Aalto-yliopisto
Aalto-universitetet
Aalto University

ELEC-E7130

Internet Traffic Measurement and Analysis

Traffic with Probe Packets

Assignment #4

Aitor Urruticoechea Puig
aitor.urruticoecheapuig@aalto.fi
Student N°101444219
October 2023

Contents

List of Figures

List of Tables

Task 1: Packet capture with probe packets

1.1 Set up active measurements

Set up active measurements by running scripts based on the table provided for selecting servers related to the "Basic measurements" assignment 2 but use shorter interval such that you will have multiple measurements within measurement period in step 2.

Active measurements have been set up for research and iperf servers using the same code as in Assignment 2, for that will also allow for a reuse of the code to analyse the data. The crontab settings, however, have been changed to report data every minute, instead of every 10. This should be okay because the measurement will only be taken for a relatively short timespan.

```
* * * * * /bin/bash /home/aurruti/Aalto/InternetTraffic/HW2/research_latency.sh >>
  /home/aurruti/Aalto/InternetTraffic/HW4/active/research.log
* * * * * /bin/bash /home/aurruti/Aalto/InternetTraffic/HW2/iperf_latency.sh >>
  /home/aurruti/Aalto/InternetTraffic/HW4/active/iperf.log
```

1.2 Capture for a duration of minimum 15 minutes

This includes regular activities such as web browsing, checking emails, watching videos, listening to music, and completing assignments, as well as the active measurements at background.
Note: Record interface counters and overall statistics at the beginning and end of the packet capture as well as store the result of these active measurements (the command outputs) for the next task.

On Wednesday, October 18, measurements were carried starting from 18:00 hours onwards for a bit over 45 minutes. During this time, crontab was active, as well as packet capture using Wireshark. This time was mostly spent watching video via streaming and occasional messaging, which means expected traffic should be quite regular.

1.3 First Sanity Check

Once the packet capture is complete, do the first sanity checks on captured data for

- *Size of trace file.*
- *Number of packets in trace file.*
- *Total size of packets.*
- *Compare values from interface counters to capture file. Is there any difference?*

Full results of the sanity check performed to the .pcap file can be consulted in Table 1. Interestingly, the file size appears to be roughly 100 KB larger than the space that the captured packets take. This is assumed to be okay, for the .pcap file needs also to store other data besides packets, like timestamps, error codes, and other relevant data to be analysed later.

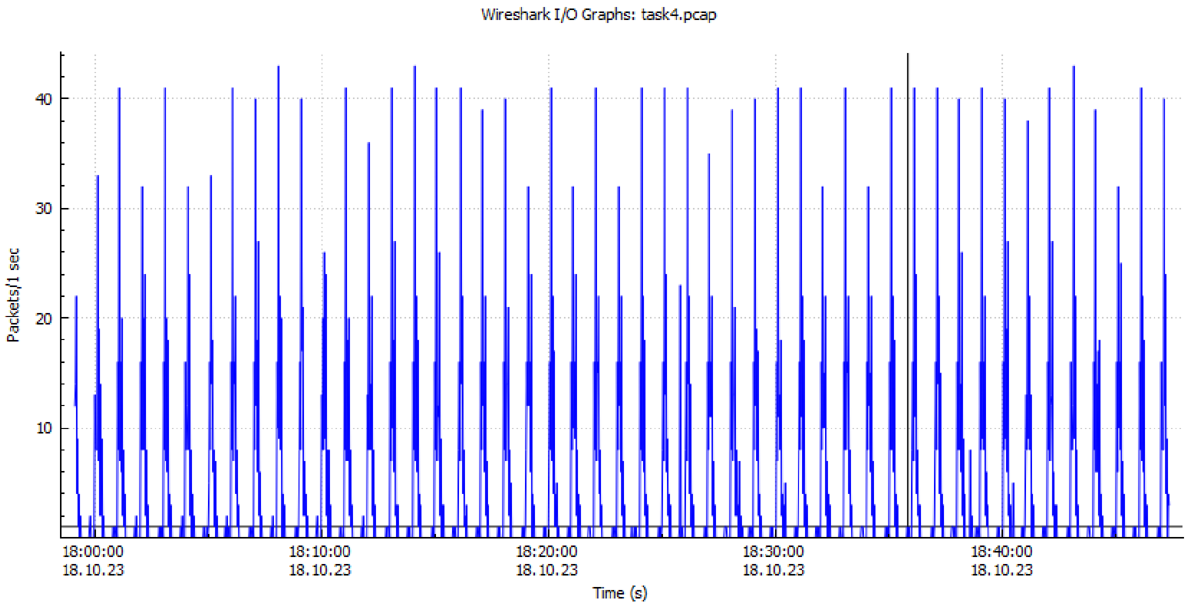
Size of track file	1.386.691 bytes
Number of packets captured	10.673 packets
Total size of packets	1.215.899 bytes

Table 1: Sanity check on the obtained data.

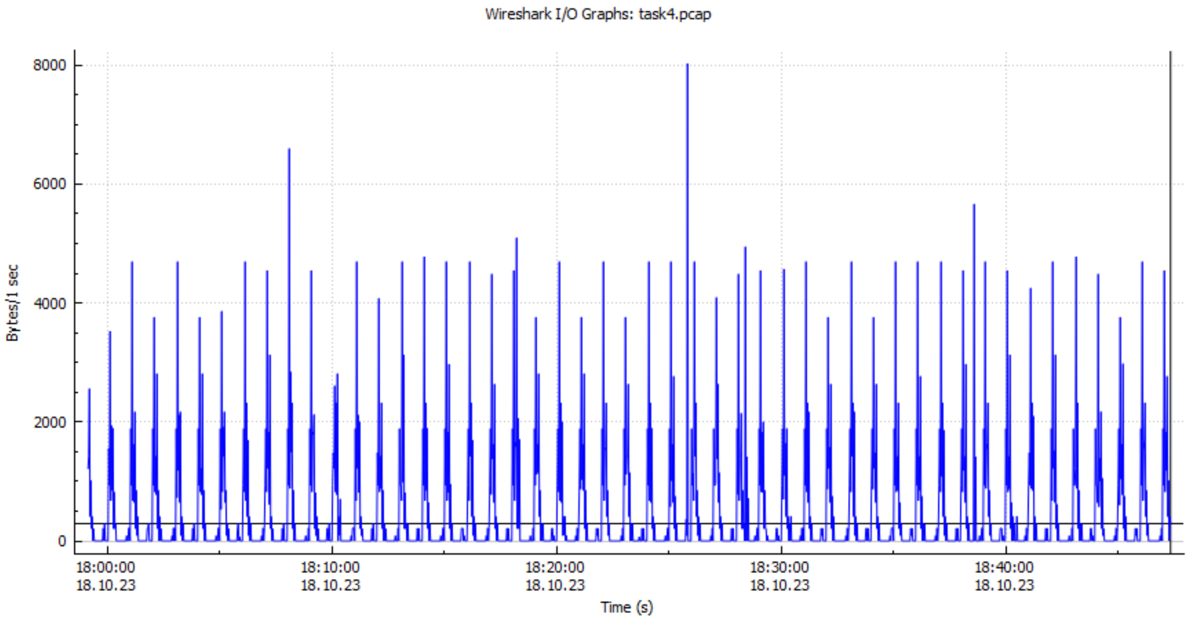
1.4 Questions for data analysis

1.4.1 Plot the traffic volume over time by considering all captured packets within the most appropriate time interval.

The time interval to focus on is the one where there is `ping` and `iperf` traffic which; thanks to clever setup when capturing data, it is the whole duration of the packet capture done by Wireshark. This is because the `crontab` setting specifying when to send the active measurements were started just before starting capture with Wireshark and were ended just after Wireshark capture was stopped. The plot of all captured packets can be seen in Figure 1.



(a) Packets/second over time.



(b) Bytes/second over time.

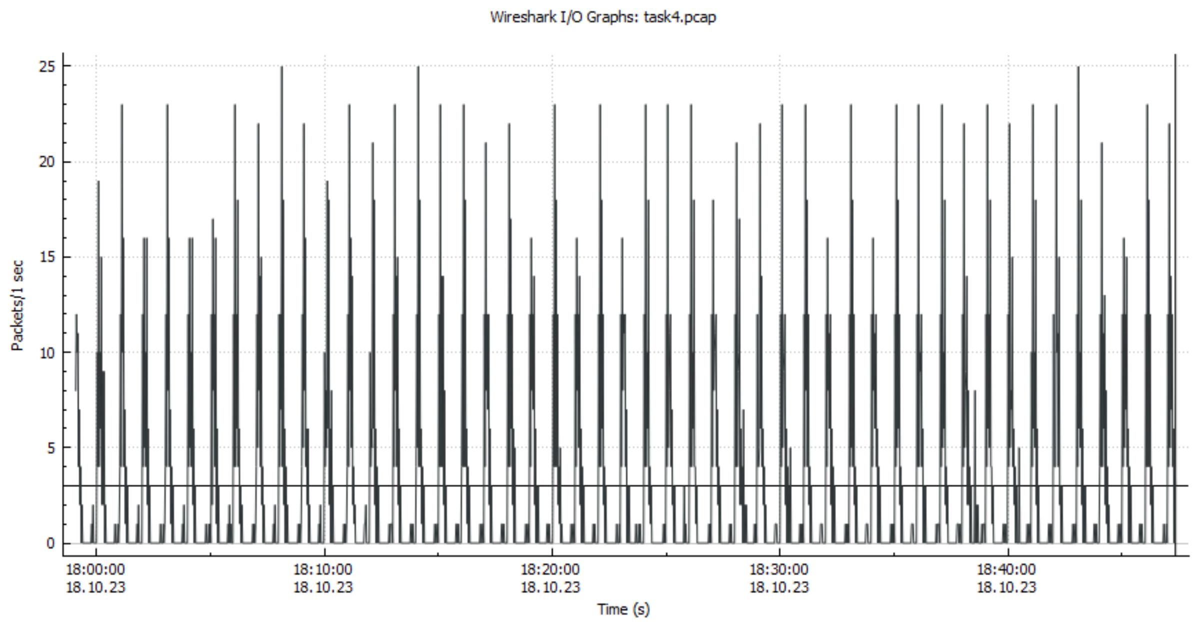
Figure 1: All captured packets over time.

1.4.2 Plot the traffic volume without the ping packets and iperf3 packets over time (select the same interval selected in the previous plot).

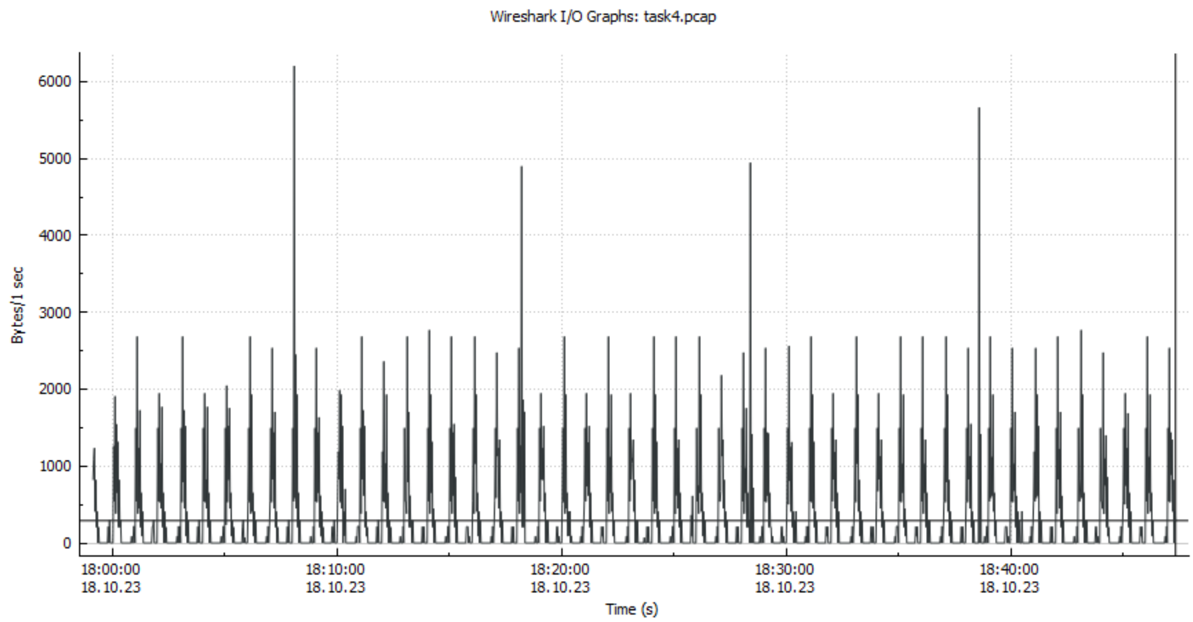
The filtering of these kinds of packets can be easily implemented in Wireshark's GUI, for it is known that ping packets will be marked by their usage of the icmp protocol, while iperf3 packets exit the computer (that has a known IP address) through port 53. These facts can be established as a "filter-out" with a "not" in front of it. This takes the overall syntax of:

not icmp and not ip.addr == 172.26.184.25 and not tcp.port == 53 or udp.port == 53

which can be directly inputted Wireshark's display filter. The resulting plots can be consulted in Figure 2.



(a) Packets/second over time.



(b) Bytes/second over time.

Figure 2: Captured packets over time, excluding those corresponding to ping and iperf3 connections.

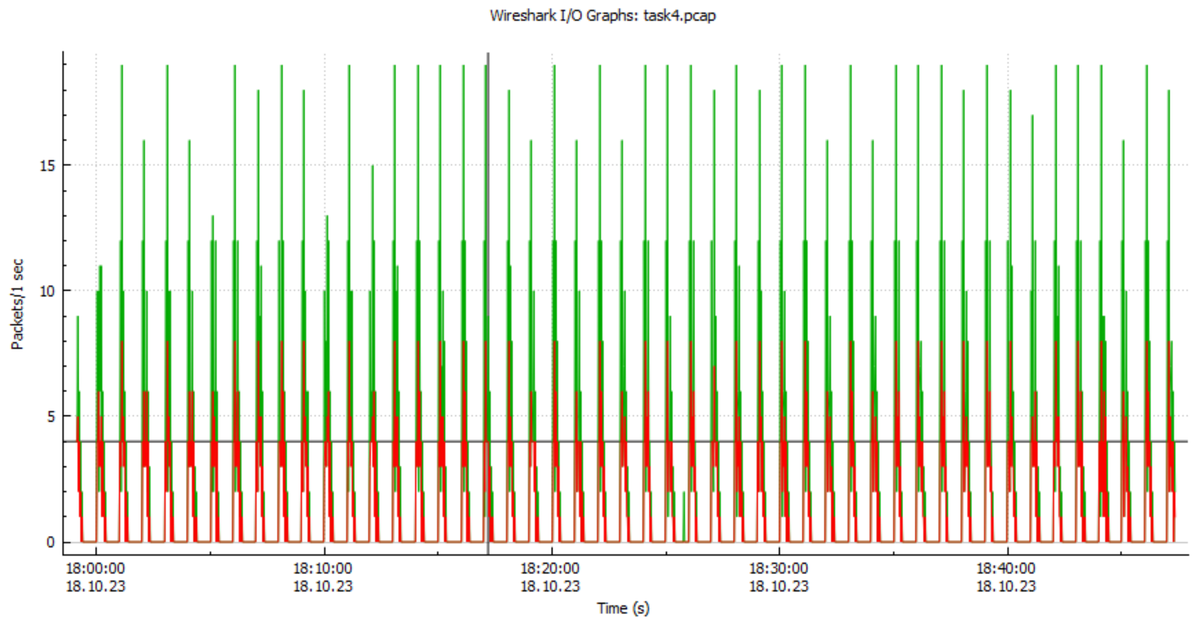
1.4.3 Plot the traffic volume comparing the ping packets with the iperf3 packets over time (keeping the same interval).

In the same fashion as the previous section, using very similar syntax the traffic volume coming from the ping and iperf3 connections can be "filtered-in":

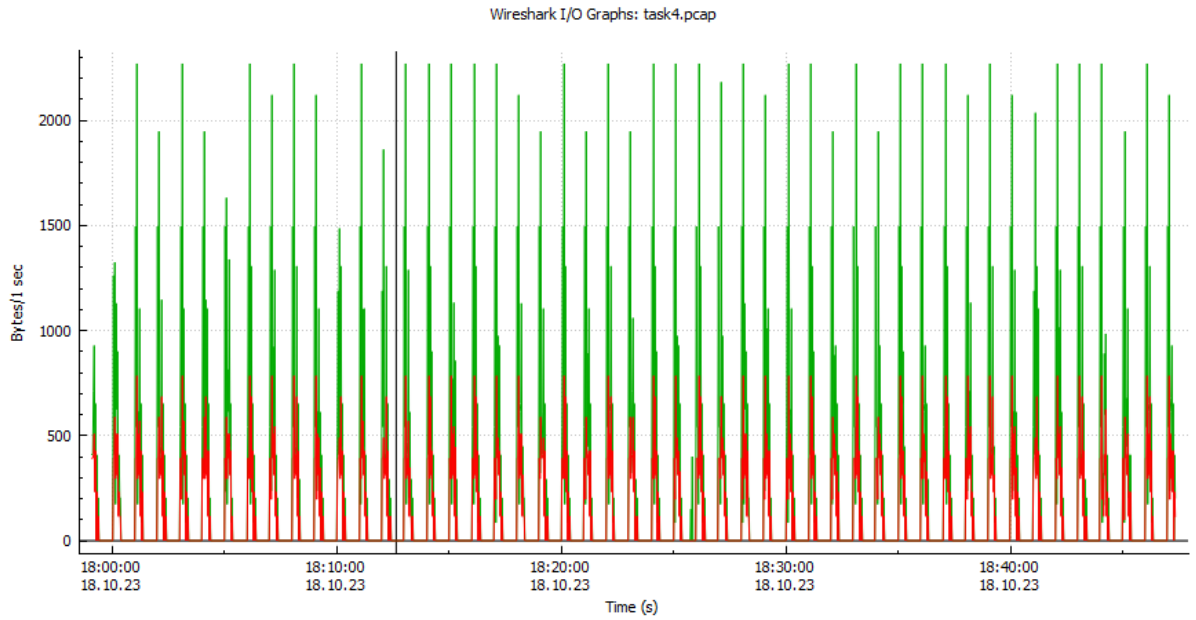
icmp

ip.addr == 172.26.184.25 and tcp.port == 53 or udp.port == 5

for ping and iperf3 respectively. Again, inputting this into Wireshark's display filter results in the desired plots (Figure 3).



(a) Packets/second over time.



(b) Bytes/second over time.

Figure 3: Captured packets corresponding to ping (red) and iperf3 (green) connections, over time.

1.4.4 Provide the average throughput.

The average throughput can be directly consulted in Wireshark's GUI, going to Statistics > Capture File Properties > Statistics. There, the average throughput is stated as 3.361 bytes/s.

1.4.5 Do you have any observations from the above plot of network traffic?

From all the previous plots, several conclusions can be drawn. Firstly, it is clear that due to heavy traffic (streaming video at Full HD quality), ping and iperf3 measurements represent but a fraction of the total captured data. The consistency observed in the overall graphic (Figure 1) also clearly indicates this overarching trend: with active measurements taken each minute and the bandwidth that was taken by constant video streaming, the overall behaviour seems predictable and stable.

When packets generated by active measurements techniques are out of the equation, a similar graph emerges (Figure 2). Interestingly, some peaks in traffic do appear very regularly every 10 minutes; which can be attributed to either some background process by the computer, like installing updates or something similar; or more likely, the need for buffering larger chunks of the video for streaming. Alternatively, for the plots of only the active measurements (Figure 3), it can be seen that in some occasions some of the ping and/or iperf3 measurements do not seem to go through, resulting in error. This can be observed in the fact that, while many do reach the very top of the graphic, there are moments where either the green, red, or both lines do not manage to reach the very top, resulting in less packets transmitted and less throughput for that connection; clearly, thus, indicating some kind of error in the attempted connection.

Task 2: Compare active and passive measurements

2.1 Extract information appropriately from the iperf3 and ping sessions

This time, and in order to experiment with other tools, pyshark has been used. This is notably less efficient than transforming the captured data into flows using CoralReef, but does provide for easier coding. This library allows for initial direct conversion of the .pcap file into a Python object. However, to transform this into a more readable and operable DataFrame, one must be careful: depending on the flow and the protocol used, different data will be available, and this should be taken into account to create an overall DataFrame with all the flows. This necessarily forces iteration through the object:

```python
# File task2.py
# Aitor Urruticoechea 2023
import pyshark
import pandas as pd
from datetime import datetime
import numpy as np
from matplotlib import pyplot as plt

filepath = "task4.pcap"

# From .pcap to DataFrame
packet_data = []
capture = pyshark.FileCapture(filepath)
for packet in capture:
    packet_info = {
        "Timestamp": packet.sniff_time,
        "Source IP": packet.ip.src if "ip" in packet else None,
        "Destination IP": packet.ip.dst if "ip" in packet else None,
        "Protocol": packet.transport_layer,
        "Length": packet.length,
    }
    if 'TCP' in packet:
        packet_info['Transport Protocol'] = 'TCP'
        packet_info['Destination Port'] = int(packet.tcp.dstport)
        packet_info['ICMP Type'] = None
        packet_info['ICMP Id'] = None
    elif 'UDP' in packet:
        packet_info['Transport Protocol'] = 'UDP'
        packet_info['Destination Port'] = int(packet.udp.dstport)
        packet_info['ICMP Type'] = None
        packet_info['ICMP Id'] = None
    elif 'ICMP' in packet:
        packet_info['Transport Protocol'] = 'ICMP'
        packet_info['Destination Port'] = None
        packet_info['ICMP Type'] = int(packet.icmp.type)
        try:
            packet_info['ICMP Id'] = packet.icmp.ident
        except:
            packet_info['ICMP Id'] = None
    else:
        packet_info['Transport Protocol'] = 'Other'
        packet_info['Destination Port'] = None
        packet_info['ICMP Type'] = None
```

```python
        packet_info['ICMP Id'] = None
    packet_data.append(packet_info)
ws_df = pd.DataFrame(packet_data)
ws_df["Timestamp"] = pd.to_datetime(ws_df["Timestamp"])
```

Once this is done, however, the required data for ping and iperf3 flows can be extracted quite easily by applying the same filters as before with WireShark's GUI. Before getting into the results, it is important to note that, when calculating throughput, some extra grouping is needed. It is not enough with grouping the flows by IP pairs, for there are many such flows with iperf3 requests every minute. They shall also be grouped by timestamp, so actual throughput is obtained for each relevant flow. Needless to say, since all of this data is already in a pandas DataFrame, this is quite quick to do:

```python
iperf3_flows = iperf3_flows.sort_values(by=['Timestamp'])
time_interval = pd.Timedelta(minutes=1)
groups = iperf3_flows.groupby(['Source IP', 'Destination IP', pd.Grouper(key='Timestamp',
↪    freq=time_interval)])
```

On the flip side, for the ping flows, delay is easier to compute because one can group them by ICMP ID, making timestamp grouping not needed. For iperf3, a snippet of the obtained throughput when observing the same IP pairs can be consulted in Figure 4. Similarly, a snippet of the obtained delay and packet loss corresponding to ping flows is included in Figure 5. Interestingly, one can observe orders of magnitude more ping related flows than iperf3 ones, since ping packets are being sent 5 times each minute and that to both server types; while iperf3 are done more sparsely and only to the two iperf3 servers.

```
throughput_df
         Source IP   Destination IP  ...              End Timestamp Throughput (bps)
0   172.26.184.25  188.166.241.168  ... 2023-10-18 17:59:12.615291           4040.0
1   172.26.184.25  188.166.241.168  ... 2023-10-18 18:00:14.163488           4040.0
2   172.26.184.25  188.166.241.168  ... 2023-10-18 18:01:13.375346           4040.0
3   172.26.184.25  188.166.241.168  ... 2023-10-18 18:02:12.848471              NaN
4   172.26.184.25  188.166.241.168  ... 2023-10-18 18:03:13.313517           4040.0
..            ...              ...  ...                        ...              ...
93  172.26.184.25   195.148.124.36  ... 2023-10-18 18:44:05.167163              NaN
94  172.26.184.25   195.148.124.36  ... 2023-10-18 18:45:05.511527              NaN
95  172.26.184.25   195.148.124.36  ... 2023-10-18 18:46:05.750465              NaN
96  172.26.184.25   195.148.124.36  ... 2023-10-18 18:47:05.292045              NaN
97  172.26.184.25      54.247.62.1  ... 2023-10-18 18:25:48.100169          12320.0
```

Figure 4: Observed throughput for captured iperf3 flows.

```
icmp_merged
              Timestamp_request Source IP_request  ...     Delay Packet Loss
0    2023-10-18 17:59:07.633248     172.26.184.25  ...  0.210690       False
1    2023-10-18 17:59:07.633248     172.26.184.25  ...  1.211678       False
2    2023-10-18 17:59:07.633248     172.26.184.25  ...  2.213452       False
3    2023-10-18 17:59:08.634934     172.26.184.25  ...  0.790996       False
4    2023-10-18 17:59:08.634934     172.26.184.25  ...  0.209992       False
...                         ...               ...  ...       ...         ...
6028 2023-10-18 18:47:14.532074     172.26.184.25  ...  3.778236       False
6029 2023-10-18 18:47:14.532074     172.26.184.25  ...  2.778255       False
6030 2023-10-18 18:47:14.532074     172.26.184.25  ...  1.777856       False
6031 2023-10-18 18:47:14.532074     172.26.184.25  ...  0.777876       False
6032 2023-10-18 18:47:14.532074     172.26.184.25  ...  0.222540       False
```

Figure 5: Observed delay and packet loss for captured ping flows.

2.2 Analyze the captured data

2.2.1 How much traffic was there that was not iperf or ping traffic?

As already seen in Figure 2, there is quite a bit of traffic not corresponding to measurements. These flows not corresponding to active measurements can be easily filtered out by applying the same filters as before, in reverse this time, to the DataFrame with all the flows. When all of it is summed, it corresponds to 7.086.256 bits of captured packets. This is compared to 403.816 bits worth of iperf3 flows and 2.640.936 bits of ping flows. As seen in Figure 6; the trend observed in the previous task is seen again, with the overwhelming majority of flows being related to activities other than measurements.

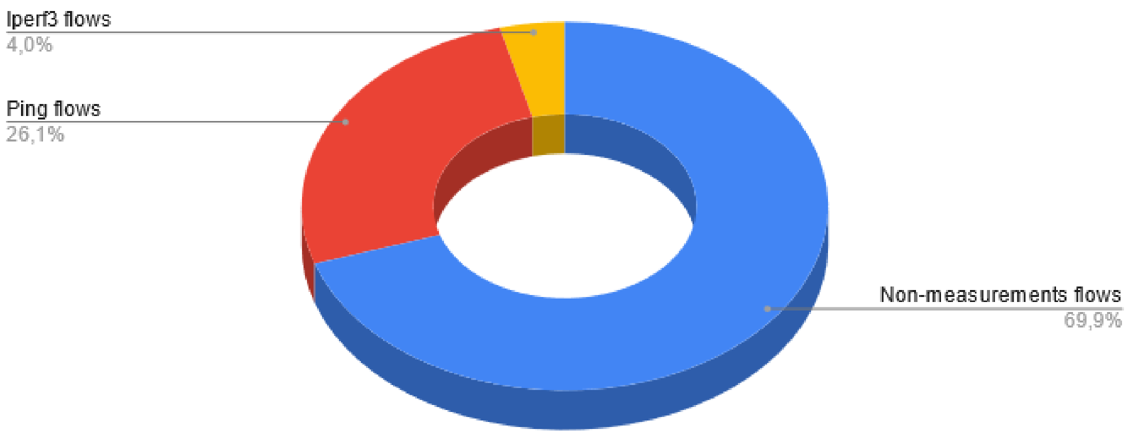


Figure 6: Flow distribution between active measurements and non-measurement activities.

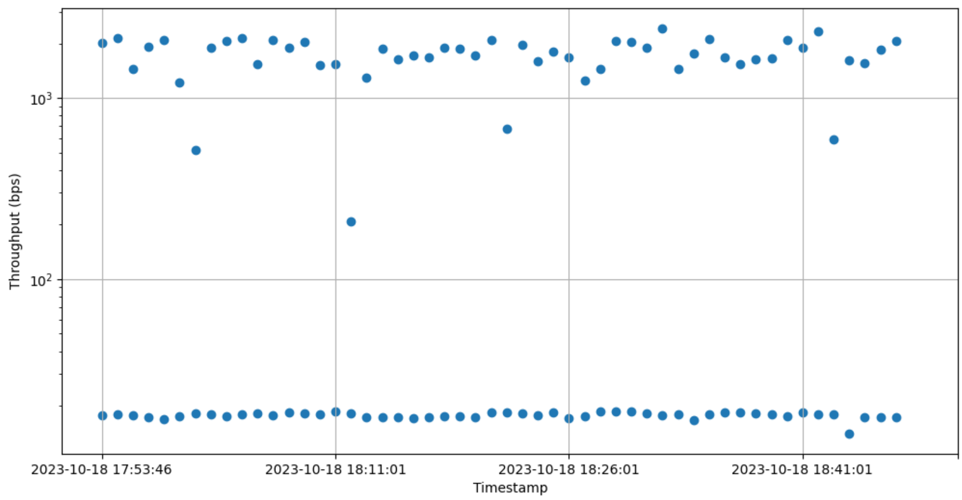
2.2.2 Compare iperf results from active and passive measurements. Provide a table and plot a time series.

The same code used for Assignment 2 to analyse active measurements using iperf3 can be reused to do the same. Note that throughput can be easily calculated having the connection time when the size of the packets transmitted is known (10 Kb). This provides with the very same data obtained by analysing the flows, which in turn allows for a comparison between methods. Nevertheless, differences are quite obvious. While Figure 4 has already shown the obtained results for passive measurements, Figure 7 shows a snippet of the obtained data for the active measurements. Once this is plotted as a time series, Figure 8 clearly showcases how the passive measurements have only managed to reflect the traffic of one of the two iperf connections, the active measurements reflect also the slower traffic of the second connections - besides being more precise.

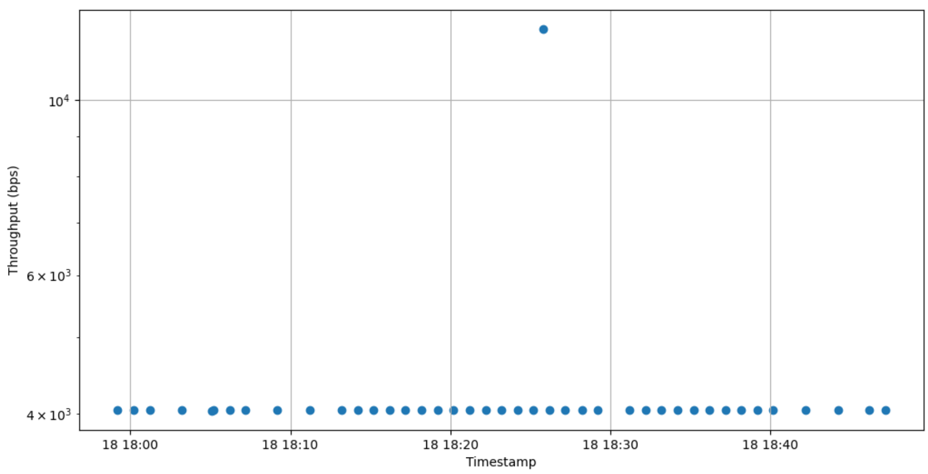
	Timestamp	Iperf server	Namelookup (ms)	Connect Time(ms)	Start Time (ms)	Total Time (ms)	Total Latency	Throughput (bps)
0	2023-10-18 17:53:46	ok1	0.001968	0.004937	0.008937	0.008986	0.002969	2025.521572
1	2023-10-18 17:53:46	sgp1	0.291658	0.566008	0.838793	0.838858	0.274350	17.667595
2	2023-10-18 17:57:01	ok1	0.001431	0.004649	0.008666	0.008714	0.003218	2151.000215
3	2023-10-18 17:57:01	sgp1	0.284361	0.557856	0.830666	0.830726	0.273495	17.925773
4	2023-10-18 17:58:02	ok1	0.003115	0.006856	0.010386	0.010457	0.003741	1458.576429
...	...	...	...	...	...	...	...	...
99	2023-10-18 18:45:01	sgp1	0.305123	0.577878	0.850094	0.850199	0.272755	17.304691
100	2023-10-18 18:46:01	ok1	0.002389	0.005385	0.008788	0.008847	0.002996	1857.010214
101	2023-10-18 18:46:01	sgp1	0.313630	0.581605	0.849370	0.849521	0.267975	17.193800
102	2023-10-18 18:47:01	ok1	0.001548	0.004805	0.008120	0.008168	0.003257	2081.165453
103	2023-10-18 18:47:01	sgp1	0.303954	0.577710	0.850170	0.850298	0.273756	17.309723

104 rows × 8 columns

Figure 7: Observed throughput (and other relevant data) with active iperf3 measurements.



(a) Plot for active measurements' data.



(b) Plot for passive measurements' data.

Figure 8: Time series for iperf measurements.

2.2.3 Compare ping results from active and passive measurements. Provide a table and plot a time series.

In a similar fashion to the previous subsection, for ping measurements, the code used for Assignment 2 has been reused with very few modifications to obtain the delay observed for the active measurements. This time, however, one must observe that the data set includes the delay for research servers and iperf servers, meaning that operations will need to take place in two separate DataFrames. Figure 9 reflects this reality by showing snippets of the two obtained DataFrames. This is in contraposition to the already discussed Figure 5 corresponding to active measurements, which does not reflect this reality. When observing the obtained plots for both measurements (Figure 10), something very interesting can be observed. While passive measurements seem to be able to more or less keep up in reflecting the same trends shown by active measurements; they seem to replicate them an order of magnitude smaller. This can be explained by either or both of the following explanations:

- Passive measurements have involved processing to remove unwanted data. These processing steps can sometimes affect the scale of the measurements.
- Active and passive measurements might not always be perfectly synchronized, leading to discrepancies that ultimately originate in using different clocks.

2.3 Make a table comparing the active and passive measurements

The comparison has been compiled into Table 2. In short, however, active measurements provide for more concrete details regarding individual connections as long as the right tools are used in the right conditions. On the flip side, passive measurement collect broader sets of data. Their use does entail some privacy concerns which should be explored in-depth before using, and produce aggregates of data that generally require more analysis for concrete conclusions to be drawn from.

		Measures	**Passive tools**	***which allow for measuring***	**Problems & Limitations**
Active Measurements	Ping	Latency, Packet Loss	-	-	ICMP traffic prioritization, Firewall blocking
	Iperf	Throughput, Bandwidth	-	-	Server-client setup required, adds extra network traffic
	Traceroute	Network Path, Hop Delays	-	-	Not all routes allow for trace requests
	Speedtest	DL/UL Speed, Latency	-	-	Testing server may not represent the entire network
Passive Measurements	WireShark	Network Traffic Analysis	tshark/pyshark	Traffic Flows, Protocols, ...	Limited details for individual packets, privacy issues
	NetFlow	Network Traffic Flows	SiLK	Flow Records, Hosts, ...	Limited details for individual packets, privacy issues
	PcapNG	Network Packet Capture	tcpdump	Packets, Protocols, ...	Storage minimums, resource intensive, large datasets.

Table 2: Pro-con table for active and passive measurements.

	Timestamp	Iperf server	Packets Transmitted	Packets Received	Packet Loss %	Time (ms)	RTT Min (ms)	RTT Avg (ms)	RTT Max (ms)	RTT Mdev (ms)
0	2023-10-18 17:53:46	ok1	5	5	0.0	4008	3.171	3.691	4.219	0.385
1	2023-10-18 17:53:46	sgp1	5	5	0.0	5275	272.701	273.335	274.046	0.468
2	2023-10-18 17:57:01	ok1	5	5	0.0	4007	2.566	2.995	3.520	0.327
3	2023-10-18 17:57:01	sgp1	5	5	0.0	5132	272.658	273.031	273.444	0.310
4	2023-10-18 17:58:02	ok1	5	5	0.0	4007	2.739	2.911	3.212	0.161
...	...	...	...	...	...	...	...	...	...	...
99	2023-10-18 18:45:01	sgp1	5	5	0.0	6201	272.723	273.101	274.076	0.493
100	2023-10-18 18:46:01	ok1	5	5	0.0	4006	3.313	4.592	6.233	1.004
101	2023-10-18 18:46:01	sgp1	5	5	0.0	5212	272.951	274.287	276.661	1.304
102	2023-10-18 18:47:01	ok1	5	5	0.0	4007	2.219	3.317	5.417	1.094
103	2023-10-18 18:47:01	sgp1	5	5	0.0	5195	272.634	273.345	274.010	0.517

104 rows × 10 columns

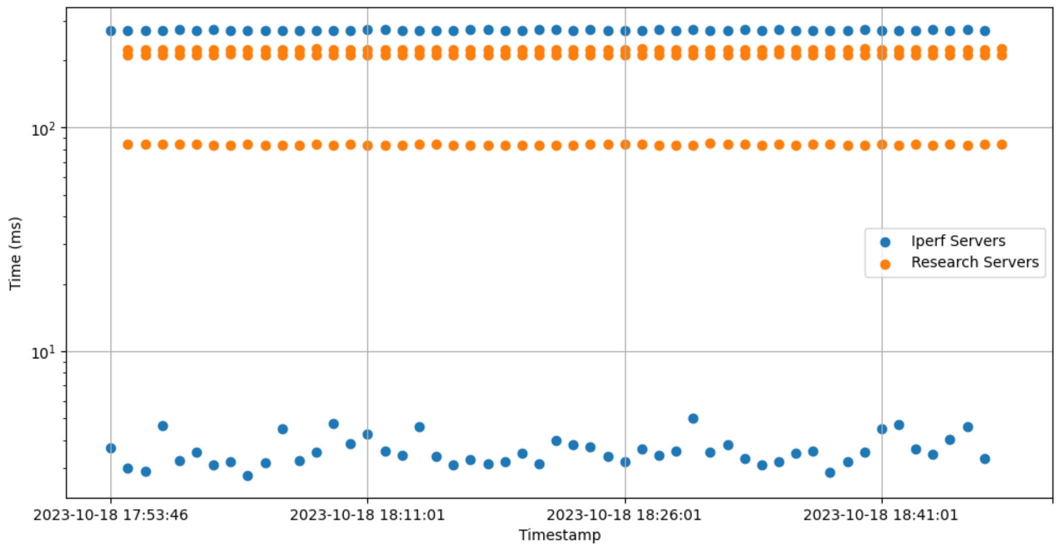
(a) DataFrame corresponding to iperf servers.

	Timestamp	Research server	Packets Transmitted	Packets Received	Packet Loss %	Time (ms)	RTT Min (ms)	RTT Avg (ms)	RTT Max (ms)	RTT Mdev (ms)
0	2023-10-18 17:52:29	bcn-es	5	5	0.0	4006	83.672	84.149	84.620	0.364
1	2023-10-18 17:52:29	mnl-ph	5	5	0.0	17332	210.801	211.141	211.452	0.208
2	2023-10-18 17:52:29	hnl-us	5	5	0.0	5014	222.540	225.525	230.541	2.741
3	2023-10-18 17:57:01	bcn-es	5	5	0.0	4007	83.239	84.763	87.939	1.710
4	2023-10-18 17:57:01	mnl-ph	5	5	0.0	4213	210.881	211.494	213.076	0.799
...	...	...	...	...	...	...	...	...	...	...
151	2023-10-18 18:46:01	mnl-ph	5	5	0.0	4004	211.102	211.728	212.614	0.585
152	2023-10-18 18:46:01	hnl-us	5	5	0.0	4005	222.816	223.015	223.546	0.279
153	2023-10-18 18:47:01	bcn-es	5	5	0.0	4005	83.299	84.118	84.656	0.467
154	2023-10-18 18:47:01	mnl-ph	5	5	0.0	4257	210.858	211.535	212.885	0.706
155	2023-10-18 18:47:01	hnl-us	5	5	0.0	4001	222.867	223.042	223.270	0.174

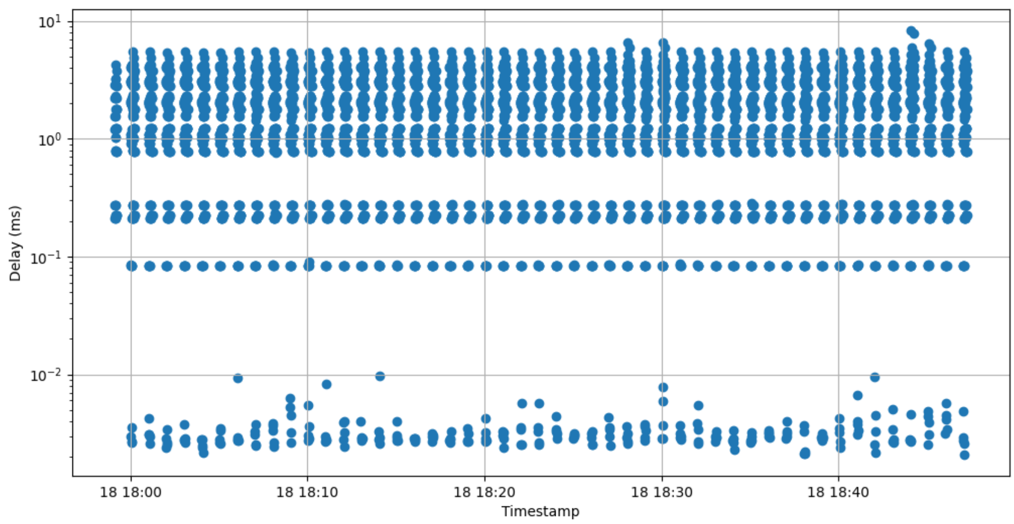
156 rows × 10 columns

(b) DataFrame corresponding to research servers.

Figure 9: Snippets of the obtained DataFrames for ping measurements.



(a) Plot for active measurements' data.



(b) Plot for passive measurements' data.

Figure 10: Time series for ping measurements.