



Aalto-yliopisto
Aalto-universitetet
Aalto University

ELEC-E7130

Internet Traffic Measurement and Analysis

Basic Measurements

Assignment #2

Aitor Urruticoechea Puig

aitor.urruticoecheapuig@aalto.fi

Student N°101444219

September 2023

Contents

1 Measurement metrics	3
1.1 Explain the concept of network packet loss, its definition, calculation method, and its impact on network performance.	3
1.2 Compare the concepts and characteristics of mean and median, and discuss their advantages and disadvantages.	3
1.3 What is network stability? How can network stability be measured using various indicators or metrics?	3
2 Measuring Latency	4
2.1 Choose 3 name servers, 3 research servers and 2 iperf servers based on instructions. Take measurements for at least 24 hours.	4
2.2 Report Table	6
2.3 Conclusions	8
3 Measuring Throughput	9
3.1 Take measurements for at least 24 hours	9
3.2 Basic Statistics	10
3.3 Conclusions	11
References	12

List of Figures

1	Infographics on the assigned overseas territory, extracted from Wikimedia.	4
2	Sample of nameservers latency data obtained using dig method.	6
3	Sample of research latency data obtained using ping method.	7
4	Sample of Iperf latency data obtained using curl method.	7
5	Snippet of the DataFrame obtained from Method 1.	10
6	Snippet of the DataFrame obtained from Method 2.	10
7	DataFrame obtained from Method 1.	11
8	Throughput results for server ok1, using iperf3.	12
9	Throughput results for server sgp1, using iperf3.	12

List of Tables

1	Latency data after a 28h period of data collection of the selected servers.	7
2	Throughput measurements results, all in Mb/s	11

Acronyms

DL Download. 9–11

DNS Domain Name System. 8

HTTP Hypertext Transfer Protocol. 8, 9, 11

ICMP Internet Control Message Protocol. 8

std. Standard Deviation. 10, 11

TCP Transmission Control Protocol. 3

UL Upload. 9–11

WSL Windows Subsystem Linux. 5, 9

Task 1: Measurement metrics

1.1 Explain the concept of network packet loss, its definition, calculation method, and its impact on network performance.

Packet loss is a phenomenon that may happen when transmitting packets of data through a network. If some or all of them fail to reach their destination, packet loss is happening. It is typically quantified using a percentage $(100 \cdot \frac{\text{packets lost}}{\text{packets sent}})$.

This phenomenon is detected by the Transmission Control Protocol (TCP) using one or several of its dedicated algorithms to this task (RTO, RACK...) which allow it to perform a retransmission of the packets to ensure continuous communications. These retransmissions can have mounting effects on overall network performance, starting at slower internet speeds due to having to resend the same packages, and escalating from here on. Connection interruptions can happen with high packet loss, which can necessarily result in a degradation in application performance. In the worst of cases, complete data loss and thus shutdown of communication can happen if the rate of packet loss is such that the packets that are transmitted are just insufficient on their own, rendering them unusable.

1.2 Compare the concepts and characteristics of mean and median, and discuss their advantages and disadvantages.

While mean and median are both common measures of a general tendency, the typical value if you wish, a given data set might have; they are widely different by concept and definition. On one hand, the mean is obtained by dividing the sum of all the points in the sample by the number of points the sample has. The mean is thus highly sensitive to each point of the data set, while being easy to compute. Nevertheless, the mean can be, in return, misleading for non-normal data distributions. The fact that it is so sensitive to every point makes it quite vulnerable to outliers, that can push the mean to take a value far from the general tendency the rest data set has.

On the other hand, the median is calculated by arranging the data in an ascending or descending order. The mean will be the middle value of this ordered data sequence (or the average of the two middle values if the sequence has an even number of values). This definition points clearly to one of its clear advantages over the mean: it makes it quite resistant to outliers, for the influence they have in this value is limited by the orderly fashion of its calculation. It can be, by the same principle, useful as an indication of a tendency even for non-normally distributed data sets. However, it is important to note that ordering the values of a data set can be more computationally challenging than summing them; while its robustness sacrifices representativity for every point — something that the mean achieves better.

1.3 What is network stability? How can network stability be measured using various indicators or metrics?

Network stability is a broad concept that refers to the ability a network has to remain stable when the unexpected happens. It is achieved to a certain degree of success by a series of processes, combined with physical infrastructure, that makes sure that every necessary device between the terminal and the server are working properly and as intended.

Key metrics or indicators to keep into account when discussing network stability include (but are not limited to):

- Latency, or delay in the network.
- Packet Loss, or which percentage packets of information actually do not arrive at their destination.
- Throughput, or the amount of data that is successfully transmitted in a given amount of time.
- Bandwidth, or the maximum possible data that can be transmitted in a given amount of time through a concrete path in the network.

Task 2: Measuring Latency

2.1 Choose 3 name servers, 3 research servers and 2 iperf servers based on instructions. Take measurements for at least 24 hours.

By running the required script, the *country* that has been assigned is the French Southern Territories (also known as French Southern and Antarctic Lands); which is not technically a country according to most standards, for it is a French overseas territory. It consists of the Adélie Land (the French claim on Antarctica), and the Corzet, Kerguelen, Scattered, Saint Paul, and Amsterdam Islands. None of them have no permanently settled inhabitants, and are instead home to a set of research and military stations [1]. Its exact geography and flag can be seen in Figure 1.



(a) French Southern Territories map.



(b) French Southern Territories flag.

Figure 1: Infographics on the assigned overseas territory, extracted from Wikimedia.

The chosen “nameservers” from the assigned ones to this territory are g.ext.nic.fr, e.ext.nic.fr, and f.ext.nic.fr, which will in turn will work with the official government webpage of the territory, <https://taaf.fr/> when using the dig command.

The “research servers” that have been assigned following the instructions are bcn-es.ark.caida.org, mnl-ph.ark.caida.org, and hnl-us.ark.caida.org; located in Barcelona, Manila, and Honolulu respectively.

Finally, the “lperf servers” to work with are ok1.iperf.comnet-student.eu and sgp1.iperf.comnet-student.eu, which will be referred to as ok1 and sgp1 respectively from now on, for simplicity's sake.

Recursive requests for information will be performed at the 59th minute for hourly requests, and at every minute ending in 9 for 10-minute frequent requests, following the guidelines specified. In order to set up the periodic requests for information, three scripts have been designed and implemented, one for each type of server. They are quite simple scripts that will generate log files with data that has seen very little processing; for the heavy lifting in that sense will be done in Python, a language the author is more proficient in.

Nameservers measurements

The script that is to be hourly executed to measure the connection to the assigned nameservers uses the `ping` command for basic measurements, but also the provided `http - dig` file to get a different set of results. `ping` has been implemented with the `-c 5 -O -D` options, while `http - dig` has been implemented as is. To make a very rough clean-up of the data, `fgrep` has been used for both commands, keeping only the lines where “packets” or “rtt” is mentioned for `ping`, and the lines where “curl” or “query” is mentioned for `http - dig`.

Research servers measurements

The same setup deployed for the `ping` measurements for the nameservers has been used for the research ones, resulting in a very similarly looking `.sh` file and `.log` of results.

lperf servers measurements

Again, the `ping` measurements have been implemented in the same way as in the other two cases. Additionally, the `curl` command has also been used to retrieve key data. For this one, `fgrep` is not needed. Instead, the desired fields (time namelookup, time connect, time starttransfer, time total) are specified when calling the command.

In all cases, the server name is printed before its measurements, and the timestamp is printed at the very beginning of the execution of every `.sh` file. All in all, the crontab setting for these measurements takes this form:

```
1 59 * * * * /bin/bash /home/aurruti/Aalto/InternetTraffic/HW2/nameservers_latency.sh >>
   ↪ /home/aurruti/Aalto/InternetTraffic/HW2/latency-log/nameservers.log
2 9,19,29,39,49,59 * * * * /bin/bash
   ↪ /home/aurruti/Aalto/InternetTraffic/HW2/research_latency.sh >>
   ↪ /home/aurruti/Aalto/InternetTraffic/HW2/latency-log/research.log
3 9,19,29,39,49,59 * * * * /bin/bash
   ↪ /home/aurruti/Aalto/InternetTraffic/HW2/iperf_latency.sh >>
   ↪ /home/aurruti/Aalto/InternetTraffic/HW2/latency-log/iperf.log
```

This has been executed in a desktop computer executing from Windows Subsystem Linux (WSL). The computer is located in Leppavara, Espoo; permanently connected to the internet via Wi-Fi provided by DNA.

2.2 Report Table

It shall include the following metrics for each of your target servers, including the results obtained using all methods (ones for method 1 and others for method 2).

- Median delay with lost packets with delay of infinity, thus if more than 50% of packets are lost, then consider as infinity.
- Mean delay.
- Loss ratio
- Delay spread as the difference with 75th and 25th percentiles. If more than 25% of packets is lost, then consider as infinity.

After letting the measurements run for roughly 28 hours, they have been analysed using Python and Pandas. Due to the structure of the log files, their conversion into a DataFrame has proven not-trivial at some points, something that in hindsight should have been predicted and taken into account better when generating these log files. Namely, the fact that every line has a different length, with lines with only one parameter (the timestamp or the server name) and lines with all the ping data in them. Nevertheless, the challenge is not incomprehensibly hard, and has been solved with some extra lines of code, which can be consulted in the code files related to this very same assignment. Some sample results after data treatment can be seen in Figures 2, 3, and 4.

In contrast with the first assignment, an array of functions included in the Pandas library have been implemented, which help computationally and simplify the calculation process for the basic statistical data that is to be obtained. The final results of such data can be seen in Table 1.

	Timestamp	Nameserver	Time Start Transfer	Time Pre Transfer	Time Connect	Total Latency
0	2023-10-01 12:59:01	e.ext.nic.fr	15.185816	0.101080	0.007113	15.286896
1	2023-10-01 12:59:01	g.ext.nic.fr	15.158561	0.096054	0.013725	15.254615
2	2023-10-01 12:59:01	f.ext.nic.fr	15.152546	0.091008	0.005340	15.243554
3	2023-10-01 13:59:01	e.ext.nic.fr	0.151087	0.082987	0.004795	0.234074
4	2023-10-01 13:59:01	g.ext.nic.fr	0.172418	0.108303	0.004754	0.280721
...
76	2023-10-02 13:59:01	g.ext.nic.fr	15.145637	0.077215	0.004094	15.222852
77	2023-10-02 13:59:01	f.ext.nic.fr	15.143426	0.079045	0.004423	15.222471
78	2023-10-02 14:59:01	e.ext.nic.fr	15.187591	0.099501	0.016740	15.287092
79	2023-10-02 14:59:01	g.ext.nic.fr	15.132162	0.079596	0.005161	15.211758
80	2023-10-02 14:59:01	f.ext.nic.fr	15.124099	0.077313	0.004862	15.201412

81 rows × 6 columns

Figure 2: Sample of nameservers latency data obtained using dig method.

nameservers_digdata
✓ 0.0s

	Timestamp	Nameserver	Time Start Transfer	Time Pre Transfer	Time Connect	Total Latency
0	2023-10-01 12:59:01	e.ext.nic.fr	15.185816	0.101080	0.007113	15.286896
1	2023-10-01 12:59:01	g.ext.nic.fr	15.158561	0.096054	0.013725	15.254615
2	2023-10-01 12:59:01	f.ext.nic.fr	15.152546	0.091008	0.005340	15.243554
3	2023-10-01 13:59:01	e.ext.nic.fr	0.151087	0.082987	0.004795	0.234074
4	2023-10-01 13:59:01	g.ext.nic.fr	0.172418	0.108303	0.004754	0.280721
...
76	2023-10-02 13:59:01	g.ext.nic.fr	15.145637	0.077215	0.004094	15.222852
77	2023-10-02 13:59:01	f.ext.nic.fr	15.143426	0.079045	0.004423	15.222471
78	2023-10-02 14:59:01	e.ext.nic.fr	15.187591	0.099501	0.016740	15.287092
79	2023-10-02 14:59:01	g.ext.nic.fr	15.132162	0.079596	0.005161	15.211758
80	2023-10-02 14:59:01	f.ext.nic.fr	15.124099	0.077313	0.004862	15.201412

81 rows × 6 columns

Figure 3: Sample of research latency data obtained using ping method.

nameservers_digdata
✓ 0.0s

	Timestamp	Nameserver	Time Start Transfer	Time Pre Transfer	Time Connect	Total Latency
0	2023-10-01 12:59:01	e.ext.nic.fr	15.185816	0.101080	0.007113	15.286896
1	2023-10-01 12:59:01	g.ext.nic.fr	15.158561	0.096054	0.013725	15.254615
2	2023-10-01 12:59:01	f.ext.nic.fr	15.152546	0.091008	0.005340	15.243554
3	2023-10-01 13:59:01	e.ext.nic.fr	0.151087	0.082987	0.004795	0.234074
4	2023-10-01 13:59:01	g.ext.nic.fr	0.172418	0.108303	0.004754	0.280721
...
76	2023-10-02 13:59:01	g.ext.nic.fr	15.145637	0.077215	0.004094	15.222852
77	2023-10-02 13:59:01	f.ext.nic.fr	15.143426	0.079045	0.004423	15.222471
78	2023-10-02 14:59:01	e.ext.nic.fr	15.187591	0.099501	0.016740	15.287092
79	2023-10-02 14:59:01	g.ext.nic.fr	15.132162	0.079596	0.005161	15.211758
80	2023-10-02 14:59:01	f.ext.nic.fr	15.124099	0.077313	0.004862	15.201412

81 rows × 6 columns

Figure 4: Sample of Iperf latency data obtained using curl method.

Type	Server	Method	Median delay	Mean delay	Loss ratio [%]	Delay spread
name server 1	e.ext.nic.fr	ping	33, 158	33, 4997	3, 8462	1, 9650
name server 2	f.ext.nic.fr	ping	4, 043	4, 5491	0, 0000	2, 5865
name server 3	g.ext.nic.fr	ping	34, 247	37, 0190	0, 7463	1, 9715
name server 1	e.ext.nic.fr	dig	0, 287	6, 6254	3, 7037	15, 0159
name server 2	f.ext.nic.fr	dig	15, 247	13, 4396	0, 0000	0, 0360
name server 3	g.ext.nic.fr	dig	3, 421	7, 3150	0, 0000	14, 9692
research server 1	bcn-es	ping	78, 465	79, 2328	0, 1292	10, 5895
research server 2	hnl-us	ping	220, 880	221, 4396	4, 0269	1, 2200
research server 3	mnl-ph	ping	291, 769	292, 0801	0, 2587	0, 7250
iperf server 1	ok1	ping	3, 435	3, 6305	0, 9115	0, 6505
iperf server 2	sgp1	ping	278, 096	278, 2798	0, 2587	0, 4455
iperf server 1	ok1	curl	0, 0032	0, 0037	0, 6452	0, 0008
iperf server 2	sgp1	curl	0, 2742	0, 2641	0, 0000	0, 0083

Table 1: Latency data after a 28h period of data collection of the selected servers.

2.3 Conclusions

Finally, make conclusions about stability of network delay. Were some of the hosts different from the others? Could you observe any daytime variations? Do the timezones where target servers (or you) have an impact?

Overall, both the desktop computer used for measurements and the target servers seem to have quite stable network connections. Overall, there is very little packet loss, which can mean that both the origin and destination servers have overall reliable connections.

Regarding nameservers, there is something very interesting happening: while when using `ping e.ext.nic.fr` and `g.ext.nic.fr` end up accumulating more latency, the inverse happens when using `dig`, and `f.ext.nic.fr` is the one with more latency. This is surprising, but not unexplainable. They are methods that work on different levels: Internet Control Message Protocol (ICMP) and Domain Name System (DNS) respectively. Due to this, they might take different routes to their destinations, and be vulnerable to different things. For instance, `dig` can benefit from DNS cache keeping data in some cases and speeding up communications. Alternatively, different servers may prioritize different communication protocols, which could also explain this observed behaviour. The Delay Spread observed is also very different, with it being quite larger when using `dig`, which probably come from the inherently more complex workings of DNS queries. Having to perform multiple jumps and interacting with different cached data can result in widely varying data, specially when compared to a simpler ICMP protocol.

For research servers, such comparison is not possible, for only `ping` is used. Here, instead, it is a lot more interesting to see how the location of the targets easily explains the results obtained. The research servers located in Manila and Honolulu, being at the antipodes of the origin desktop computer, have a substantially larger latency; while the one located in Barcelona has less than half the latency, very probably due to it being in the same continent. Interestingly, though, those larger latencies are quite more stable, with less delay spread than the one observed for the server located in Barcelona, which could be explained by potentially higher fluctuations in server loads. Alternatively, it could come to a higher number of route changes that can happen through a denser network, such as the one existing in continental Europe. Such route changes would be more limited for the undersea cables that connect the servers in Manila and Honolulu.

Finally, for the `iperf` servers, the key difference seems to be distance again, for the “far away” server `sgp1` higher latency is observed when compared to the “close” server `ok1`; both when using `ping` and `curl`. The staggering difference, though, is the one observed between using `ping` and `curl` which can probably be explained due to the intrinsic differences in the workings of the two commands. The fact that `curl` is sending and Hypertext Transfer Protocol (HTTP) request against an ICMP packet can result in very different latency measurements, as well as which parts of the HTTP request are actually being accounted for latency following the instructions given. Nevertheless, it is interesting to see that, despite the difference in order of magnitude, the very same tendency and two-zeros difference is observed in the behaviour of the `iperf` servers.

Task 3: Measuring Throughput

3.1 Take measurements for at least 24 hours

Start measurement and collect data for at least 24 hours using a shell script and crontab, and describe your measurement setup in the report.

Following the given instructions, the iperf servers to be analysed are, again ok1.iperf.comnet-student.eu and sgp1.iperf.comnet-student.eu, which again are to be referred to as ok1 and sgp1 and to be contacted hourly on the 59th minute. In the same way as the previous task, the actual shell scripts are quite simple, leaving the cleaning of the data for the Python part.

Method 1: curl

This method is quite straight forward: the curl command has been implemented, storing the resulting HTTP code and Upload (UL) and Download (DL) speeds. Note that the HTTP code is interesting in case of error, for this allows for identification of such errors. At the very beginning of each hourly measurement, the timestamp is applied similarly to the previous task.

Method 2: iperf3

For the iperf3 method, the required package has had to be installed in WSL. In the .sh script, the way the data is stored is, again, as straight-forwards as it gets. For each iperf server, the iperf3 command is executed twice, one for each direction, and taking into account that the destination port has to be chosen at random as specified in the instructions. All the parameters and how the obtained data is saved in .json format can be more easily seen from the provided code snippet:

```
1 iperf3 -c ok1.iperf.comnet-student.eu -p $((RANDOM % 11 + 5200)) -J >
  ↪ /home/aurruti/Aalto/InternetTraffic/HW2/throughput-log/ok1N$(date +%s).json
2 iperf3 -c ok1.iperf.comnet-student.eu -p $((RANDOM % 11 + 5200)) -R -J >
  ↪ /home/aurruti/Aalto/InternetTraffic/HW2/throughput-log/ok1R$(date +%s).json
3 iperf3 -c sgp1.iperf.comnet-student.eu -p $((RANDOM % 11 + 5200)) -J >
  ↪ /home/aurruti/Aalto/InternetTraffic/HW2/throughput-log/sgp1N$(date +%s).json
4 iperf3 -c sgp1.iperf.comnet-student.eu -p $((RANDOM % 11 + 5200)) -R -J >
  ↪ /home/aurruti/Aalto/InternetTraffic/HW2/throughput-log/sgp1R$(date +%s).json
```

Method 3: speed test

For this method, three manual internet throughput tests have been performed using a popular online tool [2]. These measurements have been taken at three notably different times of the day in order to take into account possible variances in connectivity quality due to network availability.

Again, all three methods have been executed from a desktop computer's WSL located in Leppavara, Espoo; and with internet services provided via Wi-Fi by DNA. The crontab lines corresponding to the periodic execution of method 1 and 2 are as follows:

```
1 59 * * * * /bin/bash /home/aurruti/Aalto/InternetTraffic/HW2/throughput1.sh
2 59 * * * * /bin/bash /home/aurruti/Aalto/InternetTraffic/HW2/throughput2.sh
```

3.2 Basic Statistics

Make a table where your results from these 3 methods and calculate basic statistics, such as mean, median, max, min and average deviation (mean absolute deviation). Note that for methods 2 and 3 you will receive UL and DL readings. Report them separately.

Once again, the data has been processed using Python and Pandas, now with Pandas integrated functions - unlike the previous assignment. While the processing of method 1 has proven equally challenging to task 2, method 2 and 3 have actually been more direct, due to the simplicity of reading a .json file or a .txt one. Notably, the exception due to a failed connection has had to be included in the reading of method 2, for this results in a mostly empty .json file. However, the fact that the timestamp is included in the file's name making it possible to include that 0 bps connection into the wider DataFrame. Some snippets of the resulting DataFrames can be consulted as examples of the obtained data in Figures 5, 6, and 7; while the results after the analysis of the data has been performed are available in Table 2, including minimums, maximums, mean and median, and Standard Deviation (std.).

method1data				
✓	0.0s			
	Timestamp	Server	Http Code	Download Speed (bps)
0	2023-10-01 10:59:01	ok1	200	21427146
1	2023-10-01 10:59:01	spg1	200	2313931
2	2023-10-01 11:59:01	ok1	200	6607329
3	2023-10-01 11:59:01	spg1	200	587263
4	2023-10-01 12:59:01	ok1	0	0
5	2023-10-01 12:59:01	spg1	0	0
6	2023-10-01 13:59:01	ok1	200	9686238
7	2023-10-01 13:59:01	spg1	200	2595451

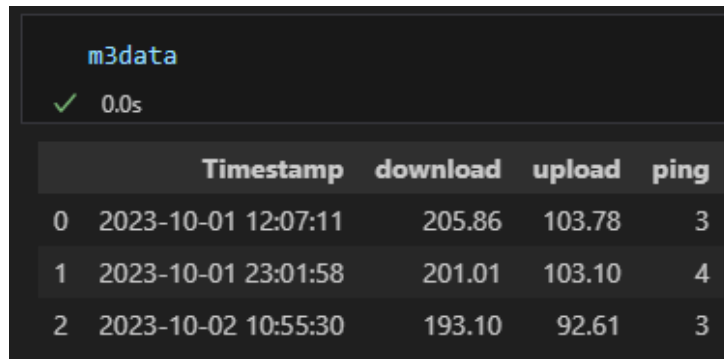
Figure 5: Snippet of the DataFrame obtained from Method 1.

ok1Ndata			
✓	0.0s		
	Timestamp	Sender_bps	Receiver_bps
0	2023-10-01 10:59:01	9.425578e+07	9.342369e+07
1	2023-10-01 11:59:01	9.447507e+07	9.351662e+07
2	2023-10-01 12:59:01	9.550994e+07	9.508848e+07
3	2023-10-01 13:59:01	9.551697e+07	9.417785e+07
4	2023-10-01 14:59:01	9.186577e+07	9.103340e+07
5	2023-10-01 15:59:01	9.447058e+07	9.286788e+07
6	2023-10-01 16:59:01	9.262769e+07	9.179849e+07
7	2023-10-01 17:59:01	9.316878e+07	9.272477e+07
8	2023-10-01 18:59:01	9.410431e+07	9.316105e+07
9	2023-10-01 19:59:01	9.482194e+07	9.344635e+07
10	2023-10-01 20:59:01	9.395762e+07	9.308215e+07

spg1Rdata			
✓	0.0s		
	Timestamp	Sender_bps	Receiver_bps
0	2023-10-01 10:59:35	3.397802e+07	3.236945e+07
1	2023-10-01 11:59:26	2.475915e+07	2.263881e+07
2	2023-10-01 12:59:56	2.397403e+07	2.264898e+07
3	2023-10-01 13:59:35	2.693522e+07	2.526887e+07
4	2023-10-01 14:59:35	2.631637e+07	2.473370e+07
5	2023-10-01 15:59:34	2.020253e+07	1.932860e+07
6	2023-10-01 16:59:34	2.980249e+07	2.788717e+07
7	2023-10-01 17:59:50	3.283616e+07	3.129807e+07
8	2023-10-01 18:59:35	3.010865e+07	2.846124e+07
9	2023-10-01 19:59:35	3.458684e+07	3.357534e+07
10	2023-10-01 20:59:35	2.480921e+07	2.254886e+07

(a) Connecting to server ok1 in normal operations. (b) Connecting to server spg1 in reverse operations.

Figure 6: Snippet of the DataFrame obtained from Method 2.



```
m3data
✓ 0.0s
```

	Timestamp	download	upload	ping
0	2023-10-01 12:07:11	205.86	103.78	3
1	2023-10-01 23:01:58	201.01	103.10	4
2	2023-10-02 10:55:30	193.10	92.61	3

Figure 7: DataFrame obtained from Method 1.

Key	Min.	Min. (> 0)	Max.	Mean	Median	std.
HTTP ok1	0, 00	6, 42	22, 52	15, 14	17, 94	6, 5146
HTTP sgp1	0, 00	0, 44	2, 60	1, 89	2, 18	0, 7363
Iperf UL ok1	0, 00	91, 87	96, 08	90, 63	94, 31	18, 5158
Iperf DL ok1	0, 00	183, 98	205, 70	195, 43	204, 61	40, 0969
Iperf UL sgp1	0, 45	0, 45	21, 00	9, 59	10, 76	6, 6084
Iperf DL sgp1	20, 20	20, 20	47, 83	29, 73	27, 65	6, 0361
Speed Test UL	92, 61	92, 61	103, 78	99, 83	103, 1	6, 2619
Speed Test DL	191, 1	191, 1	205, 86	199, 99	201, 01	6, 4409

Table 2: Throughput measurements results, all in Mb/s

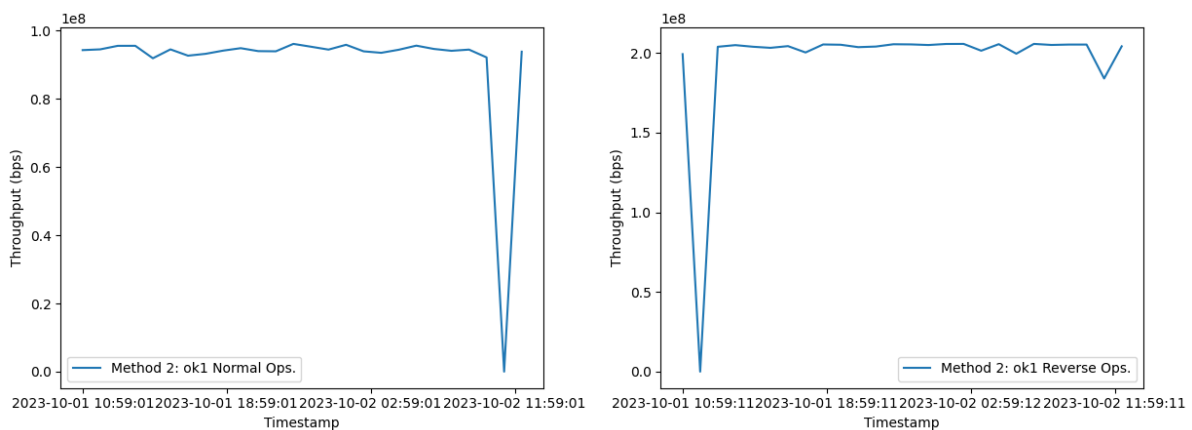
3.3 Conclusions

Make conclusions about the methods of network throughput answering for at least the following topics. It may be beneficial to graph results to identify some trends.

- Are the results between methods in line with each other?
- Did some method have a lot of deviation? What do you think might cause this?
- Was there some method that gives higher values than others? What do you think might cause this?
- Did you observe any variation in throughput based on the time of day? For example, did you get higher throughput during the day or night?
- Were there any anomalies observed? For example, no connection or very different capacity.

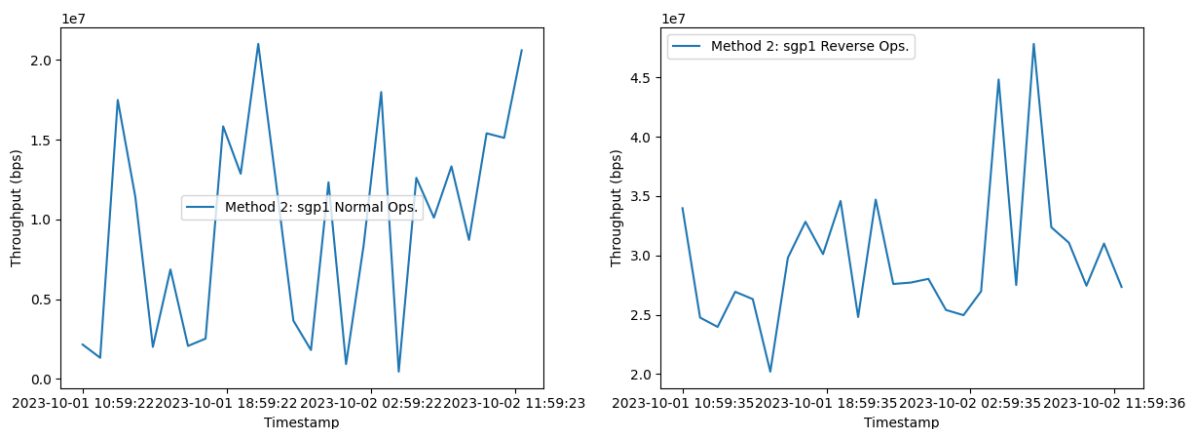
From Method 1 and Method 2, it is clear that, overall, ok1 server has a higher quality connection from throughput to latency. In this case, it achieves consistently almost an order of magnitude more Mb/s when handling HTTP requests; and more than double the throughput, both in UL and DL when handling the iperf requests. There seems to have been some moments with a loss of connection, which could help explain the high variance values and the zero minimum (this latter is notably nowhere close to the non-zero minimum). Despite this fact, it remains true that the iperf3 method seems to display more variance, even more than the manual speed tests. This might be due to server handling worse the iperf3 requests or due to server load. Geography, on the flip side, seems to again be the reason for worse throughput when dealing with sgp1.

Alternatively, hourly fluctuation does not seem to play a particularly important role in the measurements for ok1 with the important exception of the moment where connection to the servers was not possible (see Figure 8, which includes for illustrative purposes the results of method 2 - note that method 1 yields similar data on its own scale). This is nothing comparable to the results obtained when connecting with sgp1, that instead vary wildly with the time of the day (see Figure 9). This server seems to support better reverse operations during the early morning, and worse ones in the afternoon and evening, where it drops to almost a third of its maximum throughput. For normal operations, however, it becomes a lot harder to draw any conclusions: its operations seem quite erratic, and its availability seems to hardly depend on the time of the day. Note that this cannot be really attributed to the connection of the desktop computer from which measurements have been made: ok1 seems to not have had this kind of problems, and the results from method 3 corroborate that there is little variance in the connection quality no matter the time of the day.



(a) Connecting to server ok1 in normal operations. (b) Connecting to server ok1 in reverse operations.

Figure 8: Throughput results for server ok1, using iperf3.



(a) Connecting to server sgp1 in normal operations. (b) Connecting to server sgp1 in reverse operations.

Figure 9: Throughput results for server sgp1, using iperf3.

References

1. *Présentation des Territoires* [online]. TAAR, 2023 [visited on 2023-10-01]. Available from: <https://taaf.fr/collectivites/presentation-des-territoires/>.
2. *Speedtest* [online]. Ookla, 2006-2023 [visited on 2023-10-01]. Available from: <https://www.speedtest.net/>.