Network Logs Analysis

1. Introduction and goal of the work

In this work a data collected from university campus network from day 08-06-2015 (Monday) have been analyzed.
The goal of this analysis is to present some trends in network traffic, however based on the data from only one day those trends will be obviously limited and the analysis is not that accurate as it might be when more data is analysed. Nevertheless, due to performance limitations I have decided to focus on only one particular day.

The analysis has been done via python script, which code can be found under the following link: https://github.com/krzysiekpagacz/nst_praca_koncowa/tree/2.0
It was assumed that this script takes input data, do the analysis and prepare the report in form of PDF file.

Raw data have been gathered by nfdump tool and uploaded into internet in the binary form. nfdump reads the netflow data from one or many files stored by nfcapd (netflow collector daemon). nfdump displays netflow data and/or creates top N statistics of flows, bytes, packets. nfdump has a powerful and flexible flow aggregation including bi-directional flows. The output format is user selectable and also includes a simple csv format for post processing.[1]

1.1 Files used for analysis

- nfcapd.201506080000.csv
- nfcapd.201506080005.csv
- nfcapd.201506080010.csv
- nfcapd.201506080015.csv
- nfcapd.201506080020.csv
- nfcapd.201506080025.csv
- nfcapd.201506080030.csv
- nfcapd.201506080040.csv
- nfcapd.201506080050.csv
- nfcapd.201506080100.csv
- nfcapd.201506080110.csv
- nfcapd.201506080115.csv
- nfcapd.201506080120.csv
- nfcapd.201506080125.csv
- nfcapd.201506080135.csv
- nfcapd.201506080145.csv
- nfcapd.201506080150.csv
- nfcapd.201506080205.csv
- nfcapd.201506080210.csv
- nfcapd.201506080220.csv
- nfcapd.201506080225.csv
- nfcapd.201506080230.csv
- nfcapd.201506080240.csv
- nfcapd.201506080245.csv
- nfcapd.201506080250.csv

2. Script architecture

In this chapter detailed architecture of python script has been described.

2.1 Collecting network logs

Network logs have been downloaded by the module: logs_collector.py. It saves log files into the folder ./resources/netflow_raw.
In the second step the log files have been converted into CSV files by means of shell script: to_csv.sh. This script makes a CSV file from every single binary file located in directory: ./resources/netflow_raw. The output in CSV format have been stored in ./resources/netflow_csv and used for further processing as described below.

2.2 Module core.py

This is the starting point of the script. It contains the following methods:

get_data(files_count=3)

looks up the data in the directory specified by the variable: folder_with_csv_files (the directory where CSV files have been stored) opens each file and converts the data into pandas dataframe type. After that those data are being manipulated, so that the needed input is obtained.
Attribute files_count, which is set by default to 3 indicates how many files from folder ./resources/netflow_csv should be taken into consideration. It is particulary useful during testing phase, where one would not like to wait until the whole scope of files in the directory have been processed.
The output of this method is a dictionary object in the form of:
raw_data[date] = [file_name, dict_summary, protocols, ports_df]
where:
date - key of the dictionary
file_name - name of the CSV file (data source)to
dict_summary - each of CSV file has got a summary raw at the bottom, with the following information:
- flows
- bytes
- packets
- avg_bps - average bytes per seconds
- avg_pps - average packets per seconds
- avg_bpp - average bytes per packets

protocols - dataframe object, which contains the relation of a protocol and corresponding number of input bytes
ports_df - dataframe object, which contains information about destination port and the number of connections to each of this port.
Please note that some ports have been skipped: 0, 21548 and ports > 49152. The connections with duration time equal to 0 have been also omitted.
Example of the output:
'201506080120': ['nfcapd.201506080120.csv',
{'ts': '625029', 'te': '49853438740', 'td': '56879420', 'sa': '92856', 'da': '13', 'sp': '876'},
{'ESP': {'ibyt': 5061114.0}, 'GRE': {'ibyt': 11557475.0}, 'ICMP': {'ibyt': 21667274.0}, 'ICMP6': {'ibyt': 853767.0}, 'IGMP': {'ibyt': 52272.0}, 'IPIP': {'ibyt': 1979963.0}, 'IPv6': {'ibyt': 107684.0}, 'OSPF': {'ibyt': 155468.0}, 'PIM': {'ibyt': 1180.0}, 'TCP': {'ibyt': 43637160330.0}, 'UDP': {'ibyt': 6174842213.0}},
{53.0: {'sa': 78651}, 443.0: {'sa': 58524}, 80.0: {'sa': 54954}, 6881.0: {'sa': 31196}, 445.0: {'sa': 20380}, 5060.0: {'sa': 13145}, 23.0: {'sa': 5358}}]

get_destination_ports_df(data)

takes dictionary returned by get_data function and returns pandas dataframe with port number and number of source addresses. The number of different ports is limited to some values represented by NUMBER_OF_PORTS. In this work NUMBER_OF_PORTS has

been set to 10, so only top 10 ports, ordered by the number of the connection have been returned, example (for 2 files):

```
    80.0    443.0   445.0   53.0    ...  993.0   22.0    6667.0  5938.0
sa  75430   60667   35105   33120   ...  1774    1637    1463    1290
```

get_bytes_per_protocols_df(data)

takes dictionary returned by get_data function and returns 1-raw pandas dataframe with protocol names and summary number of inputted bytes, example (for 2 files):

```
            ESP         GRE         ICMP  ...  PIM          TCP          UDP
ibyt  9862084.0  22151985.0  58651084.0  ...  2360.0  9.507028e+10  1.460152e+10
```

get_summary_df(data)

takes dictionary returned by get_data function and returns pandas dataframe with last raw from each CSV file, thus with the following information for the 5 minutes time interval: flows, bytes, packets, average bytes per seconds, average packets per seconds, average bytes per packets. Index of this dataframe is datetime object in format: %Y%m%d%H%M, example (for 2 files):

```
                     ts      te      td      sa  da  sp
2015-06-08 00:25:00  798345  59915617091  70472445  111585  16  850
2015-06-08 01:20:00  625029  49853438740  56879420   92856  13  876
```

get_input_file_names(data)

takes dictionary returned by get_data function and returns list of file names, which have been taken into consideration by the script, sorted by the name ascending, example (for 2 files):

```
['nfcapd.201506080025.csv', 'nfcapd.201506080120.csv']
```

main

function, which runs other functions from core.py module and other imported functions from other modules.

```
if __name__ == '__main__':
    input_data = get_data(files_count=288)
    destination_ports_chart(get_destination_ports_df(input_data))
    protocols = get_bytes_per_protocols_df(input_data)
    bytes_per_L4_protocol_chart(protocols)
    for summary in SUMMARY_OPTIONS:
        get_summary_chart(get_summary_df(input_data), option=summary)
    files = get_input_file_names(input_data)
    generate_pdf_file(files)
```

2.3 Module charts.py

This module is responsible for creating charts based on the input data retrieved by function get_data() from core.py module. It contains the following methods:

bytes_per_L4_protocol_chart(data)

uses data provided by get_bytes_per_protocols_df(data) from core.py module and creates bar chart, which shows bytes distribution between TCP and UDP protocols

destination_ports_chart(data)

uses data provided by get_destination_ports_df(data) from core.py module and creates a chart, which shows the relation between

number of connections and destination port number.

get_summary_chart(data)

uses data provided by get_summary_df(data) from core.py module and creates 6 charts (detail description in Chapter 5)

2.4 Module pdf_generator.py

This module generates an output in the form of PDF file. It contains class PDF, which inherites from FDPF module. PDF class sets some basic parameters of the file, like: header, footer, chapter title, chapter body. It has also functions print_chapter() and print_image(), which adds a text and an image (e.g. chart) respectively.

this function creates an object of PDF class. By means of this object the PDF file is being created in this function. Parameter input_files is needed only to display which files have been considered in this report.

generate_pdf_file(input_files=None)

2.5 Module config.py

Some configuration variables have been extracted from other chapters and gather in config.py module, like for example: charts' names, mapping of port names and numbers etc.

3. Transport Layer Protocols

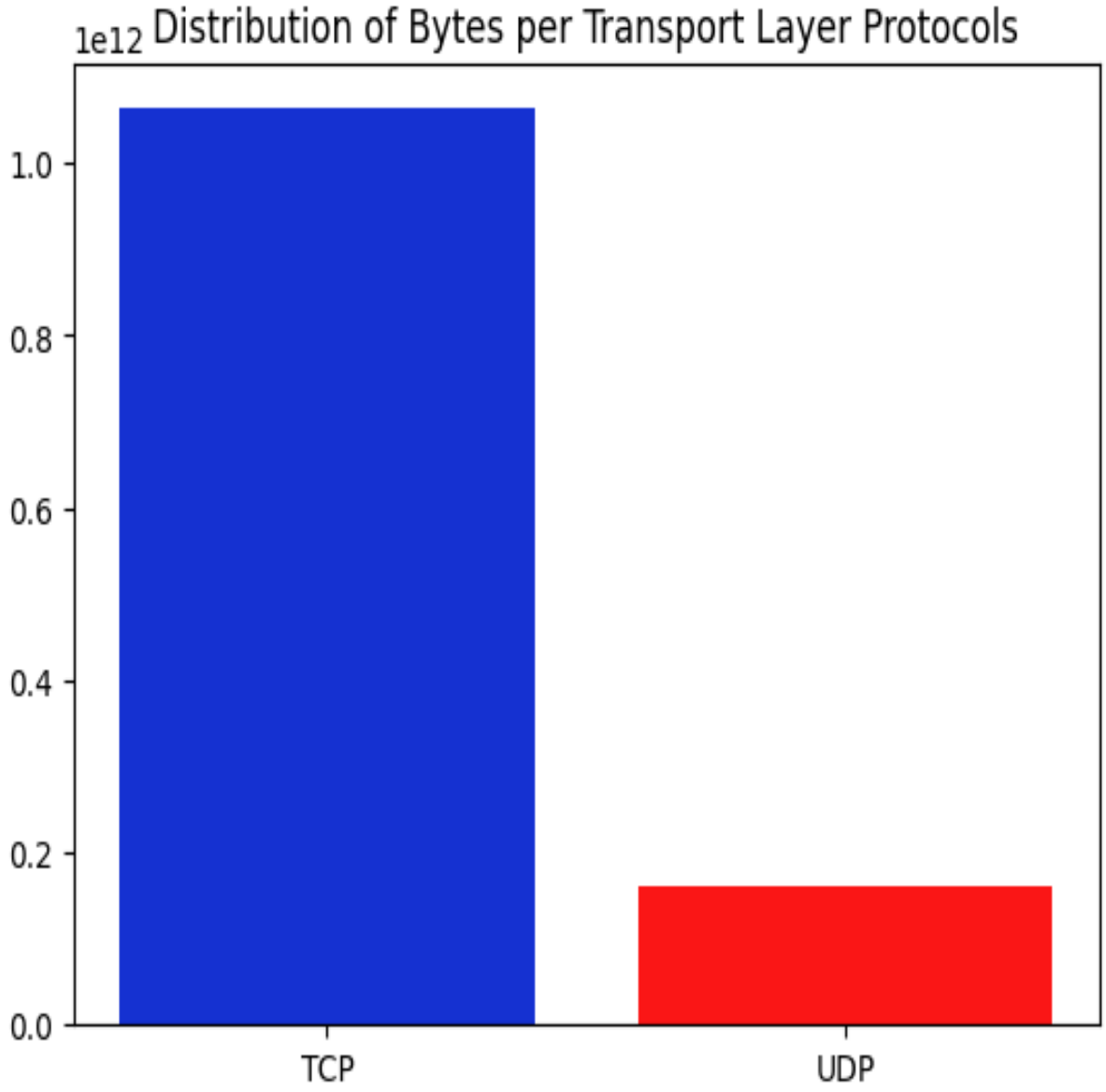
Transport layer in ISO/OSI model is also knwon as Layer 4. It is responsible for:
- establishing a temporary communication session between two applications and delivering data between them
- tracking the individual communication between applications on the source and destination hosts
- segmenting data
- identifying the proper application for each communication stream [2]

The best-known transport protocol of the Internet protocol suite is the Transmission Control Protocol (TCP). It is used for connection-oriented transmissions, whereas the connectionless User Datagram Protocol (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its stateful design incorporating reliable transmission and data stream services. Together, TCP and UDP comprise essentially all traffic on the internet and are the only protocols implemented in every major operating system. [3]

TCP is best suited to be used for applications that require high reliability where timing is less of a concern.
- World Wide Web (HTTP, HTTPS)
- Secure Shell (SSH)
- File Transfer Protocol (FTP)
- Email (SMTP, IMAP/POP)

UDP is best suited for applications that require speed and efficiency.
- VPN tunneling
- Streaming videos
- Online games
- Live broadcasts
- Domain Name System (DNS)
- Voice over Internet Protocol (VoIP)
- Trivial File Transfer Protocol (TFTP) [4]

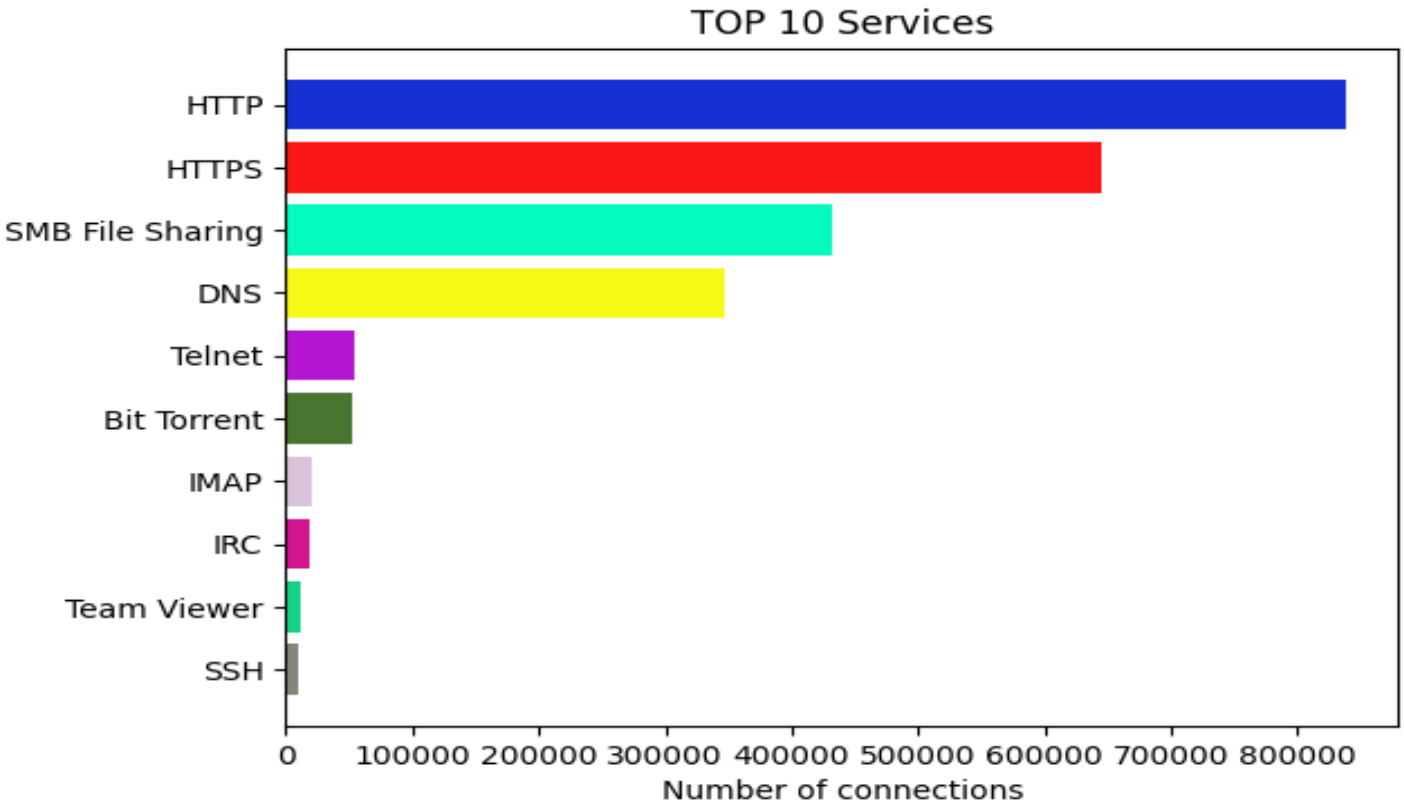
In the chart below we can see the distribution of bytes between both transport protocols: TCP and UDP. As assumed share of TCP usage is significantly bigger.
As it was noticed by the scientists from University of Auckland in their work "Observations of UDP to TCP Ratio and PortNumbers" the ratio (UDP/TCP) is rather dependent on application popularity and, consequently, on user choices. the majority of volume, however is dominated by TCP [6]


Distribution of Bytes per Transport Layer Protocols
1e12
1.0
0.8
0.6
0.4
0.2
0.0
TCP
UDP

4. Services

Each application is identified by the unique number assigned - Port. It is used by protocols of transport layer to identify the proper application, to which data has to be sent. IANA (ang. Internet Assigned Numbers Authority) has divided ports into the following groups:
- Well Known Ports range (0-1023)
- Registered Ports range (1024-49151) - can be used as service identifiers upon successful registration through IANA
- Dynamic Ports range (49152-65535) - as it is dynamically assigned it cannot be used as service identifier [5]

Known that port number up to 49151 can be used to identify the service it was decided to find top 10 used services in the analysed period of time.
In the chart below we can see that HTTP and HTTPS (services used by internet browsers) are the most commonly used. As those logs have been collected on the university campus it was also not surprise that file transfer services are on the top (SMB, torrent).

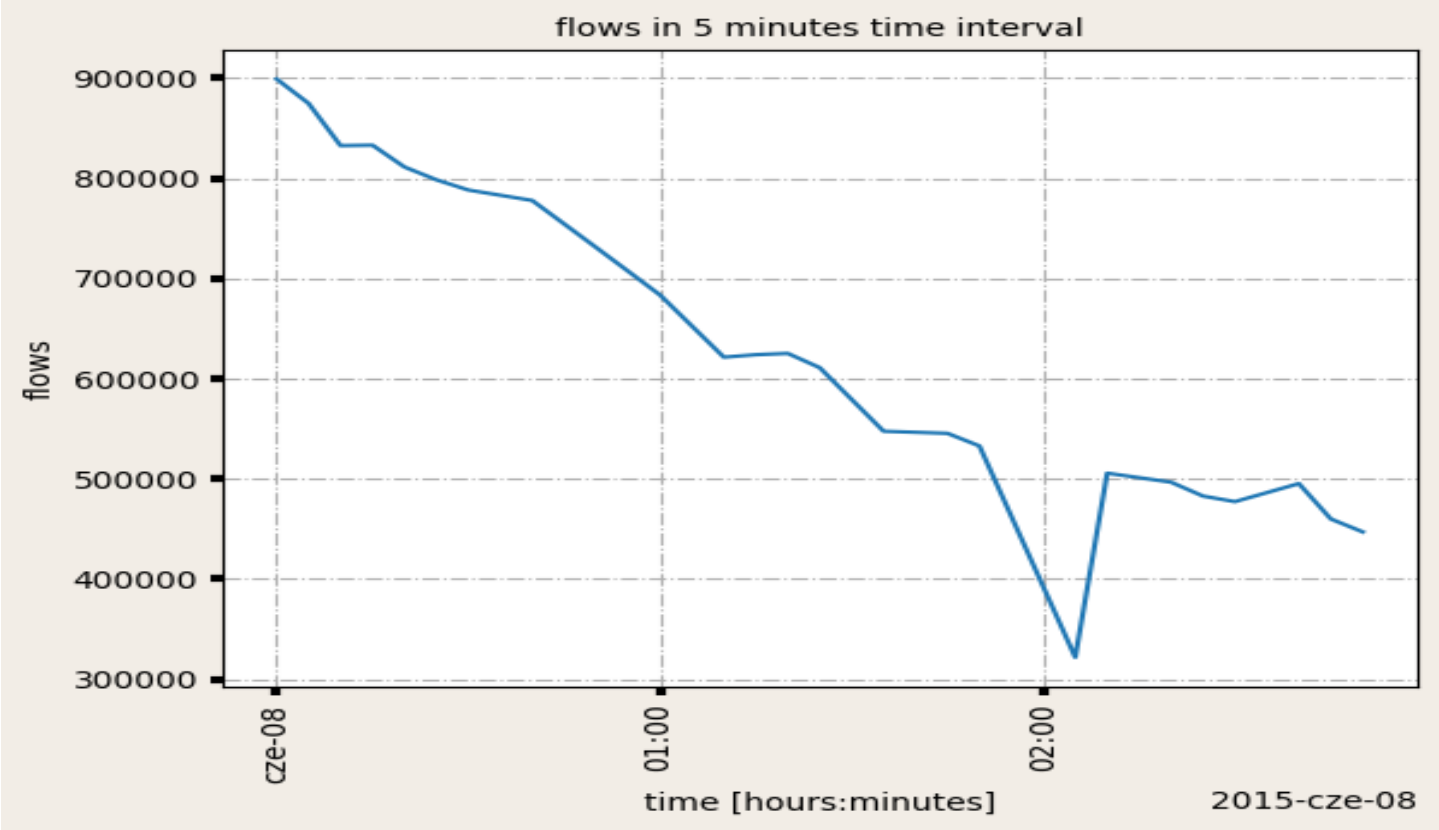


5. Summary of data traffic

5.1 Flows

The most universal definition of traffic flow in packet network is a sequence of packets which share a common property. In this work flow is defined as a unidirectional sequence of packets that share the same five-tuple: IP source address, IP destination address, source port, destination port, and transport layer protocol type. [7]

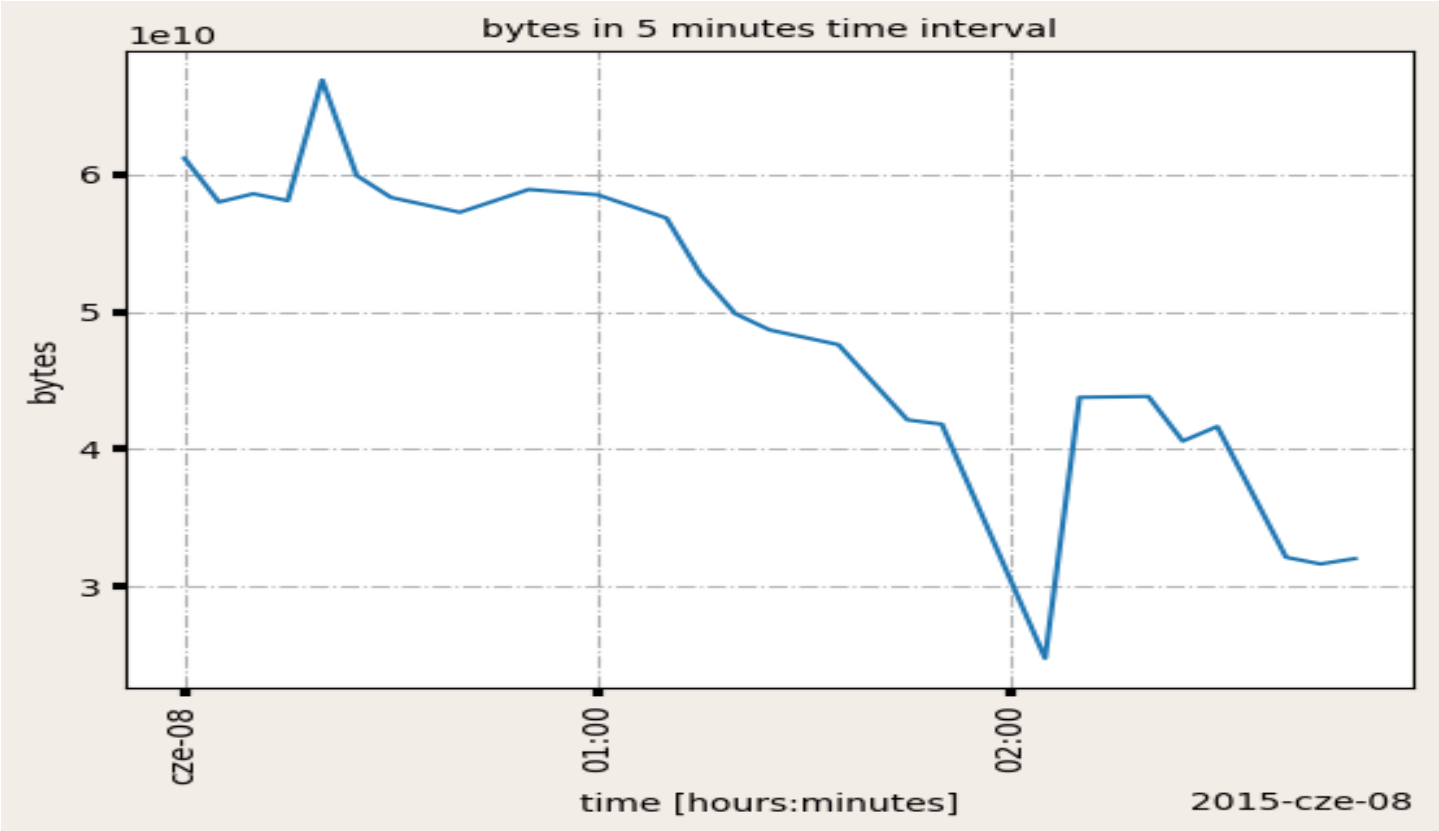
In the below presented chart we can see the relation between time and number of flows. Starting from midnight 08-06-2015 the amount of flows descreasing untill 6 a.m. after that we can observe constant growth and the pick is located somewhere about 1 p.m. By the midnight at the next day (09-06-2015) the amount of flows drops 40% compares to the highest amount at 1 p.m.



5.2 Bytes

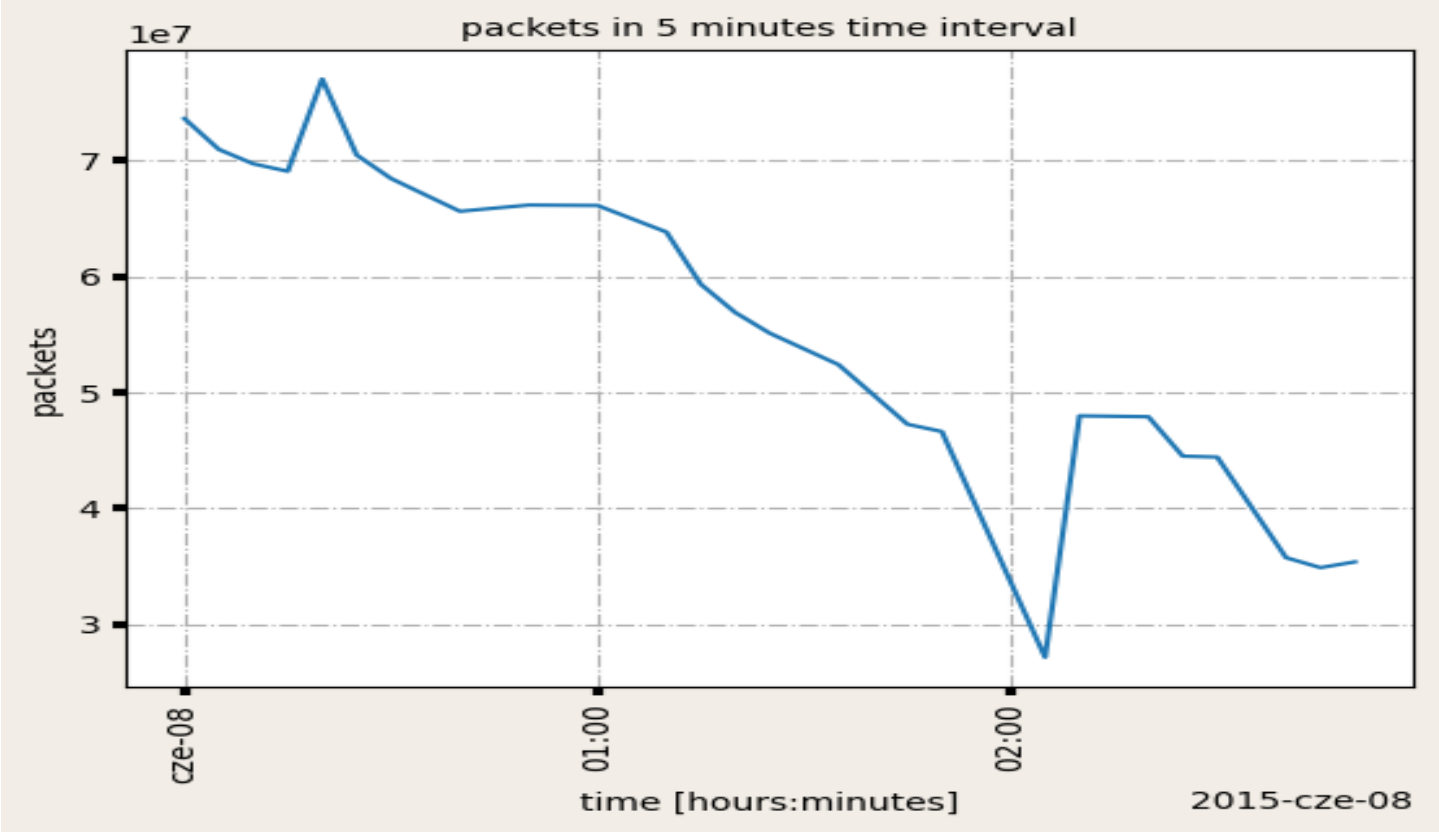
Byte is a unit of digital information that consists of eight bits. A single byte can be used to represent 256 different values.

Similary to the chart presented for flows we can observe the same shape for the chart ilustrated number of bytes per time relation. However here we can notice 4 picks between 12:00 and 18:00. The first one occured somwhere around 1 p.m., the second around 3 p.m. and the last 2 picks can be observe shrtly before 6 p.m. On the chart presented number of flows in time those picks were not so clearly presented. I suspect that this is because of the granularity of the data. Amount of bytes is by 5 orders of magnitude more than the number of flows for the same time (bytes - 10^11, whereas flows are 10^6).



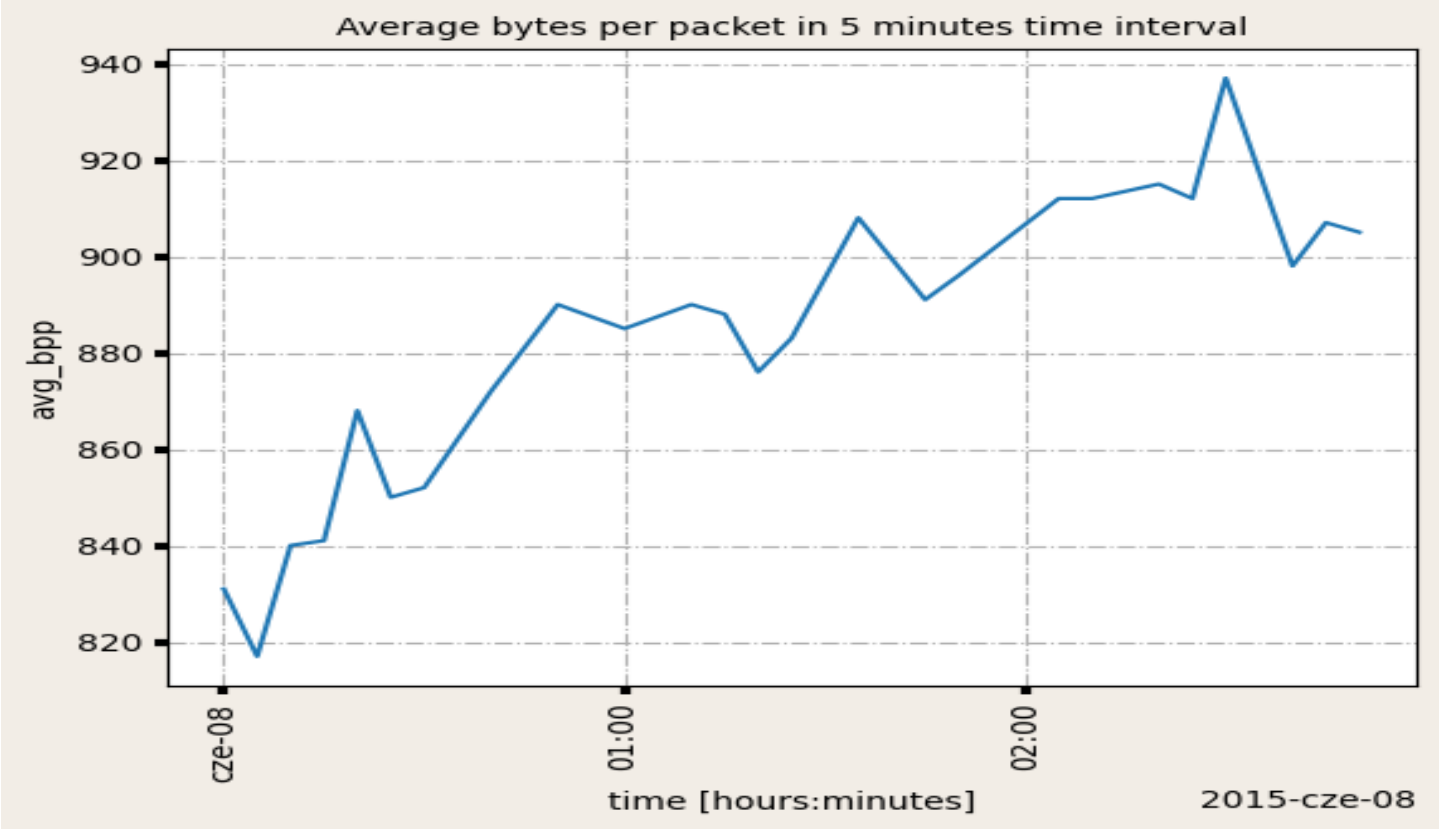
5.3 Packets

Packet is a collection of data that can be used by computers which need to communicate with each other, usually as part of a network. [8]
Each packet includes a source and destination as well as the content (or data) being transferred. The structure of the packet depends on the protocol used to transfer it.



5.4 Average bytes per packet

The chart below presents average number of bytes per packets in time. It is different then previously described charts, as between 6 a.m. and 12 a.m. grows, at 12 a.m. we can observe point of inflection after that average value of bytes per packets drops.



6. Conclusion

The relation between used data (define as number of flows, bytes or packets) in time interval for a regular working day is not typical from the perspective of household or corporate network. The first network might have thier pick in data traffick after working hours, where all are at home, where the second would have the top volumes between 9:00 and 17:00 where the most employees are at work. Students life varies from the other 2 mentioned here. They have classes in a different times therefore from 12:00 till midnight we can observe relatively constant and on high level data traffick in the university campus.

It is important to measure and understand what happens in the network, what are the volumes and types of traffic in order to properly manage it, for troubleshooting, detecting security incidents. This work definitely did not answer all those questions but it shows that with use of some tools like nfdump (to gather network logs) and python modules together with reliable hardware, which can handle such a big amount of data we are able to find some trends in usage of the network under our management.

7. References

[1] http://nfdump.sourceforge.net/
[2] Cisco Networking Academy, 7.1.1.1 Role of the Transport Layer
[3] https://en.wikipedia.org/wiki/Transport_layer
[4] https://www.privateinternetaccess.com/blog/tcp-vs-udp-understanding-the-difference/
[5] IANA Allocation Guidelines for TCP and UDP Port Numbers, February 18, 2008
[6] DongJin Lee, Brian E. Carpenter, Nevil Brownlee, "Observations of UDP to TCP Ratio and PortNumbers"
[7] Piotr Jurkiewicz, Grzegorz Rzym, Piotr Borylo, "Flow length and size distributions in campus Internet traffic"
[8] https://simple.wikipedia.org/wiki/Packet_(computing)