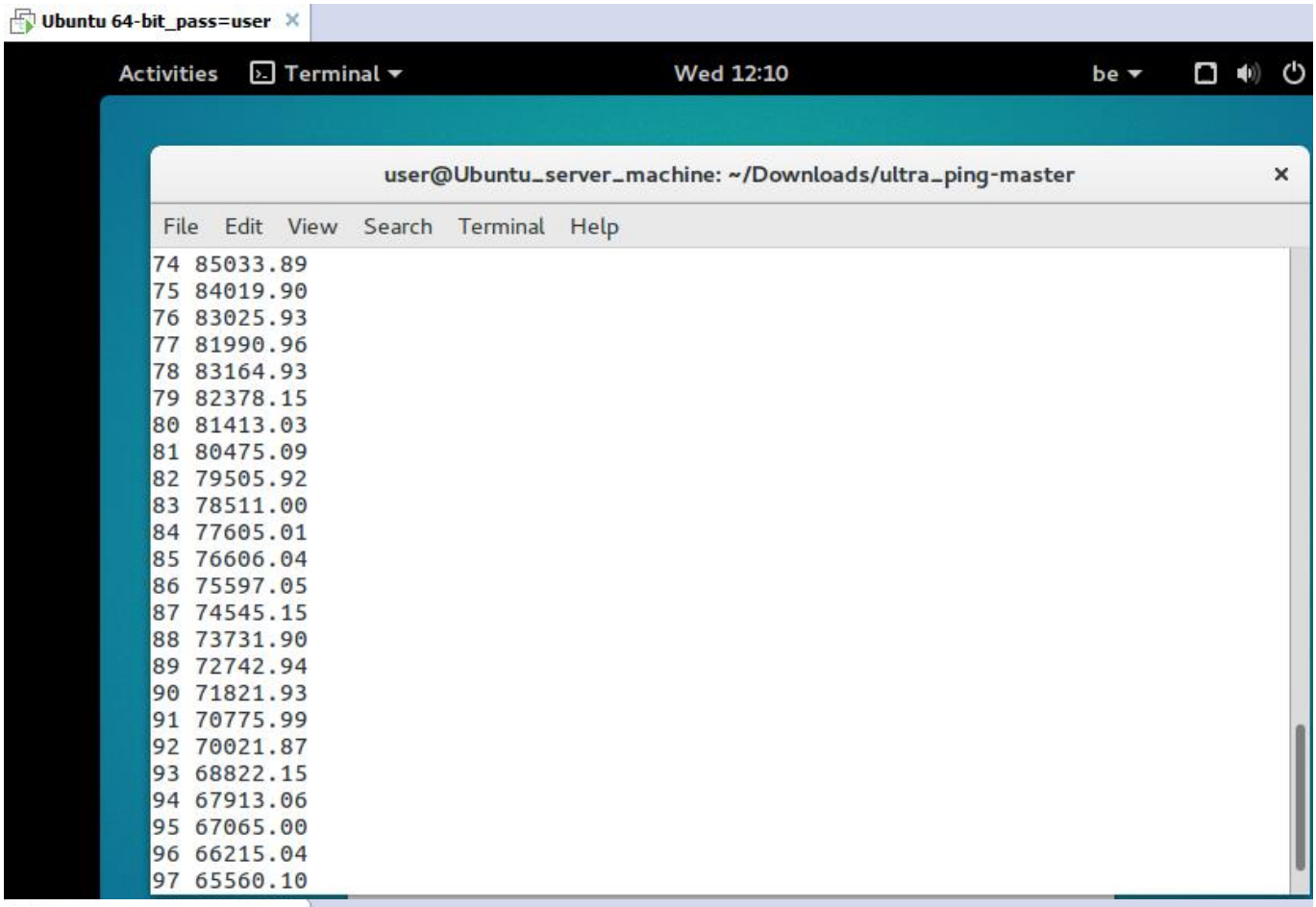

Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 11:58
be
user@Ubuntu_server_machine: ~
File Edit View Search Terminal Help
user@Ubuntu_server_machine:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
roup default qlen 1000
link/ether 00:0c:29:f9:f5:18 brd ff:ff:ff:ff:ff:ff
inet 192.168.0.17/24 brd 192.168.0.255 scope global dynamic ens33
valid_lft 6546sec preferred_lft 6546sec
inet6 fe80::637d:8f3:8a47:4bb9/64 scope link
valid_lft forever preferred_lft forever
3: wlxc04a001a96f5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
link/ether c0:4a:00:1a:96:f5 brd ff:ff:ff:ff:ff:ff
inet 192.168.168.1/24 brd 192.168.168.255 scope global wlxc04a001a96f5
valid_lft forever preferred_lft forever
inet6 fe80::3954:4bce:c6f7:aeaa/64 scope link
valid_lft forever preferred_lft forever
user@Ubuntu_server_machine:~$

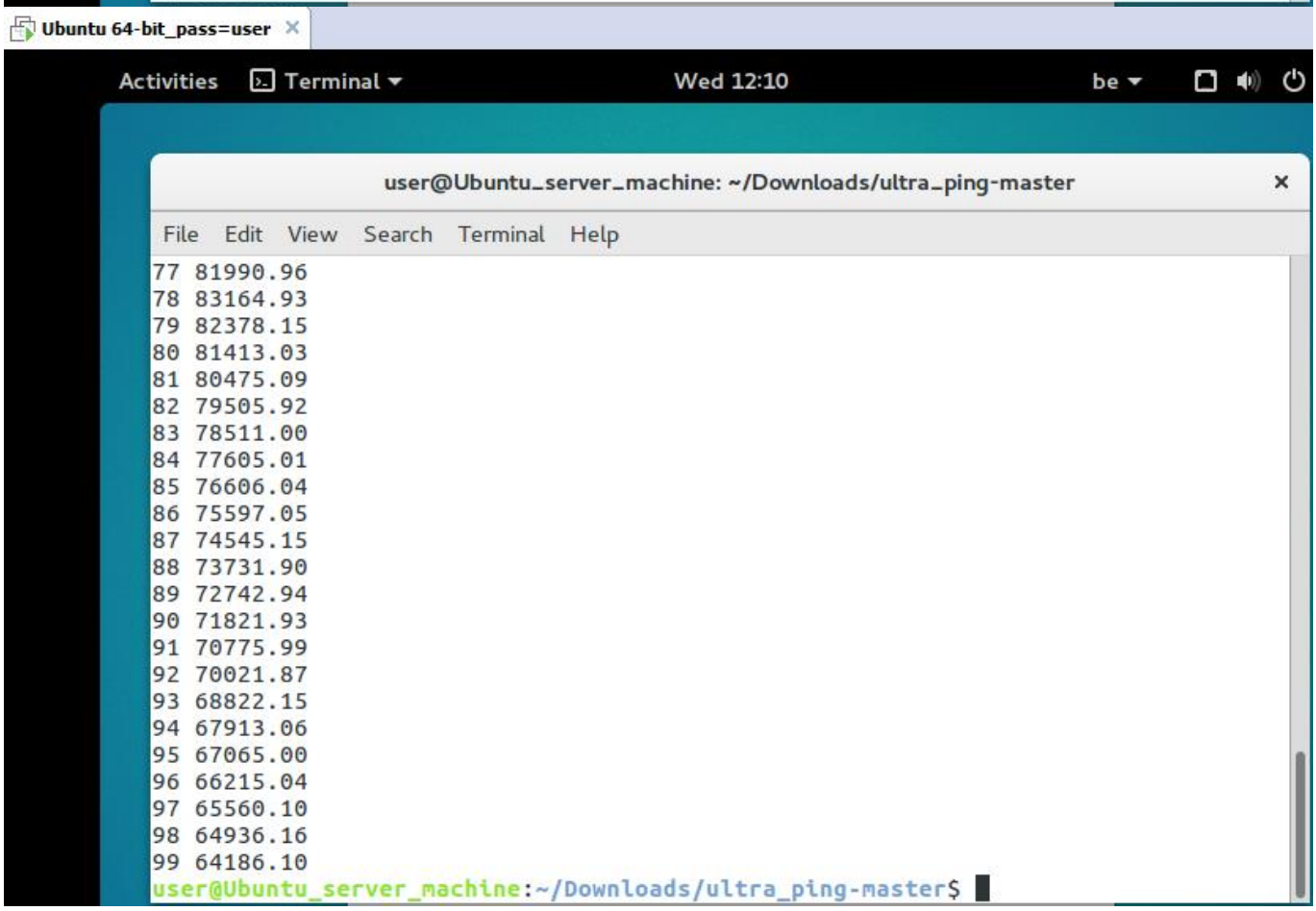

Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 12:02
be
user@Ubuntu_server_machine: ~/Downloads/ultra_ping-master
File Edit View Search Terminal Help
user@Ubuntu_server_machine:~/Downloads/ultra_ping-master$ python echo.py --clien
t 192.168.168.70
Sending 100 256-byte packets at about 400 kB/s to 192.168.168.70:8888...
Finished sending packets!
(Actually sent packets at 254 kB/s)
Note: timed out waiting to receive packets
So far, had received 64 packets
Received 64/100 packets back from server
user@Ubuntu_server_machine:~/Downloads/ultra_ping-master$

Results


Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 12:08
be
user@Ubuntu_server_machine: ~/Downloads/ultra_ping-master
File Edit View Search Terminal Help
user@Ubuntu_server_machine:~/Downloads/ultra_ping-master$ ls
analysis logi_pi_timer.pyc quack.py
common.py measurement.py README.md
common.pyc measurement.pyc roundtripmeasurement.py
echo.py onewaymeasurement.py roundtripmeasurement.pyc
img onewaymeasurement.pyc udp_packetn_latency_pairs
logi_pi_timer.py __pycache__
user@Ubuntu_server_machine:~/Downloads/ultra_ping-master$ cat udp_packetn_latenc
y_pairs
100
36 111972.81
37 120997.91
38 120049.00
39 119091.99
40 118113.04
41 117118.84
42 116089.82
43 115150.93
44 114199.88
45 113199.00
46 112320.90
47 111341.00
48 110351.09
49 109374.05


Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 12:09
be
user@Ubuntu_server_machine: ~/Downloads/ultra_ping-master
File Edit View Search Terminal Help
50 108402.01
51 107454.06
52 106377.84
53 105461.12
54 104446.89
55 103489.16
56 102467.06
57 101478.10
58 100416.90
59 99385.98
60 98473.07
61 97555.16
62 96688.99
63 95668.79
64 94822.17
65 93842.03
66 92853.07
67 91887.95
68 90902.81
69 89923.86
70 88809.01
71 87898.97
72 86946.01
73 85996.15


Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 12:10
be
user@Ubuntu_server_machine: ~/Downloads/ultra_ping-master
File Edit View Search Terminal Help
74 85033.89
75 84019.90
76 83025.93
77 81990.96
78 83164.93
79 82378.15
80 81413.03
81 80475.09
82 79505.92
83 78511.00
84 77605.01
85 76606.04
86 75597.05
87 74545.15
88 73731.90
89 72742.94
90 71821.93
91 70775.99
92 70021.87
93 68822.15
94 67913.06
95 67065.00
96 66215.04
97 65560.10


Ubuntu 64-bit_pass=user
Activities
Terminal
Wed 12:10
be
user@Ubuntu_server_machine: ~/Downloads/ultra_ping-master
File Edit View Search Terminal Help
77 81990.96
78 83164.93
79 82378.15
80 81413.03
81 80475.09
82 79505.92
83 78511.00
84 77605.01
85 76606.04
86 75597.05
87 74545.15
88 73731.90
89 72742.94
90 71821.93
91 70775.99
92 70021.87
93 68822.15
94 67913.06
95 67065.00
96 66215.04
97 65560.10
98 64936.16
99 64186.10
user@Ubuntu_server_machine:~/Downloads/ultra_ping-master$

In images above the results are shown from a file generated by the piece of code. Of the 100 packets send there are only 64 packets which have been successfully transmitted in both ways. The first column shows the number of packed which has passed and the second shows the transmission time in nanoseconds.

Information & conclusion

After conducting this experiment several times I have seen that there is a packet loss of around 30 % when sending 100 packets simultaneously. The average transmission time fluctuates between 60 to 80 milliseconds.
The packet loss might be a deciding factor for the success of our project however this will be determined only upon the actual experiment with the cars.

Referencelist

https://github.com/mrahtz/ultra_ping