

CZECH TECHNICAL
UNIVERSITY IN PRAGUE

FACULTY OF MECHANICAL
ENGINEERING



BACHELOR THESIS

2022

ARTA RIZVANOLLI

CZECH TECHNICAL UNIVERSITY IN PRAGUE FACULTY OF
MECHANICAL ENGINEERING

Department of Information and Automation Technology

Application to monitor the availability of remote
servers

Arta Rizvanolli

Supervised by: Ing. Matouš Cejnek, Ph.D.

2021/2022



BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Rizvanolli Arta** Personal ID number: **476522**
Faculty / Institute: **Faculty of Mechanical Engineering**
Department / Institute: **Department of Instrumentation and Control Engineering**
Study program: **Bachelor of Mechanical Engineering**
Branch of study: **Information and Automation Technology**

II. Bachelor's thesis details

Bachelor's thesis title in English:
Create an application to monitor the availability of remote servers

Bachelor's thesis title in Czech:
Vytvořte aplikaci monitorující dostupnost vzdálených serverů

Guidelines:
• Create a configurable script that can run nonstop to monitor availability of remote servers, store the observed data in logs
• Create a framework for mathematical evaluation of the stored data - statistical analysis of the network integrity in various time windows
• Create simple user interface to access the data evaluation and export activity reports

Bibliography / sources:
[1] Fielding, Roy, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. "Hypertext transfer protocol—HTTP/1.1." (1999).
[2] Gralla, Preston. How the Internet works. Que Publishing, 1998.

Name and workplace of bachelor's thesis supervisor:
Ing. Matouš Cejnek, Ph.D., U12110.3

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **29.10.2021** Deadline for bachelor thesis submission: **20.01.2022**

Assignment valid until: _____

 _____
Ing. Matouš Cejnek, Ph.D.
Supervisor's signature


 _____
Head of department's signature

 _____
prof. Ing. Michael Valášek, DrSc.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

_____ Date of assignment receipt

 _____
Student's signature

Statement

I declare that I have worked out this thesis independently assuming that the results of the thesis can also be used at the discretion of the supervisor of the thesis as its co-author. I also agree with the potential publication of the results of the thesis or of its substantial part, provided I will be listed as the co-author.

Prague,

.....
Signature

Acknowledgement

I would like to thank my supervisor Ing. Matouš Cejnek, Ph.D. for his constant support and guidance. I also want to thank my family, for their emotional and material support during my studies.

Abstract

This paper describes the creation of a fully extensible Python 3 application to monitor the availability of remote servers. The motive is to offer an alternative solution within the field that can be easily customized. The monitoring is done by utilizing ping. The application is also programmed to store and analyse the data obtained. This paper can also be useful as a guide for the application.

Table of Contents

1.	Introduction.....	8
2.	Theoretical part.....	10
2.1.	How does the internet work?.....	10
2.2.	Client-Server model	14
2.2.1.	Server Monitoring	14
2.3.	Internet Control Message Protocol (ICMP)	15
2.3.1.	Ping as a monitoring tool.....	15
2.4.	Storing data	16
2.4.1.	JavaScript Object Notation (JSON).....	16
2.4.2.	Comma Separated Values (CSV)	17
2.4.3.	Database	18
2.5.	Data Analysis	19
2.5.1.	Graphic display/data presentation	19
2.6.	User interface	19
2.6.1.	Hypertext Markup Language (HTML).....	19
2.6.2.	Cascading Style Sheets (CSS)	20
3.	Practical part	21
3.1.	Overview	21
3.2.	Data Acquisition	22
3.2.1.	Methods	22
3.2.2.	Core loop.....	22
3.3.	Storing of the data	25
3.3.1.	CSV Writer----	26
3.3.2.	CSV Reader.....	27
3.3.3.	Engine.....	28
3.4.	Graphical Analysis of data	29
3.4.1.	Histogram.....	30
3.4.2.	Linear plot.....	31
3.4.3.	Bar chart.....	33
3.4.4.	Interactive map.....	33
3.5.	User interface	36
	Discussion.....	38
	Conclusion.....	40
	Further work	41
	References	42

1. Introduction

When managing remote servers, it is important to monitor their performance and availability. There are many different hardware or software issues that can affect them, and there is a multitude of software tools out there which offer monitoring services to better manage the health of servers. They can offer a variety of features such as managing server uptime and determining when problems in connectivity occur. The focus of this thesis will be monitoring remote servers using ping. There are many different tools that offer this, for that reason I have created a small table which includes different types of the most used tools, for easy comparison.

Name	Reference	Features
SolarWinds Engineer's Toolset	https://www.solarwinds.com/	60+ tools, costs \$1495, multiplatform, offers graphing and logging of data, real-time monitoring and alerting
EMCO Ping Monitor	https://emcosoftware.com/ping-monitor	Free edition: only allows monitoring up to 5 hosts, does not allow any specific configuration for hosts, 1ms precision, only one connection to a server at a time
Nagios Core	https://www.nagios.org/projects/nagios-core/	open-source, only runs on Linux, several community-developed plugins

The first two tools are commercial, and the third tool is open-source. The SolarWinds Engineers Toolset [1] offers more features than the rest but also has a high price. It is more geared towards businesses and offers a fully intuitive user interface. The EMCO Ping Monitor [2] also offers two types of paid versions, but to add more variety to this table, I will focus on their free edition. Although the paid versions offer some good features, their free edition limits the user a lot by only allowing up to 5 hosts to be monitored. The Nagios Core [3] is one of the most common open-source tools for monitoring with ping. Being open-source allows this tool to have one of its best features which is the fact that it offers several

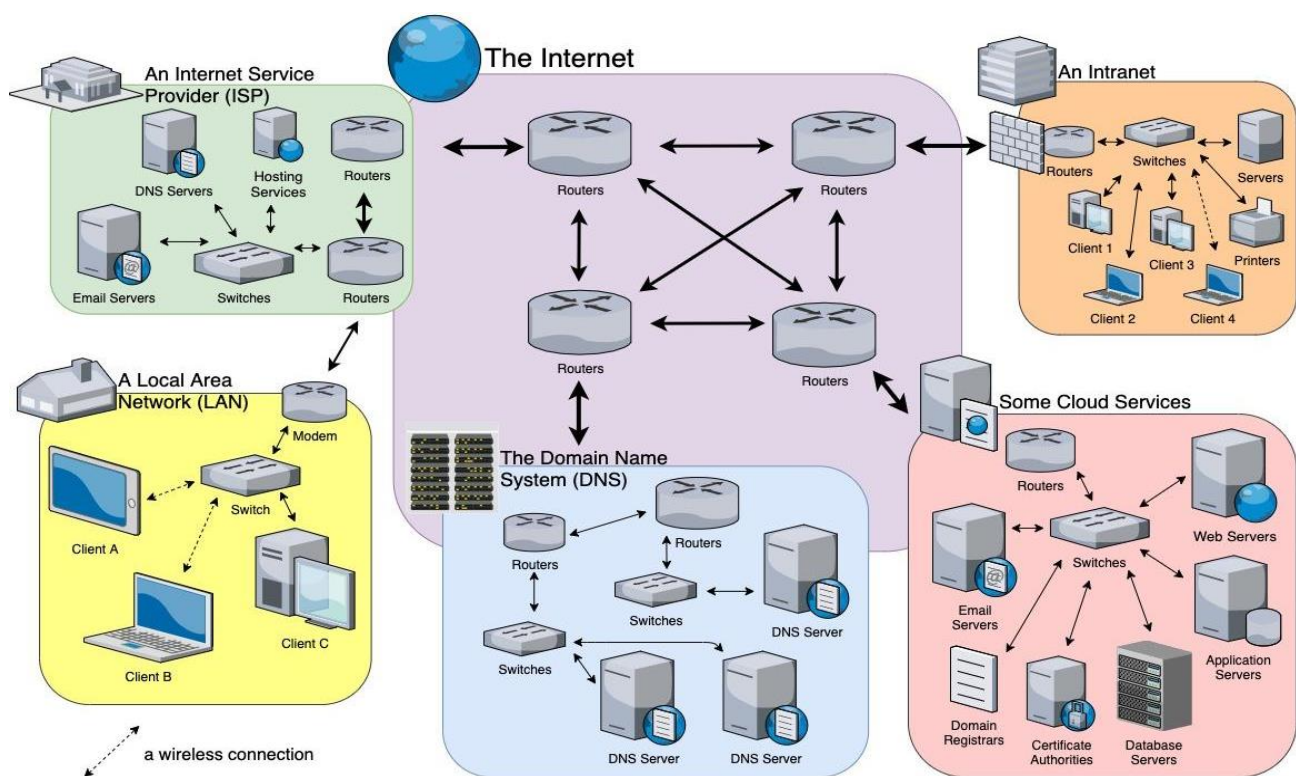
community-developed plugins. Besides the fact that it is free, it also allows you to customize your tool to your own wants and needs.

This thesis focuses on building a fully extensible, open-source tool which would allow users to monitor remote servers. The practical part of this thesis will show how everything was implemented, while the theoretical part will explain the general logic behind it. The tool is multiplatform, uses Python 3, offers data storing and analysis and allows the user to customize the configuration file (config). Since it is written in Python, which is a very common programming language, it increases the chances of people using it and potentially adding more features to it.

2. Theoretical part

2.1. How does the internet work?

The internet is a worldwide computer network that transmits a variety of data and media across interconnected devices. It operates on the basis of a packet-switched network that adheres to the Internet Protocol (IP) and Transport Control Protocol (TCP); commonly referred to as TCP/IP [4]. In a packet-switched network, there is no single, unbroken connection between sender and receiver. Instead, when information is sent, it is broken into small packets, and then reassembled at the receiving end. Those packets are sent by your computer to your local network, Internet service provider (ISP), or online service. The packets then travel via several layers of networks, computers, and communications lines before arriving at their final destinations, which might be to a neighbouring country or throughout the world. Figure 1 shows a birds-eye view of this process.



Source: Vahid Dejwakh, 2020

Figure 1. This picture depicts the infrastructure of the internet. [5]

TCP/IP are used to transport packets from one source to the next. By utilizing a numerical address (IP Address), the IP system obtains additional instructions on how the data should be transferred. TCP collaborates with IP to guarantee that the data transfer is safe and secure. This ensures that no packets are lost, that packets are reassembled in the correct sequence, and that data quality is not harmed by any delays [6]. Various pieces of hardware process the packets and route them to their intended destinations. This hardware is meant to transport data between networks and constitutes a significant portion of the glue that keeps the Internet together [4]. Hubs, switches, repeaters, routers and modems are five of the most important pieces of hardware.

Hubs are vital because they connect groups of computers and enable computers to interact with one another. Switches are responsible for sending data coming from various input ports into a particular output port (usually a router) which will then transmit the data to the desired destination. Therefore switches can be used to connect multiple devices together on a single computer network. [7] When data travels across the Internet, it frequently covers large distances, which may be problematic since the signal that sends the data might diminish over time. To address this issue, repeaters are used to magnify the data at regular intervals so that the signal does not weaken.

Routers are critical components in the management of Internet traffic. It is their responsibility to guarantee that the packets always reach their intended destination. They consider the volume of traffic on the Internet and send the packet to another router that is closer to the packet's ultimate destination. When data is transmitted between computers on the same Local Area Network (LAN), routers are typically unnecessary since the network can handle its own internal traffic. However, when data is transmitted between two or more networks, routers come into play [4]. For instance, figure 2 shows how routers can connect two networks together as well as connect them to the internet.

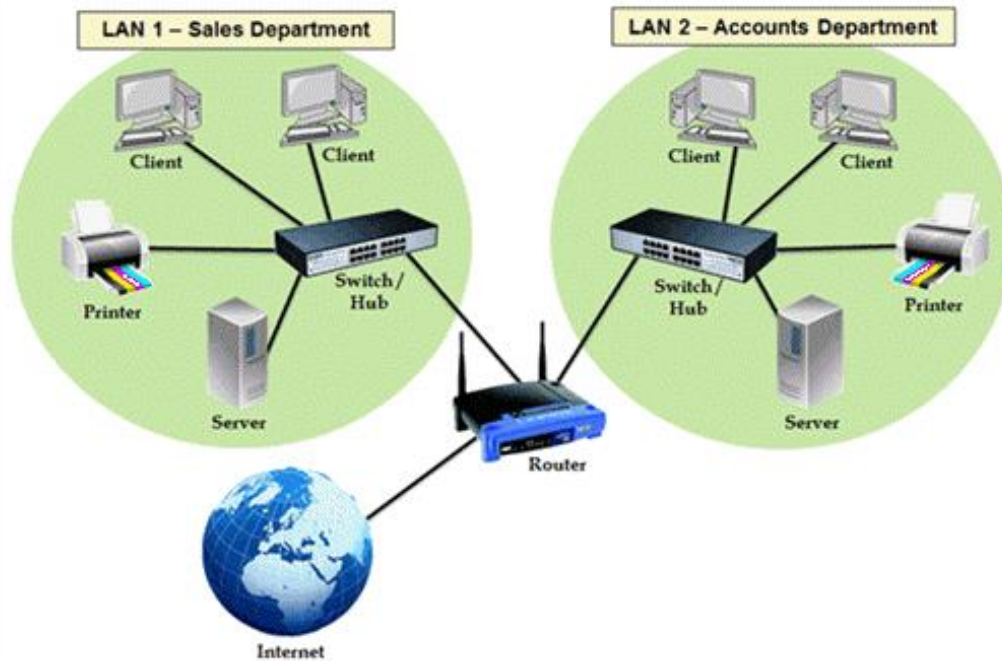


Figure 2. Router connection. [8]

A modem (Modulator/Demodulator) is the intermediary between your computer and the ISP, and it is usually built into the router. ISPs allow you to connect to the World Wide Web, most commonly through Cable Modems and Digital Subscriber Line (DSL) connections. When you use your modem to access the Internet, you typically dial in to an ISP, such as USP or O2. When you dial into and connect to your ISP, you are actually connecting to a modem that is tethered to a more powerful computer known as a server. ISPs typically have banks of hundreds or thousands of modems that accept dial-in connections from users [4]. A collection of commands, known as the Hayes Command set, are used by computers and communications software to control modems. It is a language that tells the modem what to do at various moments during a communication session, such as establishing a line and sending out tones that the phone system understands.

To access the World Wide Web, you must run a web browser on your computer, such as Internet Explorer or Chrome. Using TCP/IP, the web browser makes a Hypertext Transfer Protocol (HTTP) request to a host server. The browser can request a specific web page or request that the host server run a database query [7]. In either case, the request is divided into HTTP packets and delivered to the host over the Internet's TCP/IP communications infrastructure. The host server locates the information and then delivers it to the web browser, which shows the findings.

When web browsers contact servers, they request pages mostly built in Hypertext Markup Language (HTML) to be sent to them. Browsers then translate those pages and present them on your computer in a readable format [9]. The web pages and hosts that comprise the World Wide Web must have distinct locations in order for your computer to locate and retrieve the content. The Uniform Resource Locator (URL) is accepted as a unique identifier for a webpage [4]. A URL, among other things, shows the location of the host computer, the position of the website on the host, the name of the web page, and the file type of each document.

`http://www.example.com/search?item=vw+beetle`

Protocol	Domain	Path	Parameters
----------	--------	------	------------

Figure 3. Example of a URL broken down into individual parts. [10]

When someone on the Internet wants to visit a website, they type in an address, such as “www.cvut.cz”. For TCP/IP to be able to identify the location of that website, the alphanumeric address must be converted into an IP address by the Domain Name System (DNS). The query goes through various levels of DNS, until the IP address of the relevant host server is found and a connection is established [11]. This system works in a hierarchical order, where the DNS resolver is your ISP. The figure below explains the process clearly.

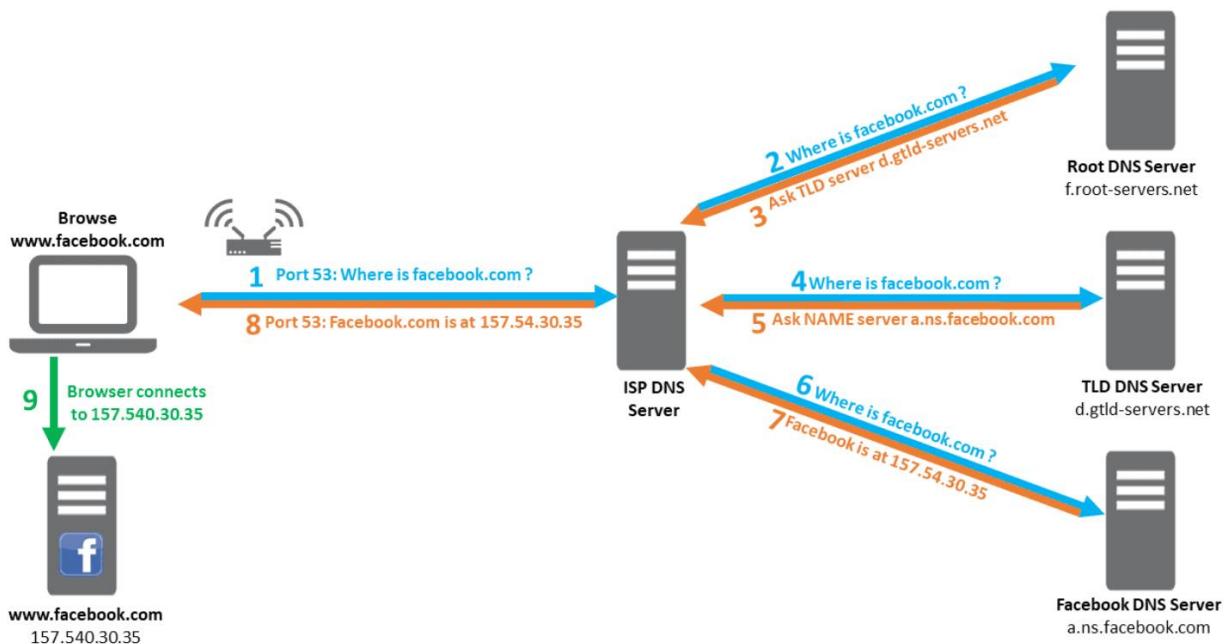


Figure 4. Hierarchy of DNS servers. [12]

2.2. Client-Server model

The basic architecture of the Internet is what is known as the client/server model [13]. A server, a computer running special software capable of sharing files with other systems, hosts a range of services that are accessed by client software running on a computer. Sometimes, clients can also be servers themselves; for example, in a distributed network where all members of the network are connected to each other non-hierarchically, the members can communicate directly with each other without having to connect to a central server [14]. Some of the standardized protocols that client and servers use to communicate with themselves include File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and most commonly Hypertext Transfer Protocol (HTTP).

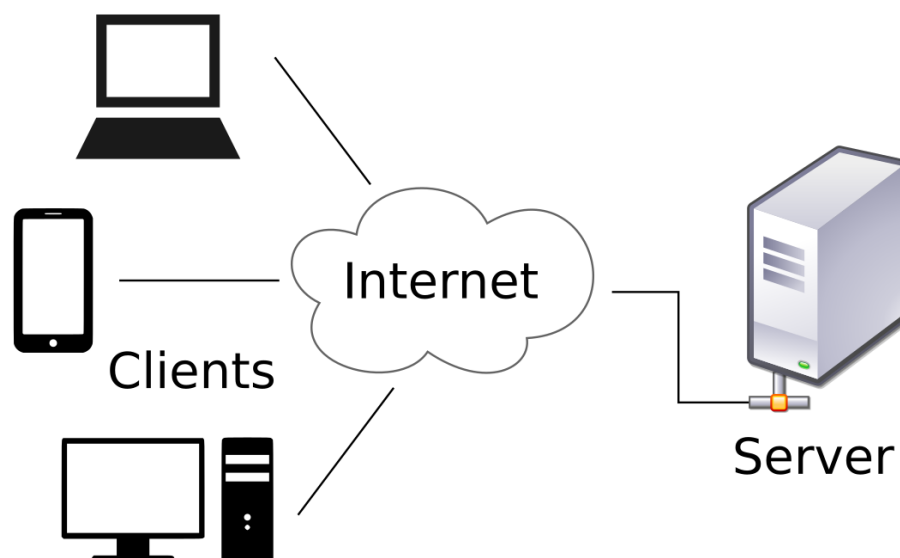


Figure 5. How clients are connected to a server. [15]

2.2.1. Server Monitoring

Monitoring servers is the process of gaining insight into the activities of your servers, either physically or in virtual form. Server monitoring is important because it allows you to see changes in the servers' status. One server can handle hundreds, or even thousands, of simultaneous requests. Therefore, making sure your servers are functioning as expected is important.

2.3. Internet Control Message Protocol (ICMP)

ICMP is used for network diagnostics; the widely used terminal applications, traceroute and ping, both employ ICMP. Traceroute is a utility that displays the routing path between two Internet devices. The routing path is the actual physical path of linked routers that a request must traverse before arriving at its destination. This can be helpful in discovering the causes of network delays [16]. Ping (Packet InerNet Groper) is a streamlined form of traceroute. A ping will measure the speed of a connection between two devices and report how long it takes for a packet of data to go to its destination and return to the sender's device. Even though ping does not give routing information, it is a very helpful statistic for determining the connection strength between two devices.

2.3.1. Ping as a monitoring tool

Ping is also used for diagnostic purposes to confirm that the target host which the user is trying to reach exists and can accept requests. Ping may be used by any operating system (OS) with networking capabilities, including most embedded network administration software. It operates by sending an ICMP Echo Request (usually 32 bytes per packet) to a specific TCP/IP node and waiting for a response. When the target host gets the echo request, it sends an echo reply packet in response. If the message does not come back, you can deduce that the network or host is down or that the packet was lost in transmission [17]. Ping commands, by default, send many requests (typically four or five) and show the results. It also contains the amount of bytes received and the time it takes to obtain a response, which is referred to as latency [7]. The picture below depicts a user pinging a specific IP address.

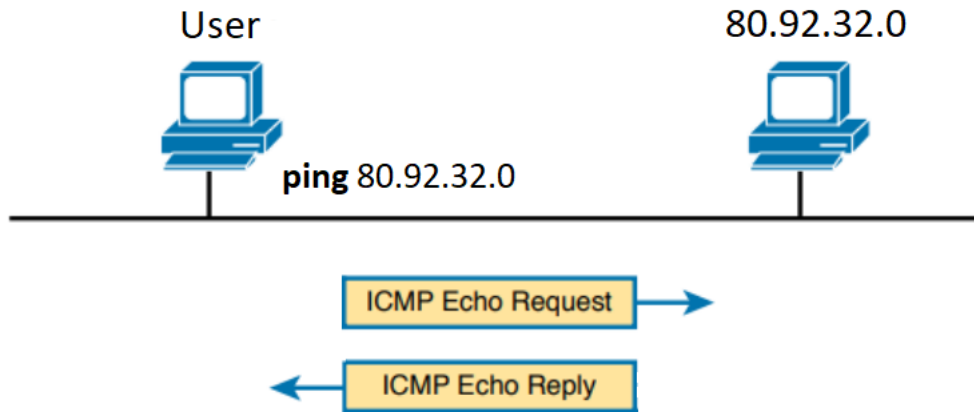


Figure 6. Example of a user pinging a specific IP address.

The results you get from ping allow you to infer certain things about the target host which you are pinging. For example, if the target web site is unavailable but the web host responds to pings, you can conclude that the host application has crashed but the server is operational. If the server does not react to pings, it is safe to believe that something is wrong with the hardware. Likewise, if your server replies by IP address but not by domain name, you might assume that the DNS is malfunctioning. Further inquiry is required to discover whether the domain name has expired or whether something else is going on [18]. Additionally, the data that ping provides allows you to estimate the quality of a network connection or a system's health by storing and analysing the data.

2.4. Storing data

Data storage fundamentally implies that information and documents are digitally stored and kept in a storage system for future use. Data storage can take place on physical hard drives, disk drives, Universal Serial Bus (USB) drives, or virtually on the cloud [19]. One of the easiest, most common ways of storing data is using a comma-separated values (CSV) file [20].

2.4.1. JavaScript Object Notation (JSON)

JSON is a data exchange format that stores and transmits data objects made up of key-value pairs and arrays using human-readable language. It is a basic, standard data format

with several applications in electronic data transfer, including web applications with servers. When writing key-value pairs, the keys must be strings, whereas the values can be numbers, strings, booleans, arrays, objects, or null. independent format. The snippet below [21] shows a possible JSON representation describing a person's details.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

2.4.2. Comma Separated Values (CSV)

A CSV file is a delimited text file in which values are separated by commas. Each row of the file represents a data record, and each record has one or more fields separated by commas. A CSV file generally carries tabular data (numbers and text) in plain text, with the same number of fields on each line [20]. Even though CSV uses commas as a delimiter by default, when storing your data in CSV, you can customize the delimiter to your own needs. Some examples include semicolons, tab-separated values, and space-separated values.

CSV files are practical because they are human-readable and very simple to create. They can be read easily and are easy to parse due to the simplicity of their format. The biggest advantage of using CSV files is that they are very widespread and can be manipulated simply, as well as can be opened in Excel for better visual representation of the data.

2.4.3. Database

A database is a structured collection of data that is stored and retrieved electronically. Small databases, such as CSV, can be saved on a file system, but big databases are housed on computer clusters or cloud storage [22]. When designing a database, many things must be taken into account, such as: data modelling, query languages, efficient data presentation and storage, sensitive data security and privacy, and computing challenges such as concurrent access and fault tolerance.

The software that interacts with end users, applications, and the database itself is known as a database management system (DBMS). The DBMS software records, and analyses data and it also includes the main database administration tools. The key features of DBMS are: data definition, maintenance, manipulation, display and integrity [23]. Some of the most common DBMS software used are SQLite, Oracle, Microsoft SQL Server, etc. The term "database" is frequently used informally to refer to any of the DBMS, a database system, or an application linked with the database. Users access databases through servers as shown in figure 7.

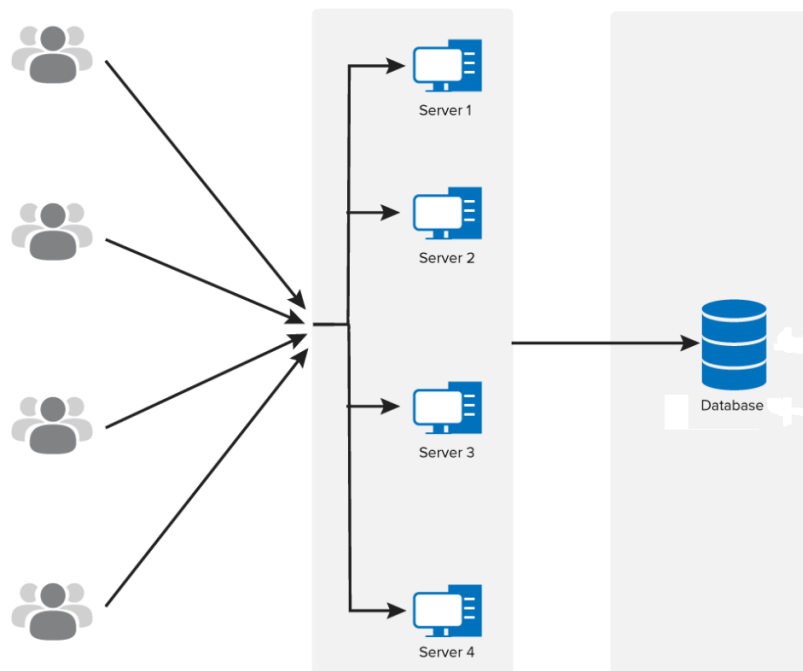


Figure 7. Modified figure shows database connection [24].

2.5. Data Analysis

To better understand the meaning of data, it must be analysed using mathematical operations. Key figures such as the count, mean, median, etc. help us grasp what our data is showing us. One of the best ways of comprehending big sets of data is utilizing graphic display, as it converges all of your data into one image.

2.5.1. Graphic display/data presentation

The goal of visually displaying data is to take use of the power of visual presentation to effectively communicate information. This is crucial for both how we convey our results to others and how we comprehend and analyse the data. Patterns will frequently emerge in a data visualization that would go entirely overlooked if statistical analysis were performed alone. There is a relatively small amount of graph types used when visually displaying data; among the most common include the scatter plot, line plot, histogram and bar chart [25]. Data is usually displayed as raw numbers, however sometimes it is useful to represent data in terms of the proportion or percentage of different elements making up a sample, such as in a stacked bar chart. Overall, it is important to make sure that the message you are trying to convey through your data visualisation can be seen clearly, your axes have a sensible start and finish point, and that the nature of the graph is in line with what you are trying to convey.

2.6. User interface

The user interface (UI) is the point at which the user interacts with a computer, website, or application. The purpose of a UI is making the user's experience simple and straightforward, requiring as little effort on the user's side as possible to get the desired result.

2.6.1. Hypertext Markup Language (HTML)

HTML is the standard markup language for documents designed to be displayed in a web browser. It gives content meaning and structure by telling your browser how to display text, graphics, and multimedia files. It also contains commands for linking the page to other pages and to other Internet resources which is a very important feature [26]. HTML files are simply text documents, and it is one of the reasons why the web works so well. Any text file you create on any operative system, including HTML, works equally well on other operating systems. An example of a HTML code is shown below.

```
<html>
<head>
<style>
  h1 {color:green;}
  p {color:pink;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

2.6.2. Cascading Style Sheets (CSS)

While HTML is the main language used in web-based user interfaces, CSS is presentation language which is used to determine the appearance of content, say through colours or fonts. It is important to note that HTML and CSS are independent of one another; HTML should never be written inside of a CSS document and vice versa [27]. A CSS class is an attribute that is used to designate a set of HTML elements so that CSS can apply custom styling and formatting to them. In the example above CSS classes are used to assign colours to the heading and paragraph.

3. Practical part

The goal of this thesis is to create an application that is fully extensible and allows the user to monitor multiple servers through ping, store the data in a csv format, visualise the data and show that through a simple user interface.

3.1. Overview

The diagram scheme of the problem explaining the whole process was created to break down the task into smaller pieces. This layout was picked so it would also be easy to add more features to the app in the future.

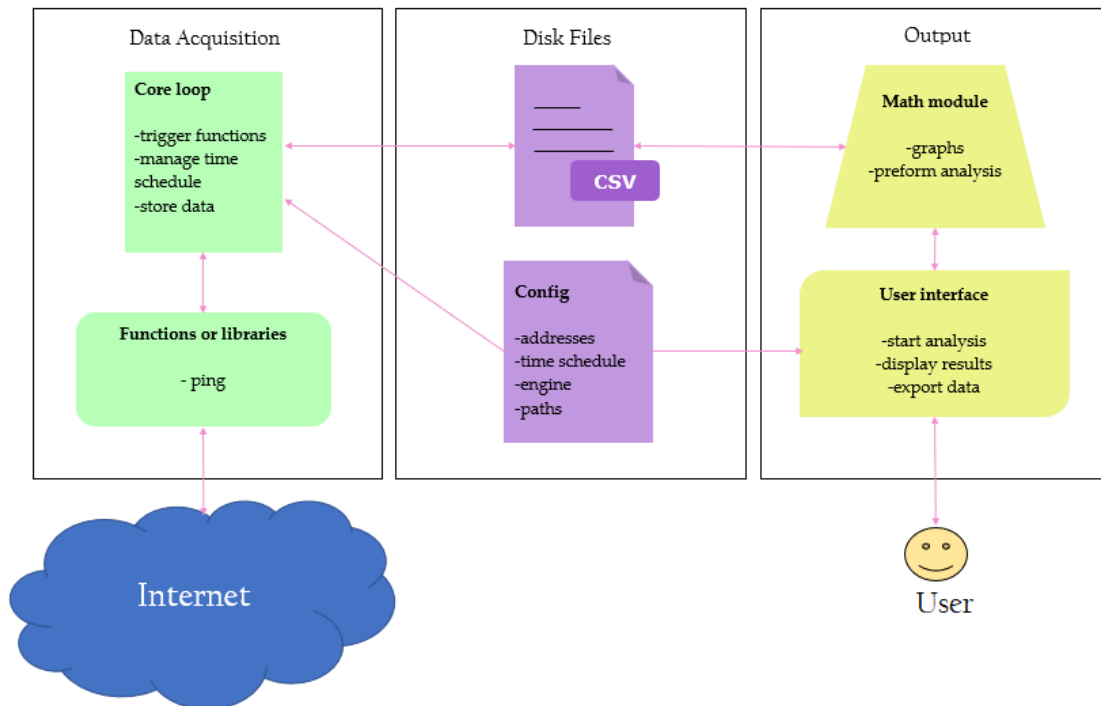


Figure 8. Diagram Scheme of the task.

There are 3 main parts to the diagram, data acquisition, disk files, and output. The first part is responsible for obtaining data over time for some addresses according to a time schedule. The next part is the disk files, which are the config file and the logs created. The config file which is a JSON file is connected to every part of this tool, and for that reason it will be broken down into smaller parts for better understanding. JSON was picked due to is

features mentioned in the Theoretical part. The final part, which is the output, will be a mathematical analysis that the user can access. Each part will be discussed in more detail below.

3.2. Data Acquisition

3.2.1. Methods

Ping was chosen as the main operation for this application as it is one of the easiest ways to monitor servers. There are multiple ways to ping, either through the built-in ping that is part of all operating systems that have networking capability or using Python libraries. There is a lot of different libraries to choose from but the one I picked is ping3 as it was one of the simplest ones. Ping3 is a pure python3 version of ICMP ping implementation using raw socket, it is also multiplatform [28]. However, when running this tool on Windows, a 0.0 value might be returned due to the delay being lower than the precision of “time.time()”. It should be noted that Windows has this error for all measurements with timestamp and it is roughly 16ms. The time.time() resolution will only become larger (worse) as years pass since every day adds 86,400,000,000,000 nanoseconds to the system clock, which increases the precision loss [29]. For this reason, it is recommended to run this tool on Linux or Mac, for better measurement precision or to only ping servers with longer distances to avoid smaller values.

3.2.2. Core loop

This loop is expected to trigger functions, manage the time schedule, and store the data. This script is connected to the config, where the user must provide each address they want to monitor, and its corresponding time schedule. There are two options for the time schedule, frequency, and daytime. Frequency in the format of S for seconds, M for minutes, H for hours, and D for days, and daytime in the format of H:M.

```
"tasks": [  
  {  
    "type": "ping",  
    "address": "ut1-time.colorado.edu",  
    "freq": "5S"  
  },  
  {  
    "type": "ping",
```

```

    "address": "216.58.223.255",
    "daytime": "11:12"
},
{
    "type": "ping",
    "address": "23.223.89.218",
    "freq": "15M"
},
{
    "type": "ping",
    "address": "95.163.141.82",
    "daytime": "12:00"
}

```

There is a function to deal with the timestamp precision, and the None values that ping3 returns when there is no reply (offline). For easier manipulation of the data, this function converts the None values to -1, and the 0.0 values to 0.016. Another function will convert the frequencies written in H, M, S, D format to timestamp to manage the time schedule. The False return will not be converted as it is currently expected for the user to only ping addresses that they know exist.

Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching) [30]. Running multiple threads is equivalent to running multiple applications at the same time, but also with the following advantages:

- Multiple threads within a process share the same data space as the main thread, making it easier for them to share information and communicate than if they were separate processes.
- Threads, also known as light-weight processes, do not require significant memory overhead and are less expensive than processes.

A thread has a start, an execution sequence, and a conclusion. It has an instruction pointer that maintains track of where it is currently running in its context. It can be averted (interrupted) and it can be put on pause (also known as sleeping) while other threads are running, which is referred to as yielding [31].

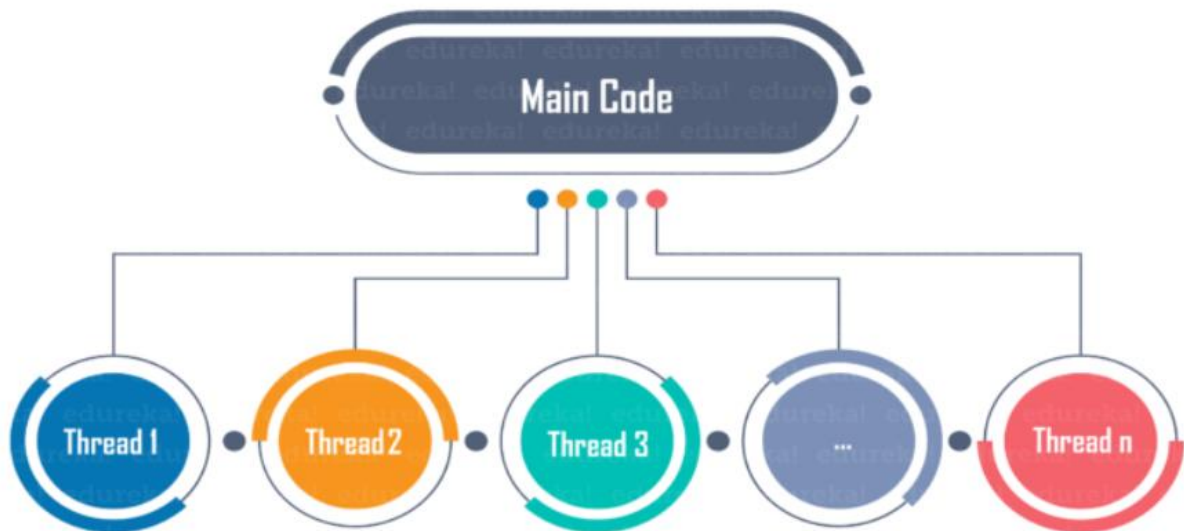


Figure 9. Multithreading. [32]

Threading is used for this tool to improve application performance, by allowing Python to execute other code while waiting. By running multiple threads, the tool is able to execute tasks, and store logs at the same time. Although there is a limit to how many threads can be created, it is not expected that the user can exceed it. For that reason, a new thread is created for each task execution. If no data is available, the thread should "suspend" itself until data is available so that it avoids wasting CPU cycles. One simple way to do this is to go to sleep for a short period of time before waking up again to check if data has arrived.

The following function creates a thread for each new execution, it also adds a dummy column of "next_run" and "last_run" to the config, which will help manage the time schedule. For writing the response in the file, the threading queue is used as the response queue to avoid overwriting. The writer loop uses a while loop to fill the queue with responses when not empty, it also checks which engine was picked for storing (currently there is only csv).

```
response = {
    "type": task_type,
    "address": address,
    "time": int(time.time()),
    "value": ping(task["address"])
}
```


The core loop checks which type of time schedule has been provided. If the user has used frequency and they have just started the script, the task will be executed from start when there is still no “last_run” added. After that it checks if the seconds have passed using the granularity function + the time that has elapsed from the “last_run”. If the user has used daytime and they have just started the script, it will constantly check if the current time in the format H:M is the same as the daytime provided, and once it is, it will execute it. After execution a next_run will be added which is calculated by:

```
task["next_run"] = time.time() + get_granularity("1D") - 60
```

Where the “time.time()” is the timestamp, “get granularity(“1D”)” is a day in seconds and the - 60 is added as a security buffer. Once this has been added, it will be used to see if more time has passed than the “next_run” and if the current time matches the daytime, so that it can execute the task the following days.

3.3. Storing of the data

This part is responsible for storing and reading the logs which is done by the “storage_csv” script. This script also uses Pandas [33] for the reading of the data. Below are the options provided to the user for customization.

```
"writer_settings": {
  "sleep": 3
  "engine": {
    "type": "csv",
    "path": "data",
    "delimiter": ",",
    "extension": ".csv"
  }
}
```

Through the config the user can choose the time sleep for the writer loop (mentioned above), the engine type for storing and reading, the path where the files storing the data will be created, the delimiter for separating the columns in the logs, and the extension of the files.

This script consists of 2 classes, CSV writer and CSV reader.

3.3.1. CSV Writer

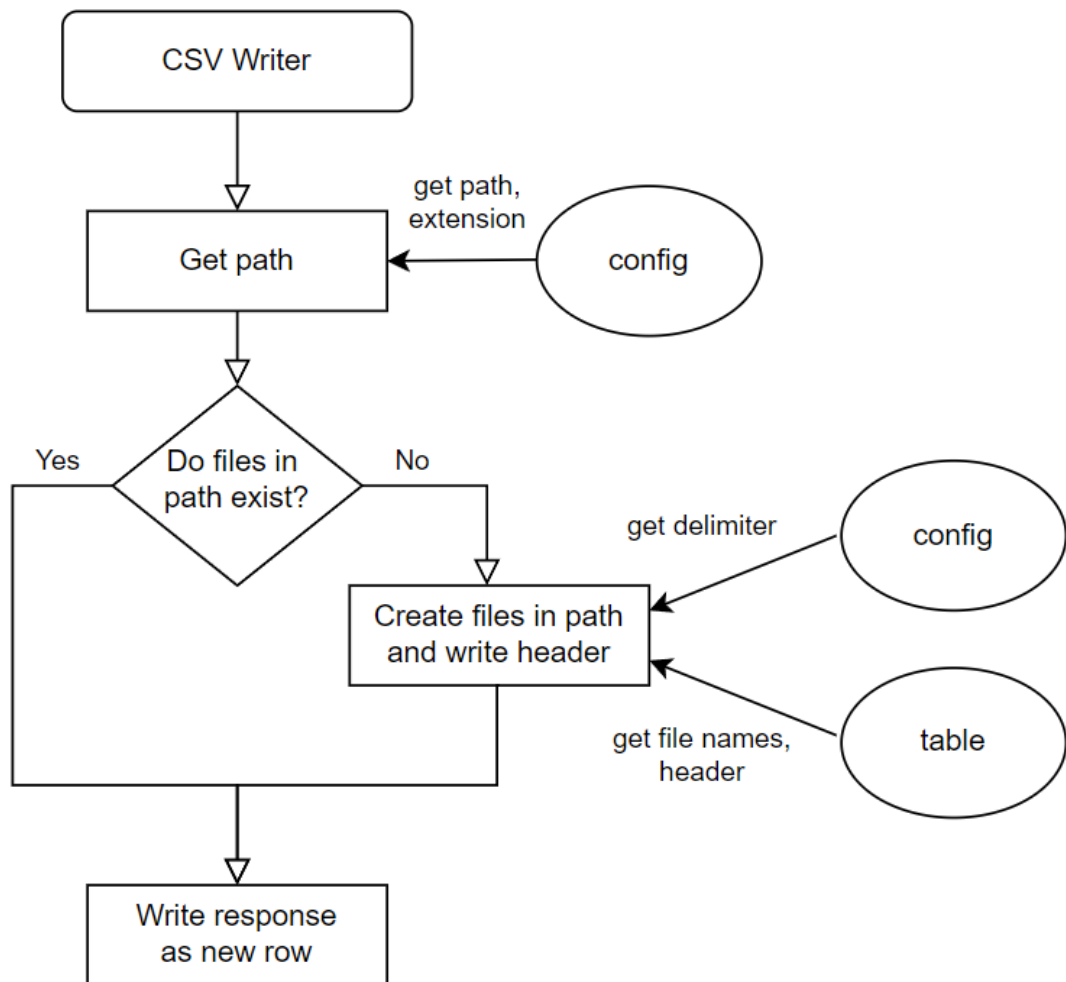


Figure 10. Simple diagram showing how the CSV writer class works.

The CSV writer class is responsible for creating the files and storing the response in those files. It does that by first getting the path where the files should be created, and the extension, from the config. Then, checks if the files exist, if not it creates those files according to the TABLES' keys, with headers from their columns, respectively.

```
TABLES = {  
  "ping": {"columns": ["address", "time", "value"],  
          "index": "time"},
```

```
"errors": {"columns": ["time", "message"],
           "index": "time"}
}
```

Once the files are created, and their headers written, each new response will be appended to its respective file as a new line. The columns in the file will be separated, by the delimiter provided in the config. An example of the csv logs can be seen in figure 11.



```
ping.csv x
1 address,time,value
2 gjrifa50.com,1643699877,0.022913455963134766
3 23.223.89.218,1643699877,0.04560732841491699
4 redstation.com,1643699877,0.048366546630859375
5 91.132.197.219,1643699877,0.049581050872802734
6 95.163.141.82,1643699877,0.0620424747467041
7 216.58.223.255,1643699877,0.12194061279296875
8 ut1-time.colorado.edu,1643699877,0.1800062656402588
9 23.35.40.237,1643699877,0.20316386222839355
10 210.0.255.251,1643699877,0.2637064456939697
11 91.132.197.219,1643699882,0.05401730537414551
```

Figure 11. Example of a ping.csv file.

3.3.2. CSV Reader

Just like the CSV Writer, the CSV Reader first gets the path, and the extension of the files, provided in the config. It then reads the file and changes the default index to the one provided in the TABLES. It does that by first creating a data frame of the current file, adding a dummy column of the TABLES index, then changing the index of the data frame to the dummy column. Before returning the new data frame the dummy column is deleted. This helps prepare the data frame for the data analysis part. The diagram below explains this in a simple and straight forward way.

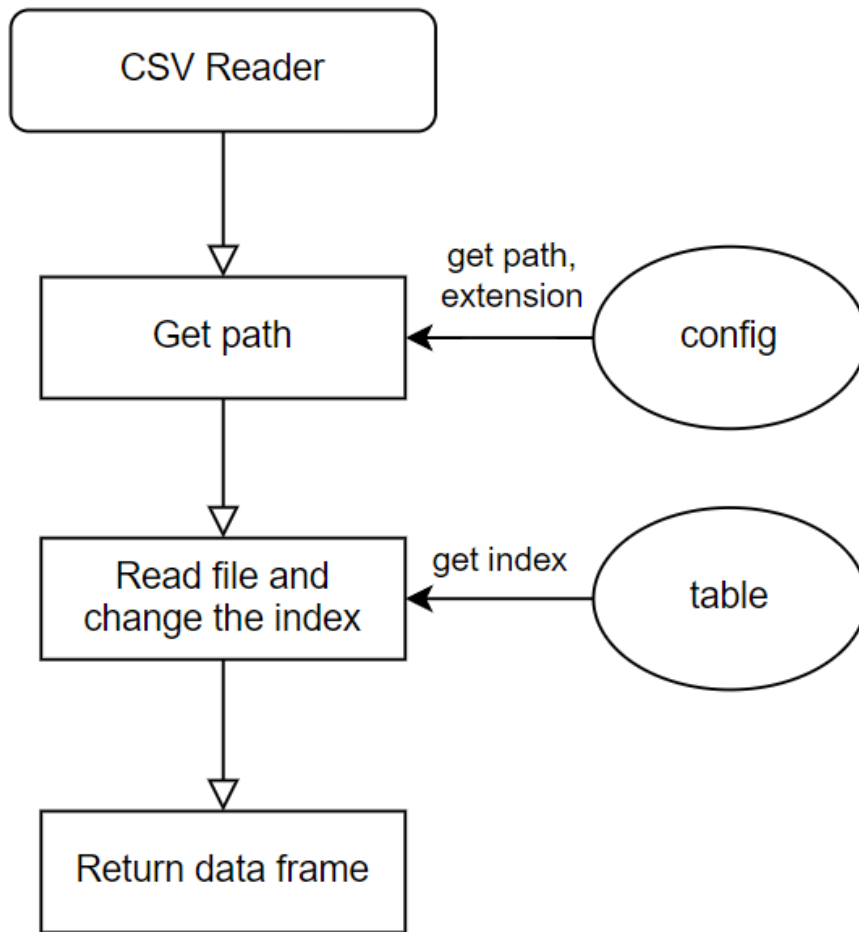


Figure 12. Simple diagram showing how the CSV reader class works.

3.3.3. Engine

The engine script allows the user to pick the engine for storing of the data. It does that by getting the writer engine and the reader engine from the ENGINES. This script will be very useful in the future when more engines are provided. It is also where the dictionaries, ENGINES and TABLES, are located.

```

ENGINES = {
    "csv": (storage_csv.CSVWriter, storage_csv.CSVReader),
    # "sqlite3": (storage_sqlite3.Sqlite3Writer,
    storage_sqlite3.Sqlite3Reader)
}
  
```

The “storage_sql” script is not used at all as the sqlite3 engine has not been implemented yet. But the logic used should be the same as the one from the “storage_csv”.

3.4. Graphical Analysis of data

The first part of this section was choosing the right analysis for the given data. The goal of this part is to offer the user insight about their servers through graphical analysis. The graphs picked for this are the histogram, representing the distribution of ping values, a linear plot with ping as a function of time, and a stacked bar chart to display the successful and the unsuccessful pings. An interactive map showing the location of each server is also provided for better visualization. The user can customise the graphs with the parameters provided in the config which are:

```
"graph_settings": {  
  "path": "figures",  
  "hist_const": "300",  
  "hist_outlier": "0.98",  
  "round_digits": "4",  
  "lin_const": "40",  
  "percentage_digits": "2",  
  "open_browser": true
```

The path is where the figures of the graphs will be saved. Histogram constant represents the number of minimum required pings per address. This constant is provided so the data analysis does not involve graphs with little amount of data. There is a minimum requirement for the user to record 10 values for every address, according to the bin formula:

$$\text{bins} = \min(100, \text{len}(\text{subset}) // 10),$$

where the subset is each unique address.

This formula was created to try and automate the number of bins, based on the amount of data provided. Histogram outlier constant is responsible for removing the outliers from each histogram, and the round digits is the number to which the values will be rounded to which are displayed on the user interface. The linear constant, same as the histogram constant, represents the minimum number of responses to avoid showing graphs with very little observations. The percentage digits are the constant responsible for the bar chart, it is the number to which the percentages are rounded to. The last parameter is open browser which has a Boolean value. It can be switched off with false if the user would not want the local file to open in a web browser automatically when running this script.

The graphs are all made using the following libraries: Pandas, Matplotlib [34], and Scipy [35].

3.4.1. Histogram

To create the histograms, first the filename is assigned, each graph is saved using this filename and the variable i , where i represents every unique address. The first graph is a histogram overlay of all the addresses and is shown below.

All plots are saved both as a svg and png file using the path, filename, and i .

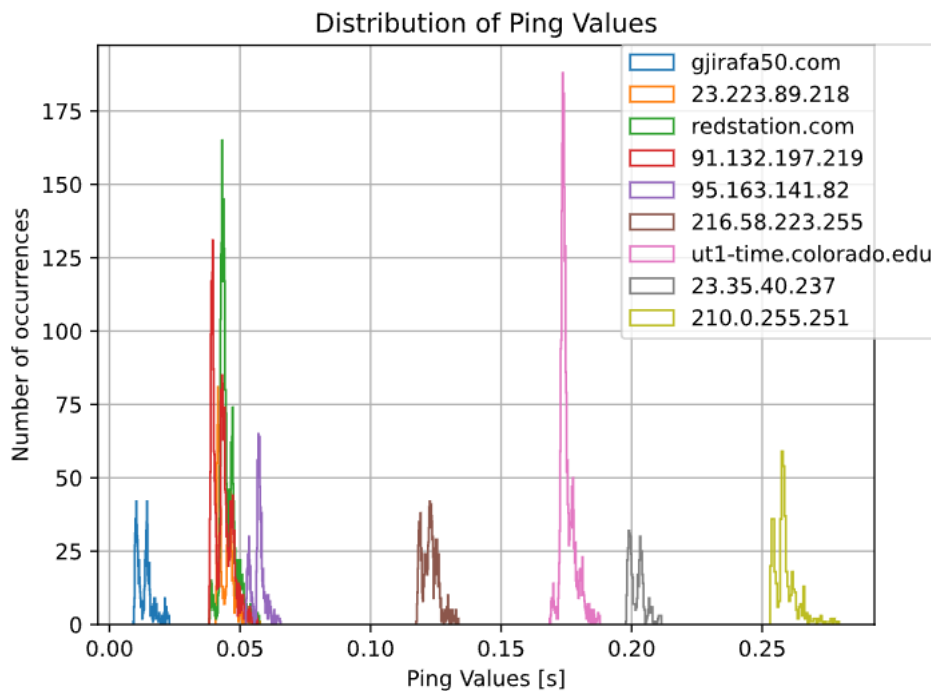


Figure 13. Histogram overlay of all the addresses.

The second part of this function is creating a histogram for each unique address, using a for loop to iterate through the Data Frames address column for all unique addresses. There is also some descriptive statistics provided for the user to help understand the distribution better, such as the median value, minimum value, maximum value, and the offline count for each address.

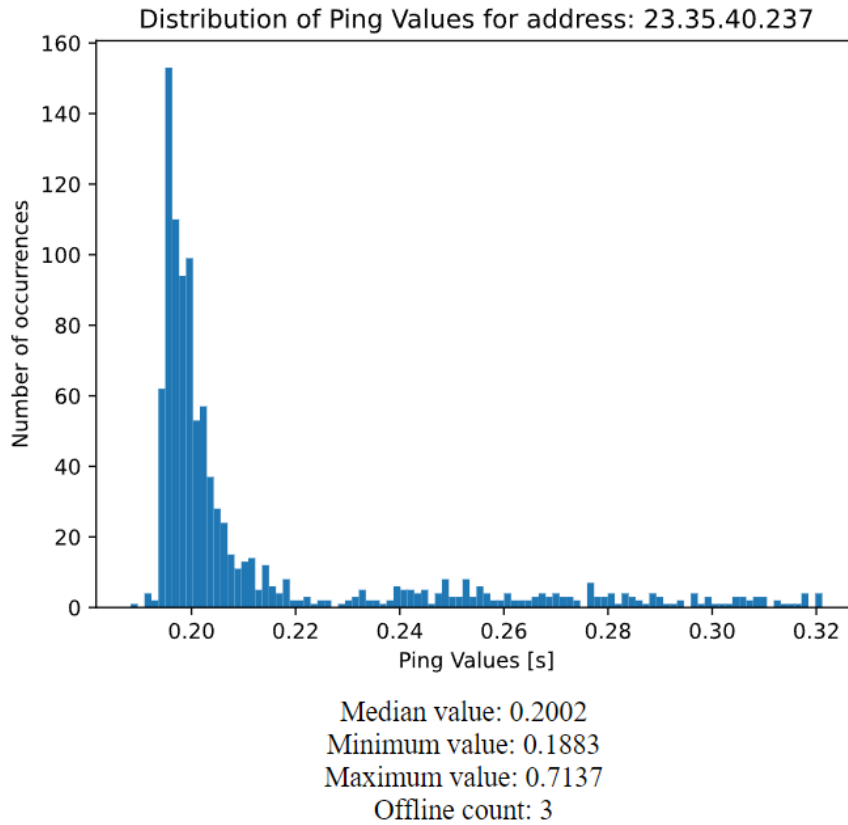


Figure 14. Histogram with some descriptive statistics for 23.35.40.237 address.

3.4.2. Linear plot

The linear graph with ping as a function of time also provides a linear approximation for each graph. The approximation is made using optimize from Scipy. The rest follows the same logic as the histogram, where the first plot shows all the addresses, and the rest of the graphs will represent each unique address. The index, which is the timestamp, is converted to datetime for easy readability.

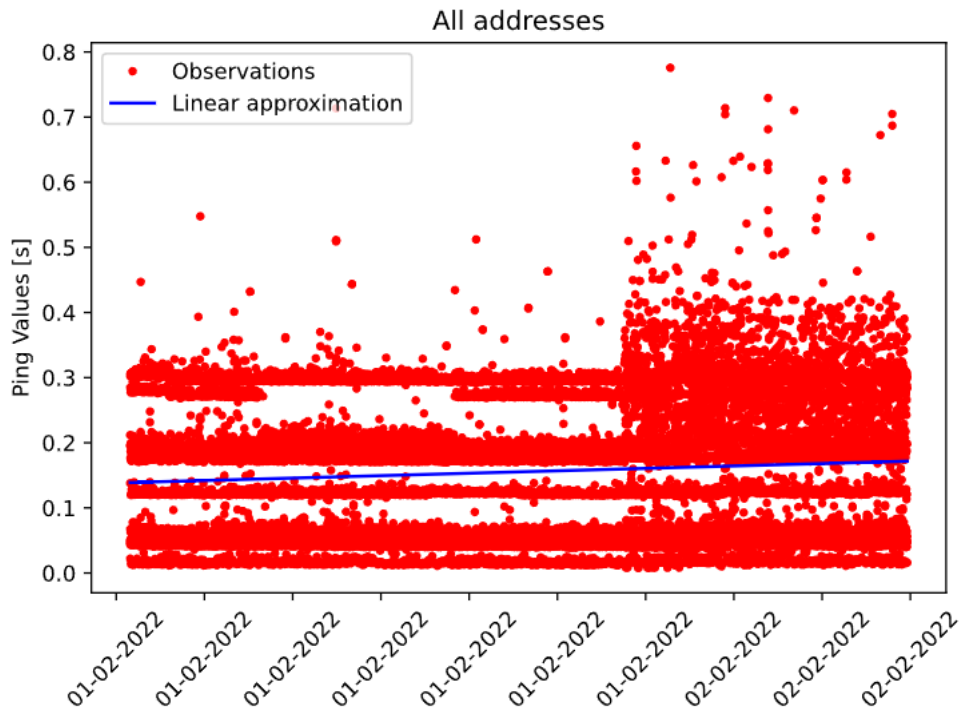


Figure 15. Linear plot with linear approximation for all the addresses.

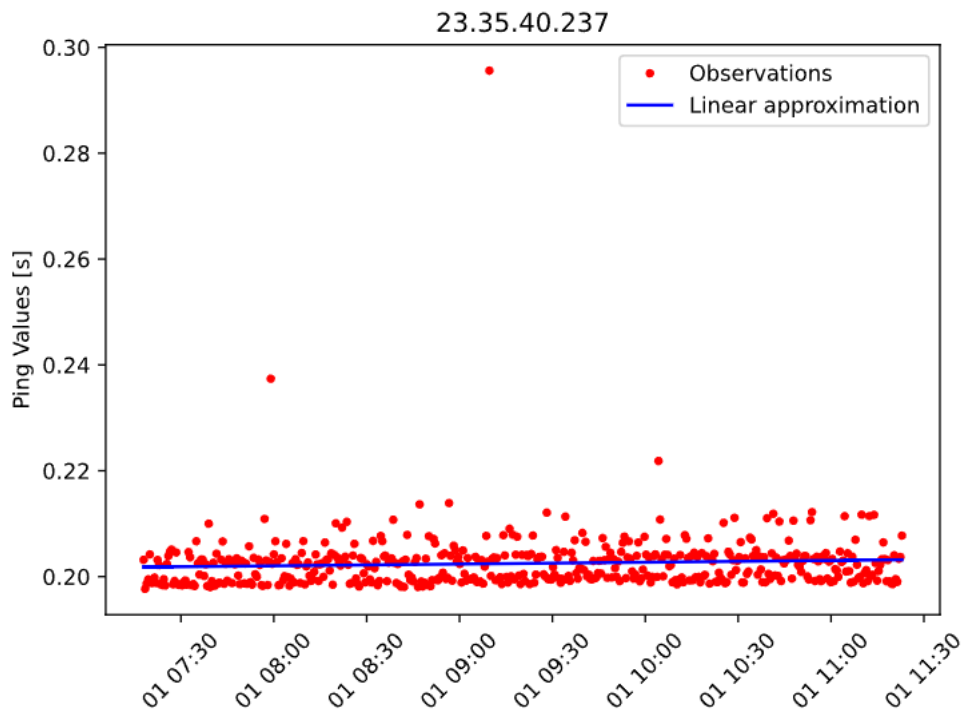


Figure 16. Linear plot with linear approximation for 23.35.40.237 address.

3.4.3. Bar chart

The stacked bar chart will only be a single plot where all the addresses provided represent an independent bar which is split into an offline and online percentage. The plot also shows the number of the actual counts for both the online, and offline values.

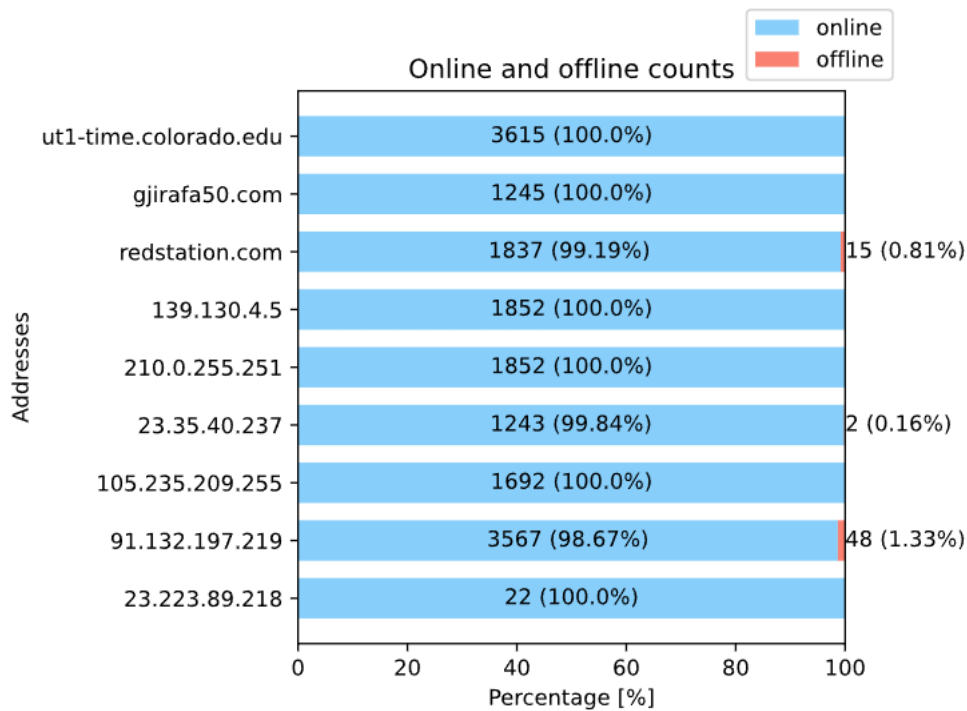


Figure 17. Stacked bar chart with number of counts and percentage for both bars.

3.4.4 Interactive map

Another method for better data visualisation is provided, in the map script. This script uses the library “folium” which makes it easy to visualise data that has been manipulated in Python on an interactive leaflet map. The point of this map is to see how the geographical position of a server affects the ping value.



Figure 18. Interactive leaflet map.

In order to use the map, the user must provide the location for each address in the config. The name, latitude and longitude of every city is expected in the server info section.

```
"server_info": {
  "ut1-time.colorado.edu": {
    "location": "Boulder",
    "latitude": "40.014984",
    "longitude": "-105.270546"
  },
  "216.58.223.255": {
    "location": "Johannesburg",
    "latitude": "-26.2041",
    "longitude": "28.0473"
  },
  "139.130.4.5": {
    "location": "Melbourne",
    "latitude": "-37.840935",
    "longitude": "144.946457"
  },
  "23.223.89.2186": {
    "location": "Madrid",
    "latitude": "40.416775",
    "longitude": "-3.703790"
  }
}
```

This information is then used to create each marker in blue, representing every address. Every marker also has a popup and tooltip parameter to provide more information. The tooltip will show the address, last ping value and the median ping value. And the popup of the marker will show the name of the city.

The user location, latitude and longitude also need to be provided in the map settings. The round digits will round the last ping and median value from the tooltip. A grey marker is created to represent the user based on the info provided. Lines were chosen to represent the connection between the user and each address. These lines can be customized with different colours based on the range of values provided.

```
"map_settings": {
  "user_location": "Prague",
  "user_latitude": "50.0755",
  "user_longitude": "14.4378",
  "round_digits": "3",
  "open_browser": true,

  "warning_colors": {
    "green": {
      "min": 0,
      "max": 0.09
    },
    "orange": {
      "min": 0.09,
      "max": 0.5
    },
    "red": {
      "min": 0.5,
      "max": 10
    }
  }
},
"default_color": "black"
}
```

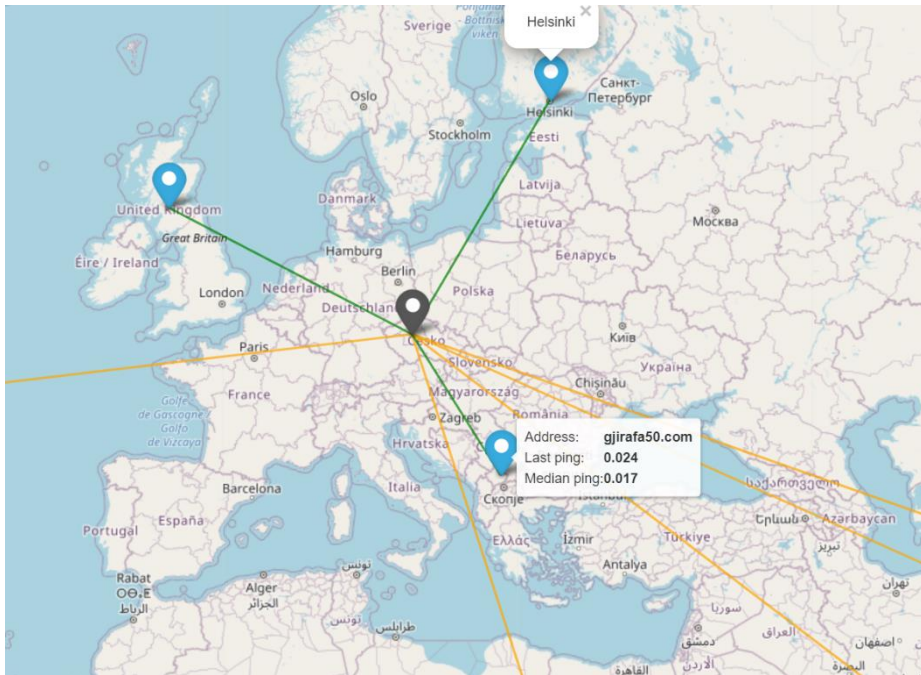


Figure 19. A more detailed picture of the interactive leaflet map.

The current warning colours which are green, orange and red represent different boundaries. Green being the lowest range of values and red the highest. Default colour is used whenever there is a value outside of the bounds provided. In this case the black lines would mean the server is offline.

3.5. User interface

The user interface currently requires the user to manually run the script to see the visualization of their data. Once the user runs the “data_analysis”, or the “map” script, their web browser will automatically display the results after execution. This can be turned off in the config, as mentioned for the graphs.

In order to display the results in a user-friendly format, a HTML template was used. Each new image created, is appended to a html_content list. In the end a new index file is created which shows all the html_content, in the same format as the template.

```
html_content = ["<h2>Histograms<br>(Distribution of pings)</h2>"]
html_content.append("""
<a download="{1}{2}.png" href={0}/png/{1}{2}.png" title="{2}">
<br>
""").format(path, filename, i))
```

The new HTML index file is built by replacing non-html syntax from the template with the `html_content`.

The template also uses some CSS classes for better formatting of the graphs which is shown below.

```
.format {
    text-align: center;
}
.format img {
    width: 500px;
    height: auto; /*this makes sure to maintain the aspect
ratio*/
}
.caption {
    font-weight: normal;
    font-size: 90%;
    font-color: black;
    text-decoration: none;
    pointer-events: none
}
```

Discussion

This section will show and explain some of the most interesting graphs obtained with this tool. The observations will be discussed briefly with possible explanations of the behaviour.

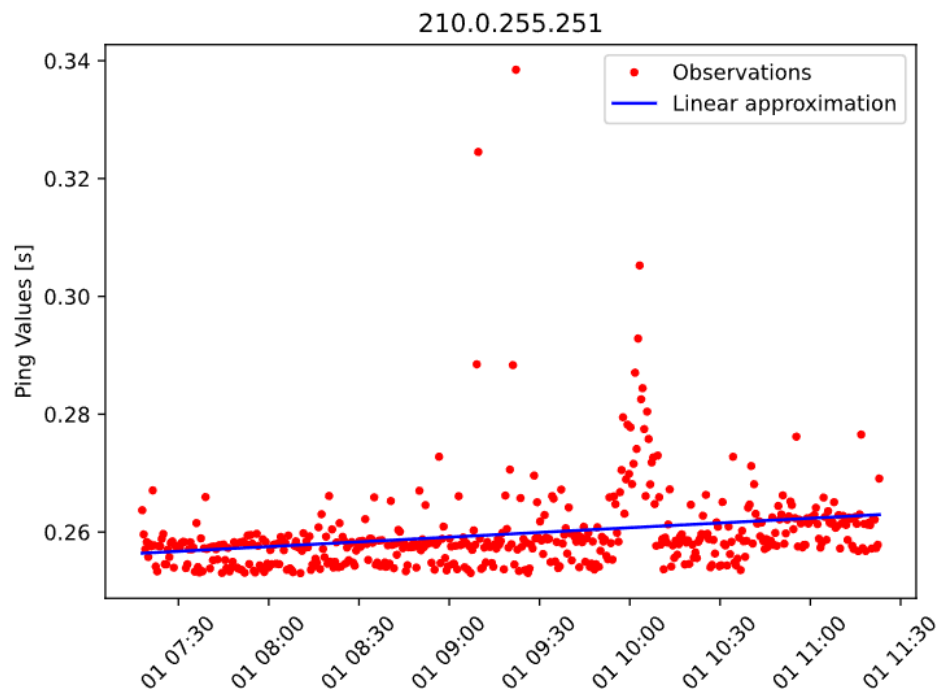


Figure 20. Linear plot with linear approximation for 210.0.255.251 address.

In this graph we can clearly see when there was something wrong with this server. The ping values went higher and then dropped down again around 10:00. The linear approximation also shows how the overall connection got worse over time. Key point of this tool is being able to see when a problem occurs which is perfectly visible here.

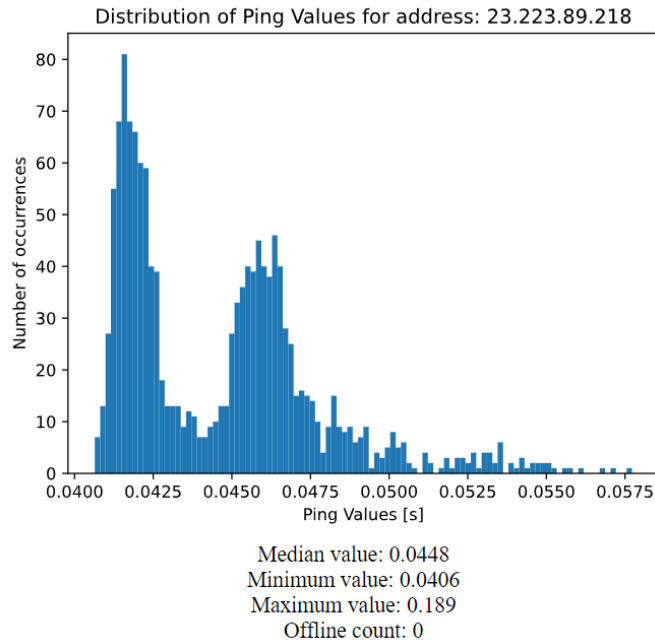


Figure 21. Histogram with some descriptive statistics for 23.223.89.218 address.

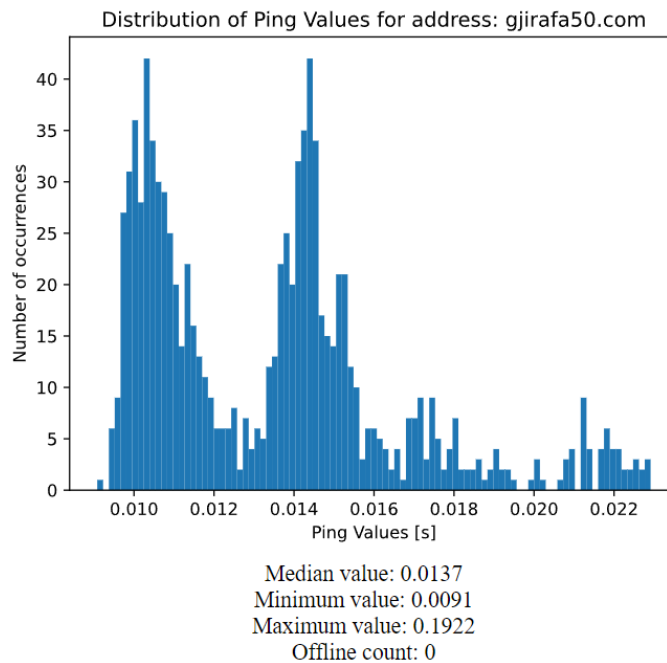


Figure 22. Histogram with some descriptive statistics for gjirafa50.com address.

These graphs were chosen as they both have multiple peaks, which is unusual. A possible reason for this could be the different paths the ping can take throughout the day, resulting in different values. This shows that the tool can also be used to analyse these routes.

Conclusion

Overall, this tool proved to be useful for monitoring remote servers. Through the data analysis it should be obvious to the user when something unusual is happening with their servers. It also managed to prove how useful ping is as a utility and how it can also be used to analyse other things such as the route.

The tool currently can only be used by someone that has at least some basic knowledge about Python since it does require for the user to manually run it. However, that also allows the user to implement even more features very easily.

The main goal of this tool was to be fully extensible, making it easy to implement more complicated features. Even though the tool only offers ping it is built in a way that more things can be added easily. The config also allows the user to customize most of the stuff available to them without having to even change the script.

Further work

This tool can improve by adding more features and options in the future. In this part, I will go over some of those features in hopes of saving some time for the next person who chooses to take this further.

Some features that can be added to this tool are the traceroute, standard HTTP requests, the upload and download speed, and many more depending on what the goal is. I did already find some libraries that I did not have the time to implement like the “icmplib” which basically offers both ping and traceroute, and the “speedtest-cli” which measures the upload and download speed.

Another important thing to add would be more storage engines. I would start with the SQLite engine as it is already mentioned in the script and thesis, which would make adding it a lot easier. Different types of databases can also be implemented.

For better data visualization a “day-time-plot” should be created to show if the ping has repeating patterns throughout the day. Something that would be very easy to add to the scatter plot is a different approximation, such as the polynomial curve.

References

- [1] “Network Engineer Software Tools for Remote Management | SolarWinds.” <https://www.solarwinds.com/engineers-toolset> (accessed Feb. 07, 2022).
- [2] “Compare Editions - Ping Monitor.” <https://emcosoftware.com/ping-monitor/feature-list> (accessed Feb. 07, 2022).
- [3] “Nagios Core” *Nagios*. <https://www.nagios.org/projects/nagios-core/> (accessed Feb. 07, 2022).
- [4] P. Gralla, *How the Internet Works*. Que Publishing, 1998.
- [5] V. Dejawakh, “How the Internet Works, Part I - The Internet Infrastructure,” *Vahid Dejawakh*, Dec. 15, 2020. <https://vahid.blog/post/2020-12-15-how-the-internet-works-part-i-infrastructure/> (accessed Feb. 07, 2022).
- [6] W. Goralski, *The Illustrated Network: How TCP/IP Works in a Modern Network*. Morgan Kaufmann, 2009.
- [7] D. E. Comer, *The Internet Book: Everything You Need to Know about Computer Networking and How the Internet Works*. CRC Press, 2018.
- [8] “All About Routers: Types of Routers, Routing Table and IP Routing,” *Software Testing Help*, Feb. 07, 2022. <https://www.softwaretestinghelp.com/types-of-routers-routing-table/> (accessed Feb. 07, 2022).
- [9] J. Duckett, *HTML and CSS: Design and Build Websites*. John Wiley & Sons, 2011.
- [10] “An introduction to HTTP: everything you need to know,” *freeCodeCamp.org*, Sep. 11, 2019. <https://www.freecodecamp.org/news/http-and-everything-you-need-to-know-about-it/> (accessed Feb. 07, 2022).
- [11] P. V. Mockapetris and K. J. Dunlap, “Development of the Domain Name System,” p. 11.
- [12] R. Cinkais, “Encrypted DNS - The good, the bad and the ugly.,” *3Key Company*, Apr. 06, 2020. https://www.3key.com/encrypted_dns (accessed Feb. 07, 2022).
- [13] S. Sulyman, “Client-Server Model,” *IOSR J. Comput. Eng.*, vol. 16, pp. 57–71, Jan. 2014, doi: 10.9790/0661-16195771.
- [14] Article 19, *How the Internet Really Works: An Illustrated Guide to Protocols, Privacy, Censorship, and Governance*. No Starch Press, 2020.
- [15] “Client–server model,” *Wikipedia*. Feb. 07, 2022. Accessed: Feb. 07, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Client%E2%80%93server_model&oldid=1070444597
- [16] Noite.pl, *ICMP: Network Basic. AL0-023*. NOITE S.C.
- [17] K. S. Siyan, *Windows 2000 TCP/IP*. Sams Publishing, 2000.
- [18] M. Lee and S. Meyers, *Learn Mac OS X Snow Leopard*. Apress, 2011.
- [19] A. Khurshudov, *The Essential Guide to Computer Data Storage: From Floppy to DVD*. Prentice Hall Professional, 2001.
- [20] A. Bertram, *PowerShell for Sysadmins: Workflow Automation Made Easy*. No Starch Press, 2020.
- [21] “JSON,” *Wikipedia*. Feb. 04, 2022. Accessed: Feb. 07, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=JSON&oldid=1069850883>
- [22] D. M. Negi, *Fundamental of Database Management System: Learn essential concepts of database systems*. BPB Publications, 2019.
- [23] S. Maheshwari and R. Jain, *DBMS – Complete Practical Approach*. Firewall Media, 2005.

- [24] “Application Scaling using Microsoft SQL Server,” Feb. 07, 2022. <https://www.manageengine.com/privileged-access-management/help/application-scaling.html> (accessed Feb. 07, 2022).
- [25] D. Z. Childs, P. H. Warren, and B. J. Hindle, “APS 240: Data Analysis and Statistics with R - Graphical Representation of Data,” p. 19, Jan. 2021.
- [26] E. Tittel and J. Noble, *HTML, XHTML and CSS For Dummies*. John Wiley & Sons, 2011.
- [27] S. Howe, *Learn to Code HTML and CSS: Develop and Style Websites*. New Riders, 2014.
- [28] Kyan, *Ping3*. 2022. Accessed: Feb. 07, 2022. [Online]. Available: <https://github.com/kyan001/ping3>
- [29] “PEP 564 -- Add new time functions with nanosecond resolution,” *Python.org*. <https://www.python.org/dev/peps/pep-0564/> (accessed Feb. 07, 2022).
- [30] “Multithreading in Python 3 - Javatpoint,” *www.javatpoint.com*. <https://www.javatpoint.com/multithreading-in-python-3> (accessed Feb. 07, 2022).
- [31] “Python - Multithreaded Programming.” https://www.tutorialspoint.com/python/python_multithreading.htm (accessed Feb. 07, 2022).
- [32] “What is Multithreading in Python and How to Achieve it? Edureka,” *Edureka*, May 17, 2019. <https://www.edureka.co/blog/what-is-multithreading/> (accessed Feb. 07, 2022).
- [33] “pandas documentation — pandas 1.4.0 documentation.” <https://pandas.pydata.org/docs/> (accessed Feb. 07, 2022).
- [34] “Matplotlib — Visualization with Python.” <https://matplotlib.org/> (accessed Feb. 07, 2022).
- [35] “SciPy.” <https://scipy.org/> (accessed Feb. 07, 2022).
- [36] W. Chun, *Core Python Programming*. Prentice Hall Professional, 2001.
- [37] M. Lutz, *Programming Python*. O’Reilly Media, Inc., 2001.
- [38] G. van Rossum, “The Python Library Reference,” p. 1898.
- [39] M. Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*. Addison-Wesley Professional, 2010.
- [40] M. Pilgrim, *Dive Into Python 3*. United States, 2009. [Online]. Available: <https://link.springer.com/content/pdf/bfm%253A978-1-4302-2416-7%252F1.pdf>
- [41] G. van Rossum, “PEP 8 -- Style Guide for Python Code.” 2013. [Online]. Available: http://cnl.sogang.ac.kr/cnlab/lectures/programming/python/PEP8_Style_Guide.pdf
- [42] G. van Rossum, “The Python Language Reference Release 3.6.0.” 2017. [Online]. Available: <https://scicomp.ethz.ch/public/manual/Python/3.6.0/reference.pdf>
- [43] G. van Rossum, “Python Tutorial Release 3.8.1.” 2020. [Online]. Available: <https://www2.karlin.mff.cuni.cz/~halas/IT/tutorial.pdf>