Masaryk University
Faculty of Informatics

Behavior-based Identification of Similar Hosts in Computer Network

Bachelor's Thesis

Pavel Novák

Brno, Spring 2020

Masaryk University
Faculty of Informatics

Behavior-based Identification of Similar Hosts in Computer Network

Bachelor's Thesis

Pavel Novák

Brno, Spring 2020

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Pavel Novák

Advisor: RNDr. Tomáš Jirsík, Ph.D.

Acknowledgements

Abstract

This work is devoted to explain one possible way of searching for similar devices on the network - similarity search based on network flows profiles. The first part focuses on the theoretical background. This is necessary for the implementation of a simple tool for similarity searching. This section likewise explains the fundamental principles of network flows. Here are similarity search techniques like local outlier factor or K - nearest neighbours explained as well. The second part of the thesis explains the implementation of a tool, which can find similar devices to a given one or, on the contrary, identify significantly differently behaving devices, based on their communication profiles.

Keywords

network flows, similarity search, local outlier factor, distance function, host profile, network traffic analysis

Contents

Introduction

Networks and network communication play a crucial role in today's world. They allow us to communicate and share information over long distances in real time. Almost every larger organization manages its network. Such a network can contain arbitrary many devices. These devices, like desktops, mobile phones, laptops, tablets or servers, are primarily designed to enable employees to do their jobs or provide many types of services. However, very often, they are used differently. Whether intentionally or unintentionally, these devices can be used for harmful and often illegal activities. Such devices typically exhibit behaviour on the network that is unusual and suspicious. Identifying such devices on the network is not easy at all. Network monitoring is one possible way to detect such behaviour. Through network monitoring, we determine who communicates with whom. Although the content of the communication is usually encrypted and hence unknown, even the fact that communication takes place can be suspicious. Likewise, excessive traffic or fluctuations in communication during the day or week may be considered suspicious. One possible way of network monitoring is a network flows monitoring. Aggregating and subsequently processing the obtained data creates a unique profile for each device in the network that characterizes it. The profiles thus obtained are easy to store and update. Subsequent comparison of these profiles can identify very differently behaving devices or groups of devices that behave similarly.

This thesis deals with ways of identifying devices in the network whose behaviour can be considered different, as well as searching for groups of similar devices. For this purpose are used so-called communication profiles. The way of creating these profiles is introduced in this thesis as well. For the purpose of this work were used an anonymized profiles obtained from the Masaryk University network. The metric space and a distance function is used to look for similarities in this data. Thanks to it, we can define the concept of distance between two vectors. This concept is essential as their distance determines the similarity of the two devices. A Local outlier factor (LOF) algorithm is used to identify different behaving devices in a dataset. Its output is a number that shows its difference from the devices in its vicinity.

It makes it very easy to identify a device that is not different within the whole dataset but only within the devices that should behave similarly.

The content of this work will also be the analysis of profiles using the tool created as a practical part of this thesis. Finally, the tool will be tested by experiments. This tool then could help, for example, network administrators to detect faulty devices in a network.

The thesis is organized as follows. In chapter 1 is described a network flows theoretical background. Network flows are a data sources to creating communication profiles, that is used in this work. Definition of network flow or other ways of network monitoring is described here.

Chapter 2 deals with creating communication profiles from network flow data. Definition of communication profile, possible monitored properties or profile updating is described here.

Chapter 3 covers topic of similarity search. Metric space and distance function concepts and K-NN or LOF algorithms are introduced here. Also, ways of measuring the distance between profiles is depicted here.

Chapter 4 describes the implementation of a tool to locate and compare similar devices and identify different devices. Introduction of used technologies such as work with databases or implementation of selected algorithms are there.

Chapter 5 shows the use cases and the experiments that were carried out using the created tool.

Summary of all the findings are then in the final Chapter 6.

1 Network Monitoring

Network monitoring is essential in today's world. It is used for pricing, usage analysis and last but not least also for security control. There are several approaches to network monitoring, each with its advantages and disadvantages. We can divide network monitoring into two basic classes.

Active monitoring is mainly focused on network scanning. Tools like Nmap or ping are used here. The disadvantage is that active monitoring increases network traffic. The advantage, on the contrary, is to get far more detailed data using this method. Usually, this type of monitoring is not very often used to monitor the network continuously because it increases network traffic too much. Active monitoring methods can be used for network topology discover or network asset management. Also can be used by attackers to scan the target network. Another option where active monitoring can be used is a network scan within an organization to determine the status of hardware or software.

Passive monitoring methods do not increase network traffic as much as active monitoring methods. For this reason, they are better suited to continuous network monitoring. Passive monitoring methods can be, for example, Deep Packet Inspection (DPI) or network traffic flows monitoring. DPI is a technique of seeing the payload of IP packets [1]. Deep packet inspection is time-consuming and computationally intensive, but on the other hand, it often provides valuable information that cannot be obtained in any other way. Network flow monitoring, on the other hand, is a useful technique even for large networks with massive traffic as it focuses on the analysis of communications, rather than the content of individual packets. [2]. This method is much more effective because only packet headers are processed. In our thesis, we use flows as a data source. Hence, we describe network flows in detail in the following sections.

1.1 Network Flows

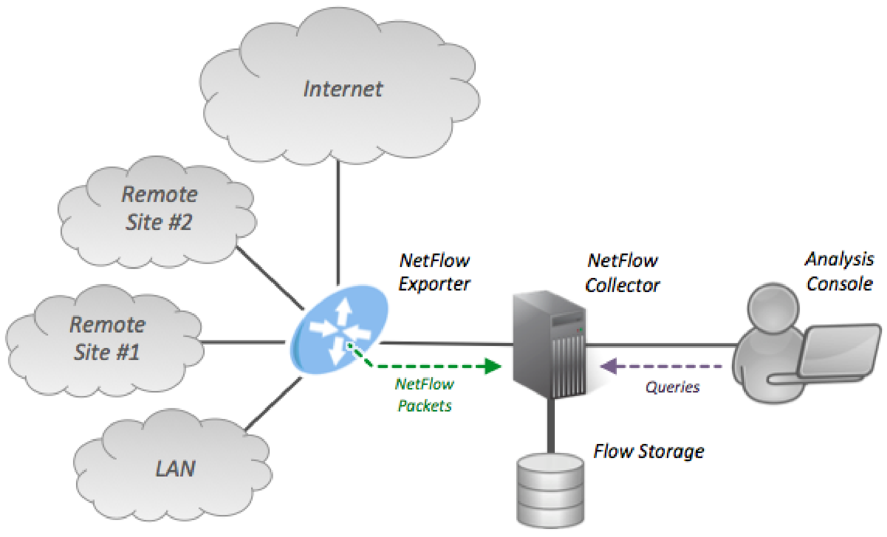
According to the RFC 7011, a network flow is defined as a set of IP packets passing an observation point in the network during a specified time interval. All packets belonging to a particular flow have a set of common properties [3]. These properties usually includes the source and destination addresses, the source and destination ports, or the protocol used. Data such as the number of bytes transmitted, the number of packets transmitted or the duration of the flow can be then collected.

Packets are captured at the observation point. Generally, the source from which the data is obtained is called the exporter. It is typically a probe that is connected with a device that duplicates traffic on the line (Traffic Access Point). However, also other network devices such as a router can do the probe function, but it is not customary to add a probe function to internal network elements (routers, switches) as it increases the performance requirements of the device. The probe then collects flow information and sends it to the collector at periodic intervals. Collector is usually a device with large storage capacity that collects data from one or several exporters. NetFlow v9 protocol is most commonly used to collect network flow records. It is the ninth version of the protocol developed by CISCO company. Other examples are NetStream, JFlow or IPFIX protocols[4]. For sending data from exporter to the collector, TCP or UDP protocols are used. Once data are sent to the collector, they are deleted from exporter's cache.

1.2 Capture of Network Flows

As already mentioned, a device called an exporter is used to capture data. It analyzes all packets and assigns them to flows. The flow is created when the exporter intercepts the first packet. Each additional packet that has the same set of common properties is then assigned to the same flow. The following data are usually recorded about the flow:

- Source and destination IP addresses
- Source and destination ports
- Flow duration
- Number of packets and bytes



[5]

Figure 1.1: The schema of NetFlow architecture

This approach of network monitoring does not interfere with network traffic at all. This is an advantage and disadvantage in once because anybody cannot find out that the traffic is monitored. Advantage because if someone knew that the network is being monitored, traffic could be directed differently and no undesirable behaviour could have been detected. Disadvantage because of privacy issues.

When the flow is terminated, it is sent to the collector. There are three options of how the flow can be terminated.

- Packet with FIN or RST flags is captured.
- *Inactive timeout* - Flow is terminated after a specific time when the probe does not capture any packet belonging to the flow.
- *Active timeout* - Active timeout is used in case of very long-lasting flows. The very long-lasting flows may never be exported, so usually after 300 seconds, every active flow is terminated and sent to the collector. Next incoming packet then starts the new flow. This ensures that each communication will be recorded as a flow

and sent to the collector no matter how long the communication takes.

Table 1.1: Network flows example

Start	Duration	Proto	Src IP : Src Port	Dst IP : Dst Port	Packets	Bytes	Flows
2019-01-21 14:00:00:156	0.156	UDP	239.36.55.235:55864	239.36.36.52:80	50	560	1
2019-01-21 14:00:00:228	0.000	UDP	239.36.36.36:4859	239.36.54.12:78555	1	48	1
2019-01-21 14:00:00:548	0.255	UDP	239.36.36.52:80	239.36.55.235:55864	120	10080	1

1.2.1 Export of Records

Recorded flows are sent to the collector as soon as enough records to fill in packets is exported, or a predefined timeout is triggered. The collector is a device with a large storage capacity that receives data from an exporter and stores and process them. Usually, one collector receives data from several exporters. The NetFlow standard (RFC 3954) does not specify a specific NetFlow listening port. The sending from the exporter to the collector is usually done on ports 2055 or 9555 or 9995 using User Datagram Protocol [6]. On the collector, the data is further processed by tools such as nfdump or nfsen or others.

1.3 Network Monitoring at Masaryk University

The Masaryk University network is an example of a vast network. Monitoring such a network is not easy at all, but the smart deployment of probes can monitor traffic even in such a big network. Figure 1.2 shows a scheme of the network monitoring in the MUNI network. By clever placing a few probes, it is possible to monitor the entire network effectively. Two probes that monitor a large part of the traffic are located on the connection of the Masaryk University network to the Internet. All traffic that is led from or to the university network is monitored on these probes. There is also a probe in each faculty in the MUNI network. This makes it possible to monitor faculty operations, even those that are conducted only within the faculty. Last but not least,

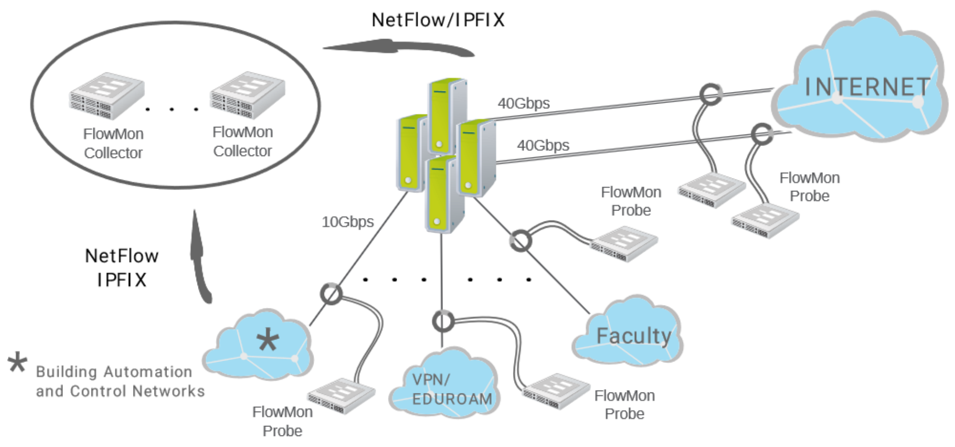


Figure 1.2: Network Monitoring On MUNI

communication via VPN and Eduroam is also monitored. All probes then send data to the collectors. The data from these collectors are then used for research purposes, traffic monitoring to ensure network security or network usage monitoring. Data that are used in the scope of this work is collected from the probe which takes place at the border of Masaryk University network and Internet, so communication from and to the network is visible here.

2 Communication Profiles

As mentioned in the previous chapter, network flow monitoring is a good way to monitor a vast network. The output is a large amount of data that should be processed so that it can be considered as communication profiles. A communication profile is a set of characteristics that determine the behaviour of a given device over a long period. We calculate these characteristics from the raw data that we obtain from the exported records of IP flows.

From the obtained data, it is then possible to calculate almost arbitrarily many characteristics, depending on which we are interested in. One of the possible ways to achieve various characteristics so that they can describe the device is to proceed through some layers of the ISO / OSI model. In the network layer category, we can, for example, collect information about the number of communicating partners or the number of countries to which the device communicates. Here we can also monitor how much the IP protocol version 4 and version 6 is used. In the transport layer, we can watch the characteristics such as the number of flows where one of the protocols of the transport layer was used. In particular, we get information for TCP, UDP and others. The application layer can then be represented by characteristics such as the top N ports used or use of specific ports.

Data aggregation is essential when creating communication profiles. Thanks to this, we obtain other characteristics such as sums, minima, maxima and averages. We receive these by simple aggregation of raw data. We can, of course, aggregate over any period. It is especially advisable to arrange so that the morning, afternoon and night periods are set aside separately. This gives us a good overview of the behaviour of the device during the day, and thanks to this, we can also determine which device category it can be.

Each type of device will show different behaviour during the day. For example, most workstations, where routine work is performed, should have the most traffic during the morning and afternoon, but at night, their network activity should be minimal. On the other hand, the server station will not show such significant fluctuations between day and night. Of course, you can aggregate not only over different sections of the day but also over the whole days. Here we can observe differences between working days and weekends, or between a typical day within the semester and the holidays.

From the characteristics obtained, we can then deduce more. The use of ratios between the acquired characteristics seems to be very suitable. In this way, we can get the average number of packets per flow, the ratio between the amount of communication during the day and at night, or the averages between the use of individual protocols of the network or transport layer and many others. For example, finding out that one device communicates for the most part using only the UDP protocol can tell us a lot, and it can differentiate the device in comparison with others.

To be able to use the communication profile created in this way, it is necessary to normalize it. The total number of transferred bytes will usually be many orders of magnitude higher than the number of flows. Such differences would then cause us problems when comparing communication profiles. Therefore, it is necessary to normalize each characteristic in the range 0 to 1. The standardization should then take place in the range in which we want to search.

3 Similarity Search

Similarity search is used in many places. It is used when searching for similar images, music, or documents. Similarity search is also used in criminal analysis when comparing fingerprints or faces. The goal of similarity search is not to find an exact match, but to compare objects and find the most similar. To achieve this, the concept of metric space and distance between two objects must be explained. Then the data can be searched, for example, to find K nearest objects to the selected one. The K-NN algorithm can be used for this purpose. The Local Outlier Factor (LOF) algorithm is then based on K-NN to identify outliers in the data file.

3.1 Metric Space

The first is to define the term metric and metric space. Metric can then be used to determine the distance between two vectors. Vectors do not have to have only numerical values but also discrete values (e.g. logical). Also for that vectors is useful to determine their distance.

Metric space is a pair $(\mathcal{M}, d)$ where $\mathcal{M}$ is a set of objects and $d : \mathcal{M} \times \mathcal{M} \to \mathbf{R}$ is a metric or a distance function on $\mathcal{M}$. The following axioms and constraints apply to the distance function d:

For any x, y, z $\in \mathcal{M}$:

- Non-Negativity $d(x,y) \geq 0$
- Symmetry $d(x,y) = d(y,x)$
- Identity $x = y \Leftrightarrow d(x,y) = 0$
- Triangle Inequality $d(x,z) \leq d(x,y) + d(y,z)$

3.2 Distance function

Distance is an essential concept to define the similarity between two objects. Distance function or a metric is function $d : \mathcal{M} \times \mathcal{M} \to \mathbf{R}$ as explained before. Using the distance function, we can measure the distance between two objects of a particular domain. The distance between two identical objects is zero. The higher the distance, the more different the two objects are. There are many different types of distances, here are some of them.

3.2.1 Minkowski Distance

Minkowski distance is the most general distance function and also the generalization of Euclidean and Hamming metrics. The p parameter is optional, and the higher p is, the higher emphasis is on more significant differences between the individual components of the vectors. If parameter p = 1, it is a Hamming metric. For p = 2, it is the Euclidean distance.

$$\mathcal{D}_m(x_1, x_2) = (\sum_{n=1}^{n} |x_{1i} - x_{2i}|^p)^{\frac{1}{p}} \tag{3.1}$$

3.2.2 Euclidean Distance

A basic metric that is probably the easiest to interpret geometrically. All points that have the same distance from point X are a circle centred at point X in 2D space. In 3D space, points with the same distance from point X make sphere centred at point X. vectors.

$$\mathcal{D}_e(x_1, x_2) = \sqrt{\sum_{n=1}^{n} (x_{1i} - x_{2i})^2} \tag{3.2}$$

3.2.3 Squared Euclidean Distance

Less computationally time-consuming variant of Euclidean distance, because there is no need of square root computation. However, it is not a metric in the real sense, mainly because it does not meet the triangular inequality. Thus, the quadratic Euclidean distance can be used when the relative comparison of the two values is decisive (which is, for example, the minimum distance classification), not the absolute value.

$$\mathcal{D}_{se}(x_1, x_2) = \sum_{n=1}^{n} (x_{1i} - x_{2i})^2 \tag{3.3}$$

3.2.4 Hamming Distance

Hamming distance is a linearized euclidean distance. This metric does not place as much emphasis on the significant differences in the individual vector components as the euclidean distance. It is also significantly computationally less time consuming than the Euclidean distance. This metric can be used to compare two binary vectors as well.

$$\mathcal{D}_h(x_1, x_2) = \sum_{n=1}^{n} |x_{1i} - x_{2i}| \tag{3.4}$$

3.2.5 Chebyshev Distance

This is the limit case of the Minkowski metric. It is used in computationally demanding cases where the computation of the Minkowski metric is unacceptable because it is too much time-consuming [7].

$$\mathcal{D}_c(x_1, x_2) = \max \forall i |x_{1i} - x_{2i}| \tag{3.5}$$

The distance function can then be used to implementation of several methods for finding similarity between two sets of vectors. In this work, the calculation of the Minkowski distance is implemented, so it is possible to calculate the Hamming, Euclidean and Chebychev distances using the parameter p. A square Euclidean distance is also implemented, but it is not used.

3.3 Query Types

There are many methods for measuring distances between two sets of vectors. Here are mentioned some of them.

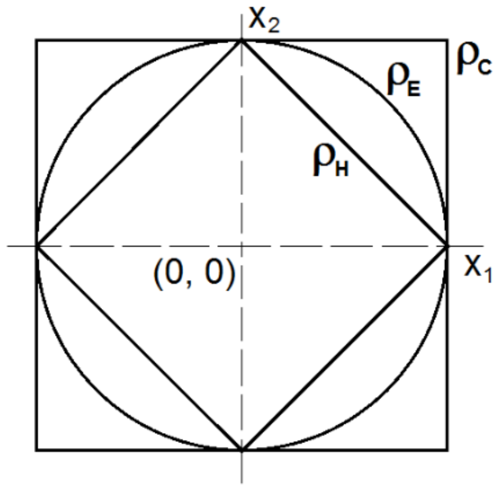


Figure 3.1: Unit circles of Minkowski distance for various parameter p [7].

The nearest neighbour method takes the smallest distance between x and y as the distance between the sets so that x belongs to one set and y belongs to the other set. The disadvantage of this approach is that this method is very prone to inaccuracies.

The furthest neighbour method is the exact opposite of the previous method. The distance between the sets is here determined by the maximum distance of the members x and y such that x belongs to one set and y to the other.

K - nearest neighbours is a method that defines as the distance between sets as the sum of the shortest distances x and y so that x belongs to one set and y belongs to the other set. It is a generalization of the nearest neighbour method.

In the *average distance method*, as the name suggests, the distance of two sets of vectors is defined as the average value of all distances between all members of both sets.

The last-mentioned method is the *centroid method*. Here the distance is determined as the distance of the so-called representative vectors. A representative vector may be a centroid. This is determined by the mean, median, or other quantity that in some way characterizes the entire set [7].

K - nearest neighbour is an algorithm that finds the nearest neighbour for a given point. It is a method that is primarily used for supervised learning in machine learning approach. This method finds the nearest neighbour for the object using the specified metric. In machine learning, it is then used to classify an object. However, this algorithm can also be used differently for anomaly detection within a given dataset, since the local outlier factor algorithm uses the K - NN to detect outliers.

3.4 Anomaly Detection

The ability to detect strange or different behaviours is useful in many human activities. We can see it in banking, where it can be used to detect frauds or thefts of bank accounts. In medicine, it can be used to detect health problems based on unusual symptoms. It can also be used to detect erroneous measurements of devices such as sensors. The possibility to detect these outliers then gives us a considerable lead, and it is possible, for example, in the case of banking, to take actions to mitigate the damage.

Outlier is defined as an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism [8].

There are a number of methods to detect outliers. In general, these methods can be divided into the following categories.

- *Statistical Methods* are based on the assumption that the data is generated by a statistical model and has, for example, a normal distribution.

- *Proximity - Based Methods* use the assumption that an object is an outlier if the proximity of the object to its neighbours significantly deviates from the proximity of most of the other objects to their neighbours in the same data set.

- *High - Dimensional Methods* - For example, an ABOD method - an object is an outlier if most other objects are located in similar directions.

3.4.1 Local Outlier Factor

The method used to detect outliers (and also inliers) in a given dataset. It uses the K - NN algorithm to calculate it. Its most significant advantage is that it does not (or does not have to) take into account the entire dataset, but only the immediate vicinity of the point. This also detects points that behave normally throughout the dataset and therefore would not be detected by other methods but are very different within their surroundings.

The result is any positive number for the given point. The significant value of this number is 1. If the result is greater than 1, it is an outlier. If it is less than 1, it is an inlier. The disadvantage of this method is that it is not possible to say which value exactly is already an outlier. Of course, this problem can be partly solved by statistical processing.

To understand the calculation of the local outlier factor, it is necessary to explain the following terms.

- $dist_k(o)$ - *K - Distance* is distance from o to its K-th nearest neighbour.
- $N_k(o)$ - Set of k nearest neighbours of o
- $reachability - distance_k(o, p) = max\{dist_k(p), d(o, p)\}$ - distance of o and p but at least k-distance of p
- $local - reachability - density_k(o) = \frac{|N_k(o)|}{\sum_{p \in N_k(o)} reachability - distance_k(o,p)}$ - an inverse of the average reachability distance of o from its k - nearest neighbours.
- $local - outlier - factor_k(o) = \frac{\sum_{p \in N_k(o)} \frac{lrd_k(p)}{lrd_k(o)}}{|N_k(o)|}$

The definition of local outlier factor implies that it is a fraction of the average local reachability density of the nearest neighbours, and the local reachability density of the object o. The optional parameter k determines how big the surrounding area is taken into account when calculating the local outlier factor. Figure 3.2 shows the LOF calculation for point A. For this point, the nearest neighbouring points are found, three in this case, which are those that are connected to point A by a red line. For each of these points, the density of its surroundings is then calculated and then compared with the density of the surroundings of point A. As can be very clearly seen, the density of points around point A is significantly less than the density of its nearest neighbours. Therefore, this point is considered to be an outlier.

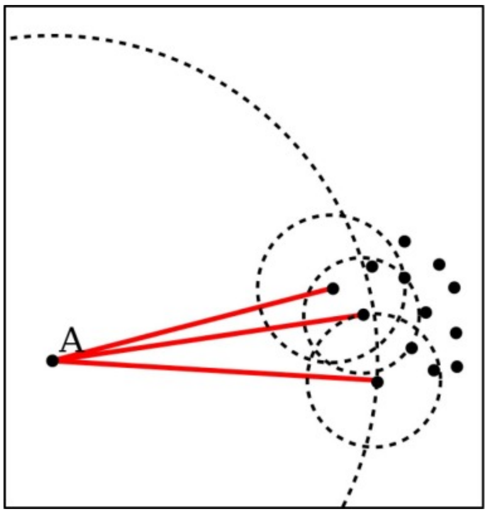


Figure 3.2: Local Outlier Factor.

4 Implementation

As mentioned earlier, it would be useful for many people managing large networks to have a tool that can monitor the network and search for suspicious-behaving devices. These devices could then be analyzed in more detail by the administrators and, for example, look for the most similar to these devices. Precisely such a tool was implemented in this thesis. The tool can analyze individual IP addresses within specified ranges. It is also able to search within the specified ranges for the strangest behaving devices, or, conversely, create a characteristic device for a given range. This tool works with communication profiles of individual devices, which are processed using algorithms K-Nearest Neighbors and Local Outlier Factor. Based on the results of these algorithms, similar or remote devices are then identified. This chapter deals with the implementation details of the tool itself, the technologies and procedures used. Examples of practical use of such a tool are presented in the next chapter.

4.1 Design

Within this work, a tool that offers the similarity search functionality mentioned earlier 3 was created. This tool processes computed communication profiles, which are stored in a database. A specific description of the database is in the following section. The tool accepts queries through the REST API and offers a basis for three use cases. The first use case is a KNN query or a query to find the most similar devices. This query then returns the most similar devices. The second use case is to search for the outliers. This search is performed using the Local Outlier Factor (LOF) algorithm. The result of this query is devices that behave very differently within a given IP range. The LOF values can then be used to determine which deviates the most from the normal. The last use case is a range query. The result is a set of devices, which are not far than the specified distance.

Since the calculation of communication profiles is not part of this work, it is necessary for proper functionality that the database contains already calculated profiles. Above them, it is then able to search for and identify both similar and different devices. In our work, we defined the

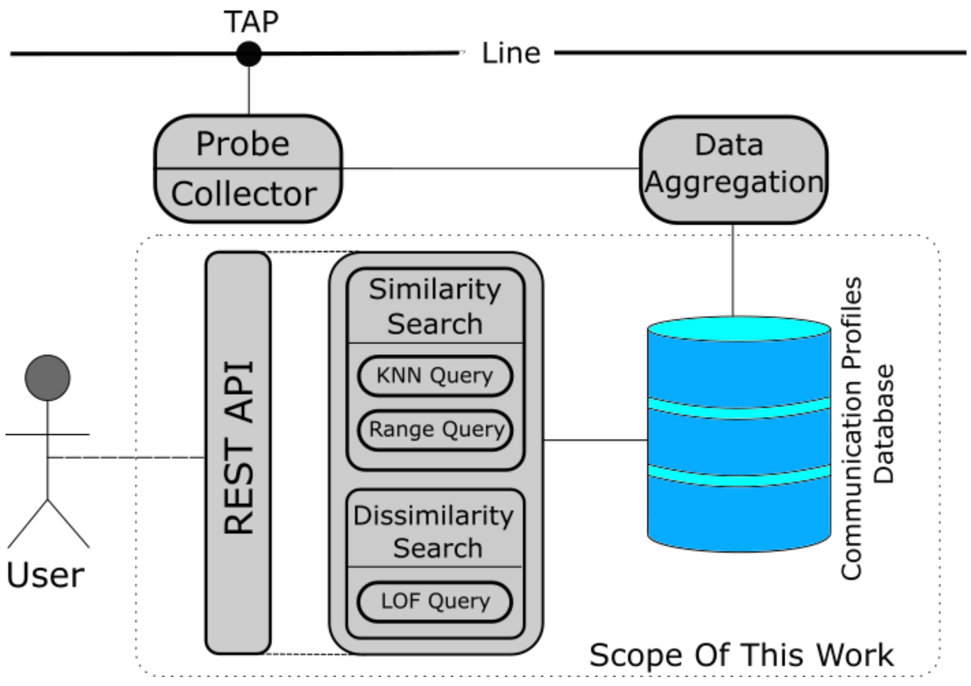


Figure 4.1: Scheme of the tool.

characteristics of host profile and based on the definition, a framework for the profile computation was implemented as part of another project. The scheme of a design is shown in figure 4.1.

4.2 Communication Profiles

Communication profile characterizes a device's behaviour. The profile is aggregated data from raw network flows. Characteristics showed below is calculated every 4 hours. A sliding window that contains 4-hour characteristics from the previous 24 hours then defines the resulting communication profile. An example of profiles is shown in the Table 4.1.

- Flows sum
- Packets sum
- Bytes sum

- Communication peers sum
- Average duration of a flow
- Used ports sum

Thanks to the calculated 4-hour profiles, it is possible to include in the resulting profile also various derived statistical quantities. Features included into the final profile are shown below. Profiles are calculated separately for in and out directions. Resulting communication profile is shown in table 4.1.

The characteristics shown here are used for this work, but they are far from exhaustive, and many can be added. Among the options that can be added to the profiles appears the ratio between weekend and business days, or the rate of behaviour change of the device over time.

- Flows sum
- Packets sum
- Bytes sum
- Flows average
- Packets average
- Bytes average
- Flows max
- Packets max
- Bytes max
- Flows std. deviation
- Packets std. deviation
- Bytes std. deviation
- Flows 95 centile
- Packets 95 centile
- Bytes 95 centile
- Communication peers average
- Ports average
- Communication peers max
- Ports max
- Communication peers min
- Ports min

4.2.1 Normalization Of Communication Profiles

For practical use, it is necessary to normalize each property of this profile to the interval $[0, 1]$ using the formula

Table 4.1: Communication profiles example

IP	Direction	Flows sum	Flows average	Avg communication peers	Average duration
239.36.255.11	out	20795	594.142	6.342	14816.805
239.36.255.11	in	20233	578.085	6.371	14788.522
239.36.255.88	out	4827	160.9	1.4	1378.695
239.36.255.4	in	137	7.611111	1	16.86

$$a_i = \frac{a_i - min(a)}{max(a) - min(a)} \tag{4.1}$$

where a is a specific column and a_i is a value of column a at row i.

The normalization is realized by the materialized view of the used database.

There are many possible characteristics from which the resulting profile can be calculated. However, they differ mainly in the ease of obtaining them. Obtained characteristics can be divided into groups according to which part of the communication it concerns.

- General
  - Total number of communicating partners
  - Average bytes per second
  - Average packets per second
  - Average flows per second
- Advanced / Timing
  - Morning/afternoon/night % of traffic by flows (0-8h; 8-16h; 16-0h)
  - Morning/afternoon/night % of traffic by packets (0-8h; 8-16h; 16-0h)
  - Morning/afternoon/night % of traffic by bytes (0-8h; 8-16h; 16-0h)
  - Average duration of outcoming flows
  - (Average time when host doesn't communicate)

  - Morning/afternoon/night ratio
  - Business days/weekend ratio

- Network layer
  - % of traffic to local network
  - % of traffic to external network
  - Total number of in-coming traffic
  - Total number of out-coming traffic
  - In/Out sent ratio

- Transport layer
  - % of TCP
  - % of UDP
  - % of ICMP
  - % of other protocols

- Application layer
  - Top ports
  - % of in-coming traffic to top ports
  - % of out-coming traffic to top ports
  - % of in-coming traffic to well known ports
  - % of out-coming traffic from well known ports

As already mentioned, not all such characteristics can be easily obtained only by observing network flows. There are also mentioned ratios, which are generally a useful characteristic for monitoring the behaviour of the device and are very easy to obtain and compute them. The division into following categories then allows us to view the communication profiles from many different perspectives, depending on what we are interested in.

4.3 Views

The view-based approach is very beneficial as it allows us to view data from different views, depending on what is important to us. Each predefined view gives a different weight to the monitored properties of the communication profile. Such views can be defined arbitrarily many, and also give each property any weight. The distance of the two devices is then measured as the weighted average of the distances between the individual groups of properties described in Table 2.4. Three views are predefined in the scope of this work. The overall view gives all categories the same weight and is default. The traffic volume view puts most emphasis on the amount of traffic, especially the number of communication partners, and the amount of data sent and received. The last view focuses on the application layer, giving the most weight to features such as the amount of communication on known ports.

Table 4.2: Predefined views - weights 4.2

	General	Advanced	Network	Transport	Application
Overall	20%	20%	20%	20%	20%
Traffic	20%	10%	50%	10%	10%
Application	5%	5%	10%	30%	50%

4.4 Database

Due to the amount of data, it was necessary to use a database to store communication profiles. For this purpose, the PostgreSQL version 12 database was used, which runs on the same machine where the tool is running. The raw data were first normalized by column. Subsequently, the profiles were linked so that each row in the table corresponds to one row in the database. For devices where only one direction was recorded, the corresponding missing values were replaced by 0. Thus, the resulting table had 25,856 rows (number of unique IP addresses) and 63 columns (31 for each direction and IP address).

The psycopg2 [9] package was used to connect to the database server from the program and execute arbitrary SQL statements over the database using the dbexecute method. There were predefined methods in the class *SQLCommands*, which performed the SQL queries and returned the result.

The predefined SQL commands are in class *sql_commands*. Here are examples of some methods in this class.

```
class sql_commands:

    def __init__(self):
        self.db = database.db_connection()

    def get_unique_ip(self):
        return self.db.get_data(
        "SELECT␣DISTINCT␣hosts_ip_address␣"
        "FROM␣host_profile")

    def get_ip_range(self, ip_range):
        return self.db.get_data(
        "SELECT␣DISTINCT␣ip_address␣"
        "FROM␣final_profiles␣"
        "WHERE␣ip_address␣LIKE␣'{}%';"
        .format(ip_range))

    def get_host_profile(self, host):
        return self.db.get_data(
        "SELECT␣*␣"
        "FROM␣final_profiles␣"
        "WHERE␣ip_address␣=␣'{}';"
        .format(host))
```

For ease of adding other profiles, all database elements except the base table with communication profiles were implemented using views. This made it very easy to add, remove or change communication profiles. Since the program used data only from views, it was always up-to-date and took into account the current state of the underlying table with communication profiles.

4.5 Similarity Search Functions

4.5.1 K-NN Implementation

The K nearest neighbours algorithm is primarily used in the supervised learning method of machine learning. Use it to classify patterns into categories according to the nearest neighbours. However, it is also the basis for the Local Outlier Factor algorithm. In this work, the K-NN algorithm is used to find the closest and therefore, the most similar devices based on their communication profiles. The implementation of this algorithm is relatively easy. It is a simple measurement of the distance using the selected metric from Chapter 3 and then sorting these values from the smallest to the largest. When determining the K - nearest neighbours, it is necessary to measure the distance to all devices, which affects performance and speed. It can also be used the views when measuring the distances, which are described in detail in the following section.

4.5.2 Local Outlier Factor Implementation

Local outlier factor is an algorithm that is primarily used to identify so-called outliers or devices that differs very much from others in a dataset. This program using the Local Outlier Factor algorithm precisely for this purpose. However, since the calculation of the Local Outlier Factor is time-consuming, this algorithm can only be used for searching in a limited number of devices. However, this does not matter for practical use. The local outlier factor can be successfully used, as shown in the next chapter. Although there are libraries that already implement the local outlier factor, for example, the PyOD library, within this thesis, was created its own implementation that implements the procedure described in Chapter 3. The reason was to compare performance with library implementations.

4.6 REST Api

Flask Restful [10] was used as a user interface. It is a Representational State Transfer architecture that allows access to resources on the server. Usually, it uses four basic methods - GET, PUT, POST and DELETE.

Table 4.3: Example of LOF values

IP address	Local Outlier Factor
239.36.249.57	5.53
239.36.249.168	1.07
239.36.249.207	1.14

Table 4.4: Comparison of communication profiles from the table 4.4

IP address	Flows sum in	Flows sum out
239.36.249.57	979 313	975 190
239.36.249.168	7798	0
239.36.249.207	7919	0

In the implementation of this tool, only get methods were used, and access to different data was realized using different endpoints. The basic endpoints are the following:

- **KNN endpoint** - This endpoint perform K-NN query for a given host. The query is in form:

  / knn / view / host / ip-range / k / t

  where view is one of "overall", "traffic", "application". Host is a host IP address, IP range is a range that is considered when computing KNN, K specify the number of the nearest neighbours and t is a threshold or maximal distance that is included into the result.

- **LOF endpoint** - This endpoint count Local Outlier Factor for a specific host in a given IP range. The query is in form:

  / lof / host / ip-range

  where the host is a host IP address, and IP range is a range that is considered when computing LOF.

- **Detail endpoint** - Shows a communication profile of a specific device. The query is in form:

  / detail / host

- **Range endpoint** - Shows all devices that are closer than a specified distance (range). The query is in form:

  / range / host / range

5 Experiments

This chapter presents several experiments that have been performed with the created tool to verify its effectiveness. To perform the experiments, it was necessary to deanonymize the dataset partially. Two ranges were identified, one containing workstations and the other containing servers. Subsequent experiments were performed over these two ranges. For practical use, it was necessary to automate the search for devices in some way. This was accomplished by so-called scanning. Local outlier factor for a small k value for each address in the range is counted. Suspicious devices whose local outlier factor values were high were then tested for a larger k value and then identified as the most suspicious device.

5.1 Dataset

The data used to implement and test the tool was collected from Masaryk University's network. The captured data originates from probes located on the connection of the Masaryk University network to Internet service provider. It captures traffic to and from the university network. Thus, communication within the faculties of the university is not recorded. In total, the dataset contains profiles of 25 856 unique IP addresses. Because of privacy issues, all addresses are anonymized. Each device has its own communication profile, as is described in chapter 2. Although the communication profiles are anonymized, IP address ranges have been preserved so that experiments can be performed. This was used in the experiments described in the next chapter.

5.2 Identification of the most similar devices

Identification of the most similar devices was carried out using K-NN query. The K-NN query returns other devices sorted by distance. Since the numerical values of distance are not very telling, distances within the range are classified into four categories based on distances between every couple of devices.

- Very close - Distance is closer than 25% of all distances.
- Close - Distance is between 25% and 50% of the closest distances.
- Far - Distance is between 50% and 75% of the closest distances.
- Very Far - Distance is far than 75% of all distances.

This categorization of distances tells us much more about how similar devices really are. Table 5.1 shows us 3-NN of device 239.36.249.57, which is considered to be an outlier. This device also has a very high Local Outlier Factor value. Table 5.2, on the contrary, shows 3-NN of device 239.36.249.170, which is considered to be an inlier.

Table 5.1: 3-Nearest Neighbours Of 239.36.249.57 - Outlier

IP Address	Distance
239.36.249.247	Very Far
239.36.249.61	Very Far
239.36.249.60	Very Far

Table 5.2: 3-Nearest Neighbours Of 239.36.249.170 - Inlier

IP Address	Distance
239.36.249.145	Very Close
239.36.249.14	Very Close
239.36.249.111	Very Close

5.3 Identification of the most different devices

The identification of the most different devices within the IP range was realized using the Local Outlier Factor. Therefore, the scan function was implemented for this purpose. This function first calculates a local outlier factor with a small k value for each address. Every suspicious device is then verified for a larger k.

5.3.1 Server Identification In Work Stations IP Range

In this experiment, a dummy device was added to the workstation IP range. The profile of this device was exactly the same as the device that was in the IP address range of the servers and showed quite usual behaviour here. The profile of this device is shown in 5.3 and its anonymized IP address is 239.36.249.52. The Local Outlier Factor of this device among servers was 1.1114. A dummy device with the same profile as the server was added to the IP range of the workstations. Then the entire range of workstations was scanned using the scan function. Quite clearly, this device was identified as suspicious, with a very high Local Outlier Factor. The scan function output is shown in 5.4.

Table 5.3: Profile Of Server Device

239.36.249.52	Flows sum	Packets sum	Bytes sum	Communication peers avg
IN	313 325	8 673 620	1 062 180 000	599.57
OUT	286 137	17 552 000	23 058 300 000	409.62

Table 5.4: Scan Work Stations

IP	Local Outlier Factor
239.36.202.145	189.67
239.36.202.151	147.43
Fictive device	37.23

First two devices from 5.4 has very high LOF value also within server (239.36.249/24) range, so these devices behave very unusual.

5.4 Comparison of devices in a different range

To verify the functionality of the tool, machines from the server IP range have been tested in the context of workstations. For each server-like machine, the LOF was calculated as if it were within the workstation range. It has been assumed that the vast majority of such machines have been identified as outliers, with very high LOF coefficient.

5.5 Performance

Performance is crucial for practical use, whether calculation time, accuracy, or both, are considered as performance. In this section, the performance of the created tool will be analyzed, both in terms of time and accuracy. Since many external libraries implement the algorithms in this work exists, the tool will also be compared with them.

5.5.1 Time Complexity - KNN

The query time of a KNN depends mostly on one factor, which is the number of devices between which a similar device is sought. Since it is not possible to determine whether the nearest neighbours have already been found at any stage of the calculation, it is necessary to calculate the distance to all other devices. Thus, the complexity increases linearly with the number of devices and is constant for a given number of devices.

5.5.2 Time Complexity - LOF

The identification of a point as an outlier depends on the size of the neighbourhood that is taken into account in the calculation. This is called the K-neighbourhood and contains the K - nearest neighbours of the given point. However, computational complexity increases with the size of the K-neighbourhood.

Considering that the dataset size is 256 addresses, the LOF calculation for a point A takes *KNN-QueryTime* to find the K-neighbourhood of A. Then for each point B in K-neighbourhood of A, K-neighbourhood of B is counted, which takes k**KNN-QueryTime*. Finally, for each point C in K- neighbourhood of B, K-neighbourhood of C is counted, which takes also k**KNN-QueryTime*. Overall, the total time-complexity is $\mathcal{O}(k^2)$

5.5.3 Accuracy

As in the time complexity, where the k parameter has a significant effect on the calculation time, in case of accuracy plays an important role as well. Depending on the magnitude of the parameter k and hence the neighbourhood, the remoteness of the point changes.

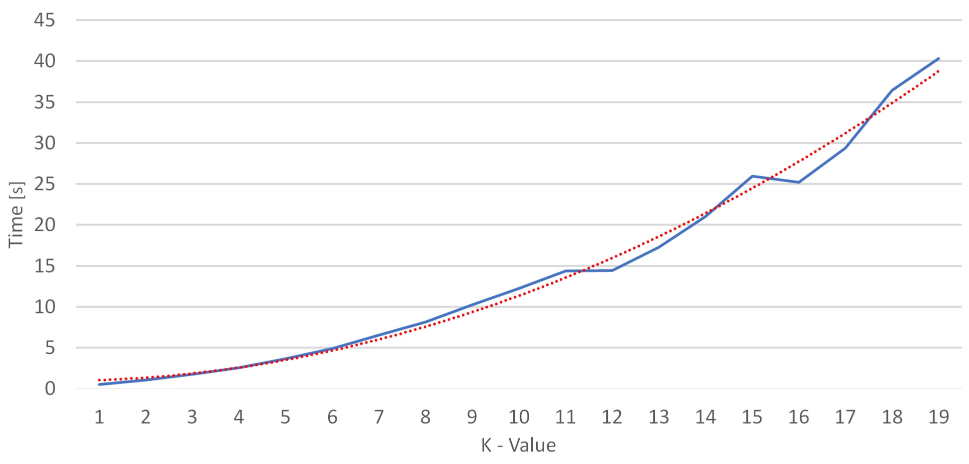
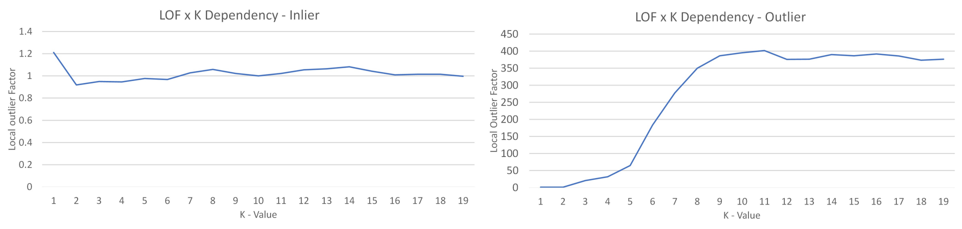


Figure 5.1: Lof Caluculation Time Dependency



(a) Lof Accuracy - Inlier (b) Lof Accuracy Outlier

Figure 5.2: Lof Accuracy - Comparison

Figure **??** shows the dependence of LOF on the K parameter for the point that is considered to be an outlier, and below for point that is considered to be an inlier. There is a clear trend for both points. While the LOF value for the inlier does not change much with the k parameter, in case of an outlier, it is different and increases very quickly for the first few values for LOF. This may not always be true. As the parameter k increases, the LOF value of the outlier may start to decrease over time, and the LOF value of the inlier may increase. The value of K determines how much context of the surrounding points we take into account when we consider whether it is an outlier or an inlier. Furthermore, while one point may be inlier within a small neighbourhood, it may be an outlier within the entire dataset and vice versa.

5.6 Evaluation

Tests have shown that similar and also different devices can be detected based on communication profiles. The accuracy of such a determination depends very much on the characteristics of the communication profiles used. There can be many of these characteristics, and the accuracy varies depending on which ones are used. Within the experiments, three tests were performed. The first test was to identify the most similar device to the given one. Using the KNN algorithm, devices most similar to the given one were found, which was also shown in the comparison of communication profiles. This test was performed both for the device that was known to be inlier and also for a device that was considered to be an outlier. The test results also corresponded to this, as the most similar device for the inlier was significantly closer than for the outlier.

The second test was aimed at verifying the ability to identify a different device. Knowledge of two IP ranges was used for this, one containing work stations, other containing servers. One device from the range of the server, which behaves normally within its IP range, was inserted into the IP range of workstations. This device was then correctly identified as a suspect using the Local Outlier Factor algorithm, with a high LOF value.

The last test, which aimed to verify the correct functionality of the tool, was a test in which the LOF algorithm was applied to all devices in the server range. The result was unambiguous, namely that the average LOF value in this test was significantly higher than the average LOF value for a workstation in the same range.

6 Conclusion

This work aimed to investigate and put into practice procedures for identifying similar devices on the network using network flows and communication profiles. This approach could greatly facilitate the way a computer network is monitored. The simplicity and speed of network flows also allows this approach to be applied to a wide range of computer networks, be it the amount of network equipment or the volume of data transferred. The ability to automate the detection and identification of suspicious devices then allows this tool to run continuously in the background and continuously monitor the network being monitored. However, when working with such a tool, it is necessary to think about privacy issues. Although communication profiles do not follow precisely for each device, its communication partners, and so it is not possible to find out who communicated with whom it is, to a certain extent a violation of privacy. Here the problem was solved by anonymization of data, but in a real deployment, it would be necessary to think about this as well. In this work was explained the approach of computer network monitoring using network flows. Besides, other network monitoring options, such as packet depth analysis, were briefly outlined. Furthermore, the principle of communication profiles, their creation and content was introduced. This approach offers a considerable variability and very many possibilities on how to adapt these profiles to specific needs depending on the monitored characteristics included in the profile. The third part deals with the similarity search in data. The concept of metric space and distance function are explained. These two terms are essential as they allow us to define the distance of two number vectors. Furthermore, since the communication profile can also be viewed as a numeric vector, the distance between two devices can be defined based on their profiles. There are also introduced two algorithms, which are then used later in this work, namely the KNN and LOF algorithms. Chapter 4 explains the implementation details of the tool itself. There is introduced used database and work with it, as well as the implementation of the algorithms mentioned above. A crucial part of this chapter is the explanation of the concept of so-called views, which is also used in this work. Finally, the performance of the solution used is evaluated, both in terms of time performance and in terms of the ability to identify suspicious devices. The last chapter presents the experiments that were performed with the tool in this work. All experiments are focused mainly on testing the capabilities of this tool.

There are many possibilities to continue in this area. The first option seems to be the expansion of communication profiles. Here, it seems to be a promising opportunity to incorporate in the communication profiles characteristics more focused on who communicates explicitly with whom. Another possibility of further development could be the use of other, more efficient, algorithms for detecting similar or different devices.

Bibliography

1. SVOBODA, Jakub. *Analýza síťového provozu metodou hloubkové inspekce paketů*. 2014. Available also from: `https://is.muni.cz/th/ql57c/`. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Supervised by Pavel ČELEDA.
2. HOFSTEDE, Rick; ČELEDA, Pavel; TRAMMELL, Brian; DRAGO, Idilio; SADRE, Ramin; SPEROTTO, Anna; PRAS, Aiko. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*. 2014, vol. 16, no. 4, pp. 2037–2064.
3. CLAISE, B.; TRAMMELL, B.; AITKEN, P. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. 2013. Available also from: `http://tools.ietf.org/rfc/rfc7011.txt`. RFC. IETF.
4. FINDURA, Lukáš. *Analýza síťového provozu: podobnostní vyhledávání*. 2015. Available also from: `https://is.muni.cz/th/wqcnt/`.
5. CZUDEK, Marek. *Detekce síťových anomálií na základě NET-FLOW dat*. 2013. Available also from: `https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=118994`. Master's thesis. VUT, Brno.
6. *Cisco Press*. Available also from: `https://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=4`.
7. RNDR. MARTIN KOMENDA, Ph.D. *Podobnosti a vzdálenosti ve vícerozměrném prostor*. Available also from: `https://is.muni.cz/www/98951/41610771/43823411/43823458/Analyza_a_hodnoc/44563155/56049312/Vicerozmerky_-_kap4_-_metriky_final.docx`.
8. HAWKINS, Douglas M. *Identification of outliers*. Springer, 1980. ISBN 978-94-015-3994-4. Available also from: `https://www.researchgate.net/publication/203917713_Outlier_detection`.
9. *psycopg2*. Available also from: `https://pypi.org/project/psycopg2/`.

10. *RESTful*. Available also from: `https://flask-restful.readthedocs.io/en/latest/`.