BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2019

Bc. Martin Kačmarčík

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

APPLICATION FOR MONITORING OF LINUX SERVERS
APLIKACE PRO MONITOROVÁNÍ SERVERŮ S OPERAČNÍM SYSTÉMEM LINUX

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Bc. Martin Kačmarčík

SUPERVISOR
VEDOUCÍ PRÁCE
doc. Ing. Dan Komosný, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Martin Kačmarčík ***ID:*** 165394
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Aplikace pro monitorování serverů s operačním systémem Linux

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s aplikací vyvíjenou na Ústavu telekomunikací pro vzdálenou práci se servery sítě PlanetLab (www.planet-lab.eu). Tato aplikace je dostupná na adrese pypi.org/project/plbmng/. Aplikaci převeďte do jazyka Python 3 a dále proveďte její aktualizaci na repositáři PyPI. Aplikaci dále rozšiřte o možnost vyhledávaní serverů sítě PlanetLab podle jejich aktuálního stavu činnosti. Vytvořený kód vystavte pod licencí MIT. Aktualizujte popis aplikace v anglickém jazyce.

DOPORUČENÁ LITERATURA:

[1] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[2] PILGRIM, M. Ponořme se do Python(u) 3. CZ.NIC, 2010. 435 s. ISBN: 978-80-904248-2-1.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRACT

PlanetLab Server Manager is an application helping users to develop distributed network projects using the PlanetLab network. It gives an ability to search for a server by its geographical position and access these servers. It offers functionality to render servers on world map. This thesis covers details of the PlanetLab network and its infrastructure. It describes PlanetLab Server Manager (also known as plbmng) and identifies its problems in the current version. The thesis aims to address these problems and describes all the improvements made to the application. Finally, it analyzes the overall status of PlanetLab network using the newly developed tools for status monitoring. To help enable the network projects even further, this Master thesis aims to improve the current application by re-writing it fully into Python 3 language. As the current application is implemented in Bash reimplementation requires complete re-design and re-implementation of functions and will allow to fully utilize perks of Python 3 language. Thesis also aims to extend existing functionality by adding support for filtering servers by their operational status and other small improvements. Application source code is available on GitHUB under the MIT license and the application is available in the PyPI repository as well.

KEYWORDS

Critical section, Linux, Multi-processing, PlanetLab Network, PlanetLab Server Manager, Python, Virtualization

ABSTRAKT

PlanetLab Server Manager je aplikace, která pomáhá uživatelům realizovat projekty zaměřené na vývoj aplikací pro distribuované sítě. Aplikace primárně umožňuje vyhledávání serverů podle jejich geografické polohy a umožňuje přístup na tyto servery a jejich vykreslení na mapě. Práce čtenáře seznámí se sítí PlanetLab a její infrastukturou. Dále popisuje aplikaci PlanetLab Server Manager (také známou jako plbmng) a identifikuje problémy v její aktuální verzi. Diplomová práce se zaměřuje na řešení identifikovaných problémů a specifikuje provedená vylepšení do aplikace. Také je v ní pomocí nově přidané funkcionality provedena analýza stavu sítě PlanetLab. Tato práce se zaměřuje na vylepšení stávající funkcionality tím, že je aplikace plně přepsána do jazyka Python 3. Kvůli stávající implementaci v jazyce Bash, jehož vlastnosti se od jazyka Python liší, bude nutné kompletně přeplánovat stávající funkce aplikace, aby bylo plně využito výhod jazyka Python. Aplikace je také rozšířena o možnosti filtrování serverů na základě jejich stavu a další vylepšení. Zdrojové kódy aplikace jsou dostupné na repozitáři GitHUB pod licencí MIT a dále je aplikace dostupná v repozitářích PyPI.

KLÍČOVÁ SLOVA

Kritická sekce, Linux, Multi-processing, PlanetLab Server Manager, Python, Síť PlanetLab, Virtualizace

KAČMARČÍK, Martin. *Application for monitoring of Linux servers*. Brno, 2018, 54 p. Master's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications. Advised by doc. Ing. Dan Komosný, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 3.03; `http://latex.feec.vutbr.cz`

ROZŠÍŘENÝ ABSTRAKT

PlanetLab Server Manager (plbmng) je aplikace jejímž účelem je podporovat vývoj a výzkum síťových projektů, které jsou testovány na akademické síti PlanetLab. Síť PlanetLab je celosvětová akademická síť, která obsahuje přes 1300 serverů, a která nabízí uživatelům infrastrukturu pro vývoj a testování projektů. Přístup do sítě PlanetLab je zajištěn za pomocí asociace `CESNET`. Každý uživatel je přiřazen do skupiny zvané `slice`, jež definuje servery, které jsou uživateli dostupné. Jednotka serveru se v terminologie PlanetLab nazývá `node`. Infrastruktura sítě PlanetLab obsahuje virtuální servery s operačním systémem Linux. Právě pro zjednodušení práce v síti PlanetLab vznikla aplikace PlanetLab Server Manager. Tato aplikace umožňuje uživatelům vyhledání serverů v síti PlanetLab na základě jejich geografické polohy. Dále aplikace umožňuje vzdálený přístup na tyto servery pomocí protokolu ssh, případně jejich vykreslení na mapě.

Aplikace ve své přechozí verzi obsahuje několik problémů. Největšími problémy jsou programová disparita, chyby v aplikaci, nepřehledný kód, nelogická struktura, nutnost doinstalovat systémové balíčky po instalaci z repozitářů PyPI, nutnost lokalizace skriptu, náhlé ukončení činnosti při běhu a jiné. Cílem této diplomové práce je aplikaci programově sjednotit do jazyka Python 3, opravit existující chyby a rozšířit její funkcionalitu například o přidání možnosti vyhledávání serverů na základě jejich aktuálního stavu činnosti. Tato práce řeší existující problémy pomocí nového návrhu programových funkcní, úplného přepisu aplikace do jazyka Python 3.

Prvním krokem k vyřešení identifikovaných problémů bylo znovu navrhnout programové funkcne pro plné využití výhod jazyka Python 3. Jednotlivé funkce aplikace, které bylo možné oddělit do samostatných skriptů, byly odděleny do knihoven a napsány tak, aby je bylo možné importovat přímo z hlavního skriptu. Přínosem tohoto přístup je, že jsou funkce dále využitelné i mimo samotnou aplikaci. Jádro aplikace je také psáno jako knihovna a lze ji importovat do jiných aplikací v případě potřeby. Výhodou zmíněného přístupu je, že je jádro aplikace inicializováno z jednoduchého skriptu v spustitelné složce díky čemuž instalátor PyPI skript automaticky rozpozná a umístí jej po instalaci do spustitelných systémových složek. Díky tomuto vylepšení není uživatel nucen aplikace po instalaci lokalizovat. Struktura složek aplikace byla vylepsena tak, aby zjednodušila orientaci v zdrojových kódech aplikace. V rámci práce byla implementována vylepšení jako například přidání funkcionality pro filtraci serverů na základě aktuálního stavu jejich činnosti. Pro tyto účely byla použita nově přidaná interní databáze. Jádro bylo logicky rozděleno na sekce obsahující grafické rozhraní a sekce obsahující vnitřní logické funkce. V rámci práce byla provedena další vylepšení do aplikace jako například:

- Odstranění limitace výsledků vyhledávání díky použití konstrukcí jazyka Python 3, který dovoluje zjedndoušenou práci s použitou knihovnou pro uživatelské rozhraní.
- Výlepšená struktura samotné aplikace byla přepracována pro usnadnění její údržby.
- Aplikace byla vyvíjena za pomocí standardu PEP8, který definuje stylizaci kódu pro aplikace psané v jazyce Python.
- Díky programové unifikaci byla odstraněna nutnost vykonávat jakékoliv před nebo po instalační kroky.
- Vylepšení doznalo také vyplňování přihlašovacích údajů. Dřívější nefunkční pole pro jednotlivé parametry bylo nahrazeno funkčním textovým editorem se snadnou orientací.
- Byla přidána zmíněná funkce pro filtrování serverů na základě jejich činnosti a byla implementována logika pro aktualizaci této databáze pomocí multiprocessingu.
- Funkcionalita pro zobrazení serverů na mapě nyní umožňuje aplikovat filtry na servery před jejich vykreslením na mapě.
- Každý bod na mapě nyní po rozkliknutí obsahuje informaci o danném serveru.
- Byla přidána možnost rychlého přístupu na poslední zobrazený server
- V menu je nově možnost zobrazit statistiky o síťi PlanetLab.
- Byla přidána podpora operačního systému MacOS.

V rámci přepisu aplikace byly opraveny existující chyby. Výsledná vylepšená aplikace je aktualizována, včetně popisků, na příslušném repozitáři PyPI a kód je vystaven pod licencí MIT na portalu GitHUB.

DECLARATION

I declare that I have written the Master's Thesis titled "Application for monitoring of Linux servers" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Master's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno . . . .

author's signature

ACKNOWLEDGEMENT

I would like to thank my thesis advisor doc. Ing. Dan Komosný Ph.D. for his guidance, great leadership, time at consultations, patience and factual suggestions how to improve this work. Also, I would like to thank to CESNET association which has provided the necessary infrastructure to make this thesis happen. Map data copyrighted OpenStreetMap contributors and available from `https://www.openstreetmap.org`.

Brno ....

author's signature

Contents

Introduction

Developing a network project can become a challenging task. Internet is a huge worldwide network and to properly simulate usage and architecture of the internet requires at least several servers on different locations at minimum. PlanetLab Network offers a global research network that enables development of new distributed network services [29]. The goal of this Master thesis is to improve the existing tool, make it easier to use and publish the changes by updating the PyPI repositories. PlanetLab Server Manager is a tool that allows users to search for nodes in the PlanetLab network and provides ability to connect to servers using ssh protocol [2]. The current state of the application can be a barrier for more extensive usage of the application and community driven improvements due to confusing structure, program disparity, many bugs and result limitations. This Master thesis aims to solve mentioned problems by re-writing the application into Python; a popular community supported multi-platform object oriented programming language [8]. This thesis extends existing tools developed by Ivan Andrašov [2] and Filip Šuba [33].

The approach for achieving the goals of this thesis consists of using existing Bash functions, re-designing them and re-writing them to Python 3 language. During this process each functions is evaluated whether the used implementation is correct or not and function is improved if possible. During re-implementation, several enhancements to the tool will be made. Specifically, the improvements will include removing search limitation, adding library support, eliminating pre and post installations steps, improving credentials set up, writing functions with support of Windows operating system and several minor bug fixes or improvements. To achieve easier usage of the application, main focus is put on removing system package dependencies and scrapping the necessity to localize the installation folder. To achieve better readability improvements to the implementation of functions, their names and menu components are added. Special emphasis is laid on logical code structure and good programming practices to empower later community improvements to the tool. Part of this thesis is also extending existing functionality by adding server filters and other small improvements.

In the Chapter 1 the PlanetLab project will be introduced and characterized. Since PlanetLab infrastructure use Linux as the main operating system on nodes, it will be described in Subsection 1.3.1. Virtualization is described in Subsection 1.3.2 as it is the technology used for provisioning the PlanetLab nodes [29]. Since this thesis improves already existing tool created by other students, in Chapter 2 the tool and summary of previous work is reviewed. In the Chapter 3 the improvements made to the tool will be explained in detail.

1 PlanetLab Network

PlanetLab is a global research network that enables development of new network services. According to the PlanetLab project main page it was used by more than 1000 researches at top academic institutions and industrial research labs to develop a new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing since it launch at 2003 [30]. The main page description also states that PlanetLab currently consists of 1353 nodes at 717 sites and their locations worldwide can be seen in Figure 1.1.

Fig. 1.1: Map displaying location of the PlanetLab network nodes [13].

The tool's internal database created using PlanetLab API (Application Programming Interface) consists of 1000 nodes which differs from the number described on the PlanetLab website because not all of the nodes are assigned to a user's group for testing. Important aspect to mention is that not all of those nodes are accessible. As shown in Figure 1.2, in November 2018 only 196 were responding to ICMP (Internet Control Message Protocol) packets and 805 were not which is only around 19.58 % responding nodes. That means 1.62 % nodes have stopped responding since the last measurement done by Filip Šuba in his thesis [33] in 2017. Important is to mention that this does not mean that the nodes are not accessible using `ssh` protocol. In the improved version of the tool users can monitor accessibility of the nodes they always have overview which nodes can be actually used for their projects. The current committee of the project consists of members like Princeton University, Cambridge

University, Intel, Google and many more [30].

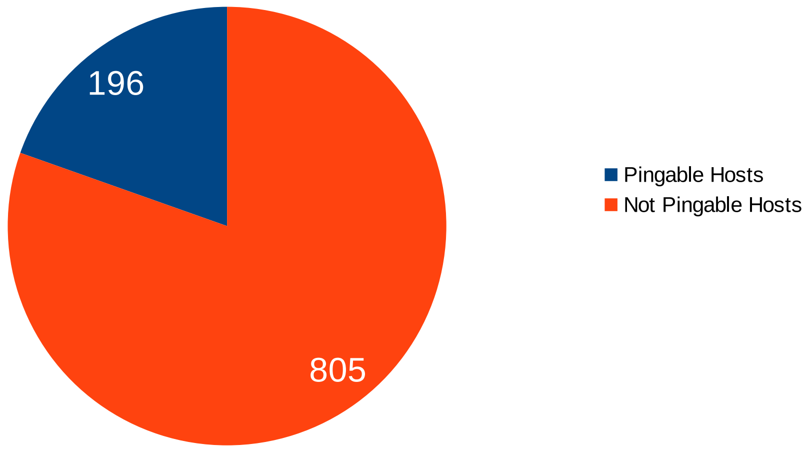

Fig. 1.2: Pie chart displaying number of nodes in PlanetLab network responding to ICMP packets. Data from November 2018.

1.1 Use and Terminology

During the initial planning of `PlanetLab network` the authors agreed on using common terminology for aspects of the network and defined them in the `Phase 0 document` [24] as follows:

- **Node:** A server machine capable of running components of PlanetLab services.
- **Site:** A physical geographical location where PlanetLab nodes are located.
- **Cluster:** The set of PlanetLab nodes located at a given site.
- **User:** An authorized human being wishing to deploy or run service over PlanetLab network.
- **Client:** A client of a service running over PlanetLab network.
- **Service:** An application running over PlanetLab network.
- **Application:** A PlanetLab service not being part of PlanetLab infrastructure.
- **Capsule:** A component of a PlanetLab service that runs on a single node.
- **Slice:** A distributed set of resources allocated to a service in PlanetLab.

1.2 Selected Projects Based on PlanetLab

In this section various projects, that PlanetLab network helped to create, will be described. All these projects wouldn't be possible without the resources PlanetLab brings. On PlanetLab site there is partial bibliography of research enabled by PlanetLab and it consist of over two hundred projects [30]. Having over two hundred projects enabled by PlanetLab network shows that PlanetLab had succeeded in their initial goals which was to provide a useful platform for networking and system research [24]. Example of projects enabled by PlanetLab are described in the following paragraphs.

Securing Web Service by Automatic Robot Detection This project is focusing on detection of automatic robots by implementing a special form of Touring test. Detection is done by comparing human versus robot behavior on the websites. According to the authors, 95 % of the human users can be detected within the first 57 requests [16].

The Design and Implementation of Next Generation Name Service for Internet Project that is aiming to solve the vulnerability of the current DNS (Domain Name System) and slow delivery of updates to the system. Project paper describes design and implementation of the Cooperative Domain Name System (CoDoNS), a novel name service, which provides high lookup performance through proactive caching, resilience to denial of service attacks through automatic load-balancing, and fast propagation of updates [19].

Slurpie: Cooperative Bulk Data Transfer Protocol Big data transfers can become problematic during peaks when huge amount of clients starts downloading the data at one point. This can occur for example during a launch of a new game or a new operating system. Slurpie is a peer-to-peer protocol for bulk data transfer that aims to reduce client download times of large popular files and to reduce load on the providing servers [26].

1.3 PlanetLab Inftrastructure

In this Section, PlanetLab infrastructure is described more in detail. The two main components of the PlanetLab infrastructure are Linux operating system, that all nodes use, and virtualization which is used to provision a encapsulated environment dedicated to users of a slice [30].

1.3.1 Operating System Linux

In this Chapter, the operating system Linux that PlanetLab nodes are running on, and that the tool is primarily developed for, will be reviewed and described.

Operating system is a connecting layer between hardware and software. It provides interface to work with system resources such as disk, processor or memory and at the same time it provides service layer for client software to run at. Linux is an open-source operating system founded by Linus Torvalds who wrote its kernel using `C language` and began the history of Linux operating system. It was originally developed for personal computers based on the Intel x86 architecture but since its creation it has been ported to many other platforms such as mobile devices, television chips and many others. A package containing Linux operating system is called Linux distribution. The defining component of each distribution is the Linux kernel [4]. Original Linux kernel has been created by Linus in 1991 [9] and since then many other forks of this kernel has emerged. Some of the most famous are `Red Hat Enterprise Linux`, `CentOS`, `Fedora`, `Debian` or `Ubuntu`. More information about mentioned distributions can be found in at the end of this Chapter. The main advantages of Linux are:

- Most of Linux distributions are available to everyone and are free to use.
- Linux is open-sourced. Source code can be inspected for potential threats.
- Linux is multi-platform and can run various devices from personal computers to televisions.
- Linux is considered to be secure. Security model of Linux is based on UNIX security principals which are considered to be robust and verified [1].
- Linux quickly adapts to changes. Since Linux is open sourced it delivers patches fast.

On other hand, Linux has some disadvantages which are summarized in the below list:

- Linux requires more technical knowledge than other systems like `Windows` or `MacOS`.
- Great number of distributions can become confusing for new users to choose from.
- Many well known applications are primarily developed for `Windows` or `MacOS` (though many open source variants of these applications exists on Linux).
- Compatibility problems. Proprietary hardware can have issues with driver compatibility. Many hardware vendors are primarily focusing on `Windows` or `MacOS`.

Linux kernel can be found in majority of devices at the moment. According to StatCounter Global Stats, 41.63 % of the machines are running on Linux kernel while 36.23 % are running on Windows-based kernel as of September 2018 [28]. The rise of Linux kernel is conditioned by raising Android popularity and in 2016 the StatCounter states Linux kernel based system had only 28.44 % market share while Windows had 48.42 % market share [28]. Linux can be found also in great amount of devices from phones, desktops, servers to television or cars. Server `opensource.com` states that more then half of all SmartTV devices runs on Linux [10]. Because Linux kernel is originally open-sourced, most of the devices worldwide runs on an open-sourced operating system which shows an interesting trend switching from proprietary software. In the following paragraphs, various popular Linux distributions will be described and their key features will be reviewed.

Red Hat Enterprise Linux Red Hat Enterprise Linux is a Linux based distribution developed by Red Hat Inc. In 1994 the first version of Red Hat Linux has been released [22]. Since its launch Red Hat became a popular enterprise Linux distribution and based on Red Hat's June 2017 statistic 100 % of all airlines, telcos and commercial banks in Fortune 500 runs on Red Hat system [23]. Red Hat advertise the system being robust and stable which is its biggest advantage for the enterprise companies. Even though Red Hat is open-sourced, it is not free as it is impossible to run it without having active subscription. Red Hat uses its own packaging system called yum (Yellowdog Updater, Modified).

CentOS The CentOS (Community Enterprise Operating System) is a Linux distribution that has been originally forked from Red Hat Enterprise Linux version 2.1 and released under name `CentOS-2` in May, 2014 [11] by John Newbigin. Since then it became a community-driven project that offers and free alternative to the Red Hat Enterprise Linux. Currently, CentOS Project is part of Red Hat while declaring to stay independent of the development of the Red Hat's Linux. Due to the distribution being free while still aiming for enterprise projects, many companies and academic institutions choose the CentOS as the primary operating system for their servers.

Fedora Fedora is a Red Hat sponsored Linux distribution developed by the Fedora Project and community around it. Currently, Fedora provides several types of system images such as Workstation for personal computers, Server for servers and Atomic for containerized applications. Fedora is distribution primarily used as a desktop operating system and shares many technologies with Red Hat Enterprise Linux and CentOS making packages compatible between these systems. Fedora is often considered to be a beta environment for changes before they make it to the more conservative Red Hat Enterprise Linux.

Debian Originally created in 1993 by Ian Murdock, Debian is a free Linux kernel based Unix like operating system developed by The Debian Project and its community. Debian in opposite to Red Hat Linux based systems uses different packaging system called APT (Advanced Packaging Tool). In February 2016 the repositories of Debian consisted of 50007 binary packages [31]. Debian is popular server distribution and in 2012 it was named by PCWorld magazine the most popular distribution for web servers [12].

Ubuntu Ubuntu is a free and open-source Linux distribution based on Debian operating system. It is being developed by Canonical and Ubuntu community. Currently Ubuntu offers three version of the system: Ubuntu Desktop for personal computers, Ubutnu Server for server and cloud and lastly Ubuntu Atomic created for IoT (Internet of Things). It is seen that Ubuntu makes the Linux more available for broader audience by having its graphical user interface similar to other operating system like `MacOS`. Ubuntu also achieved great success when Microsoft ported official Ubuntu binaries into its Windows system. Ubuntu is currently the most popular Linux distribution based on Google Trends data [7].

1.3.2 Distributed Set of Resources Using Virtualization

Every user in the PlanetLab network is assigned to a slice by slice administrator. As mentioned in Section 1.1, slice is a set of resources distributed across PlanetLab. To populate slice, nodes are assigned to it. When a node is associated to a slice, virtual servers are created on each of the assigned nodes [29]. Virtualization is therefore a vital part of the PlanetLab network and provides a way to provide users their own isolated working space.
Virtualization is an alternative to the traditional architecture. Comparison schema between virtualization and traditional architecture can be found in Figure 1.3. The traditional architecture consist of a hardware, operating system on the host and programs running on it. Application needs to be compatible with the operating system to be able to run on it. On the other hand, virtualization is a layer over the hosting operating system and offers an interface for the guest operating system to use by emulating various resources such as disk, network, USB (Universal Serial Bus) and many more. Guest operating system is a operating system of the virtual hosts running on the hypervisor. Hypervisor is providing resources to each virtual host. This enables better resource handling since all the virtual machines share same resources and theoretically have access to much higher computing power that is distributed based on real time needs. Another advantage is backward compatibility as various operating systems of a kind can be running on one host. Example is legacy version of a system that is no more compatible with current hardware but is compatible with the virtualization software. This can be particular useful for institutions that requires extremely stable well tested software running on legacy systems. Another

feature is high availability. If a virtual host crashes, many virtualization software are capable to migrate the run-time data to another virtual host live time. This can be crucial feature for any critical applications. On bare-metal hosts this can be solved by using clustered hosts.

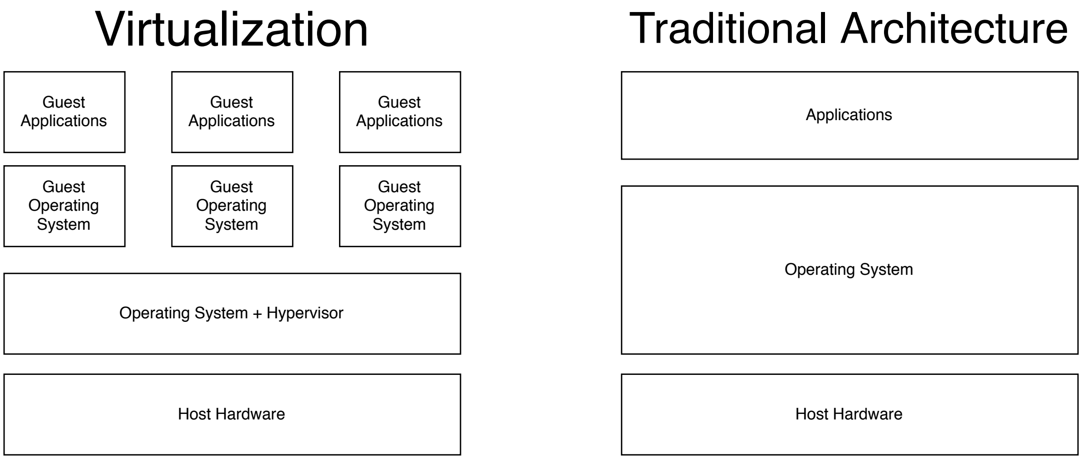

Fig. 1.3: Virtualization vs traditional architecture schema.

Currently there are many virtualization technologies available on the market. Some of the most popular ones are `VMware ESXi`, `Red Hat Enterprise Virtualization`, `Oracle Virtualbox` and `QEMU/KVM` [32]. In the following lines, some of the most popular virtualization technologies are reviewed.

VMware ESXi VMware is a company providing virtualization software. The company provides several versions of virtualization including `VMware ESXi` which is an enterprise virtualization solution. `VMware ESXi` was developed to run directly on the bare-metal hardware and doesn't require any other operating system to be installed on the host as it uses its own operating system called VMkernel [35]. This allows the software direct access to the hardware as it includes its own vital OS components. As VMware states in their architecture document [35], this also allows better hypervisor security, increased reliability, and simplified management.

Red Hat Enterprise Virtualization As Red Hat states on their website, `Red Hat Enterprise Virtualization` is an open, software-defined platform that virtualizes Linux and Microsoft Windows workloads [20]. `Red Hat Enterprise Virtualization` is a software that is developed to run on Red Hat Enterprise Linux, a Linux distribution developed by the same company. As Red Hat states in `Red Hat Enterprise`

`Virtualization` datasheet, the biggest advantage of the software is its integration with other `Red Hat` products [21]. Red Hat also offers a complete operating system designed for creating virtual machines called `Red Hat Enterprise Virtualization Hypervisor`.

Oracle Virtualbox `Oracle Virtualbox` is open source virtualization software available for both enterprise as well as home use and is developed by Oracle Corporation. As stated on the Oracle Virtualbox website: "Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD." [15] which is a considerable availability. `Oracle Virtualbox` offers many features, which are more described in the User Manual [14], such as:

- VBoxManage, a command-line interface for `Oracle Virtualbox`.
- No hardware virtualization required, supporting even older hardware without build-in virtualization support.
- Guest Additions, which is a software packages that offers features like folder sharing, automatic window focus, 3D virtualization and more.
- Multi-generation branched snapshots, which allows taking snapshot of the virtual host in any point of time completely saving the machine state allowing users to later revert machine to the saved state.

KVM with QEMU KVM (Kernel-based Virtual Machine) is a kernel module that provides a CPU (Central Processing Unit) virtualization through the use of Intel VT or AMD-V hardware extensions. KVM runs in kernel space and handles elements like processor switching or MMU (Memory Management Unit) registers, which are used to handle the virtual machines. On the other hand, QEMU is a free open source emulator that performs hardware virtualization in the user space part of the operating system. On its own, QEMU provides also CPU emulation through binary translation however the best results are delivered when combined with KVM and it can deliver near native performance [34]. During measurement of virtual machine versus host performance using the KVM technology the Geekbench's GPU test shown only 1.19 % score decrease over the bare-metal host [18].

2 Previous State Of Application Development

Application PlanetLab Server Manager, originally called `Data miner for PlanetLab`, is a supporting application for development of network projects that are using PlanetLab network. It gives its user an ability to discover and search PlanetLab servers, connect to them and upload or execute scripts. The tool is available at public PyPI repository[1]. The tool allows managing `PlanetLab` nodes, gathering information about them and pulling the latest data from the `PlanetLab` API service. Its core is written in Bash and additional modules are written in Python 3 [33]. At the moment, it is depended on both Bash and Python modules and its installation consists of several steps:

- Installing the application from PyPI repository or downloading the source codes from GitHub.
- Installing additional system packages like dialog,pssh and fping.
- Locating installation folder and putting symlink into `$PATH` directory.

Since improvement of the installation process is in scope of this Master thesis, detailed post-improvement installation steps are described later in the Chapter 3.

2.1 Description of Current Functionality

The tool consist of two layers. First one is the application logic layer and second one is graphical user interface layer, also known as menu, and consists of several options. First menu option is `Search nodes` for retrieving a node from internal database. This options allows user to either search by DNS (Domain Name System), IP (Internet Protocol) address or by node location. Second option is `Measure Menu` that allows user to schedule gathering of data about the nodes using `crontab`, select elements to monitor or start the data gathering right now. In the `Map Menu` option user has option to generate map showing location of the nodes and select map element, which in the previous versions was not working. After the first start of the application, user is required to fill credentials and `SSH` public key details to be able to access `PlanetLab` API and nodes using the menu option `Settings`. Menu is created using bash library `dialog` and is shown in Figure 2.1. Graphical interface can be run directly in terminal making it available even through `ssh` client without setting up any graphical tools.

[1] Link to PyPI repistory containing Data miner for PlanetLab tool: `https://pypi.org/project/plbmng/`

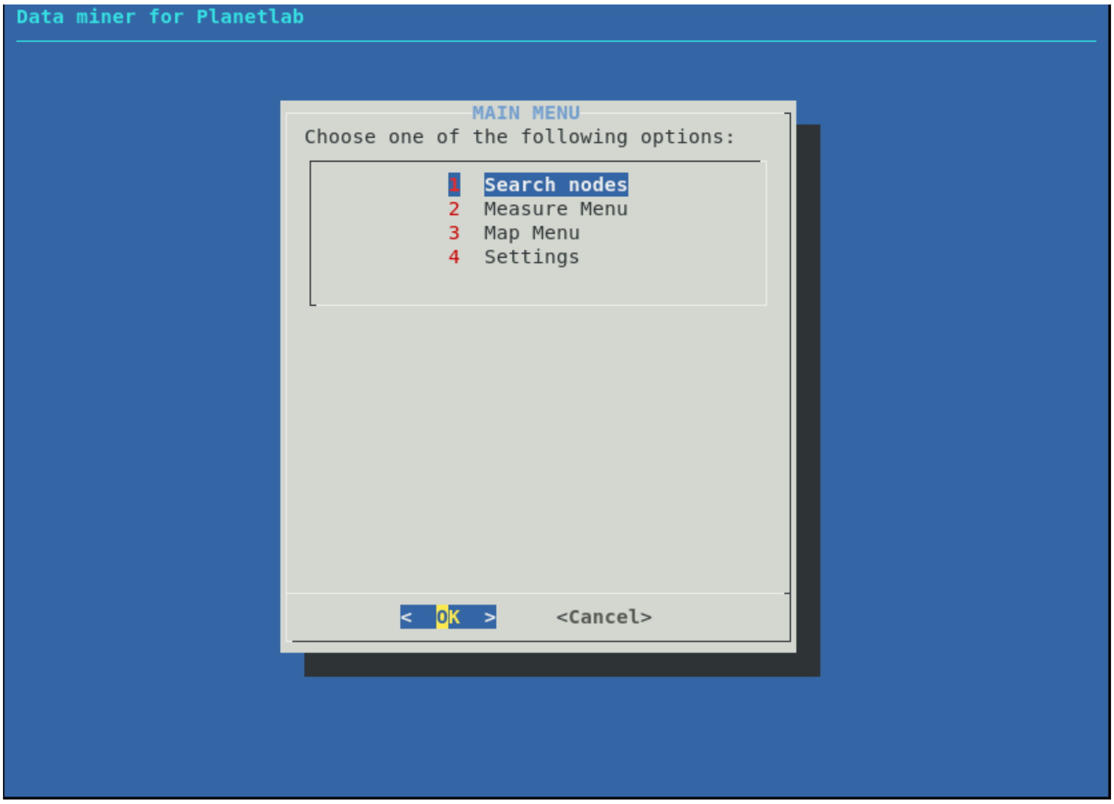

Fig. 2.1: Menu of previous version of the application called Data miner for PlanetLab.

2.2 Identified Problems

The first problem of the existing tool is language disparity having half of the functionality in Bash and half of the functionality in Python 3. This makes it difficult to make adjustment to the tool as one needs to study a vast amount of scripts that are in several different folders. Since some of the functionality is done in Python 3, which is according to portal StackOverflow fastest-growing major programming language [27], and because it is available at PyPI (Python Package Index), Python is an ideal candidate as a main language of the project. As a part of the Master thesis the existing code will be re-written into Python 3. Another great advantage of Python 3 is that it is multi-platform. As Mark Pilgrim mentions in his book [17], Python 3 is available on many platform such as `Windows`, `MacOS`, `Linux`, `BSD` and `Solaris` and their derivatives.

Second area of improvement is installation of the tool and post-installation steps. At the moment, it is required to install additional packages and tool is not automatically put into `$PATH` folders forcing its users to locate the installation folder and run the script from there. Because of the single programming language being `Python 3` the dependencies for system packages will be removed and their `Python 3` counterparts will be added as dependency for the PyPI package. PyPI installer takes care of these dependencies automatically during the installation procedure. To remove post-installation steps the tool will be written as library allowing to create a simple `Python` script in `bin` folder which is put into `$PATH` folder by the PyPI installer during the installation.

Another improvement is renaming certain menu components and adding more information to the tool itself. This change is not significant and is purely cosmetic but can make it easier for new users to get familiar with the tool. The specific rename details will be later described in Subsection 3.2.3.

The tool currently contains a lot of bugs and bad coding practices. Example of a typical bug is whole application crashing because of missing file when returning back from `Search nodes` menu. During the rewriting into `Python 3` there is space to improve certain controls to avoid these crashes and remove the need to restart the application. As for bad code practices, as an example the tool currently calls functions recursively during returning from child menu page into parent one. This means the previous function menu is stored in the `stack` waiting for the application to end before released. During rewriting of the tool these implementation details can be changed to follow good coding practices.

3 Implemented Improvements to PlanetLab Server Manager

This Chapter will discuss improvements made to the PlanetLab application. The improvements are based on the analysis made in the Section 2.2. The implementation details will be described and shown. The goal of the re-implementation is to make the `plmbng` tool simple to use, easier to contribute into by re-writing it fully into Python 3, using good coding practices, remove any post-installations steps and making small improvements. Improvements includes:

- Removing result limitation by dynamically creating menu using Python `list`.
- Adding support to use application as library by logically separating each function.
- Increasing readability and improve orientation in the application by renaming menu components and using descriptive functions names.
- Eliminating pre and post installation steps by using automatic `pip` dependency installer.
- Improving credentials set up by adding internal text editor.
- Extending Windows support to several functions by detecting operating system and dynamically changing parameters.
- Adding filtering option to limit search of nodes based on their availability.
- Displaying last accessed server.
- Adding information about nodes on a map.
- Adding status menu option to show latest statistics using internal database.
- Several bug fixes and other minor improvements.

3.1 Re-design of Application

This Section will describe the approach taken to improve the application. The issues of the application, described more in Section 2.2, are that there are two different languages, functions are scattered and application folder structure is confusing. During re-design, all these problems were taken into consideration and addressed. First problem, already well described, was solved by re-writing the application fully into Python 3 language. Second and third problem were resolved by re-designing the application folder structure and architecture. The overall architecture of re-designed application is shown in Figure 3.1 and will be described in the next paragraph.

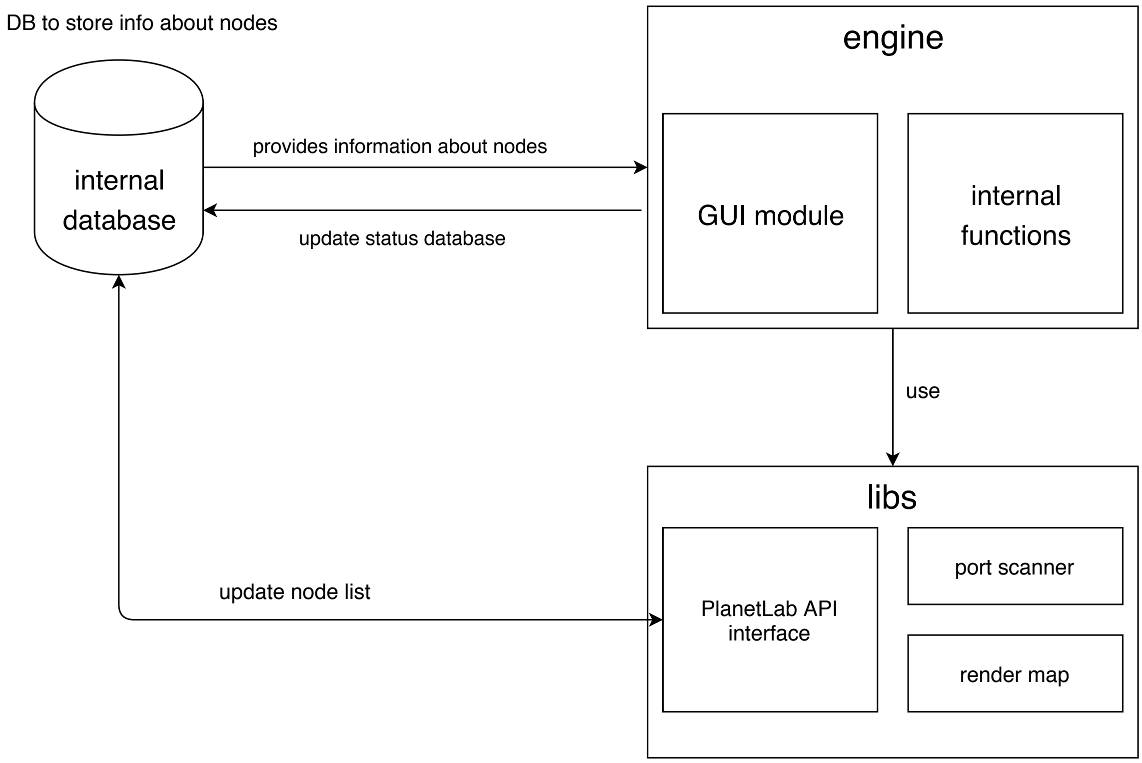

Fig. 3.1: New PlanetLab Server Manager architecture diagram.

Issue with scattered functions was solved using single file, the `engine.py`, for all the common internal functions and graphic functions and libraries for more complex functionality of the application. To make it easier to navigate inside the engine file, it was divided between graphic and internal logic sections. A question why not to split the graphical and internal logic might come to a mind. Currently, the engine module graphical and internal functions are calling each other as an reaction to certain states. Splitting them would mean importing each other in a circle which is not an ideal state. Having both sections in one file makes it one solid logical component, called the engine. However as mentioned, there are certain independent components that are split into their own files. In the Figure 3.1, these are described as `libs` and contains several utility functions that can be called independently and are imported as a library inside the `engine`. Currently, there are three of these utility modules. PlanetLab API interface, which provides data for updating the application database of servers, port scanner which allows to check port accessibility of remote host and lastly module that allows to render map of nodes using a dictionary of nodes as input. On of the challenges during re-design was a placeholder for all the information about nodes that application can later used. Previously, a text file was used to store this data. Issue with this is to implement a filter function for example, it would require to write a custom module for queries which is an unnecessary overhead as there are already solutions available to provide such module. One of these solution is relational database with SQL used as a query language. SQL is a structured query language designed to manipulate data in relational tables which are nothing more than set of related information [3]. This fits perfectly application needs as it stores a set of nodes with their related status information. In the Figure 3.1, this is shown as an internal database module. Engine triggers update of status of available nodes whereas PlanetLab API interface library updates the list of nodes. Python provides a `SQLite3` library that allows creating a custom database stored on disk. This database file is placed inside the database folder in the application configuration management. Mentioned architecture decisions will allow application to be more easily salable and maintainable in the later development phases.

3.2 Description of Improvements

In this Section, the steps to achieve goals–which were described in the Chapter introduction–will be shown in detail in their own Subsections. For each Subsection the approach, specific steps, code examples and results will be illustrated. Since the new implementation uses `pythondialog` module, at the start of the tool an instance of the `Dialog` class is spawned and will be later described just as the `instance`.

3.2.1 Removal of Search Result Limitation

The previous version of `plbmng` tool has been limited to 10 result while searching for a node. This issue was introduced due to difficulty of creating a menu based on dynamic results since author needed to always append another argument to the overall menu command. This approach can also hit limitation of characters that can be passed in a Bash command line. In Python 3 this problem does not exists since `pythondialog` module is creating menu functions based on `list`. During the search of nodes, results are added to the list which is after completed search passed to the instance which renders the GUI (Graphical User Interface). Example of this functionality is shown in Listing 3.1. Due to this improvement tool is returning all results found.

```
def searchNodesGui(prepared_choices):
  if not prepared_choices:
    d.msgbox("No results found.", width=0,height=0)
    return None
  while True:
    code, tag = d.menu("These are the results:",
                choices=prepared_choices,
                title="Search results")
```

Listing 3.1: Removing result limitation by passing choices as a list argument.

3.2.2 Writing the Application as Library

For the application to be used in other scripts and to reduce need to re-write certain code parts it is desired to write application to be able to run as a library. During the re-implementation this was considered and application is available both as library and standalone script. This will be later used in the Subsection 2.2. If the application is called as a standalone script it will trigger part of the code that is shown in Listings 3.2 which will initialize graphical interface. If imported as a library, it allows the user to call any function defined in the script.

```
if __name__ == "__main__":
  initInterface()
  exit(0)
```

Listing 3.2: Condition to recognize that application is being called as a standalone script.

3.2.3 Code Readability Improvement

Community is a powerful group that helps develop a tool and to add more functionality to it. To have community contribute to a tool, it should follow good practices and be easily readable and understandable. Previous version of the tool was using Bash script which was calling Python script and creating new Bash scripts on a disk which was merging using different pieces of code from pre-created `.dat` files in a `bin` folder. Finding a bug in this structure was difficult and navigating was not intuitive. Most functions were fully re-written into single Python script and logically divided into two sections. One section is for GUI functions and other is for logical functions. Stand-alone functions were moved to a separated scripts to be used as a library. Each functions is very descriptive in its name as shown in Listing 3.3.

```
def searchNodes(option,regex=None):
def initInterface():
def plotServersOnMap(mode):
def getPasswd():
def searchNodesGui(prepared_choices):
def printServerInfo(chosenOne):
def setCredentialsGui():
```

Listing 3.3: Example of Function Names

Each functions is trying to be as atomic as possible only having one purpose. This is helping to increase modularity of the application. Outside of this function "categories" tool is removing any `magic numbers` by defining constants at the beginning of the source code. This greatly helps to understand what is being passed as an argument and is shown in Listing 3.4 where it is descriptive what option is being passed as a search key to the `searchNodes` function. Also, the application has a block for `Initial settings` at the beginning for one single place where outside of functions definitions can be placed. Application is also honoring the conventions defined in PEP (Python Enhancement Proposal) 8 [25], like naming convention and space usage instead of tabs, as much as possible. In default, scripts like `pylint` uses snake case style naming convention however camel case that is actually described in the official pep 8 standards as a recommended descriptive naming style [25]. All these small items described are increasing the overall readability of the application for others to quickly become familiarized with it. Currently, main script `engine.py` received score of 7.38 `pylint` with main complains being about missing doc strings and certain minor warnings regarding wrong import and other small deviation from the PEP 8 standard.

```
code, tag = d.menu("Choose one of the following options:",
          choices=[("1", "Serach by DNS"),
                   ("2", "Search by IP"),
                   ("3", "Search by location")],
                   title="ACCESS SERVERS")
if code == d.OK:
  #Search by DNS
  if(tag == "1"):
    code, answer = d.inputbox("Search for:",title="Search")
    if code == d.OK:
      searchNodes(OPTION_DNS,answer)
    else:
      continue
```

Listing 3.4: Example of Constant Usage

As mentioned in the Section 2.2, renaming certain parts of the tool can improve the readability. Since the tool is not data mining rather than server manager, the tool is renamed from `Data miner or PlanetLab` into `PlanetLab Server Manager`. Version is added next to the name for users to see which they are running immediately. Another example is renaming `Search nodes` to `Access servers` since primary function of this menu item is to access the servers while search is just supporting it. The re-designed application can be seen in Figure 3.2.

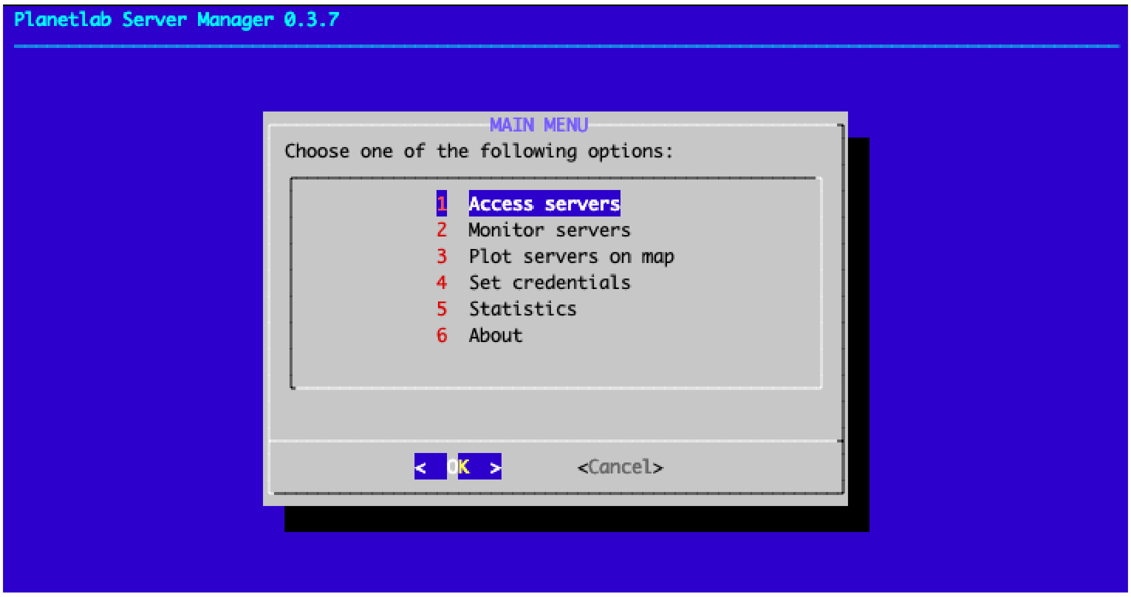

Fig. 3.2: Re-designed menu of new PlanetLab Server Manager.

3.2.4 Removal of Pre and Post Installation Steps

Previous version of application required several pre and post installation steps. In the new version developed as part of this Master thesis, all these steps were removed. Pre-installation steps were eliminated by completely getting rid of dependencies on additional system packages. All dependencies were moved into the PyPI package definition and are taken care off PyPI installer during installation of the tool. Post-installation steps were removed by adding the application into `bin` folder in the PyPI package. During installation, the PyPI installer automatically puts any scripts in the `bin` folder into a `$PATH` folder making it accessible directly from command line without the need of accessing installation folder. The contents of the main, but simple, script located in `bin` folder can be seen in Listing 3.5.

```
#! /usr/bin/env python3
from plbmng import engine
import sys

if len(sys.argv) > 1:
  if(str(sys.argv[1]) == 'crontab'):
    engine.updateAvailabilityDatabaseParent('cron')
    exit(0)

engine.initInterface()
```

Listing 3.5: Source code of the main script populated by PyPI installer into executable folders.

3.2.5 Improvement of Credentials Management

In the previous version of the tool the credentials were filled using `dialog` forms. When typing the credentials, nothing was shown except stars thus user was unaware where is the position of the cursor. Also saving the credentials was not working properly resulting into the need to adjust the configuration file on the disk which required locating the file first by inspecting the source code. In the new version settings credential is improved by creating a virtual editor in the graphical interface itself, as shown in Figure 3.3, which allows the user transparently set the credentials. One of the disadvantage of this approach is plain text visible password in the editor as it is not hidden and user needs to be careful about setting the credentials in a safe environment. To the addition of this improvement, various credentials checks were added to the application. First run of the tool will always trigger pop-up window advising user to fill credentials first. If a user chose to update a database without having credentials filled a warning message will pop-up warning the user about that fact. Also, when user wants to ssh to a server without having username and key filled he will have warning text near the ssh option. These improvements should prevent users forgetting about filling credentials and failing actions in the application.

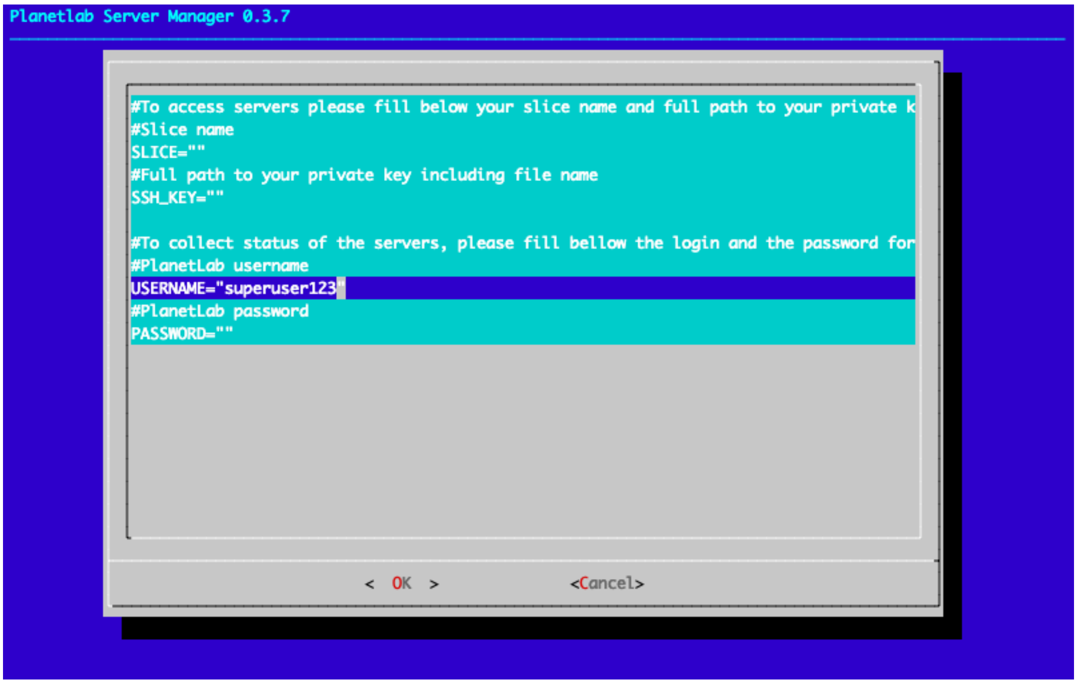

Fig. 3.3: New window for setting up credentials using internal text editor.

3.2.6 Application Folder Structure Re-design

With changes to the architecture of the application, folder management needs to be also adjusted accordingly. The root folder of the application contains two main parts, outside of necessary PyPI files. It is the `bin` folder with `plbmng` script itself and the `plbmng` folder which contains the whole application logic and graphical interface. This folder will be now described in detail. As characterized in Section 3.1, libraries are placed in the `lib` folder instead of `python_scripts` previously. In past, the configuration files were stored in `bin` folder with shell scripts. This has been renamed to `conf` folder and library functions were moved to the `lib` folder. There are information that needs to be stored persistently, like information about accessibility of nodes. For this purposes, there is a folder called `database` which contains file like `internal.db` which is already mentioned `SQLite3` database. With all these changes to the folder structure, the project should be now more transparent, clear and easier for other developers to join the project.

3.2.7 Filtering Nodes Based on Accessibility

One of the goals if this Master thesis was to add a functionality to filter nodes based on their accessibility. This Subsection will describe how the functionality was designed and implemented. The overall idea was to add a way to filter current node

based on their availability, have an option to change these settings, show these settings and have a way to update the data. In the following paragraph, all these items will be described more in detail.
As it was mentioned in Section 3.1, the core of this functionality was to have a placeholder for the data about the nodes. This was achieved using the `SQLite3` Python library. To store the information, table `availability` was created and its structure can be seen in Table 3.1. First, `nkey` column is a primary key column that contains ID of the table row. Next is `shash` that contains unique hash of the node hostname. This hash is used to find already existing records so these are not duplicated but only updated. Hash is generated using md5 function and `hashlib` Python library. Next column is a `hostname` of the node for purposes of filtering and lastly there is the double `bssh` and `bping` which are Boolean flag displaying node accessibility. During insert into the database, `nkey` is automatically incremented thus only other values needs to be specified. Before the filtering is explained, first the internal logic working with nodes needs to be characterized. When an user select to display nodes, the search function takes a dictionary of nodes as an input parameter. This simple logic is used for the filtering itself. In the function `getNodes()`, before the list of nodes is parsed, the current settings are loaded from the database from the `configuration` table which contains simply ID, name of the setting and Boolean if it is enabled or not. These settings are used to get all hostnames of the nodes that fits the current settings. For example, if user wants to filter only ssh available nodes, the query will be `select * from availability where bssh='T'`. With the list of desired nodes the function will always return only nodes that fits user's filter criteria.

nkey	shash	shostname	bssh	bping
2	fe27ca7d1707e86e1739b1819743dc79	planetlab2.fri.uni-lj.si	F	F
3	57da801bf4370f2a163a81bdf6bafa8c	ple01.fc.univie.ac.at	T	F
4	3c956d5e295f17cb303773f83c84bf17	aladdin.planetlab.extranet.uni-passau.de	T	T

Tab. 3.1: Structure and examples of availability table for filtering functionality.

Applying filters is done using `Filtering options` in the `Access servers` menu. As shown in Figure 3.4, user can choose between ssh or ping available nodes. The selection is done using checkboxes. The choices are then shown in the `Access servers` menu on the top as *Active filters* line with the listing of the active options.

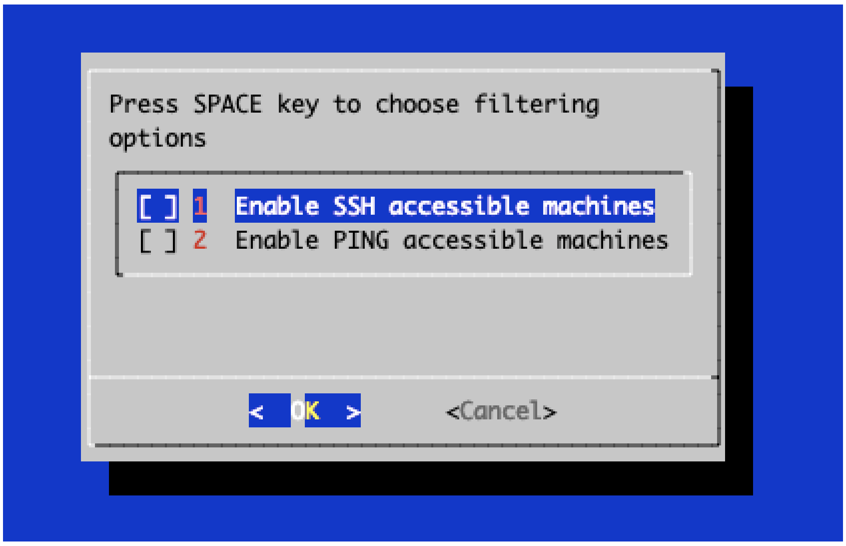

Fig. 3.4: Figure showing new menu where user can choose active filters.

3.2.8 Multiprocessing Update of Status Database

Updating the status database, or to be precise updating table inside the database, can be triggered from the `Monitor servers` menu. This will start a progress bar that takes approximately two minutes after optimization. Before optimization, the procedure took around 40 minutes. This speed was reached using multi-processing. Once procedure to update database is triggered, fifty processes are spawned at a same time using Python *multiprocessing* library. List of nodes is passed as an iterator to go over in a loop to the `Pool` object as shown in Listing 3.6. In addition, one lock is created to take over critical section of updating the shared status database and progress bar. Critical section can be defined as an area in a computer program where process operates on the shared variable by changes its value [5]. Once a process reaches the critical section, it needs to acquire the lock to enter into it. Once a process has acquired the lock, it can start writing into the shared memory without any issue. As soon as it is done, the lock is released and another process can acquire it and start writing into the critical section. To be able to show correctly the progress, once a process finishes a task, it will lock the progress bar value, increase the progress variable with an increment that is derived from total number of nodes and then unlock the progress bar iterating to another node to gather information about it. The code of entering the critical section can be seen in Listing 3.7

```
pool = Pool(50, initializer=multiProcessingInit,
        initargs=(lock,base,increment, ))
pool.map(updateAvailabilityDatabase, nodes)
```

Listing 3.6: Creating a multi-processing object with fifty processes, three shared variables and list of nodes as iterator.

```
lock.acquire()
base.value = base.value = increment.value
updateProgressBarMultiProcessing(base.value)
lock.release()
```

Listing 3.7: Entering a critical section and acquiring lock.

3.2.9 Improvements of Node Visualization

In the previous version of the application, user could render all nodes in the node database on a map. It displayed markers which, upon clicking, showed hostname of the node. During implementation of filters, maps were also enhanced. Now, user can select between three maps to render; full map, map of nodes available over ssh and map of nodes that are ping-able. In addition, upon clicking, user can see all the stored information about the node as can be seen in Figure 3.5. The library for rendering the map was changed to use the same principle as search function. Instead of reading nodes from a file, it gets them passed as a parameter. While triggering the map module, nodes are loaded using `getNodes()` function that takes filtering into consideration in default. The filter in this case is only adjusted by the user's choice. Later, in Section 4.1 the information from map filters will be used to analyze *PlanetLab Network* more in detail. For user, maps filters are good opportunity to see which server are actually available for project testing and possibly create an uniform list of candidate nodes that are spread all over the world.

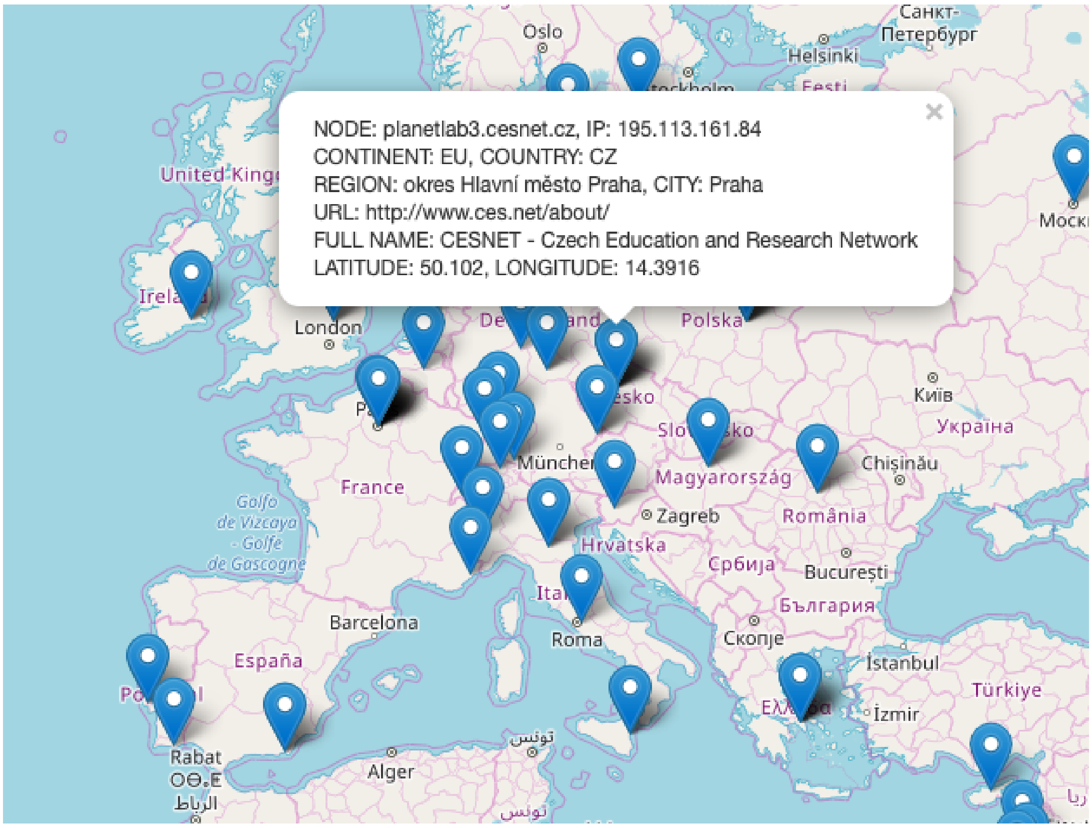

Fig. 3.5: Detail of node marker with newly added information about the particular node [13].

3.2.10 Minor Improvements

In this Subsection, minor improvements done during the re-implementation are described. All these improvements were considered minor hence they do not have separate Subsection.

Clearing of the screen after cancel When signal was send to the application using `CTRL + C` key combination, the previous version of the application was not clearing the current terminal window and the GUI. To use the same terminal user was forced to clear the window manually. In the new version, when signal is send to the application, signal handler will catch it and clean after itself.

Recursion removal for return When returning from child window to a parent page, the previous version of application was recursively calling the GUI function. This is not following good coding habits as each recursive call means storing the previous function details into the system stack, unnecessarily filling it. In the new version, while cycle is used instead and returning from a function results into new iteration of the while cycle not storing anything onto the system stack.

About is added to the menu About section is added to the menu displaying version, authors and the license.

Crontab mode created Application has option to run with `crontab` argument which will trigger just monitoring of the nodes. This is in particular useful when setting the `crontab` since the call can be simply `plbmng crontab`. This mode can be used to setup a regular updates in background to the node database.

Last server access User has now ability to return to last accessed server it the `Access servers` menu. In default, this option will just print warning message that no node was accessed. However, as soon as user access first node, this node is stored into the database in form of dump of the node dictionary variable. If user wants to access the last accessed node, this dictionary is dumped back to an internal variable and used to show this node with all the information and options.

Menu with statistics Since status database already contains all the necessary information a simple functionality regarding accessibility of the stored nodes was added to the application as can be seen in Figure 3.6.

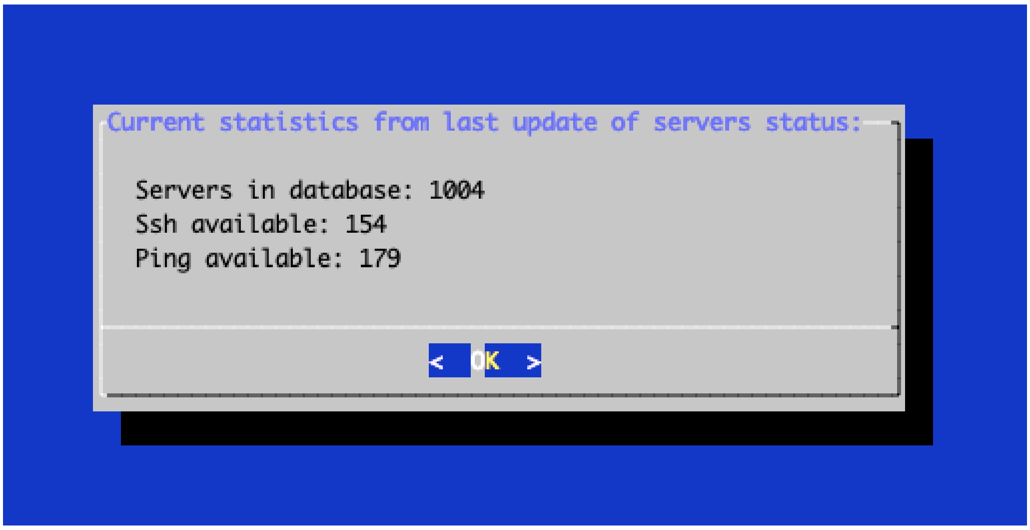

Fig. 3.6: Newly added option to display current statistics about accessibility of stored nodes.

Support of Multiple Platforms As was mentioned previously, Python is multi-platform language and brings possibility of porting the application to all kinds of different operating systems. During the re-implementation of the application this was taken into considerations and new functions were written to be able to run on Linux, Mac and possibly later on even Windows. Example of this multi-platform implementation can be seen in Listing 3.8 showing function `testPing` supporting all mentioned operating systems. As both Linux and Mac are Unix based, the implementation for these operating system is fairly simple and only minor changes are usually in several parameters. Good example is `ping` function. On Mac, time-out parameter takes one integer in millisecond as input variable. However, on Linux, the time-out parameter takes the integer variable in seconds. These little differences needs to be kept in mind when implementing platform-based functions. For Windows however, the differences are even more complex and all the libraries used in `plbmng` have to have Windows support. This was never tested yet even though most of the functions were written with Windows in mind. Adding full Windows support is a good initiative for the future development of `plbmng`.

```python
def testPing(target, returnbool=False):
  pingPacketWaitTime = None
  if system().lower() == 'windows':
    pingParam = '-n'
  else:
    pingParam = '-c'
  #for Linux ping parameter takes seconds while MAC OS ping takes miliseconds
  if system().lower() == 'linux':
    pingPacketWaitTime = 1
  else:
    pingPacketWaitTime = 800
  command = ['ping', pingParam, '1', target, '-W', str(pingPacketWaitTime)]
  p = subprocess.Popen(command, stdout=subprocess.PIPE,
  stderr=subprocess.PIPE)
  # prepare the regular expression to get time
  if system().lower() == 'windows':
    avg = re.compile('Average = ([0-9]+)ms')
  else:
    avg = re.compile(
    'min/avg/max/[a-z]+ = [0-9.]+/([0-9.]+)/[0-9.]+/[0-9.]+')
    avgStr = avg.findall(str(p.communicate()[0]))
  if p.returncode != 0:
    if not returnbool:
      return "Not reachable via~ICMP"
    return False
  else:
    p.kill()
    if not returnbool:
      return avgStr[0]+" ms"
    return True
```

Listing 3.8: Example of fully multi-platform function testPing.

3.2.11 Minor Bug Fixes

In this Subsection, minor bug fixes will be described which were not considered as enough improving to be included in separate Subsection.

Removing headers from the searches During the search, previous version of the script has also included header of the file containing information about nodes resulting into false search results. Header is now skipped and these false results are removed.

Application crashes during return When returning from a child menu window to a parent page, the application tend to crash on `grep` tool not being able to find file. During re-implementation this bug was fixed and returning now fully works.

3.3 Improved Installation and Use of Application

In this Section, the post-improvements installation procedure is described and work-flow diagram is shown in Figure 3.8. After implementing improvements, application has been updated in the PyPI repository and is available at `https://pypi.org/project/plbmng/` in version `0.3.7`. Application repository web page can be seen in Figure 3.7 and is describing the tool purpose, its Python package dependencies, installation steps, basic usage and authors. Repository page gives ability to the user to see release history or download the source files of the project. It also show the maintainers of the projects and allows users to contact them. License under which the application is written can be found the repository page as well. As part of this Master thesis, the description was changed to reflect the current settings. It was improved to be more simple for the user by reducing amount of text on the page and keeping only necessary information.

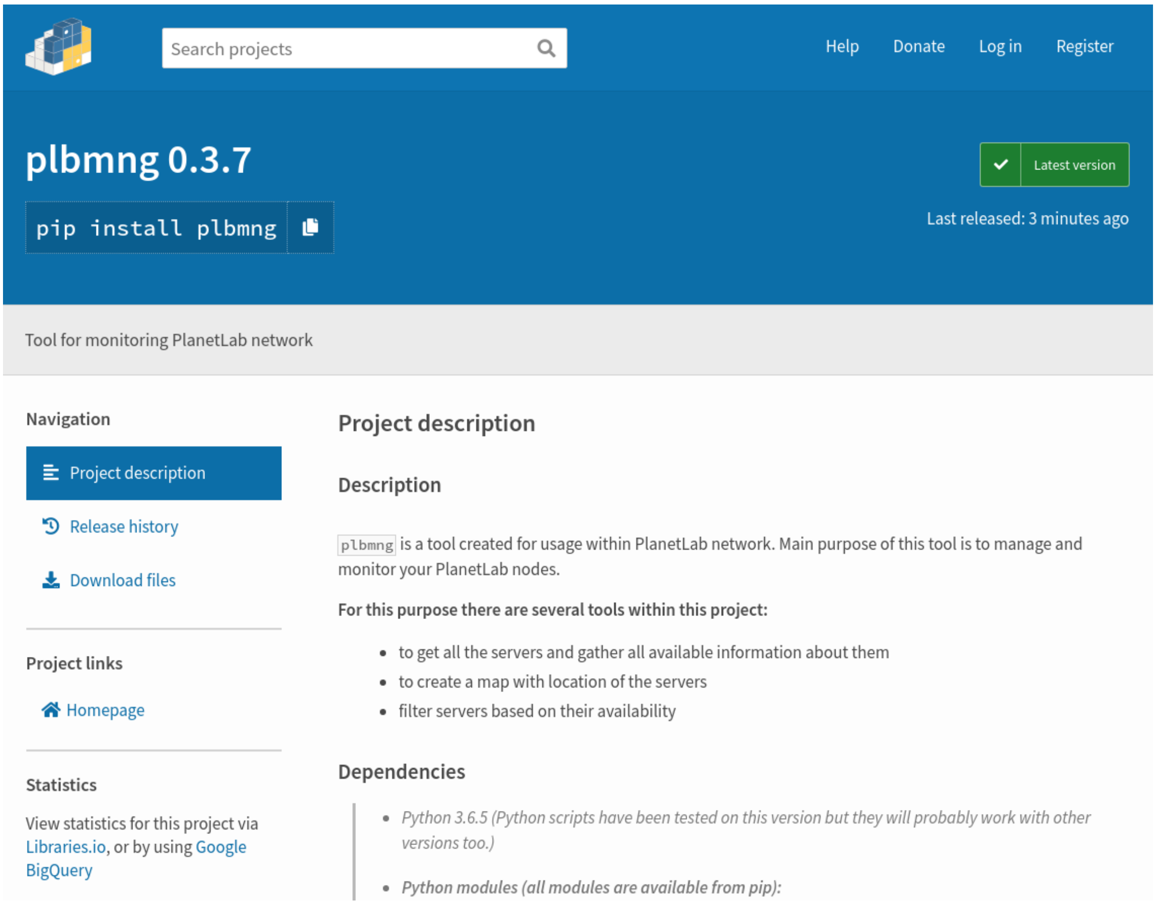

Fig. 3.7: New PlanetLab Server Manager web page in PyPI repository.

The installation steps are described in detail in the tool's repository page and consist of:

1. Installing the application using `pip3 install plbmng` command.
2. Starting the application using `plbmng` command.
3. In first start it is required to set up credentials using the `Set credentials` option in menu.

After this basic setup all the application functionality is available. It is recommended though, to update the list of nodes using `Get nodes` option in `Monitoring` menu. Installation steps are successfully reduced compared to the previous version which consisted of:

1. Installing system packages using `sudo dnf install -y dialog pssh fping` command.
2. Installing the application using `pip install plbmng` command.
3. Locating the application in a hidden folder of pip tool.
4. Finding a configuration file and using an editor to update it.
5. Running the application using absolute path.

To summarize, current application functionality workflow diagram can be seen in Figure 3.8. For sake of diagram readability, the Figure 3.8 is missing returning arrows however each node in the diagram is able to return to its parent. As seen in the Figure, first step is when menu is being initialized. From there user has option to either open `Access server menu`, `Monitor servers`, `Plot servers on map`, `Set credentials`, `About` or select `Statistics`. `Statistics` will display current data about nodes accessibility. `About` is a single window option that will show information about the software, its authors and license. `Set credentials` option will open an interactive editor where user can fill the credentials for PlanetLab network. Next option is `Plot servers on map` which offers user to choose which map elements should be displayed. After confirmation the map is generated and opened in the system default browser. `Monitor servers` option divides into three different options. First enables user to setup `crontab` to periodically scan for new nodes, second option is for updating status database and third updates list of nodes. At last, `Access servers` allows users to access the nodes by searching for them either using DNS name, IP address or location. After the search key is inserted, the available results are shown. It also offers user to get to last accessed node or select filtering option which are displayed on the top of the window. When specific node is chosen, information about the node are displayed and user has options to connect to the node using ssh, Midnight Commander or show the node on map. If credentials are not filled, warning message next to option is displayed.

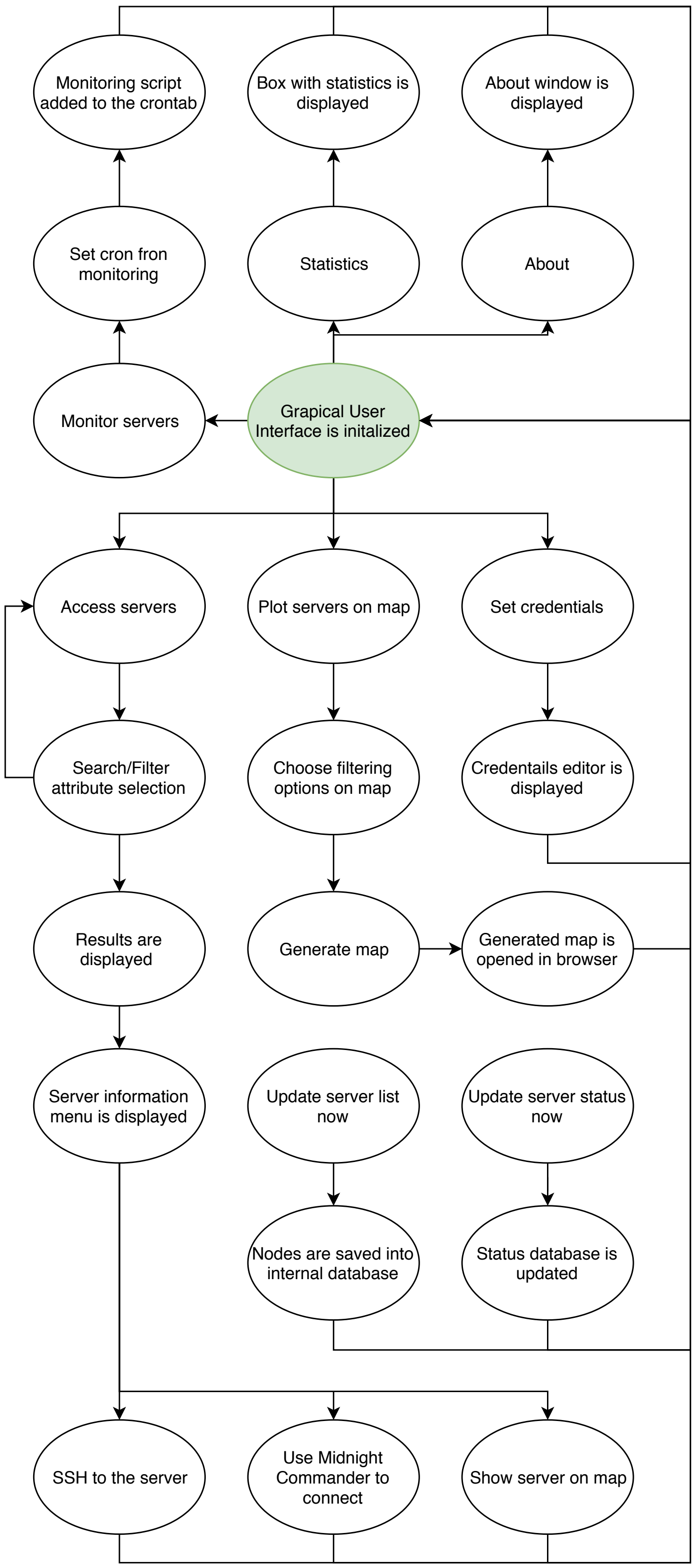

Fig. 3.8: Behavioral diagram of new PlanetLab Server Manager.

4 PlanetLab Server Manager Use Cases

In this Chapter, various use cases of the PlanetLab Server Manager tool will be revised and described. First, analysis of the PlanetLab Network will be done in Section 4.1. In Section 4.2, the very basic use case of the application, testing network projects in the PlanetLab Network, will be described and shown.

4.1 PlanetLab Network Analysis Using Plbmng Tool

In this Section, the Planet Lab network will be analyzed and discussed using the newly developed features plus using the `plbmng` database of servers pulled from the PlanetLab API. Also, data from Chapter 1 introduction will be compared to new ones with time difference of five months.

PlanetLab Network is a huge project which consists of 1353 nodes at 717 sites [30]. Currently, there are around one thousand servers available to the *cesnet_utko* user from 52 countries world wide. The Figure 4.1 shows 20 countries, represented by their country code, with the most node contribution to the network which X axis showing the country and on the Y axis showing the node count for the specific country. Most of the countries are from United States with the impressive node count of 388, second is China with its 58 nodes and third is Germany with their 49 nodes.

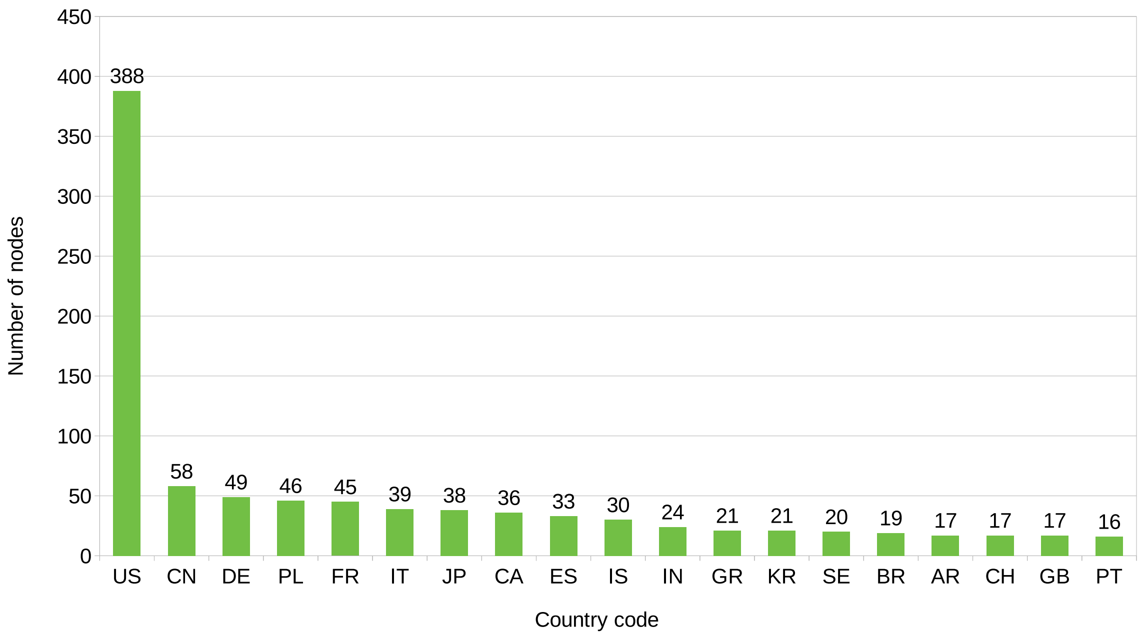

Fig. 4.1: Graph displaying top twenty countries and number of nodes they have contributed to the PlanetLab Network.

Geographically, the nodes are all over the world and are more then suitable to test large scale network project requiring multiple servers in different locations all over the world. All the nodes on a world map can be seen in Figure 4.2 which was created using the application.

Fig. 4.2: Nodes displayed on world map using the PlanetLab Server Manager map rendering functionality [13]. Data from April 2019.

However, not all the nodes are available. As seen in Figure 4.3, from the database 1004 nodes only 168 nodes are responding to ICMP packets and only 140 are listening on port 22 which is usually used for ssh. That is a very low availability percentage of only 16.7 % nodes responding to ping and even less 13.9 % listening on port 22. Interesting is also that 28 nodes stopped responding to ICMP packets since last measurement done in November 2018 as seen in Figure 1.2. That is extremely low number of available nodes and shows that the universities that are offering their nodes often do not maintain them to make sure these are still available or have left the project without decommissioning the node in the PlanetLab interface. Also, an interesting fact is that there are 28 nodes that are responding to ICMP packets but are not listening on port 22. Explanation can lie in stuck `sshd` agent or some network restrictions which are preventing to access the port 22 on specific server.

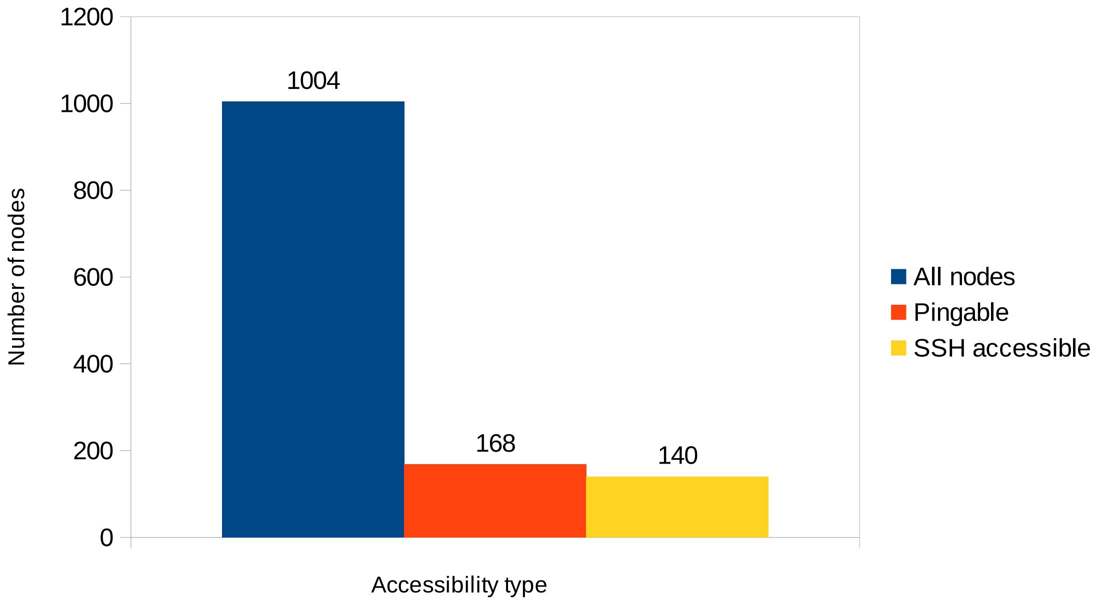

Fig. 4.3: Number of PlanetLab nodes logically divided by their accessibility. Data from April 2019.

If data are divided by continents, we will get graph with `ssh` accessible nodes in Figure 4.4. On the X axis, you can see continents and on the Y axis you can see number of accessible nodes on that particular continent. For Europe and America, the numbers are very similar. Interesting is that from South America, only 6 nodes are accessible and all are from Brazil.

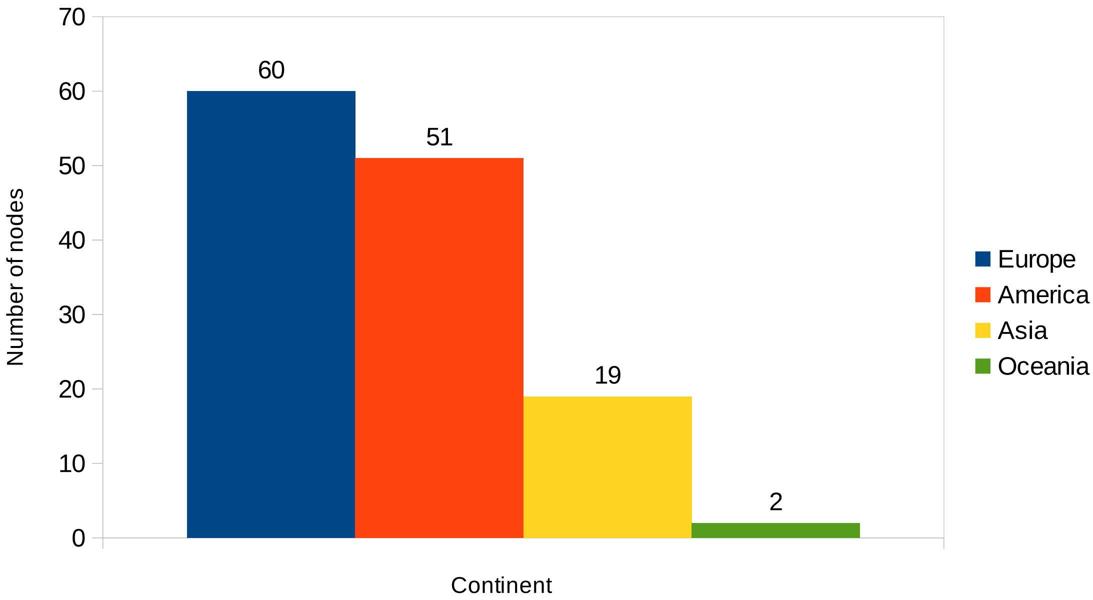

Fig. 4.4: Number of ssh accessible nodes logically divided into continents.

In more detailed breakdown, that can be seen in Table 4.1, it can be observed that Europe, with its 12 %, has the biggest percentage of ssh-accessible servers compared to overall number of nodes of all continents. Ssh accessible node means node available for project testing. On the other hand, the lowest percentage of available server has China with only 10.7 %.

Europe		America		Asia		Oceania	
All nodes	Ssh accessible	All nodes	Ssh accessible	All nodes	Ssh accessible	All nodes	Ssh accessible
332	60	422	51	177	19	18	2
100 %	18.6 %	100 %	12 %	100 %	10.7 %	100 %	11.1 %

Tab. 4.1: Table comparing number of all nodes versus ssh accessible nodes per continent using plbmng status database.

When map with all nodes, seen in Figure 4.2, is compared to map with only ssh accessible nodes, seen in Figure 4.5, the difference is even more clear. Map rendering functionality can be used to plan project testing and offers filtering only ssh available nodes.

Fig. 4.5: World map displaying only ssh-accessible nodes using application database [13]. Data from April 2019.

4.2 Testing Network Projects with Planet Lab Server Manager

In this Section, a use case of testing a network projects, on of the most crucial use cases, will be shown and illustrated. For the purpose of illustration a simple imaginary project of running a listener on port 8765 in America and accessing it from China and Europe will be described. The idea of the project is to record the times that are required to establish a connection and log necessary turn-around time for a packet to travel between continents. Mentioned project is basically a simple implementation of `ping` tool.

As a first step, credentials needs to be updated in the `Set credentials` menu options. Next, filter is turned on to show only ssh accessible servers in the `Access servers` menu section. Later, a server is found using the `Search by location` menu option and credentials are verified. If the connection is successful, testing can continue. For this purposes, simple Python script will be used. There will be a server as seen in Listings 4.1 and client as seen in Listings 4.2 which all are inspired from `cyberciti.biz` site [6]. Server will be listening on port 8765 address 0.0.0.0 and client will send a packet with message `"Test successful."` that will be send back to the client from the server as an echo reply. The results can be seen in Figure 4.6 where, in the terminal, three consecutive runs of the application were started and using the filter and `Search by location` menu option the ssh available servers from America, China and Europe were found and connected into. On the bottom, server from America is shown running the echo server script. On the top left, server from China is shown running the echo client script and on the right top corner the server from Europe is shown running also echo client script.

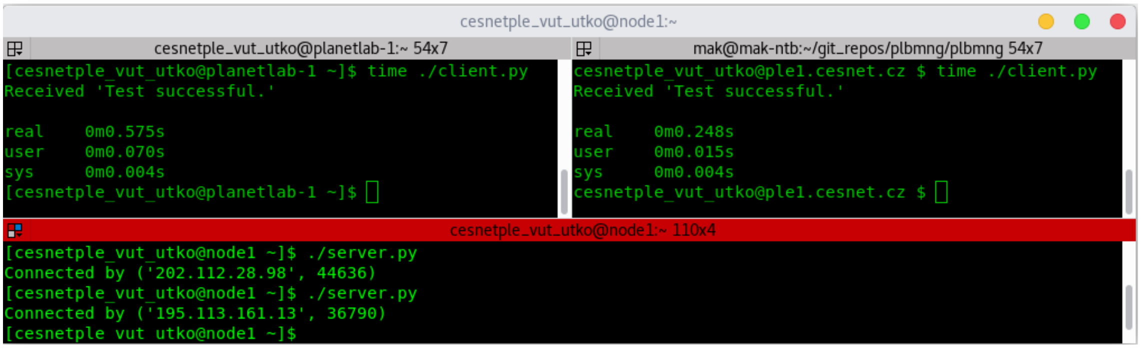

Fig. 4.6: Terminal view illustrating usage of PlanetLab Server Manager to find proper servers and run server/client echo script on them to measure round times of packets between Europe, China and America.

Script was copied into servers using Midnight Commander which provides extensive interface for managing files on both local and remote locations and can be seen in Figure 4.7. As you can see by the results, the packets were successfully delivered and the packet round time were 0.248 seconds for Europe and 0.575 second for the China. This use case demonstrate very simple project but the same use case can be applied basically to any project as long as it use existing tools available on the servers. Even though only a bit above 10 % servers are available, PlanetLab still offers a vast network of nodes for project testing.

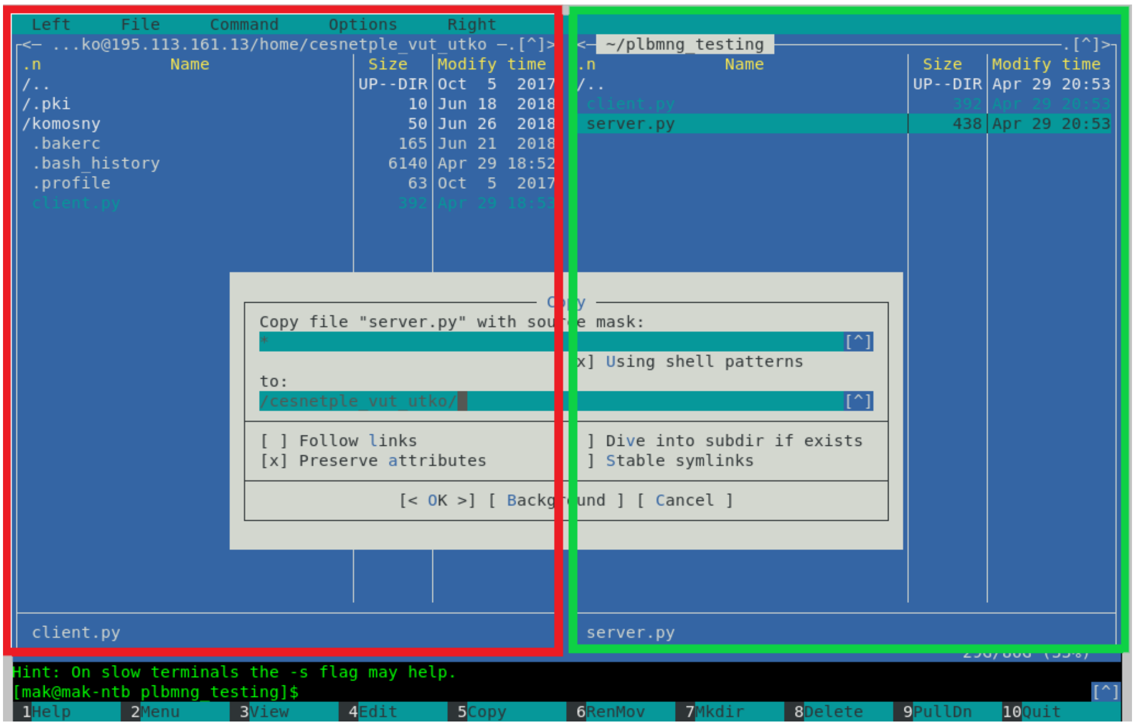

Fig. 4.7: Illustration of midnight commander used to copy scripts from localhost (green rectangle) to a remote server (red rectangle) inside the PlanetLab network.

```
#!/usr/bin/python

#Demo echo server for use case example
import socket

SOURCE_HOST = source_server_ip
PORT = 8765

s~= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SOURCE_HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected by: ', addr
while 1:
  data~= conn.recv(1024)
  if not data:
    break
  conn.send(data)
  conn.close()
```

Listing 4.1: Code of echo server.

```
#!/usr/bin/python

#Demo echo client for use case example
import socket

TARGET_HOST = target_server_ip
PORT = 8765

s~= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TARGET_HOST, PORT))
s.send('Test successful.')
data~= s.recv(1024)
s.close()
print 'Received: ', repr(data)
```

Listing 4.2: Code of echo client.

5 Conclusion

The goal of this Master thesis was to improve the current of state of the PlanetLab Server Manager (plbmng). PlanetLab Server Manager supports research and development of distributed network services. Application was described in Chapter 1. In Chapter 2, the previous version of PlanetLab Server Manager tool is reviewed, analyzed and weak points like program disparity, numerous bugs, result limitation, pre and post installation steps and others were identified and possible improvements were suggested. Since the PlanetLab network is primarily using Linux and virtualization, these technologies were covered in Subsection 1.3.1 and Subsection 1.3.2 respectively.

The improvements, as the result of this thesis, are described in Chapter 3. First, application logic was re-defined to use Python 3 advantages and new architecture diagram can be seen in Section 3.1. One of the main improvements is fast `SQLite3` database, where all data are stored, and independent library modules providing functionality to the core engine. Python usage has enabled various advancements, such as removing result limitation, removing pre and po installation steps, implementing application as a library, increasing readability by having core functions logically divided in one file instead of being composed from different files into script saved and run from disk, improving function for setting credentials, writing functions in a multi-platform way, fixing few minor bugs and adding minor improvements. Folder structure was completely re-designed to be more transparent and easily oriented. Filter function was added to help find only available nodes. Logic to update availability database in using multi-processing technology was implemented. Minor, but useful features, like accessing last server or having statistics available in the application are improving PlanetLab Server Manager usability. Full behavioral diagram with the new improvements is available in Section 3.3.

Last requirement was to update the application at the PyPI repository[1] with a new code and updated description to fit the latest information. The repository was successfully updated with the newest code increment version `0.1.10` to version `0.3.7`. The description has been updated containing latest information regarding installation process of the tool. Dependencies were removed to reflect this change. Overall, described changes might help further development of the tool and increase its usability for the PlanetLab users possibly helping develop new distributed network services.

[1]The PlanetLab Server Manager tool is available at: https://pypi.org/project/plbmng/

Bibliography

[1] *Linux: dokumentační projekt.* 4., aktualiz. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1525-1.

[2] Andrašov, I. *Měření experimentální sítě PlanetLab.* Brno: Brno, University of Technology, 2017. Bachelor thesis. Available at: https://www.vutbr.cz/studenti/zav-prace/detail/110277.

[3] Beaulieu, A. *Learning SQL.* Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN 0596007272.

[4] Eckert, J. W. *Linux+ Guide to Linux Certification.* 3rd. Boston, MA, United States: Course Technology Press, 2011. ISBN 9781418837211.

[5] Gebali, F. *Algorithms and Parallel Computing.* Hoboken, NJ, USA: Wiley, 2011. Wiley Series on Parallel and Distributed Computing. ISBN 9780470934630.

[6] Gite, V. *HowTo: UNIX / Linux Open TCP / UDP Ports* [online]. Revised: 3, September, 2010 [cit. !2019-4-29]. Available at: https://www.cyberciti.biz/faq/linux-unix-open-ports/.

[7] Google. *Google Trends* [online]. Revised: 24, November, 2018 [cit. 25. November 2018]. Available at: https://trends.google.com.

[8] Lutz, M. *Learning Python.* 2. vyd. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2003. ISBN 0596002815.

[9] Magklaras, G. *Introduction to Linux* [online]. [cit. 24. November 2018]. Available at: https://folk.uio.no/georgios/other/IntroductiontoLinux.pdf.

[10] Neary, D. *Did you know Linux is in your TV?* [online]. Revised: 08, May, 2018 [cit. 25. November 2018]. Available at: https://opensource.com/article/18/5/places-find-linux.

[11] Newbigin, J. *CentOS-2 Final finally released* [online]. Revised: 14, May, 2004 [cit. 25. November 2018]. Available at: https://goo.gl/7HsZUP.

[12] Noyes, K. *Debian Linux Named Most Popular Distro for Web Servers* [online]. Revised: 11, Jannuary, 2012 [cit. 25. November 2018]. Available at: https://goo.gl/6ZXLZU.

[13] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org* [<https://www.openstreetmap.org>]. 2017.

[14] Oracle Corporation. *Oracle VM VirtualBox User Manual* [online]. Revised: 09, November, 2018 [cit. 25. November 2018]. Available at: https://download.virtualbox.org/virtualbox/5.2.22/UserManual.pdf.

[15] Oracle Corporation. *Welcome to VirtualBox.org!* [online]. Revised: 15, November, 2018 [cit. 25. November 2018]. Available at: https://www.virtualbox.org/.

[16] Park, K., Pai, V. S., Lee, K.-W. et al. Securing Web Service by Automatic Robot Detection. *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference.* Berkeley, CA, USA: USENIX Association. S. 23–23. ATEC '06. Available at: http://dl.acm.org/citation.cfm?id=1267359.1267382.

[17] Pilgrim, M. *Ponořme se do Python(u) 3: Dive into Python 3.* Praha: CZ.NIC, c2010. ISBN 978-80-904248-2-1.

[18] Pruitt, G. *How fast is KVM? Host vs virtual machine performance* [online]. Revised: 1, December, 2016 [cit. 25. November 2018]. Available at: https://goo.gl/AUWfuH.

[19] Ramasubramanian, V. a Sirer, E. G. The Design and Implementation of a Next Generation Name Service for the Internet. *SIGCOMM Comput. Commun. Rev.* Srpen 2004, vol. 34, Issue 4, s. 331–342. Available at: http://doi.acm.org/10.1145/1030194.1015504. ISSN 0146-4833.

[20] Red Hat. *Red Hat Virtualization* [online]. Revised: 2018 [cit. 25. November 2018]. Available at: https://goo.gl/YAQ9Eb.

[21] Red Hat. *Red Hat Virtualization Datasheet* [online]. Revised: January, 2018 [cit. 25. November 2018]. Available at: https://www.redhat.com/en/resources/virtualization-datasheet.

[22] Red Hat. *Timeline: Red Hat's history* [online]. Revised: 2016 [cit. 25. November 2018]. Available at: https://goo.gl/H5w6q7.

[23] Red Hat. *Trusted* [online]. Revised: June, 2017 [cit. 25. November 2018]. Available at: https://www.redhat.com/en/about/trusted.

[24] Roscoe, T. *PlanetLab Phase 0: Technical Specification.* Princeton, New Jersey: PlanetLab Consortium, August 2002. PDN–02–002.

[25] Rossum, G. v., Warsaw, B. a Coghlan, N. *PEP 8 – Style Guide for Python Code* [online]. Revised: 1, August, 2013 [cit. 25. November 2018]. Available at: https://www.python.org/dev/peps/pep-0008/.

[26] Sherwood, R., Braud, R. a Bhattacharjee, B. Slurpie: a cooperative bulk data transfer protocol. *IEEE INFOCOM 2004*. S. 941–951 vol.2. ISSN 0743-166X.

[27] StackOverflow. *The Incredible Growth of Python* [online]. Revised: 6, September, 2017 [cit. 24. November 2018]. Available at: https://stackoverflow.blog/2017/09/06/incredible-growth-python/.

[28] StatCounter. *Operating System Market Share Worldwide 2018* [online]. Revised: October, 2018 [cit. 25. November 2018]. Available at: http://gs.statcounter.com/os-market-share.

[29] The Trustees of Princeton University. *About PlanetLab project* [online]. Revised: 29, May, 2017 [cit. 24. November 2018]. Available at: https://www.planet-lab.eu/about.

[30] The Trustees of Princeton University. *PlanetLab Main Page* [online]. Revised: 2017 [cit. 24. November 2018]. Available at: https://www.planet-lab.org/.

[31] Treinen, R. *50.000 binary packages* [online]. Revised: 8, Februray, 2016 [cit. 25. November 2018]. Available at: https://lists.debian.org/debian-devel/2016/02/msg00122.html.

[32] TrustRadius Editors. *Top Server Virtualization Software in 2018* [online]. Revised: 2018 [cit. 25. November 2018]. Available at: https://www.trustradius.com/server-virtualization.

[33] Šuba, F. *Monitorování serverů s OS Linux*. Brno: Brno, University of Technology, 2018. Bachelor thesis. Available at: https://www.vutbr.cz/studenti/zav-prace/detail/110178.

[34] Virtual Open Systems SAS. *The Virtual Open Systems video demos to virtualize ARM multicore platforms* [online]. Revised: 16, August, 2016 [cit. 25. November 2018]. Available at: https://goo.gl/Me4ikn.

[35] VMware. *Architecture of VMware ESXi* [online]. Revised: 15, October, 2008 [cit. 25. November 2018]. Available at: https://goo.gl/Gskpvf.

List of symbols, physical constants and abbreviations

DNS	Domain Name System
IP	Internet Protocol
KVM	Kernel-based Virtual Machine
MMU	Memory Management Unit
CPU	Central Processing Unit
CentOS	Community Enterprise Operating System
yum	Yellowdog Updater, Modified
APT	Advanced Packaging Tool
IoT	Internet of Things
GUI	Graphical User Interface
PEP	Python Enhancement Proposal
PyPI	Python Package Index
API	Application Programming Interface
ICMP	Internet Control Message Protocol
UML	Unified Model Language
USB	Universal Serial Bus