Network Traffic Analysis

André Rosa, André Real
DI/FCT/NOVA University of Lisbon, Lisboa, Portugal
Student № 48043, 49831
af.rosa@campus.fct.unl, a.corte-real@campus.fct.unl
Group № 1

***Abstract*—Network Traffic Analysis (NTA) is extremely important for extracting essential information from monitored networks. However, in spite of this information being highly valuable to better manage networks, processing this data is particularly challenging due to it consisting of simple traffic logs and its huge amounts.**

As such, the aim of this assignment was to process recorded traffic flows, that crossed a given network device, and extract knowledge about the network. Thus, we report in this document the strategies employed to extract information from this traffic. Finally, with the elaboration of this assignment, we gained an insight into how challenging these types of analysis can be.

***Index Terms*—NTA, traffic analysis, NetFlow, python**

I. Introduction

Network Traffic Analysis (NTA) is an essential instrument for obtaining insight of network usage, being extremely important for retrieving all sorts of crucial information from monitored networks, such as network performance indicators, reveal security breaches, or uncover the necessity of performing hardware maintenance.

Although this information is highly valuable to better manage networks, processing the monitored data is not a trivial task. Usually, this monitored data is just a simple log of the packets that crossed a specific network device. However, sometimes these packets are aggregated into sequences of packets that share common information (e.g same destination address), called *flows*. Consequently, extracting more complex information from those simple logs can be quite challenging. Furthermore, this data can be extensively vast, due to the huge amounts of traffic that cross network devices, turning processing this data even more challenging.

Taking this into account, the aim of this assignment is to process monitored flows of traffic, that crossed a given network device and extract knowledge about the network. Thus, in this document, we report and explain the strategies employed to extract valuable information from the collected traffic, as well as the results and conclusions derived from that information.

The remainder of this document is structured as follows: Section II that gives an overview of the collected traffic and briefly explains the tools employed to process each file; Section III describes how the traffic data was processed to extract useful information as well as our conclusions regarding that information; and finally, Section IV concludes this report with some final remarks.

This report was written in the context of the APRC course.

II. Collected Network Flows

In this section, we give an overview of the collected traffic and briefly explain how this data was processed. We were given two files, which we named *fctFlows* and *bigFlows*, containing all the flows that crossed a given network device, each from a distinct network, during a certain period. To simplify the exposure of information we will use the terms *fctFlows* and *bigFlows* to refer to both the files and the respective networks the traffic contained on those files was taken from. The *fctFlows* traffic was recorded from a network interface from the *www.fct.unl.pt* web server during one hour, having been registered 21362 flows. The *bigFlows* traffic was recorded from a router serving as an internet access point to a company's private network, during five minutes, which registered 42845 flows. Both of these files are in a format similar to Cisco's NetFlow [1], where each line represents a distinct unidirectional flow. Each flow contains a pair of timestamps which represent the period when the flow was active, the source and destination's IP addresses and ports, the inner protocol (e.g. TCP), a set of all the flags observed during the flow duration, and the number of packets and bytes.

Regarding the IP addresses, they can be classified into two disjoint groups: *internal* IP addresses, which belong to an entity inside of the current domain, or *external* IP addresses otherwise. In *bigFlows* we considered as an *internal* IP address any address contained in the range 172.16.0.0/16 - 172.31.0.0/16 being all the other addresses considered as *external*. In *fctFlows* we considered as an *internal* IP address any address belonging to any of the following network prefixes: 192.68.216.0/24, 193.136.120.0/21, 193.137.125.0/24, 193.137.126.0/23; being all the other addresses considered as *external*. As such, the flows can be classified according to their "direction", i.e. according to the classification of the source and destination IP addresses. Thus, a flow which has an *internal* source address and an *external* destination address has the direction *in*⟶*out*, or simply *out*. Similarly, a flow which has an *external* source address and an *internal* destination address has the direction *out*⟶*in*, or simply *in*. Additionally, there can also exist flows with the directions *in*⟶*in*, when both addresses are *internal* to the domain, and *out*⟶*out*, when both address are *external* to the domain.

Regarding the port values, they only apply to the TCP and UDP protocols, however, port 0 was assigned to IGMP and port 2048 to ICMP. Similarly, the flags field only applies to TCP.

To process each file, we resorted to a data analysis and manipulation library, written in the Python language [2], called *Pandas* [3]. This tool enables manipulating and processing tabular data, such as data stored in spreadsheets, databases, or CSVs files.

III. Knowledge Extraction

In this section, we delve into the details of how we leveraged Pandas to extract information from the gathered flows and discuss the obtained results.

A. Total Number of Packets and Bytes per Protocol

To compute the total number of packets and bytes of each file, both in and out, we first filtered all the flows to obtain only those with the intended "direction" i.e. *in* and *out*. Then, we separated the flows into groups according to their protocol, and, for the flows of each group, we summed the number of packets and bytes.

The results of these operations are presented in Figure 1, being that the axis is in logarithmic scale to improve readability.

In the *bigFlows* network, it was recorded traffic of four different protocols: ICMP, IGMP, TCP, and UDP. As expected, TCP was the protocol with the highest traffic share, both in total bytes (Fig. 1c) and number of packets (Fig. 1d), in the two "directions", i.e. *in* and *out*. In this network, the incoming TCP traffic was slightly higher than the outgoing, corresponding to 73% and 55% of the bytes and packets exchanged belonging to TCP, respectively, and 61% and 64% of the total bytes and packets exchanged, respectively. These results evidence that there was much more incoming data to the company's private network than outgoing. Regarding UDP, it was the other way around, with the outgoing traffic being higher than the incoming, corresponding to 72% and 51% of bytes and packets exchanged through UDP, respectively.

On the other hand, in the *fctFlows* network, it was only recorded traffic belonging to TCP. We believe that this might be due to security policies that block traffic of other protocols. In this network, the outgoing TCP traffic was much higher than its incoming counterpart, being 96% and 64% of the total bytes and packets exchanged, respectively. It is relevant to notice that in this network, the predominant traffic was data transferences to the outside of the domain, regarding the number of bytes, yet it was composed of fewer packets than the incoming traffic. The superior number of outgoing bytes was expected since the traffic was recorded in a web server, where incoming packets are mainly HTTP GET requests for web pages, which do not possess many bytes, while the outgoing traffic is responses containing the data of the requested web pages. Regarding the discrepancy in the number of packets, we believe that this is again due to the security policies of the domain, which prevent the servers from establishing connections or replying to some requests.

B. 50 Most Popular External IP Addresses per Flows

For both files, to determine the 50 most popular IP addresses external to the domain by the number of *out* flows, we started by filtering the other types of flow. After that, each flow was grouped according to their destination address, and we counted how many (*out*) flows each address had. The results are present in Figure 2.

In the *bigFlows* network (Fig. 2a), the most popular IP address was 8.8.8.8, with 1465 (*out*) flows, and corresponds to Google's public DNS server. The 8th most popular address, 8.8.4.4 is another Google's public DNS server. Overall, almost every address in the higher half of the chart belongs to LogMeIn, Salesforce, and AKAMAI. In particular, in this network, we can see a pattern of IP addresses with the prefix 68.64.0.0/16 with a total of 762 flows combined, which belong to LogMeIn [4], a company responsible for remote access software. All but one of all the 50 IP addresses are located in the United States of America, being the other located in India. Additionally, it is also relevant to notice that the IP address 255.255.255.255, which corresponds to the broadcast address, is the 12th most popular address, which means that in this company's network there is a significant amount of services who leverage broadcast.

Focusing on the *fctFlows* network, the most popular IP address was 154.0.164.204 with 404 flows, and belongs to a South African ISP. From the 50 determined addresses, around 80% belonged to Portuguese ISPs. However, these addresses only contained 42.64% of all the flows. The remaining addresses were from abroad, more precisely, from Africa, America, China, Russia, Brazil, and Poland.

C. 50 Most Popular External IP Addresses per Bytes

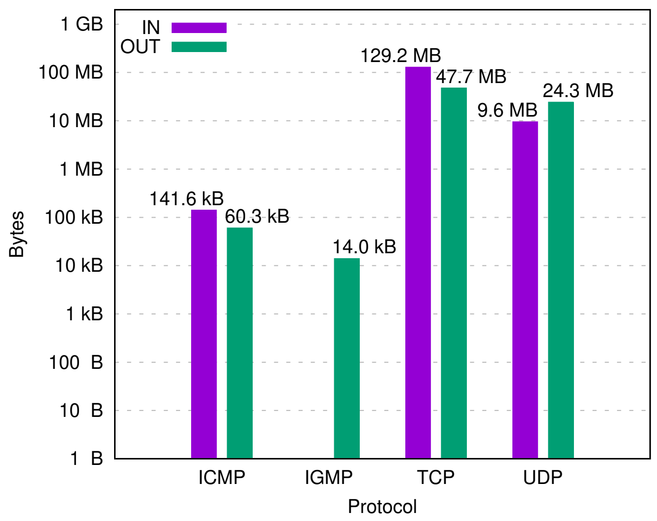
The computation of the 50 most popular external IP addresses per number of bytes is very similar to the previous one, §III-B. The only difference is that instead of counting the number of flows belonging to each address, we sum the bytes of all those flows. The results obtained are presented in Figure 3.

On the results of the *bigFlows* network (Fig. 3a), the most popular IP address belongs to *Outlook.com*, Microsoft's email service. The second belongs to a remote access service for companies, LogMeIn [4]. There are more IP addresses belonging to this company in the chart, which reveals that remote access is very popular in this company. Perhaps, this can be a consequence of COVID-19, which increased the amount of remote access work around the world.

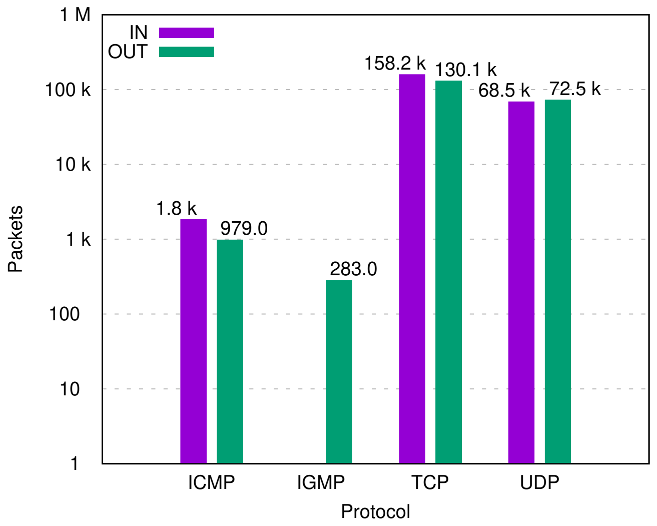
In the *fctFlows* network (Fig. 3b), the most popular IP address belongs to *ViaSat*, a global communications company. Furthermore, 78% of the IP addresses were owned by Portuguese ISPs and, these ISPs were responsible for nearly 72% of the total number bytes of the 50 addresses.

D. 50 Most Popular Applications

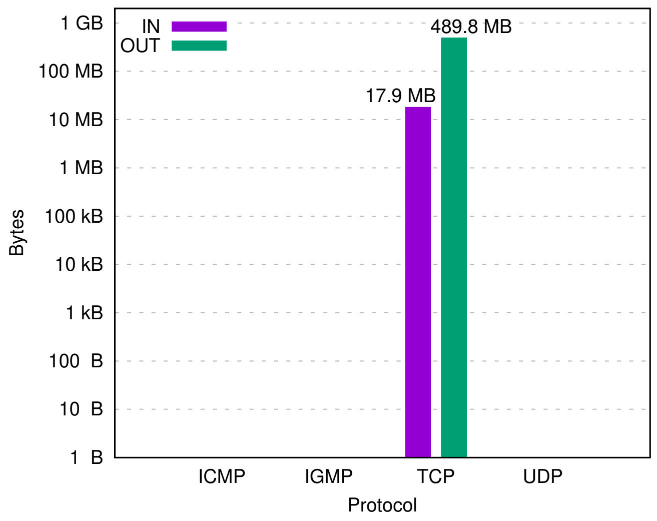
To determine the 50 most popular applications in the *bigFlows* network, we started by filtering all the flows in which the protocol was either UDP or TCP since these were the only recorded transport protocols. Each application-level service/protocol can be identified by the leveraged transport protocol and the default port assigned. As such, after filtering the flows, we grouped them by both the destination port



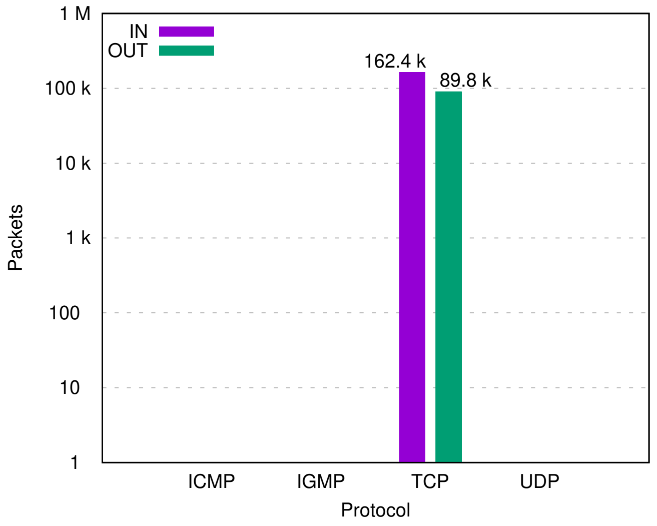
(a) *bigFlows* bytes per protocol



(b) *bigFlows* packets per protocol



(c) *fctFlows* bytes per protocol



(d) *fctFlows* packets per protocol

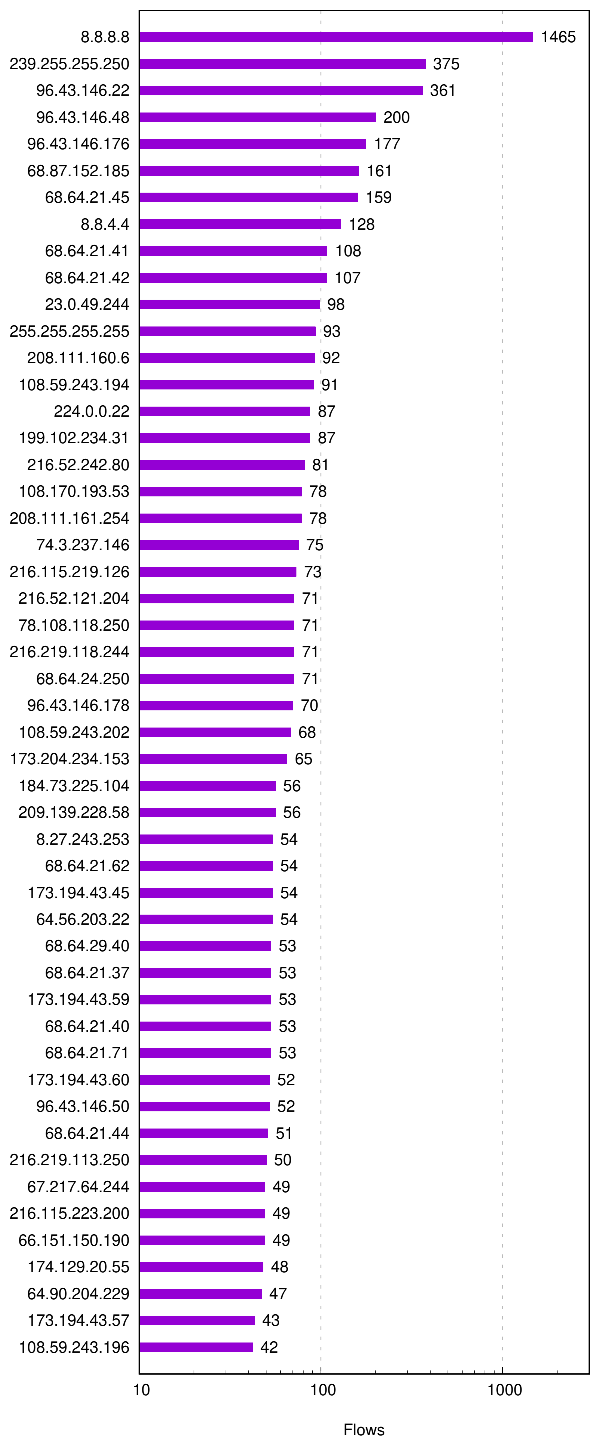
Fig. 1: (a),(b) presents the amount of bytes and packets exchanged per protocol in *bigFlows* and (c),(d) presents the amount of bytes and packets exchanged in *fctFlows*, per protocol.

and the protocol and counted how many flows each group had. Finally, we mapped each pair protocol and port to the service/protocol assigned by IANA [5], being the results presented in Figure 4.

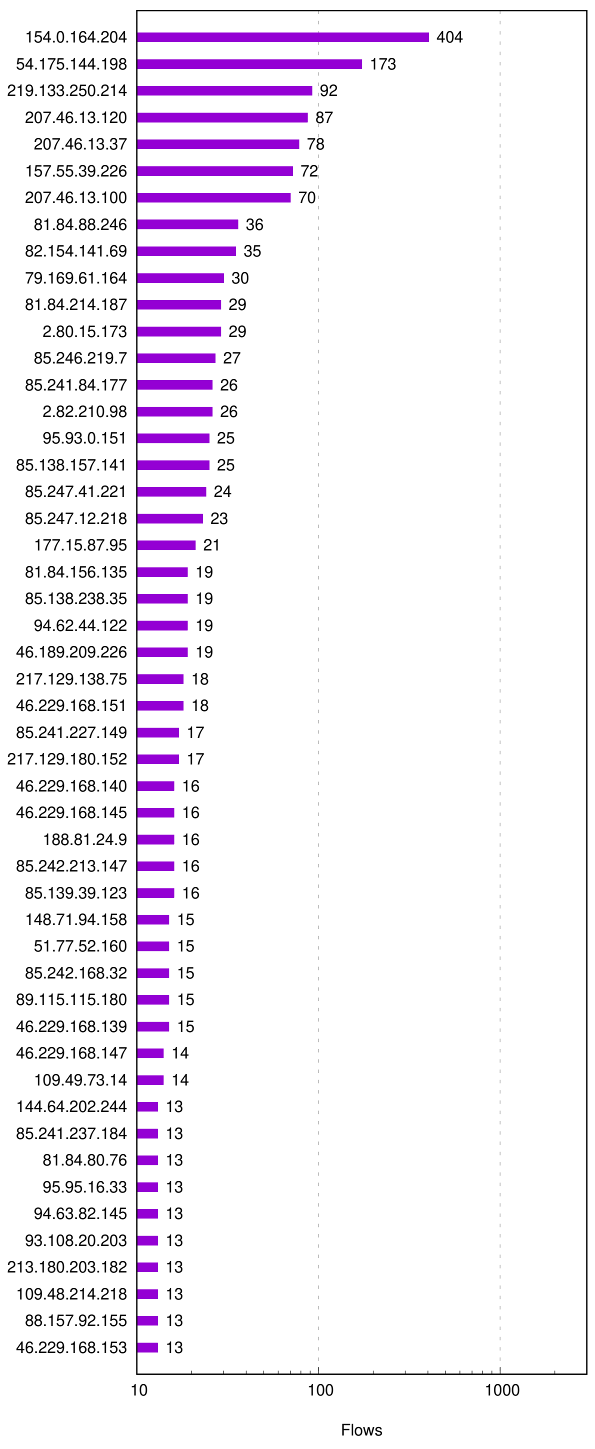
Surprisingly, only five applications leverage TCP, while the majority resorts to UDP. However, TCP has a much higher number of flows. The most popular application-level protocol was HTTP, with 4360 flows, closely followed by HTTPS, with 3750 flows. This was expected since these protocols are the base for data transference on the world wide web. At the third place, we have the DNS protocol, with 1511 flows. In the first half of the chart, there are several remote access and communication applications/protocols such as AVTP (Audio Video Transport Protocol), GoToMyPC / MiniDLNA, XMPP (eXtensible Messaging and Presence Protocol) and Skype VoIP. As mentioned before, this could be related to COVID-19. Finally, we also have several Microsoft Services as well as Traceroute Services in the lower half of the chart.

E. Total Number of TCP Connections

Each TCP connection is uniquely identified by the source address and port and the destination address and port. Typically, each connection has two flows, one in each direction. However, in the files provided there we cases where a connection could have 3 or even 4 associated flows. Some of these flows contained only a single packet and thus were discarded. The remaining were grouped and the flags of both were considered to determine the validity of a connection. To aggregate the two flows representing the same TCP connection, we first filtered all the flows where the protocol was TCP. Next, for each TCP flow we deterministically computed a connection identifier based the source address and port and the destination



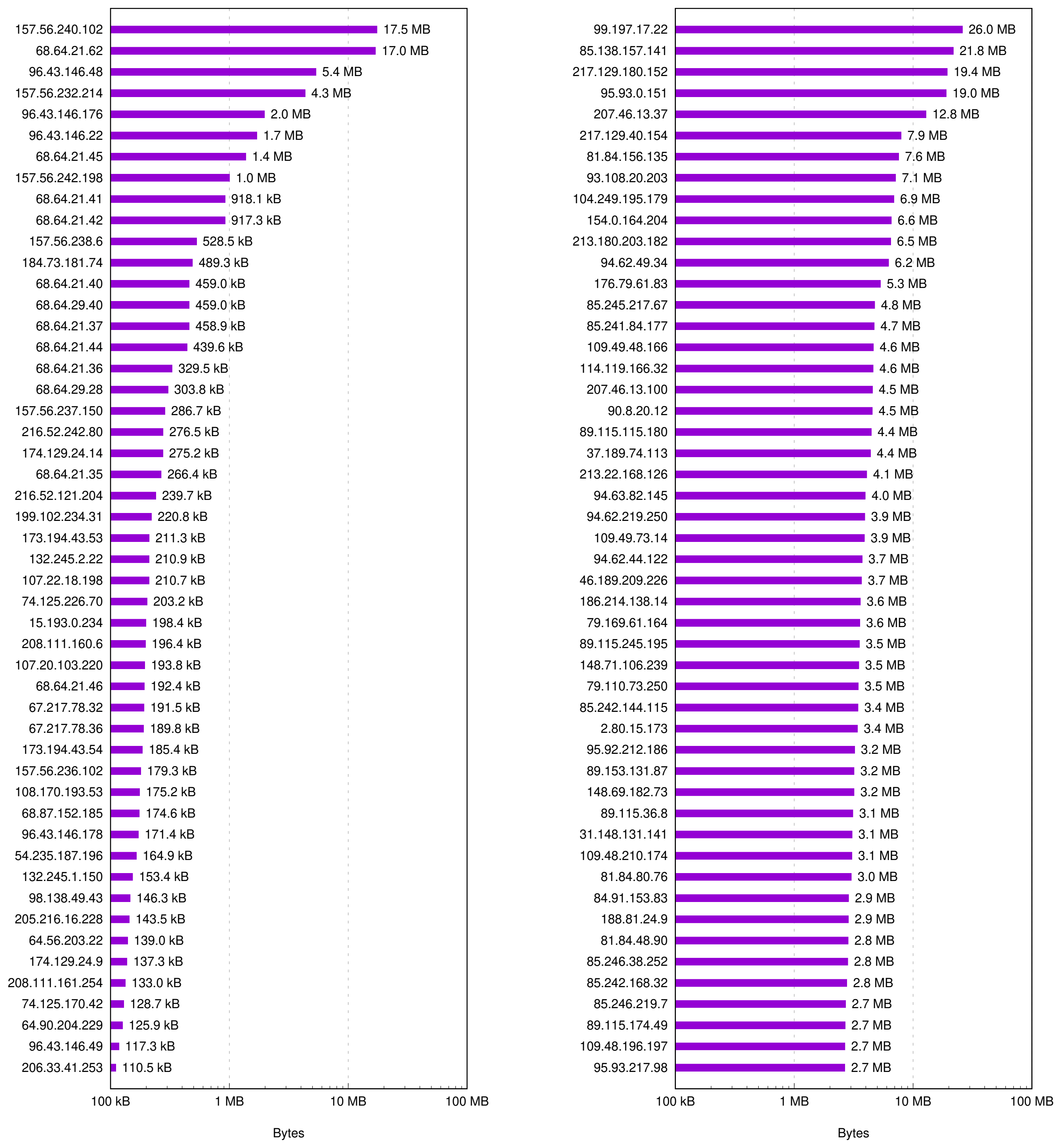
(a) *bigFlows* flows per external IP address



(b) *fctFlows* flows per external IP address

Fig. 2: (a),(b) presents the amount of out flows per external IP address in *bigFlows* and *fctFlows*, respectively.

address and port. Afterward, the flows were grouped together by the connection identifier, corresponding each group into a different connection. At this point, we computed the number of total TCP connections collected including the failed ones i.e. connections that did not successfully conclude the three-way handshake or the fin handshake. Finally, from these connections, we removed the ones which did not contain two valid flows. A valid flow corresponds to a flow with both the flags S-Syn and F-Fin. If a connection contains two of these flows, one in each direction, then both the establishment and the termination of a connection were successfully concluded. The results are presented in Table I.



(a) *bigFlows* bytes per external IP address

(b) *fctFlows* bytes per external IP address

Fig. 3: (a),(b) presents the amount of bytes per external IP address in *bigFlows* and *fctFlows*, respectively.

On one hand, in the *bigFlows* network, only 26.7% of the connections were valid. On the other hand, in the *fctFlows* network, 76.1% of the connections were valid. As previously mentioned, here are two reasons for a connection being considered invalid: no successful handshake or no successful termination. Regarding unsuccessful handshakes, they can be the result of firewalls that block connections, the destination is unavailable (e.g. changed IP or crashed), or the destination is overloaded. In relation to unsuccessful terminations, the causes can be network failure, destination or source failure, one side continuing to send data after the other closed its receiving part of the connection, a user manually interrupts the connection or

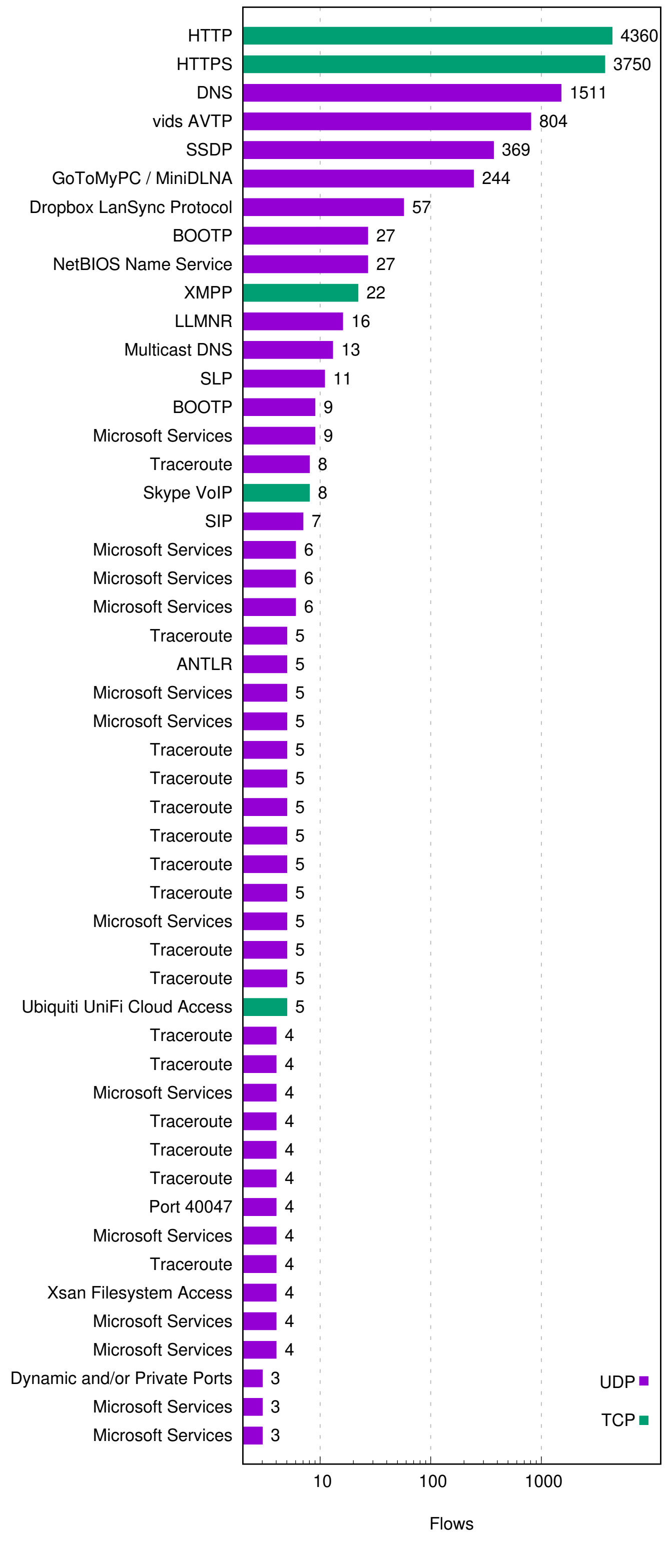


Fig. 4: 50 most popular applications in the *bigFlows* network.

lousy implementations which some send R instead of properly attempting to close a connection.

Files	Total Connections	Valid Connections
bigFlows	20657	5525
fctFlows	10687	8132

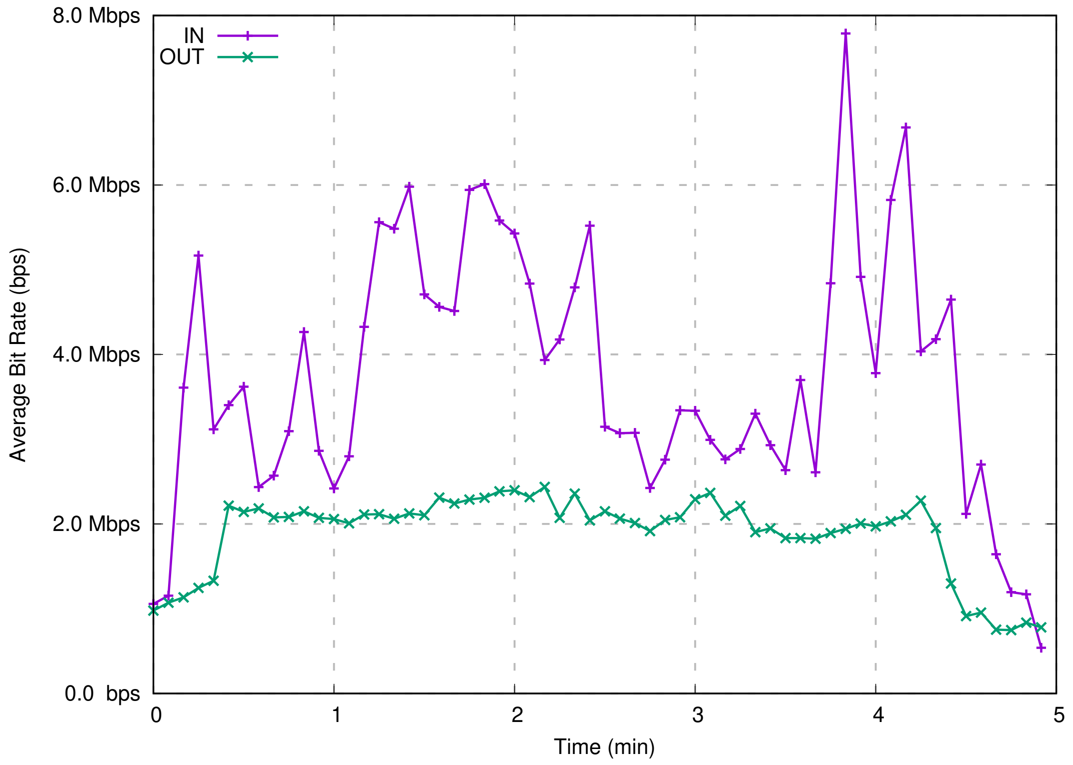
TABLE I: Total and valid connections for *bigFlows* and *fctFlows*.

F. Average Bit Rate

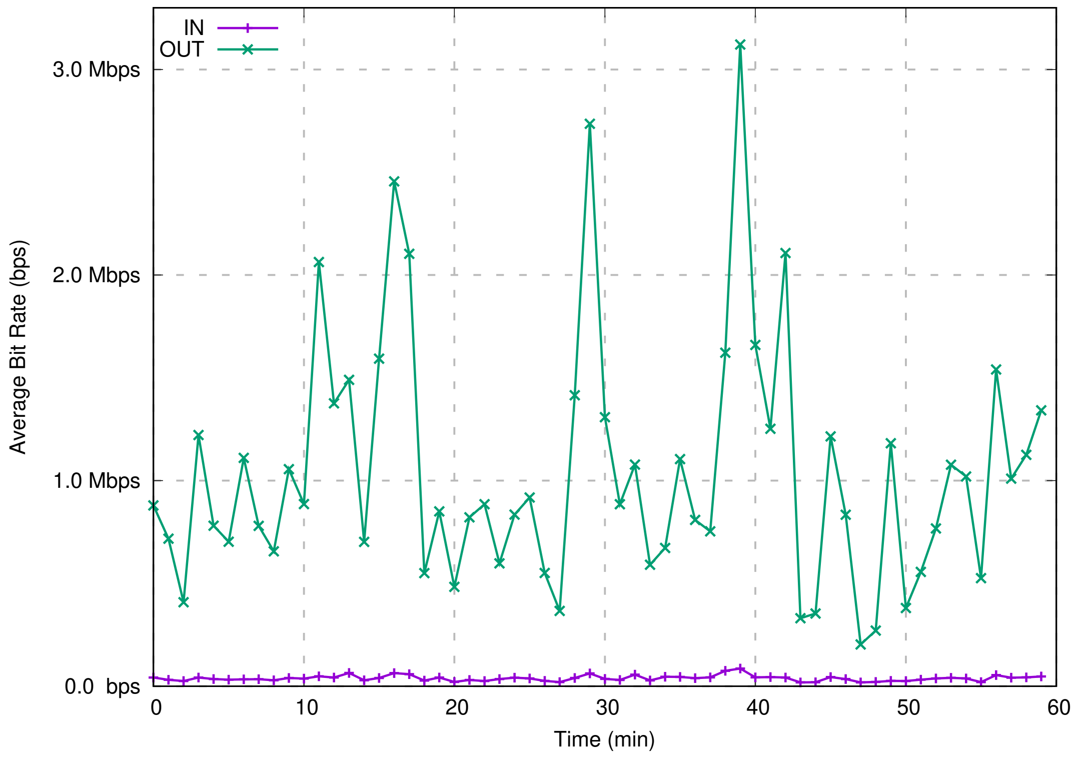
To compute the average bit rate per time unit that crossed each device, both in and out, during the collecting period, we first found the minimum start and maximum end times of the flows in order to delimit the period of data collection. Next, from these values, we computed the duration of the collection and divided this value by the number of data points we pretended to compute, which were 60 in our case. This results in the duration which each data point covers. As such, each data point in *bigFlows* corresponds to 5 s of traffic, and each data point in *fctFlows* corresponds to 1 min of traffic. Next, for each data point, we computed the average bit rate per duration of the data point, i.e. for each 5 s or 1 min accordingly to the file, both *in* and *out*. To compute these values, for each "direction" (*in* and *out*), we first filtered all the traffic to obtain only the flows in the intended "direction". Next, for each of these flows we computed the *overlap* the flow had with the current data point, i.e. how much each flow intersected with the data point interval. Thus, if a flow started before and ended after the current data point the overlap is total, and corresponds to the data point duration; if a flow started and ended before or after the current data point, then the overlap is zero; if a flow started before and ended during the data point or started during the data point and ended after, then the overlap is partial and is less than the data point duration; if a flow started and ended inside a data point then the overlap corresponds to the flow duration. Therefore, the *overlap* is a measure of the contribution of each flow to the current data point's traffic. Then, we computed the average amount of bits each flow contributed to the current data point as follows: we obtained the average bit rate of the flow, by dividing its bytes (times 8) by its duration, and then multiplied the average bit rate by the overlap computed previously. Finally, after computing the average amount of bits each flow contributed to the data point, we summed all these values and divided by the duration of each data point, obtaining the average bit rate of the data point. The results of this operation are presented in Figure 5.

Concerning the average bit rate per minute, in the *bigFlows* network the bit rate of the *in* traffic was much higher than the bit rate of the *out* traffic, as expected since the incoming traffic was much higher than the outgoing (§III-A). The *in*'s traffic bit rate was highly unstable during the collection of traffic, ranging from 540.0 kbps to 7.8 Mbps, and reaching its peak at around 3:45 min. On the other hand, the *out*'s traffic bit rate was relatively stable, averaging around 2.0 Mbps.

In contrast, in the *fctFlows* network the average bit rate of the *out* traffic was superior to the *out* traffic's. The bit rate of the *out* traffic was highly unstable, ranging from 203.0 kbps to 3.1 Mbps, and reaching its peak at around 39 min. On the



(a) Average Bit Rate per Minute *bigFlows*



(b) Average Bit Rate per Minute of *fctFlows*

Fig. 5: (a),(b) presents the average bit rate per minute of *bigFlows* and *fctFlows*

other hand, the bit rate of the *in* traffic was relatively stable and quite low, averaging around 40.0 kbps.

G. Geographic Position of the External IP Addresses

To compute the geographic position of the 50 most popular IP addresses per number of bytes, obtained in §III-C, we used a geocoding library, written in Python [6], to obtain the coordinates of each address. Next, resorting to *basemap [7]*, a tool from Matplotlib [8], we generated a 2D chart with the previous coordinates, i.e. a map. For each network, we generated two maps: world wide view (Figs.6,8) and a view focused on the locations with highest concentration of addresses(Figs. 7,9). In each map, the marker's size represents the number of bytes (from §III-C) of the correspondent IP address.

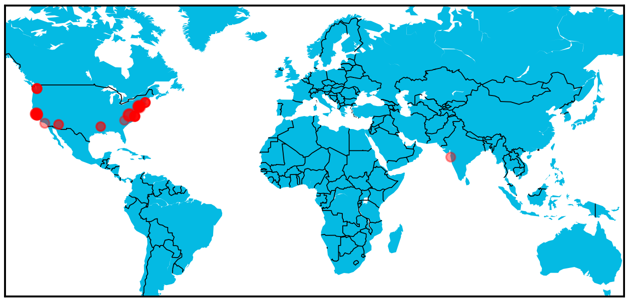


Fig. 6: Geographic position of the 50 most popular external IP addres of *bigFlows*.

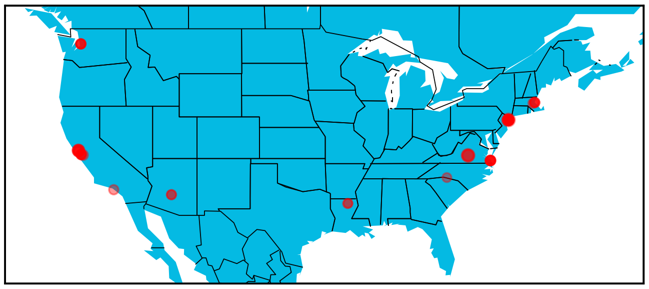


Fig. 7: Geographic position of the 50 most popular external IP addres of *bigFlows*, zoomed in on the USA.

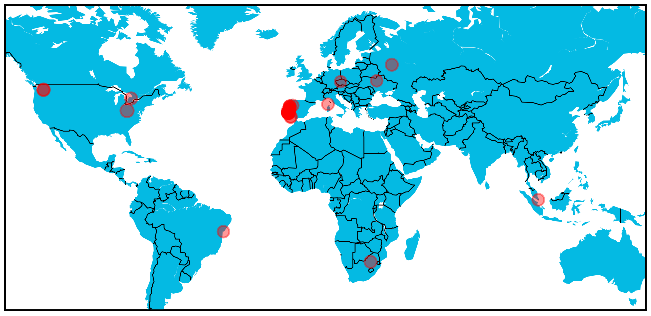


Fig. 8: Geographic position of the 50 most popular external IP addres of *fctFlows*

In Figure 6 are presented the geographic position of the most popular external IP addresses of the *bigFlows* traffic. In this, there is only one IP address outside of the USA, and it is located in India. Taking a closer look in the USA (Fig. 7),

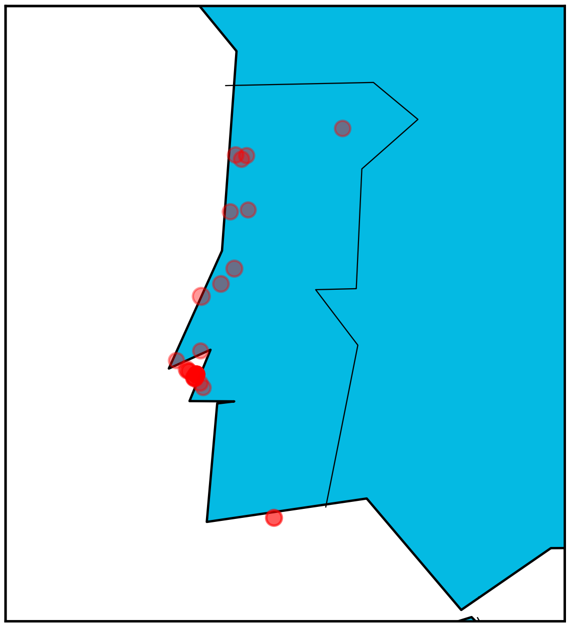


Fig. 9: Geographic position of the 50 most popular external IP addres of *fctFlows*, zoomed in on Portugal.

the addresses are concentrated near both coastlines, in the east and west.

In Figure 8, the majority of IP addresses are located in Portugal, as expected, with some points spread around the globe, e.g. France, Poland, Russia. Taking a closer look into Portugal (Fig.9), we can observe that the highest concentration of IP addresses and bytes is nearby to FCT, and some spread in the north.

IV. Final Remarks

Finally, in this project, we analyzed the traffic of two distinct networks to obtain knowledge concerning them. For this, we leveraged Pandas, a python library for tabular data manipulation, and developed some procedures to extract network statistics. The results of these were presented along with a discussion. Overall, we were surprised by the vast discrepancy between the total number of TCP connections and the number of valid TCP connections, in both networks, and that in the *fctFlows* network, the abroad IP addresses had more flows then Portuguese addresses, yet they only corresponded to around 20% of all the external IP addresses. Finally, the elaboration of this assignment provided us with an insight into how challenging traffic analysis can be.

References

[1] "Cisco IOS NetFlow," https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, accessed: 31-05-2020.

[2] "Python Programming language," https://www.python.org/, accessed: 31-05-2020.

[3] "Pandas, an open source data analysis and manipulation tool," https://pandas.pydata.org/, accessed: 31-05-2020.

[4] "LogMeIn," https://www.logmein.com/pt, accessed: 31-05-2020.

[5] "IANA: Service name and transport protocol port number registry," https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, accessed: 31-05-2020.
[6] "Geocoder: geocoding library written in python," https://pypi.org/project/geocoder/, accessed: 31-05-2020.
[7] "basemap: library for plotting 2d data on maps in python," https://matplotlib.org/basemap/index.html, accessed: 31-05-2020.
[8] "Matplotlib: a comprehensive library for creating static, animated, and interactive visualizations in python," https://matplotlib.org/basemap/index.html, accessed: 31-05-2020.