Aalto-yliopisto
Aalto-universitetet
Aalto University

ELEC-E7130

Internet Traffic Measurement and Analysis

User Traffic

Assignment #3

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

Contents

List of Figures

List of Tables

Task 1: Introduction to the Traffic Data

1.1 What is the passive measurement in terms of network traffic? What kind of information does it provide, and what is its role or significance?

The idea behind passive measurement is to monitor the network's traffic without actively partaking in such traffic. This is to mean that, unlike methods like `ping`, passive measurements will not generate its own traffic to quantify anything. Instead, a passive measurement will focus in observing the natural flow of traffic through a network as generated by the users and other agents involved in the network. Since passive measurements, by definition, are used to analyse the natural flows of a network, the insights it provides are key to network managing and optimization; for it will provide information in traffic patterns, network utilisation, protocol prevalence, latency/delay/packet loss, etc. Significantly, these insights empower network administrators to make informed decisions that can lead to more efficient network management and optimization. By understanding traffic patterns, they can allocate resources more effectively, ensuring that critical applications receive the necessary bandwidth while preventing congestion in non-essential traffic, among others. Furthermore, passive measurements aid in troubleshooting network issues. When latency, packet loss, or abnormal traffic behaviour is detected, administrators can quickly pinpoint the source of the problem and take corrective actions.

1.2 Please provide an explanation of the concepts of packet capture and flow data. What kind of information they can provide? Additionally, discuss the advantages, disadvantages, and importance of both packet capture and flow data in network analysis.

Packet capture is a process where individual packets flowing thorough a network are inspected individually. Packet capture tools, such as Wireshark, capture packets to examine their content, source, destination, protocols, etc. **Flow of data**, alternatively, refers to the movement of information within a network. Depending on the protocols used to move data around a network, it will flow differently through switches, routers, and other network equipment. Typical tools to get such insights are NetFlow or sFlow.

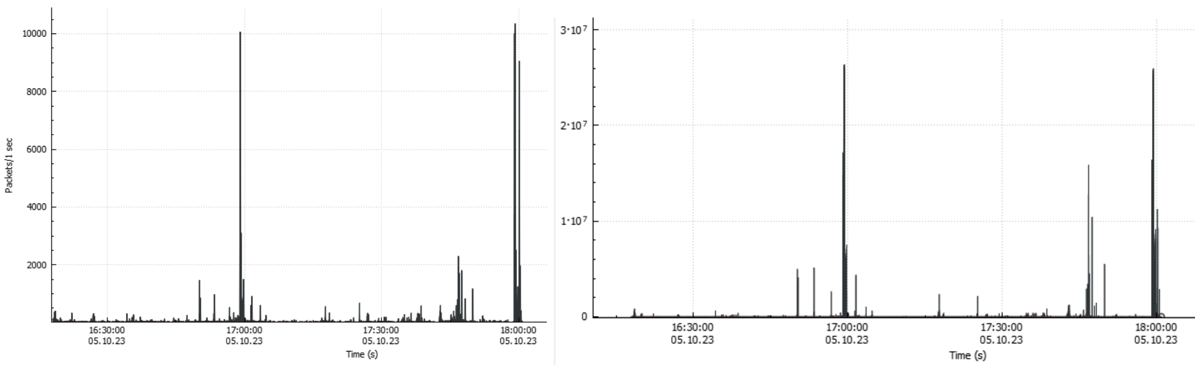
Packet capture, by itself, provides unparalleled granularity: individually capturing packets offers a depth and nuance to the investigation that flow of data analysis can simply not match. However, the amounts of data it tends to generate are too often times more cumbersome to deal with, as well as the justified privacy concerns that come from packet sniffed. On the flip side, flow of data analysis comes as more resource-friendly and thus scalable. What lacks in content detail, it gains in ease of implementation when the goal is to have an overall view at the network level, with enough data for network monitoring, prioritization, and anomaly detection.

1.3 What is hashing? How does the hash algorithm work, and what is the relation with the memory management in the large data analysis?

Hashing is a cryptographic mechanism that, by applying an algorithm to an input "message", generates a fixed-length string as an output (typically a hexadecimal number). Only by having access to the correct hash table, the original message can be decrypted from its output, making it a very efficient while secure method of storing critical data like password storage. For large scale data analysis, memory becomes a critical resource, and effective memory management strategies are essential to ensure the analysis can be conducted. Here, hashing becomes very useful to store and manage large datasets in memory. By applying a hash function to the data's keys, hash tables ensure rapid indexing and retrieval of information.

Task 2: Analyse Flow Data

In order to analyse flow of data from the home computer, WireShark has been used [1]. The computer used is a desktop pc located in Leppavara, Espoo; and connected to the internet via Wi-Fi provided by DNA. The timeframe of the analysis was of almost two hours, the evening of Thursday, October 5th (from 16:17 to 18:00). During that time, the computer was used normally: watching some online streaming, doing homework, and the like. After the analysis was complete, the data was easily stored as a .pcap file using the same program, and can actually be plotted for a general overview of the data obtained (see Figure 1).



(a) Packets per second, against time.

(b) Bytes, against time.

Figure 1: Preliminary plots by WireShark.

Once the measurements are done, the generated .pcap file needs to be processed first, before being analysed, for it alone is over 1 Gb. It has been uploaded to the provided university server `lyta.aalto.fi` using the scp command. Once there, CoralReef has been implemented as recommended to transform the .pcap file into data flows.

```
urrutia1@lyta:/var/tmp$ ./work/courses/unix/T/ELEC/E7130/general/use.sh
urrutia1@lyta:/var/tmp$ crl_flow -I -Ci=172800 -cl -Tf60 -o output-all-ended.t2 -Cai=1
↪  urrutia1-as2.pcap
urrutia1@lyta:/var/tmp$ tcpdump -r urrutia1-as2.pcap -nv > tcpdump-as2.csv
```

This set of commands ultimately generate two files: output − all − ended.t2 and tcpdump − as2.csv. Both of them are significantly smaller than the original file (by an order of magnitude at least); which makes them now operable using Python and Pandas, the environment the author is more used to. The exact code to transform these into DataFrame can be found in the attached code files, but as an example, the first file results in the DataFrame depicted in Figure 2.

	src	dst	pro	ok	sport	dport	pkts	bytes	flows	first	latest	protocol
0	192.168.1.62	224.0.0.252	17	1	52822	5355	2	118	1	1.696514e+09	1.696514e+09	IPv4
1	192.168.1.62	224.0.0.252	17	1	52821	5355	2	118	1	1.696512e+09	1.696512e+09	IPv4
2	178.79.212.129	192.168.1.177	6	1	80	56577	2	80	1	1.696514e+09	1.696514e+09	IPv4
3	192.168.1.62	224.0.0.252	17	1	52816	5355	2	118	1	1.696513e+09	1.696513e+09	IPv4
4	216.58.211.234	192.168.1.177	6	1	443	57338	11	5776	1	1.696517e+09	1.696517e+09	IPv4
...	...	...	...	...	...	...	...	...	...	...	...	...
16973	fe80::3c64:85d8:f884:544c	ff02::16	58	1	143	0	5	700	1	1.696512e+09	1.696512e+09	IPv6
16974	fe80::7e77:16ff:fea8:cf22	ff02::1	58	1	130	0	1	76	1	1.696512e+09	1.696512e+09	IPv6
16975	fe80::3c64:85d8:f884:544c	fe80::7e77:16ff:fea8:cf20	58	1	136	0	1	72	1	1.696512e+09	1.696512e+09	IPv6
16976	fe80::7e77:16ff:fea8:cf20	fe80::3c64:85d8:f884:544c	58	1	135	0	1	72	1	1.696512e+09	1.696512e+09	IPv6
16977	fe80::7e77:16ff:fea8:cf22	ff02::1	58	1	130	0	1	76	1	1.696512e+09	1.696512e+09	IPv6

16978 rows × 12 columns

Figure 2: Snippet of the DataFrame with the obtained data flows.

2.1 Provide basic statistics of flow data

Provide basic statistics of flow data, including: total number of flows, minimum, median, mean, and maximum flow sizes in bytes and packets.

16978 flows have been recorded during the analysed period. The relevant statistical data can be easily extracted from the obtained DataFrame using the integrated pandas functions. These obtained results can be consulted in Table 1.

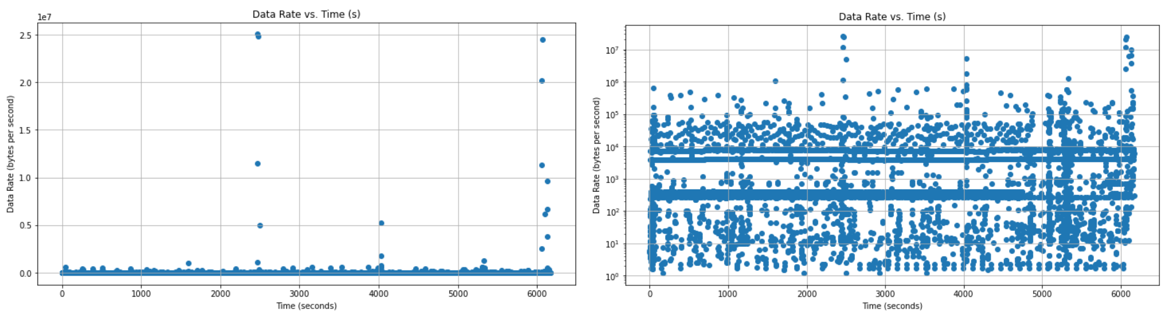
	Packets	Bytes
Minimum	1	32
Maximum	78.808	251.851.140
Median	2	134, 5
Mean	26, 6559	64.686, 1778

Table 1: Relevant statistical data from analysed data flows.

2.2 Plot the traffic volume (bytes) of the flow data file

Note: Getting traffic volume is more difficult from flow data files due to the known information are only start time, end time, and flow size (bytes). For example, if the flow contains 100,000 bytes starting at 3.4 and ending at 7.8, we can calculate that about 20,000 bytes for each second. See more information in Network capture tutorial (Traffic volume in certain interval, pp. 14).

After the necessary operations have been done to obtain data rates (as described), a plot has been obtained (Figure 3). While it is true that the fact that individual data rates are estimated linearly (i.e. no fluctuations in data rate during a single flow are considered), some values tend to repeat themselves. This aside, a plot in logarithmic scale has also been included, for the linear one gets somewhat distorted due to the peaks in data rate.



(a) In linear scale. (b) In logarithmic scale.

Figure 3: Obtained traffic volume plots.

2.3 Provide the top 5 most commonly used protocols

Please provide the top 5 most commonly used protocols, as well as the five most common source ports and five most common destination ports based on flows. Detail in a table for each one:

- *The number of flows.*
- *The number of packets.*

- *The amount of data (bytes).*
- *The application or usage.*

Hint: The column 'pro' defines the protocol used.

The required data (available in Table 2) has been extracted from the data frame using common pandas functions as well as a `for` loop that allows to only iterate through the data of the top 5 most commonly used protocols.

Protocol	Flows	Packets	Bytes	Most Common Destination
17	13.672	54.419	10.464.018	ff02::1:3
6	2.493	455	1.087.284.269	192.168.1.177
58	455	2.234	410.558	ff02::1
2	234	782	31.404	224.0.0.1
1	124	613	51.678	192.168.177

Table 2: Top 5 most used protocols by number of flows, with the total number of packets and bytes transmitted, as well as their most common destination.

2.4 Top ten host pairs

Which are the top-ten host pairs based on number of flows and number of bytes? Are there the same pairs?

These can be seen in Table 3. They are notably different, for in the captured data it seems to be the case that large transfers have happened in quite brief moments of time.

Position	Source	Destination	Flows	Source	Destination	Bytes
1	fe80::f17b:6c72:fea6:5dc5	ff02::1:3	3882	195.148.124.36	192.168.1.177	523023227
2	192.168.1.62	224.0.0.252	3881	192.168.1.177	195.148.124.36	232877667
3	192.168.1.1	239.255.255.250	1565	188.166.241.168	192.168.1.177	139853615
4	192.168.1.1	192.168.1.177	1489	216.58.210.138	192.168.1.177	35037509
5	192.168.1.177	192.168.1.1	1337	151.101.246.248	192.168.1.177	25907825
6	2001:14ba:a051:8c00:61b5:31a4:d1f8:cc83	2001:14b8:1000::1	659	192.168.1.177	130.233.229.20	25109595
7	192.168.1.177	239.255.255.250	179	130.233.229.20	192.168.1.177	16869105
8	2001:14ba:a051:8c00:ad01:ffd8:f040:eb85	2001:14b8:1000::1	144	149.154.167.99	192.168.1.177	13821836
9	192.168.1.177	192.168.1.62	119	216.58.211.234	192.168.1.177	11220798
10	192.168.1.62	192.168.1.177	119	34.104.35.123	192.168.1.177	8413258

Table 3: Top 10 pairs of hosts by number of flows and by bytes.

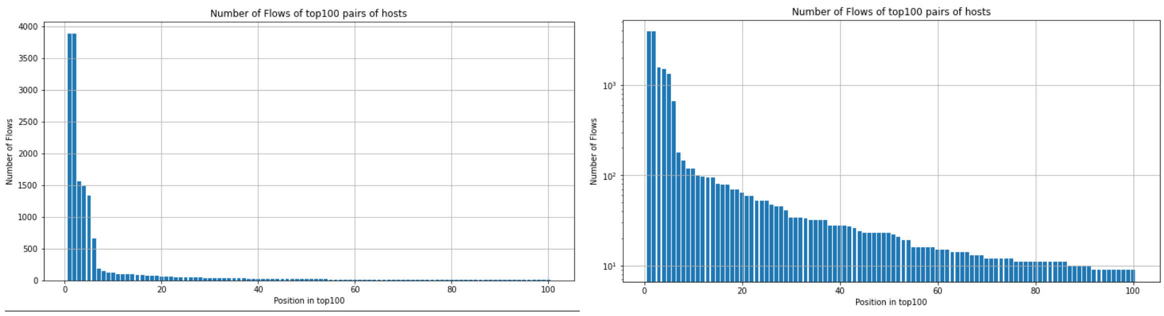
2.5 Plot the number of flows for the 100 most common pairs of hosts

Using linear scale, and using logarithmic scale.

The obtained results can be seen in Figure 4.

2.6 Plot the same plot using now time fixed size.

Repeat the previous plot (both linear and logarithmic scale) using this time fixed size (216 slots) array approach (Network capture tutorial - Large data analysis, pp. 8 and solution #2, pp. 10). What can you say about the results?



(a) In linear scale.

(b) In logarithmic scale.

Figure 4: Number of flows for the 100 most common pair of hosts.

2.7 Is there a more efficient approach in terms of running time and memory consumption to accomplish this task?

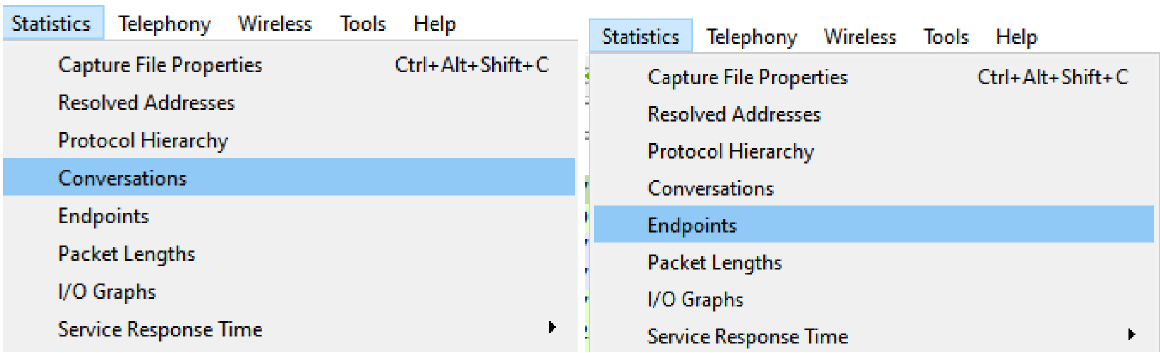
Note: You can use /bin/time command to get resource consumption o f a command, use -v for more verbose. It provided a more detailed output than shell built-in time.

Yes, there is. A common way of time optimization is tool-based: for instance, the methods utilised in this solution are based on Python (with Pandas), an interpreted language - clearly not the most optimal way of programming. Despite having avoided many for loops; working with compiled languages like C, C#, or even assembly, could potentially lead to orders-of-magnitude levels of time improvement. Alternatively, there are other ways of treating the data. As proposed in solution 2, one can use a hash function to store randomly the flows of data, relying on chance to have most common flows of data to remain uncombined. While the risk is low, it does exist and depending on the data set it should be more strongly considered.

Task 3: Analyse Packet Capture (user traffic)

Based on the traffic captured in Task 2, utilize an appropriate tool to analyze the captured data and provide answers to the following questions.

The chosen analysis tool is WireShark, the same one utilised to capture the data in the first place. It offers a wide array of analysis tools and ease to extract the treated data. While in-depth analysis are certainly more complex or sometimes not possible, the data required is quite simple, and can be extracted using filters from the conversations and destinations statistics tab (Figure 5).



(a) Conversation statistics.

(b) Destination statistics.

Figure 5: Wireshark statistics locations in the GUI.

3.1 How many IPv4 hosts (and IPv6, if any) are communicating?

208 pairs of hosts for IPv4 hosts representing $94,3\%$ of the packets; and 132 pairs of hosts for IPv6, representing the remaining $4,5\%$ of the packets. Notably, the $1,2\%$ not accounted have been handled by the address resolution protocol.

3.2 Top 5 host countries (e.g. GeoIP)

Focusing only on IPv4 destinations, the top host countries have been found by exporting the destination statistics into a .csv file and using IP Converter plug-in for Google Sheets, which allows for easy conversion from IP to location data [2]. The results are as follows:

1. **United States of America**: 85 hosts.
2. **Finland**: 33 hosts.
3. **Sweden** 19 hosts.
4. **Germany**: 14 hosts.
5. **United Kingdom of Great Britain and Northern Ireland**: 12 hosts.

3.3 Top 15 hosts by byte counts.

The results can be consulted in Table 4.

Position	IPv4 IP	Location	Bytes	IPv6 IP	Bytes
1	192.168.1.177	-	1097708926	fe80f17b6c72fea65dc5	1471382
2	195.148.124.36	Finland	759205580	ff02fb	826778
3	188.166.241.168	Singapore	145469727	200114baa0518c0061b531a4d1f8cc83	811812
4	130.233.229.20	Finland	42346129	ff0213	711548
5	216.58.210.138	United States of America	35352010	200114baa0518c00ad01ffd8f040eb85	432476
6	151.101.246.248	Finland	26097792	200114baa0518c007e7716fffea8cf20	335632
7	149.154.167.99	United Kingdom of Great Britain and Northern Ireland	16176550	fe803c6485d8f884544c	222765
8	216.58.211.234	United States of America	11389162	200114b810001	106880
9	34.104.35.123	United States of America	8518776	ff02c	105546
10	216.58.210.129	United States of America	7868658	2a0014504026808200a	86010
11	35.186.227.140	United States of America	5796606	2a0014504026805200e	85924
12	216.58.209.197	United States of America	4597925	2a0014504026808200e	78849
13	34.117.138.150	United States of America	3444761	2a00145040268082001	67823
14	40.114.178.124	Netherlands	2988267	fe807e7716fffea8cf20	58582
15	192.168.1.1	-	2486884	ff0216	38590

Table 4: Top 15 destinations, by bytes transmitted.

3.4 Top 15 hosts by packet counts

Were there any differences between the top 15 hosts in terms of byte counts and packet counts?

The required data can be consulted in table 5. While the top 4 hosts seem to remain mostly unchanged, the rest fluctuate a lot more. This is very possibly due to the fact that many of the top hosts by number of flows are reserved addresses which are usually for local use, taking a certain number of packets with little volumes of information.

Position	IPv4 IP	Location	Packets	IPv6 IP	Packets
1	192.168.1.177	-	404509	fe80f17b6c72fea65dc5	15341
2	195.148.124.36	Finland	236049	ff02fb	8222
3	130.233.229.20	Finland	24869	ff0213	7660
4	188.166.241.168	Singapore	24543	200114baa0518c0061b531a4d1f8cc83	2785
5	192.168.1.62	-	22198	200114baa0518c007e7716fffea8cf20	1213
6	149.154.167.99	United Kingdom of Great Britain and Northern Ireland	21965	fe803c6485d8f884544c	1212
7	35.186.227.140	United States of America	10797	200114baa0518c00ad01ffd8f040eb85	1069
8	192.168.1.1	-	8922	200114b810001	1031
9	35.186.224.17	United States of America	8917	fe807e7716fffea8cf20	675
10	216.58.210.138	United States of America	8723	200114b810002	356
11	224.0.0.251	-	8262	ff0216	245
12	224.0.0.252	-	7721	ff021	167
13	239.255.255.250	-	6288	ff02c	147
14	192.168.1.255	-	6041	fe801082713d3d2ca607	133
15	192.168.1.62	-	5359	fe807e7716fffea8cf22	100

Table 5: Top 15 destinations, by number of flows.

3.5 Top 10 TCP and top 5 UDP port numbers (by packet count)

While this kind of data can be extracted using an excel-like program as well, it is much easier and quicker to just transform the statistics data provided by Wireshark to a .csv and then interpret that with a few lines of Python. This is a common procedure that has been done multiple times this assignment and during previous ones, but the full code can as usual be consulted in the attached .zip file. The results are presented combined with the top UDP port numbers in Table 6.

3.6 Top 10 fastest TCP connections

The conversations statistics tab provides the data for the bytes/s in both directions of the exchange. Thus, a new dataframe has been created from that data, so the fastest connections in each direction can be easily mapped. All in all, this has resulted in these being the fastest TCP connections (see Table 7).

Port (TCP)	Packets	Port (UDP)	Packets
443	126514	5353	32644
5200	94375	5355	15375
50076	94355	137	11750
5207	92629	1900	6141
50154	92619	443	5577
5204	39250	53	4514
50100	25247	62154	1184
50144	20619	57621	464
5203	20617	59859	464
57881	19871	59860	462

Table 6: Top 15 ports by packets sent, for TCP and UDP.

Port (TCP)	Bits/s A $\rightarrow$ B	Port (TCP)	Bits/s B $\rightarrow$ A
426	384040.0	426	199600018.0
433	348981.0	433	198411193.0
1234	313632.0	1234	195932456.0
1224	359440.0	1224	160776218.0
1247	382941.0	1247	76516327.0
1248	242670.0	1248	53062182.0
1241	92280.0	1241	44177859.0
625	145207.0	625	41892426.0
450	75541.0	450	36509223.0
1226	64344.0	1226	18834760.0

Table 7: Top 10 fastest TPC connections, in both directions.

3.7 Bit and packet rate over time

This has actually been already done as a preliminary overlook at the data captured with Wireshark. They can be seen in Figure 1.

3.8 How many hosts were tried to contact to, but communication failed for a reason or another? Can you identify different subclasses of failed communications?

As provided by Wireshark, there seems to be a recurring error in contacting a set of hosts at regular intervals: every 10 minutes and every 30 (as seen in Figure 6). It is probably by some background process that is not able to get through. For instance, some failure related to trying to communicate to some overseas server required for the data collection of the final project of this course. Other than that, there does not seem to be any remarkable error, so this is probably the main issue that can be identified in the collected data.

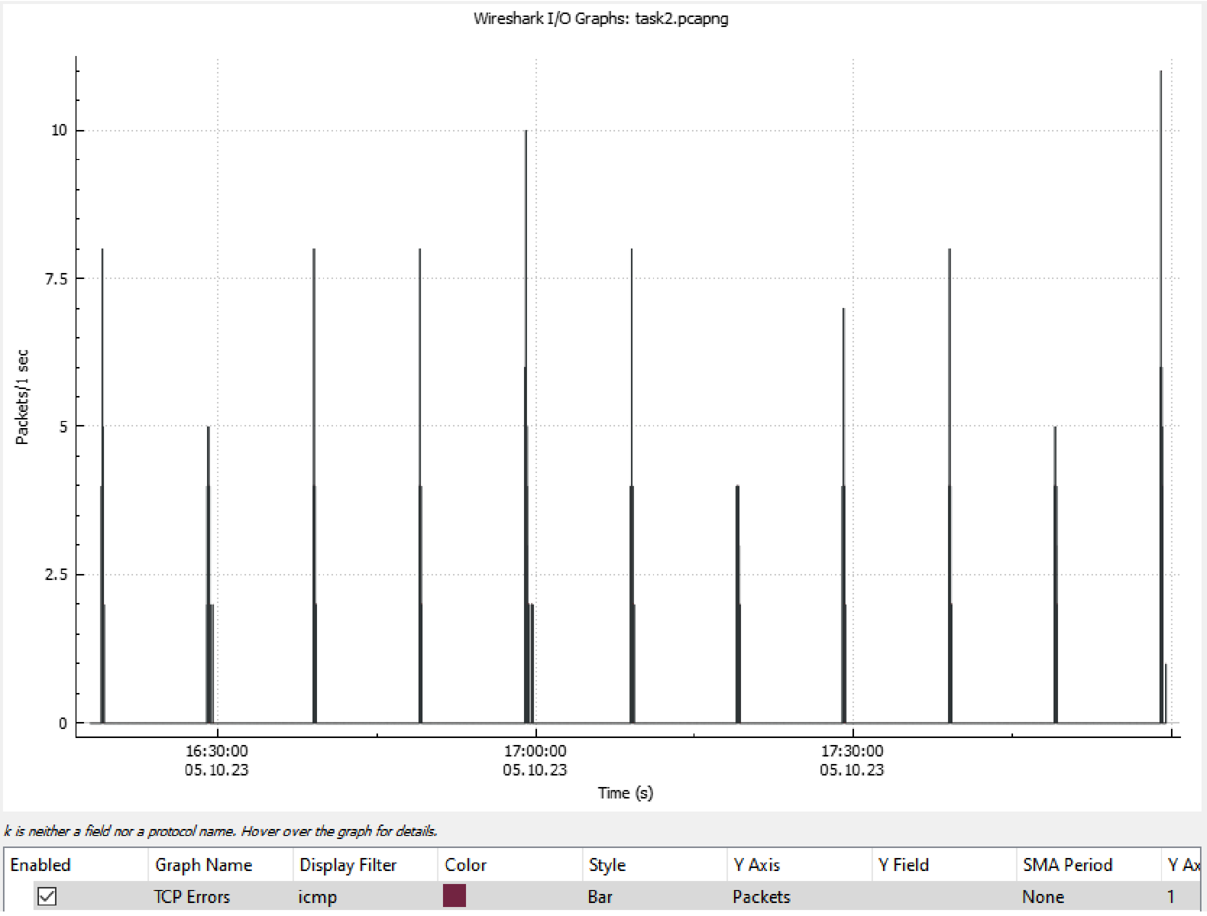


Figure 6: Failed communications over time.

References

1. *WireShark - Go Deep* [online]. WireShark Foundation, 2023 [visited on 2023-10-08]. Available from: https://www.wireshark.org/.
2. *IP Converter* [online]. 2020. [visited on 2023-10-09]. Available from: https://sites.google.com/view/ipconverter/home?authuser=0.