****

**Department of Design Engineering and Mathematics**

**Module Number: CCE3050**

**Module Title: Individual Project**

**Thesis Title: Automated VoIP Network**

**Student Name: Patryk Petryszen**

**Student Number: M00527445**

**Supervisor Name: Dr. Tuan Le**

**Submission Date: 20.04.2018**

A thesis submitted in partial fulfilment of the

requirements for the degree of Bachelor of Engineering in

Computer Communication and Networks

Contents

[List of Figures 3](#_Toc511940681)

[ACKNOWLEDGEMENTS 4](#_Toc511940682)

[Abstract 5](#_Toc511940683)

[Chapter 1. Introduction 6](#_Toc511940684)

[**1.1 Introduction** 6](#_Toc511940685)

[**1.2 Objectives ad deliverables** 7](#_Toc511940686)

[**1.3 Problem Definition** 7](#_Toc511940687)

[**1.4 Aims of this project** 8](#_Toc511940688)

[**1.5 Methodology** 8](#_Toc511940689)

[Chapter 2. Literature Review 9](#_Toc511940690)

[**2.1 Introduction** 9](#_Toc511940691)

[**2.2 Protocols Overview** 11](#_Toc511940692)

[**2.3 GNS3, Dynamips and VIRL Images** 15](#_Toc511940693)

[**2.4 Virtualization, container and docker.** 15](#_Toc511940694)

[**2.5 Software Defined Networking (SDN)** 16](#_Toc511940695)

[**2.6 Traditional vs Software Defined Networking** 17](#_Toc511940696)

[**2.7 Security** 17](#_Toc511940697)

[**2.8 Mininet** 20](#_Toc511940698)

[**2.9 Python programming language** 20](#_Toc511940699)

[Chapter 3 Design 21](#_Toc511940700)

[**3.1 Requirements** 21](#_Toc511940701)

[**3.2 Platforms** 21](#_Toc511940702)

[**3.3 Real vs Emulated** 23](#_Toc511940703)

[**3.4 Communications between clients** 24](#_Toc511940704)

[**3.5 VoIP** 25](#_Toc511940705)

[Chapter 4. Implementation and development 26](#_Toc511940706)

[**4.1 IP Addressing Scheme** 26](#_Toc511940707)

[**4.2 VMware Workstation** 27](#_Toc511940708)

[**4.3 Docker container** 28](#_Toc511940709)

[**4.4 Text editor** 29](#_Toc511940710)

[**4.5 Start-up configurations** 30](#_Toc511940711)

[**4.6 Remote configurations** 31](#_Toc511940712)

[**4.7 Problems and errors** 31](#_Toc511940713)

[Chapter 5. Evaluation 35](#_Toc511940714)

[Chapter 6. Conclusion 37](#_Toc511940715)

[References 38](#_Toc511940716)

[Appendix A – Start-up configurations with explanation 40](#_Toc511940717)

[Appendix B – Remote configuration with explanation 44](#_Toc511940718)

# List of Figures

Figure 1. OSI Model

Figure 2. A simple SIP session establishment example

Figure 3. Authentication, encryption and integrity

Figure 4. Containers vs. VMs

Figure 5. Product Development Time

Figure 6. Packet Tracer Application

Figure 7. Screenshot of GNS3 Automated VoIP Topology.

Figure 8. Examples of ISR Routers

Figure 9. Cisco Catalyst WS-C2960S 48 x POE Manageable

Figure 10. Cisco IP Phone

Figure 11. Public and Private IP Addresses

Figure 12 – Screenshot of VMware Workstation

Figure 13. Virtual Machine Settings.

Figure 14. Ubuntu docker configuration file.

Figure 15. Screenshot of Sublime Text 3.

Figure 16. IEEE 802.1Q Tag, 2018.

Figure 17. Screenshot of available network devices in GNS3

Figure 18. Screenshot of duplex mismatch error between router and the switch.

Figure 19. Screenshot of CDP neighbours issued from Switch 1.

Figure 20. Screenshot of CDP neighbours detail command issued from Switch 1.

Figure 21. Creating 2-10 VLANs on a switch using Python and paramiko library

Figure 22. Screenshot of CPU utilisation

# 

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my professor and supervisor Dr. Tuan Le for the continuous support, motivation and the freedom he gave me to do this work. He offered me valuable suggestions for this study to be successful.

I also want to say thank you to my parents and my girlfriend who have had faith in me all the time supporting me in every decision either good or bad I have made.

# Abstract

The networking world has already started to realize that network programmability more often defined as Software Defined Networking (SDN) technology is very promising and practical. SDN can solve three huge problems. Firstly, it can cut time required for setting up a network using scripts. Secondly, it can reduce number of errors caused by mistyping specific commands by a human using previously used scripts. Thirdly, it can react immediately to networks behaviour thanks to an application which communicates with an SDN controller instead of using command line for each device separately, so troubleshooting is so much easier. In this project I am going to write a script which creates VoIP topology equipped with security features. Since 1981 when first IBM PC was launched the network architecture stayed the same and we have not seen any big changes in computer networks. Even introducing IPv6 was not a huge change because it has been around for 6 years and still it did not replace IPv4. In comparison to programming where a lot of new languages have been developed through the years. Cisco for instance bought a company called Meraki in December 2012. Meraki is the company which had at that time very promising technology based on a cloud management. Today’s Cisco Meraki is the biggest cloud managed IT company which offers solutions such as wireless, switching and security. They use centralized dashboard from where they can setup the whole network topology. The concept is very similar to Software Defined Networking which is being tested and implemented right now in the industry.

The aim of this project is to create network topology with all its features such as Voice over Internet Protocol and security from a script written in Python. This is to show that networking industry will change a lot within a couple of years and every network engineer will need to learn Python programming language which is used for network automation. I will try to show that network programmability can significantly reduce the amount of time that is needed to create or modify a network. The main idea is to use same scripts for similar requirements of the network. A prototype will be shown in GNS3 emulator by using docker container and cisco IOS/VIRL images. The Python code will be run in Ubuntu docker container which will configure network topology using safe SSH protocol on port 22.

# Chapter 1. Introduction

This chapter will undertake the introduction to the Voice over Internet Protocol (VoIP) and Software Defined Networking (SDN). The aim, objectives and deliverables will be defined.

## **1.1 Introduction**

Since 1981 computer networks architecture has not changed much. The decreasing number of available IP addresses led to the introduction of IPv6 which was built to address the issues with the address exhaustion. IPv4 has 32 bits which gives us 232 available addresses. Shortage of spaces was noticed in 1992 and after ten years IPv6 was launched. It has 128 bits which gives us 2128. The digit is huge and probably we will never reach so many devices connected to the Internet. Even when IPv6 was launched it never fully replaced IPv4. According to Google statistics from 12th February 2018 we reached 18,5 percent IPv6 deployment. It hasn’t been replaced due to the introduction of Network Address Translation (NAT) which was defined in RFC 1631 standard. NAT extends the lifetime of IPv4 due to the usage of private IP addresses. Normally at home people use Port Address Translation (PAT) which is type of NAT. There is one global IP address assigned by Internet Service Provider (ISP) which represents the network in the Internet by using port numbers.

Computers networks architecture is going to change dramatically soon. The new concept of networks called Software Defined Networking (SDN) started from the company called VMware. In 1999 they released an application called VMware Workstation. Until then network, computing and storage resources had been kept separately. VMware Workstation is a hypervisor which is a small program that creates a virtual environment for a real computing environment. Thanks to this technology, there is a possibility of emulating real Linux operating system inside of the Windows software. Two operating systems could be run on the same hardware simultaneously. The usage is very convenient for big data centres. This environment is very flexible, and it gives a lot of useful features. For instance, virtual servers can be moved to any other physical device by simply copying or cloning the image to another device. At this time also, a concept of centralized control plane came back onto the scene. The network device is made from 3 layers: Management Plane, Control Plane which is the brain of the device and Data Plane. Using the concept of Centralized Control Plane, we can control our network devices from one place by using an application policy infrastructure controller (APIC). All this concepts, technologies and programs contributed to approach called Software Defined Networking which uses open protocols and is evolving and being tested now.

## **1.2 Objectives ad deliverables**

This project will be focused on automation that we can add to the network by using Software Defined Networking concept. The whole topology which includes Voice over Internet Protocol and security features will be programmed in under one minute in GNS3 emulator using the Python script which will be launched from the Ubuntu docker container. For a network engineer who uses command line interface and is not aware of network programmability this configuration would take at least 1 hour. It would have taken more if he did any mistake. In this project I would like to show how easy, fast and reliable network automation can be and how it can reduce time, errors and costs. Ubuntu client will communicate with other devices using secure socket shell (SSH) protocol which provides security because the data transmitted is fully encrypted. The whole topology will be equipped with other features like DHCP, VoIP and security. The aim of this project is to create a network topology from a script written in Python which will be equipped in security features and VoIP connection running between two virtual hosts.

## **1.3 Problem Definition**

In my project I will be focusing on Network automation. The code written in python will be responsible for configuring the topology including all its features. This can reduce time, because we can use the script on different networks that need for example voice over IP feature. The amount of errors can be reduced because we can test the code in virtual environment before implementing it to the real network. This can reduce the cost by adjusting the script to the customer’s needs. The topology will not be programmed from the beginning again. If we look at universities networks all of them are quite similar to each other. The problem is all the switches had to be programmed separately which means the process took hundreds or even thousands of hours. By implementing automation, we can reproduce the same topology and have the same network within a couple of minutes without any error caused by mistyping some commands by the human.

## **1.4 Aims of this project**

The aim of this project is to do the following:

1. Setup topology in GNS3.
2. Configure the topology in GNS3 from a script.
3. Equipped the topology with security features.
4. Establish a VoIP session between 2 hosts.

## **1.5 Methodology**

1. Required Equipment: PC, VMware Workstation, GNS3, VIRL/IOS Images, Windows Image, Cisco IP Communicator Application.
2. Design: The design will be based on Wide Area Network (WAN)
3. Testing: The testing is performed by running the code multiple times and by adding delays in the code.
4. Maintenance: The only maintenance that can be done is to save the start-up configuration of the network devices.

# Chapter 2. Literature Review

## **2.1 Introduction**

The most distant history of telephony reaches 1667 year, when British scientist Robert Hooke conveyed a voice message using 2 cans and a wire. The string was able to transmits a sound using mechanical vibrations. This method did not require any electrical current and worked only on small distances. Nowadays this setup is known as “lovers telephone”.

"The Pulsion Telephone", New Zealand: Hawke's Bay Herald, Vol. XXV, Iss. 8583, January 30, 1890, p. 3.

In 1877, when Thomas Edison constructed a device known as a phonograph. This is when analogue connections were introduced into telecommunication. An appliance was made of sound-collecting horn and a cylinder coated with tinfoil. The device was capable of recording sounds by pressing a needle into a cylinder covered with tinfoil on a rhythmic basis as a person speaks into a sound-collecting horn. Afterwards the appliance was able to play back recorded sound by changing the position of the needle at a steady speed back over the indentions made in the tinfoil (Cioara and Valentine, 2012).

A breakthrough moment happened due to invention of the phone by Alexander Graham Bell in 1876. He has experimented with the telegraph throughouthis whole life. He also invented one multiple telegraph, which could send several messages over the same line by different frequencies detected by the receiver made of tuned reeds. This led him to the idea of telephony service and by 1878 over 3000 telephones had been installed (Flood, 1987).

One of the biggest revolution in the communication occurred after the invention of the PC in 1945, which stands for personal computer. It used only 8-bit microprocessors. 16-bit microprocessors were introduced in 1981. A lot of devices that had been previously using analogue signal were converted to digital. The best example is music brand, where all the vinyl records have been replaced by compact discs commonly known as CD (Gupta and Tong, 1984).

We can recognize that the Internet was created in 1973 when the first data packet transmission occurred. It was based on ARPANET (Advanced Research Projects Agency Network) which used packet switching to allow multiple computers to communicate on a single network. It was suitable for today’s TCP/IP. That took place between University Colleague London (United Kingdom) and the Information Sciences Institute in California (United States) (Kirstein, 1999).

An important moment in the history of telecommunication took place in 1976, where voice data packet was transmitted. Thanks to Bob McAuley and Charlie Radar the first voice packet used ARPANET as a medium. It used an LPC which stands for linear predictive coder to transmit voice. In 1982 LPC was connected to PSTN (Public Switched Telephone Network) and it has been in use until now (Arroyo, 2010).

VoIP technology nowadays is used in almost every enterprise network. Many companies have decided for it because of its cheapness. Many people do not know that almost half of the phone calls are free because of the use of Local Area Network (LAN). We can define four main stages in the evolution of VoIP:

* PC to PC since 1994 – It was cheap and good only for chatting. Voice had too low quality to perform a conversation.
* PC to phone since 1996 – A lot of services were free and international calls were possible via gateway.
* Phone to Phone since 1997 – from this year a lot of people used calling cards. They were very popular in USA.
* Voice/Web integration since 1998 – calls could be made to the websites and to call centres (Minoli and Minoli, 2002).

Voice over Internet Protocol has a huge contribution in today’s network and telecommunication. The configuration requires a huge amount of commands, so it is very time-taking. Therefore, my project will be focusing on the automation of VoIP technology using cisco images and GNS3 software as an emulator by using a script written in Python.

## **2.2 Protocols Overview**

Session Initiation Protocol (SIP) has been developed by the IETF as part of the Internet Multimedia Conferencing Architecture and defined by RFC 3261.

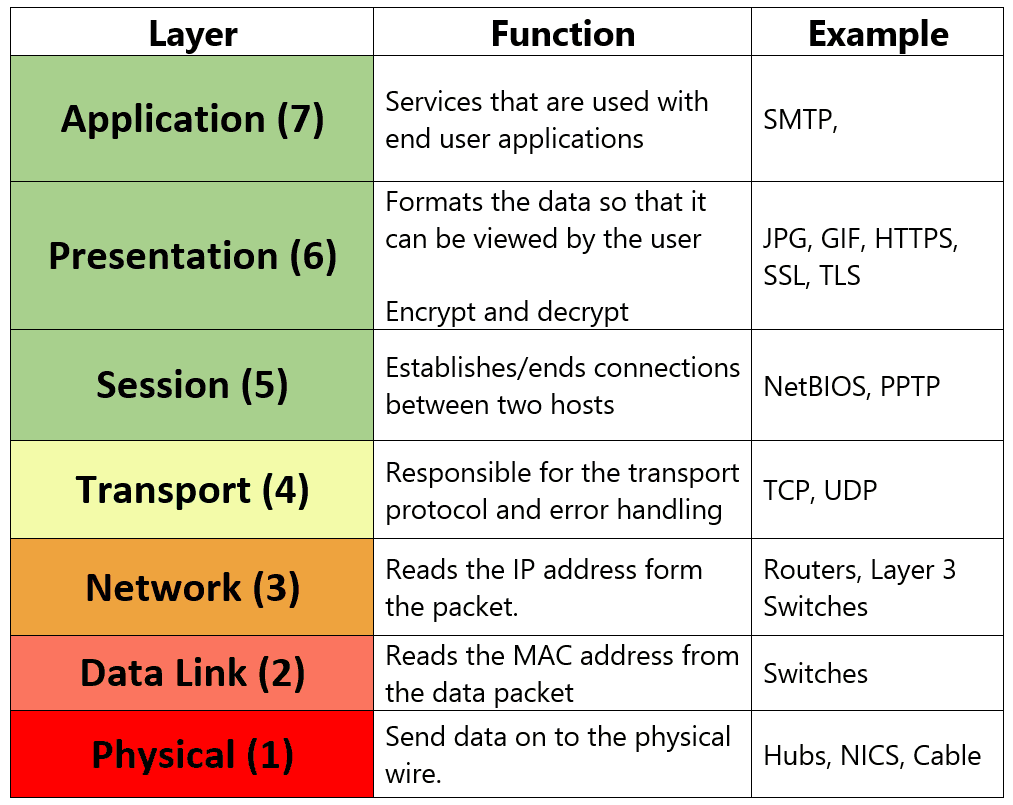
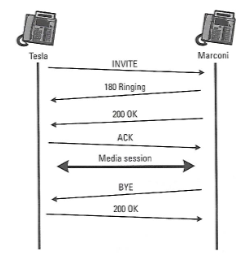
It uses a lot of protocols such as UDP, TCP, DNS and others. It is seen as the future of call signalling and telephony. It is widely used by service providers and enterprises due to the low cost of installation and maintenance. The protocols as the name suggests establishing a connection between two end points. SIP works on the seventh layer in the OSI model which is an Application Layer (see figure 1). It doesn’t provide any services, but it acts with other protocols to provide them. RTP (Real-Time Transport Protocol) for example is used to carry the voice for a call. SIP includes parts of two well-known Internet protocols: Hyper Text Transport Protocol (HTTP) and Simple Mail Transport Protocol (SMTP). From HTTP it inherited client-server design and usage of URLs and URIs. From SMTP SIP borrowed a text-encoding scheme and header style. Session establishment is very similar to 3-way handshake which occurs in TCP (see figure 2). It is good to know that one end is always User Agent Client (UAC) which sends a request and receives a response and the other User Agent Server (UAS) which receives a request and sends a response. Everything happens using proxy server which takes the request from the UAC and sends it to UAS. We can say that the role of the proxy server is very similar to that of a router. It understands a SIP request and sends it forward based on the URI. There are 2 types of proxy servers:

Figure 1. OSI Model (Policy and Policy, 2018).

Figure 2. A simple SIP session establishment example (Johnston, 2005).

* Stateful proxy server – it keeps track of all the responses and requests received which then can be used in the future.
* Stateless proxy server – it does not keep track of anything.

(Johnston, 2005).

Secure Socket Shell (SSH) it is a TCP/IP based network protocol that is used to remotely access and manage the device. It was developed by Tatu Ylönen in 1995. He developed the protocol after his university in Finland had been a victim of password sniffing attack. SSH is better than Telnet because the transmitted data is fully encrypted when Telnet transmits the data in a plain text, so everyone can read it. SSH uses Transmission Control Protocol on port 22. It uses authentication, data integrity to deliver data over unsecure network to the intendent recipient. By authentication SSH first asks for the digital proof of identity then by encryption it scrambles the data to deliver it to the intended recipient and at the end by using integrity it guarantees that the data will arrive at the destination unaltered (see figure 3).

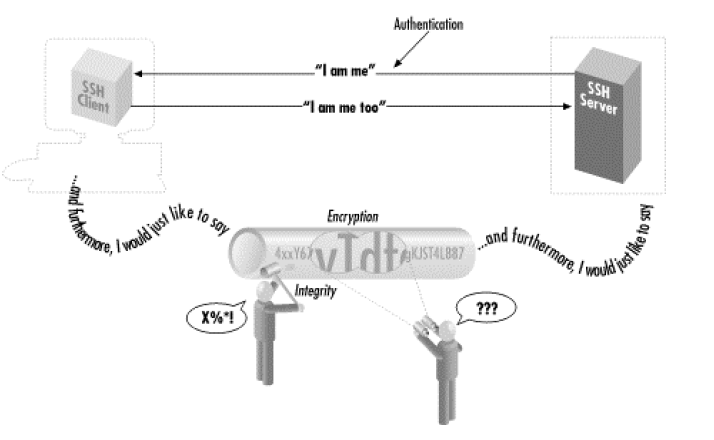


Figure 3. Authentication, encryption and integrity (Barrett and Silverman, 2001).

SSH has a lot of different authentication methods. The most secure one is based on keys instead of passwords. A key simply identifies an SSH user. If we use authentication agent together with a key, then we will be able to authenticate to all the accounts securely without the need of memorizing all the different passwords. Below we can see the following steps to accomplish this kind of authentication:

1. In advance (and only once), place special files called public key files into your remote computer accounts. These enable your SSH clients (ssh, scp) to access your remote accounts.

2. On your local machine, invoke the ssh-agent program, which runs in the background.

3. Choose the key (or keys) you will need during your login session.

4. Load the keys into the agent with the ssh-add program. This requires knowledge of each key's secret passphrase.

SSH supports several public key algorithms for authentication keys. In this project I am going to use RSA since this is the most popular algorithm (Barrett and Silverman, 2001).

RSA cryptosystem was developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977. It can be defined as a process that uses two keys (key pair) to make a secure connection. A key pair consists of one public key and one private key. The first one is used to encrypt the message and can be obtained by any host who requests it. The second one is used to decrypt the message as the name suggests and is kept secret. Exchanging the keys over a non-secure channel requires a process called Diffie-Helman. Certificate Authority (CA) has the ability to “judge” hosts whether it is legitimate or not.

To start the process both hosts need to send to the CA:

* IP Address
* Public Key
* Fully Qualified Domain Name (FQDN)

Then CA checks if the hosts are who they state they are.

Both hosts (A and B) receive digital certificate from their authority as a confirmation of what they have stated previously.

Hosts exchange their digital certificates between themselves.

Host A runs a hash algorithm against the data and he receives a message digest which is called “the hash”.

The “hash” is then encrypted with his private key and send to host B.

Host B decrypts the hash obtained by A’s public key and runs same hash against data that B’s ran.

If the result matches the signature it is authenticated.

If the received result is not equal, that could mean that someone is pretending to be host A. This attack is often called man in the middle who tries to intercept the data sent between the hosts (CCNA Security Training Videos (640-554), n.d.).

Dynamic Host Configuration Protocol (DHCP) uses User Datagram Protocol which is connectionless and does not provide reliable delivery unlike TCP. It uses port 67 and 68. DHCP provides:

* IP Address
* Subnet Mask
* Default Gateway
* DNS Server

It uses “DORA” process which consists of:

* Discover – client sends broadcast on Layer 2 and Layer 3.
* Offer – DHCP server sends broadcast on Layer 3 and unicast on Layer 2, because it knows already the MAC address of the client.
* Request – client sends broadcast to confirm the IP address that it has been offered and reject the offers from others DHCP servers.
* Acknowledgment – Unicast from DHCP server. It basically confirms that the IP address, subnet mask, default gateway and DNS server now belongs to the client (Lammle, 2011).

Point to Point Protocol (PPP) – Layer 2 (data link) protocol. Commonly used on leased lines between two routers. It supports authentication, compression, error checking and correction, and a logical multilink interface. It is open standard, so it can be used between different vendors. PPP inherits open standard High-Level Data Link Control (HDLC), because it was simple and built on the top of it. PPP starting from the bottom consists of:

- HDLC was released as an open standard but it could not support protocols.

- Link control protocol (LCP) negotiates the features of PPP.

- Network control protocol (NCP) supports layer 3 protocols.

## **2.3 GNS3, Dynamips and VIRL Images**

GNS3 stands for graphical network simulator. It can emulate real or virtual cisco images, so it is able to create complex networks which are almost the same as the real ones. It was released in 2008 to provide support for network engineers. It is used even in big companies to first check if the configuration that they have is ready to be implemented to their existing network. This can avoid introducing errors to the real topology. For comparison packet tracer is a simulator which means it is missing a lot of useful features because it doesn’t use real cisco images. We cannot then predict the behaviour of the network based on the outcome from packet tracer. GNS3 offers multiple ways to emulate IOS. In my project I will be using Dynamips and VIRL images. Dynamips uses old cisco images that are memory and CPU efficient, so they can be used directly in GNS3 software. The latest concept of emulating IOS images belongs to VIRL which stands for virtual internet routing lab. They come with a complete set of legal and licensed Cisco IOS images that are the same as those running on real physical devices. They are running inside the hypervisor. In my project routers will be Dynamips based because they are equipped with VoIP features. Switches will be running VIRL images (Gns3.com, 2018).

## **2.4 Virtualization, container and docker.**

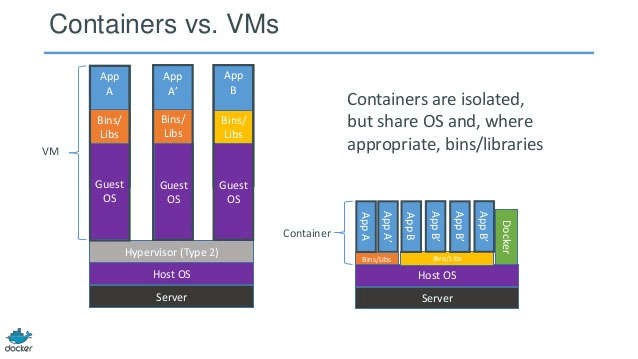
Virtual Machines consist of operating system, libraries and dependencies. On top of that there is an application running within that virtual machine. Containers have libraries and dependencies that are needed to run an application. The huge difference is that container doesn’t need operating system to host an application which makes them portable (see figure 4).

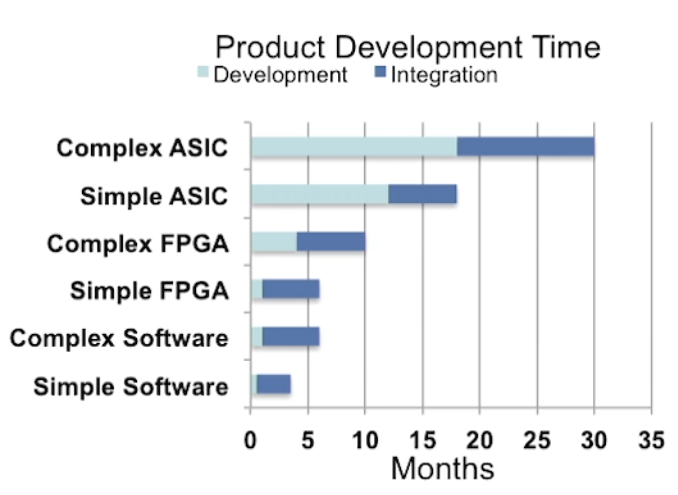
Figure 4. Containers vs. VMs, 2018.

Docker is a system that communicates with the operating system. It removes the need for a hypervisor and helps while creating a container.

The main advantage of the docker container is its portability. Thanks to them we can have one application running inside virtual environment and then we don’t need to devote our operating system to it. For example, some code needs to be written in Python 2.7 but only Python 3.0 has been installed on the operating system. In this case docker can be created and ubuntu will be installed inside of it. In this solution there is no need of deleting Python 2.7 from the machine. Imagine ruby on rails has been installed and three versions of it are needed. Without using virtual environment, 3 PC’s would be needed to accomplish this (Docker, 2018).

## **2.5 Software Defined Networking (SDN)**

When it comes to SDN we can point at one person who started this concept. Martin Casado a Senior Vice President and General Manager in the Networking Security Business Unit at VMware. He developed a protocol called OpenFlow which is the first SDN standard. It allows network devices from different vendors to be managed remotely. Devices use TCP on port 6653 to establish the connection with the controller. New concept of the network of course will need new forwarding path selection algorithms and new protocols. The hardest part in achieving this is to fully automate the network. Like OSPF has a view of entire routing topology SDN needs to have a view of everything and responds immediately when something fails or when we experience a cyber-attack. SDN doesn’t have a clear definition yet but it includes:

* OpenFlow
* Virtual Switching
* Network automation
* Device APIs
* Network Functions Virtualization
* Bare-Metal switching
* Data Centre Network Fabrics
* SD-WAN

Software Defined Networking is application based. If we compare hardware and software product development time (see figure 5) we can clearly see that development process for software is significantly shorter. That is another advantage of SDN networks over standard networks (An Introduction to Software Defined Networking (SDN), 2017).

Figure 5. Product Development Time (An Introduction to Software Defined Networking (SDN), 2017).

## **2.6 Traditional vs Software Defined Networking**

Traditional network configuration is prone to errors and time-consuming. There is a need to manually setup all the devices separately. Network engineers need extensive knowledge about devices which build the network using different vendors devices. If something happens, an administrator needs to find the root cause first and then adjusts the settings to the current problem. This sometimes leads to network outage for some time. Fully software defined networking concept gives the ability to dynamically react to any network problem. That refers to all features not only to dynamic network protocols.

## **2.7 Security**

Many of the Layer 2 protocols were created without security in mind. That is why Cisco provides many tools in its networking products so that we can add a lot of different options to secure our data.

VLANS are not considered to be a security feature but they are. In the early days of networking they were no VLANs. We used to have only one broadcast domain which means when one device sends broadcast, the rest of devices on the network had to listen to this message which caused huge CPU usage. If an attacker connects one device to the network, he could see the traffic of the whole network. We can compare this lack of segmentation to the people talking in the same room. Sometimes it is better to leave the room and talk with somebody in private rather than talk in a loud place. Large networks with no VLANs suffer from scalability issues because every host is interrupted all the time by the broadcasts messages coming from the other hosts. VLANs allow for easier management of resources by grouping hosts that they can share only what is needed. Let’s suppose we have sales finance and programming team. Second team needs access to the server where all the code is stored. For security reasons finance department is not allowed to reach that server. This is the simplest security that we can implement easily on our network.

Native VLAN – by default it is VLAN 1 and it works only while in trunking mode. It should be changed to the VLAN that is not used in the existing network since it is not tagged. Attacker knows that the default VLAN is 1 and he can try to get into our network by using double tagged frames.

Portfast should be enabled on ports connected to network endpoints. It allows a port to which the host is connected to go active almost immediately. When this feature is enabled, network engineer promises that he will not connect switch to this port otherwise it will cause broadcast storm. If he does it, Portfast will recognize BPDU’s anyway, but convergence time will be long and that’s why it is often used together with BPDU Guard which puts the port to error-disable state to avoid that.

BPDU Guard. BPDU stands for bridge protocol data unit. BDPDU’s are sent on the layer 2 using Spanning Tree Protocol. Packets contain information about addresses, priorities, ports and cost to ensure that the data arrives at the intended destination. An attacker can plug to the switch and if the switch accepts BPDU’s he can map the network. It causes a port to go into an error-disabled state if a BPDU is received.

Port Security works only on access ports. It’s a feature that allows to specify which MAC addresses are allowed from a specific switchport. They can be assigned statically or dynamically using the sticky keyword. Under the configuration the violation should be specified which is by default shutdown.

CDP – Cisco Discovery Protocol is cisco proprietary protocol. Attackers can map the topology of the network if the network engineer forgot to disable it on untrusted ports. This feature in many devices is enabled by default, hence a lot of engineers don’t remember about this feature.

DHCP Snooping protects of DHCP spoofing attack. By default, host requesting IP address by sending broadcast trusts any device from which it receives the offer. This second layer switch feature allows only trusted ports to respond to DHCP requests.

Access Control List (ACL) can be defined as a list of permissions. Based on them ACL can specify which IP addresses have access to the others. Standard Control Lists can use only destination IP address to make the decision while extended ACL’s can use source, destination IP and port numbers. Cisco recommends putting standard ACL’s as close to the destination and extended ACL’s as close to the source as possible. Each access control list can contain multiple entries but only one ACL per single interface either inbound or outbound can be applied. The entries are processed top-down. At the end of every ACL there is an implicit “deny any” statement so if we don’t permit something before that means that it will be blocked. The type of firewall called packet filter is basically made of access list. It can permit or deny traffic based on information such as source and/or destination IP address and port numbers, so it works like extended access list.

When the device comes out of the box it doesn’t have any password. First thing that should be done is to assign the passwords for virtual interfaces, console connection and enable password. If we don’t use “secret” keyword the password is not strong enough because it doesn’t use any hashing algorithm. We can see the enable password by issuing “show running-config” in exec-mode. That is why “secret” keyword is a must because it uses the message digest 5 (md5) hashing algorithm to produce a 128-bit hash value. It is default for cisco.

Challenge handshake authentication protocol (CHAP) – it is used during the initial start-up of the link. CHAP does periodic check-ups to check if the router is still in communication with the host. After the establishment has been obtained the router sends a challenge request to the remote device. This device sends back value calculated using MD5. If the value matches the connection is successful.

(CCNA Security 210-260, 2016)

## **2.8 Mininet**

Mininet it is a network emulator like GNS3 but it touches different concepts of SDN. It allows to launch virtual network with switches, hosts and an SDN controller. They can be launched with a single command. We can compare it to Powershell used in windows server where multiple commands are joined together. It supports OpenFlow protocol for custom routing and SDN. Mininet likewise GNS3 (VIRL images) is based on Linux and requires hypervisor to work. It connects switches and hosts using virtual ethernet (veth) pairs using Python (Mininet.org, 2018).

## **2.9 Python programming language**

Python is a high-level programming language created by Guido van Rossum and named after British comedy called “Monty Python”. It was released in 1991. It is open source and free. It was selected as the most popular programming language in both 2007 and 2010 year. A lot of big companies such as: Google, Facebook and Amazon use Python on a daily basis. It found its use even in artificial intelligence where we can find Pyro which stands for Python Robotics. It provides an environment for a programmer, so he doesn’t need to worry about the low low-level details of the underlying hardware. Python’s code in comparison to well-known Java’s is very readable. The latest version of Python is 3.6.5 but in production environment we can find also version 2.7.14 which is better for some particular uses (Python.org, 2018).

# Chapter 3 Design

## **3.1 Requirements**

This chapter will demonstrate design in GNS3. A few things will be taken into consideration for the project to be successful:

* Software used
* IOS/VIRL images
* Windows image
* Programming skills
* Networking knowledge
* Troubleshooting skills
* Connecting and setting up the network

## **3.2 Platforms**

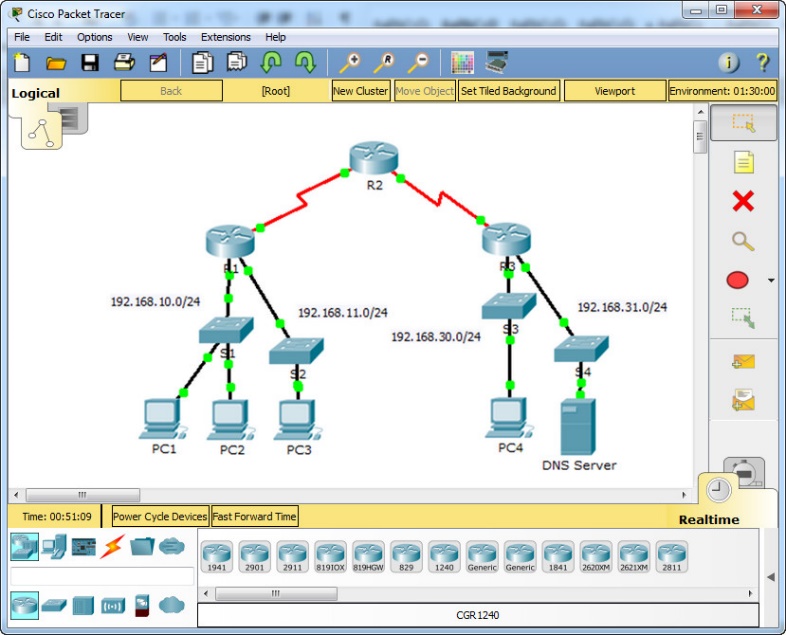
The aim of the project is to design automated, secure voice over internet protocol network. Many networking software programs have been developed through the years because networking industry is expanding. Applications such as Packet Tracer, Boston Network Simulator and GNS3 have been taken into consideration. Packet Tracer, as seen in figure 5 is a network simulator designed for students by Cisco Systems to simulate network topologies and its behaviour.

Figure 6. Packet Tracer Application, 2018.

It is available for Cisco Networking Academy students and instructors. The features are limited due to the fact that simulated images are not the real cisco images. It is recommended to use Packet Tracer on CCNA level. It is not possible to connect Packet Tracer to the real network, so it will not work together with VoIP using cisco IP communicators. We do not have any SDN concepts built in so automating the network is not possible. Boson Network Simulator on the other hand reaches CCNP concept but still doesn’t use real images so connecting existing network to it is not allowed. The only programme that can do it is GNS3, hence I am using it for the whole configuration.

To emulate cisco images and windows which have softphones installed we need to have hypervisor. Virtualbox and VMware workstation Pro had been taken into consideration. VMware workstation Pro was chosen because is less prone to errors and creators of GNS3 recommend its use rather than Virtualbox.

The following cisco images haven been used in the project:

- For the routers: c7200-adventerprisek9-mz.150-1.M.image which is running locally on the computer.

­- For the switches: vios\_l2-adventerprisek9-m.vmdk.SSA.152-4.0.55.E which is running inside the VMware Workstation Pro.

## **3.3 Real vs Emulated**

The screenshot below shows the emulated topology of the network:

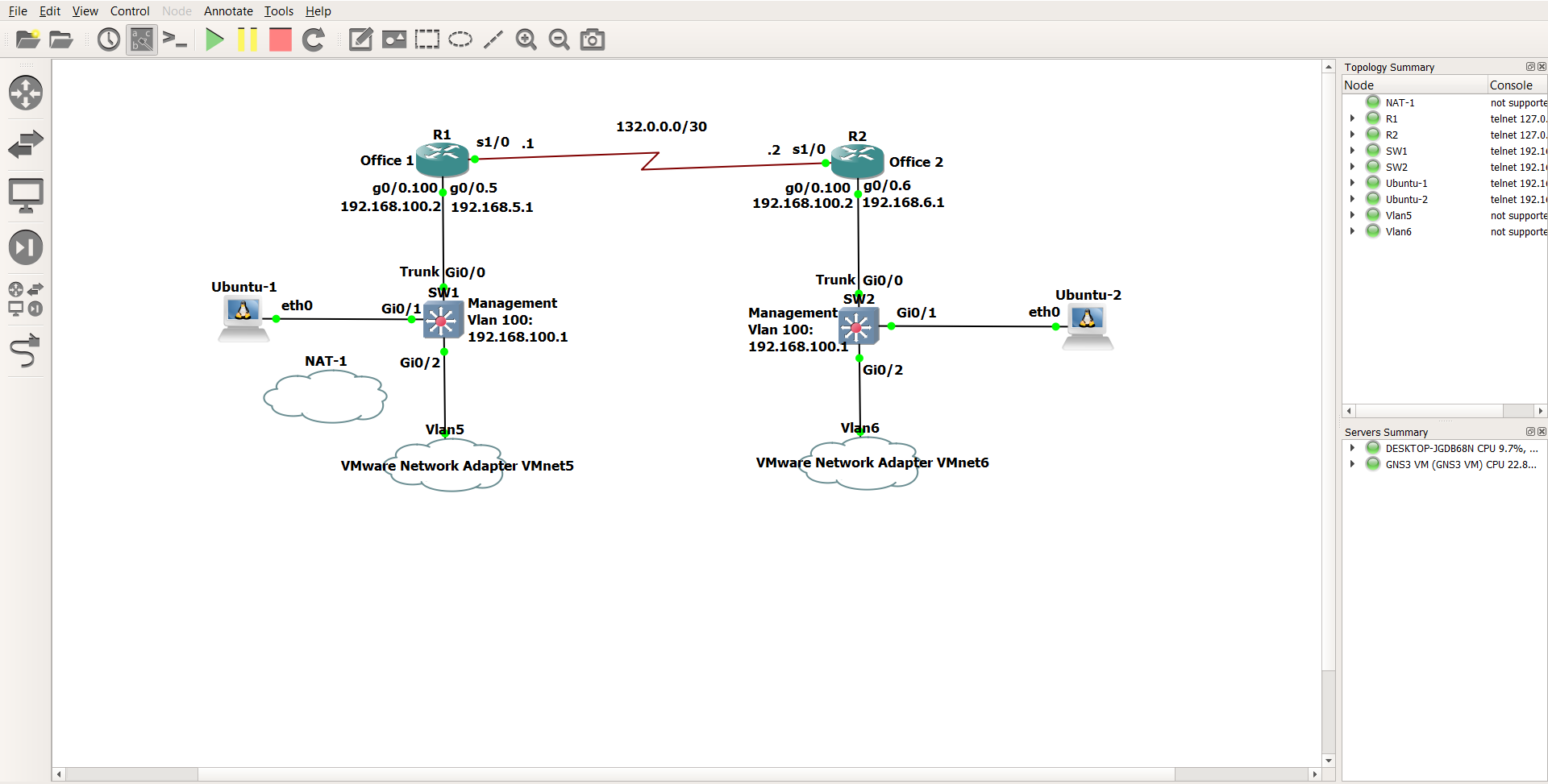


Figure 7. Screenshot of GNS3 Automated VoIP Topology.

To implement the project in the real environment routers, need to be from the group of Integrated Service Routers (ISR) which means that they are equipped with additional features like Voice Over Internet Protocol (see figure 8). In the above topology switches are L3 devices so they can work on Layer 2 and 3. They have been used because there are no L2 images that are fully compatible with GNS3. In the real environment L2 switches would be used due to the lower cost (see figure 9).

Figure 8. Examples of ISR Routers (Cisco, 2018).

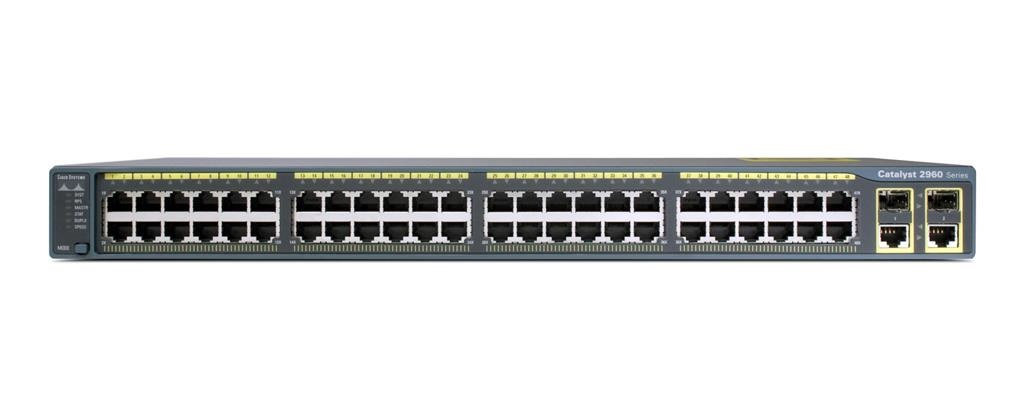


Figure 9. Cisco Catalyst WS-C2960S 48 x POE Manageable (Layer 2 Switch, 2018).

Instead of using softphones, in real environment normal phones would be used. In this case the configuration will vary, because we would have configured access port which can carry 2 VLANs at the same time (data and voice). That is the only case when access port can transmit 2 VLANs like trunk port. It’s cisco’s recommendations to configure the phone this way because we can still use port security and quality of service can be easily implemented. Cisco Phone has 2 ethernet ports. One connects the phone to the LAN and second used is used to connect the PC (see figure 10).

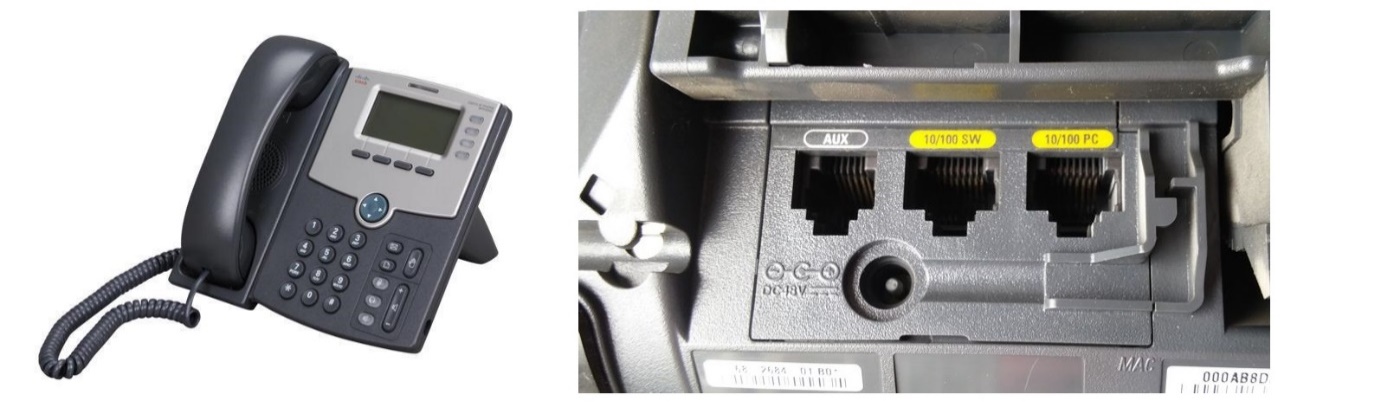


Figure 10. Cisco IP Phone, 2018.

## **3.4 Communications between clients**

There are loads of challenges when it comes to establishing secure connection between cisco soft phones that reside on Wide Area Network (WAN). First, IP addresses and subnet masks need to be assigned to devices on the network. There are 2 VLANs per Local Area Network (LAN). VLAN 5 is used to carry voice and data traffic and VLAN 100 has been created for management traffic. Routers are statically configured to pass the data over encrypted point to point protocol. Concept of router on a stick is used to separate the traffic from different VLANs. Switches have been configured with switchport security option. DHCP servers are running on both routers to distribute IP addresses to the clients inside VMware Workstation.

## **3.5 VoIP**

VoIP traffic is very sensitive in enterprise networks due to latency, jitter and packet loss. That is why QoS (Quality of Service) has been invented which can prioritize voice traffic over for example data traffic. Voice traffic requires approximately 100 kbps (kilobits per second) to achieve a clear call. QoS has not been considered while developing this project because it is not necessary within a small environment when there are no problems with quality of the voice.

There are many soft phones available on the market that are compatible with Windows. 3CX and cisco IP communicator had been taken into consideration. Since all the images in the topology are cisco’s-based cisco IP communicator was chosen as the softphone. It has only a few options available because the whole configuration is done in the router. It works as the normal Cisco Phone, so the same configuration can be done using the real equipment. In application under the preferences menu, IP address of the server which is router’s address needs to be put.

# Chapter 4. Implementation and development

## **4.1 IP Addressing Scheme**

There are 2 types of IP addresses that needs to be considered while creating the networks (see figure 11).

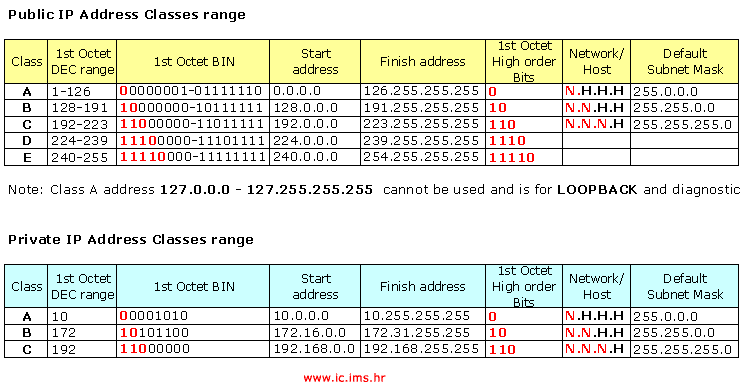


Figure 11. Public and Private IP Addresses, 2018.

Public IP addresses are used only between WAN networks when we want to connect two or more LAN offices. In the topology there are only 2 public IP addresses from the subnet 132.0.0.0/30. 255.255.255.252 subnet mask is used between point to point links to avoid exhaustion e.g. If 132.0.0.0/24 had been used, we would have lost a wide range of IP addresses that could be used on other interfaces.

## **4.2 VMware Workstation**

Two independently virtual machines with Windows installed are nested inside Hypervisor. (see figure 12).

Figure 12 – Screenshot of VMware Workstation

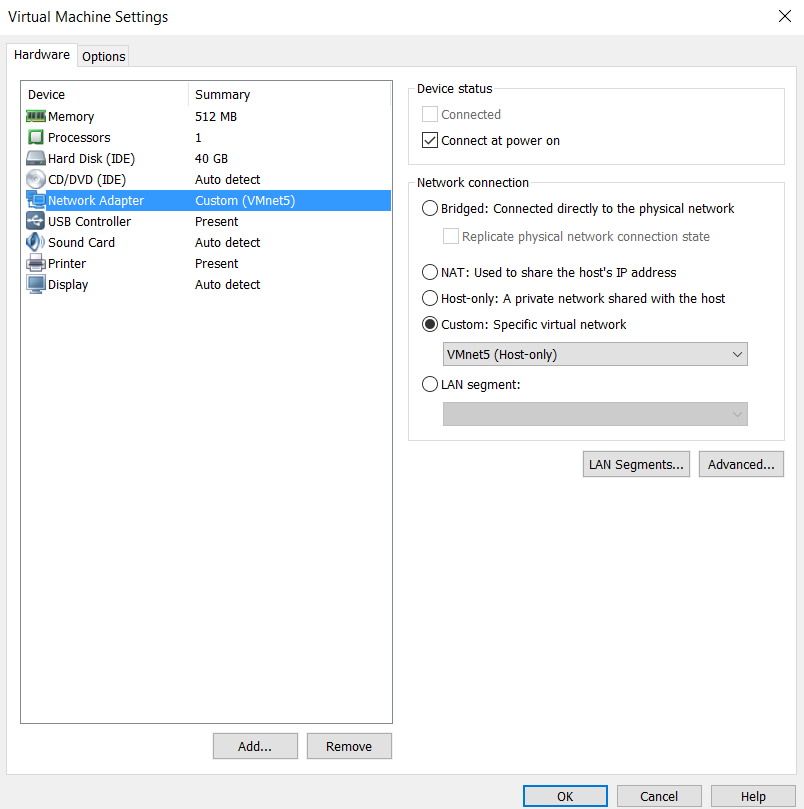
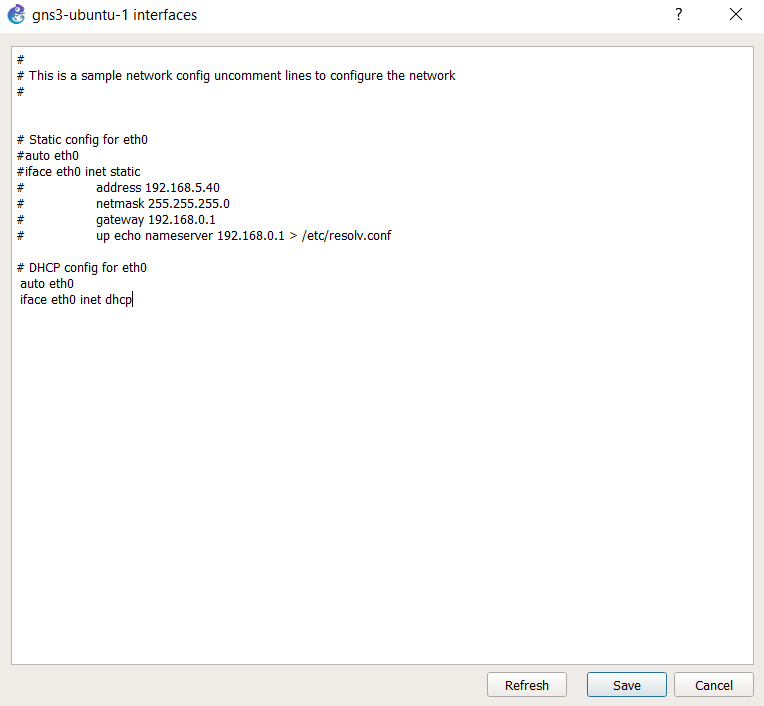
Two separates VLANs are running alongside GNS3 VM which is used only for emulating VIRL images. To assign appropriate subnets range we must go to edit 🡪 Virtual Network Editor and from there choose the VMnet (virtual interface) that we want to use, specify the subnet and untick DHCP, because the router from GNS3 will take care of assigning IP addresses. Inside windows operating system DHCP is enabled by default so we don’t need to change anything. To assign appropriate interface to VLANs shutdown the VLAN instance and go to settings and under network adapter choose the VMnet that has been previously configured (see figure 13). Since we do not need either Internet connectivity or connection with the physical device (laptop) and everything takes place inside between hypervisors. We should choose Host-only option.

Figure 13. Virtual Machine Settings.

## **4.3 Docker container**

On the left side of the topology (see figure 7) there is a docker container with Ubuntu installed. To download Python, we need to connect it first to the NAT cloud which gives Internet connectivity without advertising the topology to the Internet. It has DHCP enabled so the host needs to run as DHCP client (see figure 14).

The IP address in the range 192.168.122.0/24 will be assigned to the Ubuntu guest.

Script is created in the terminal by issuing a command: “nano NameOfTheScript”. Nano is a Linux editor. We can see all files using the command “ls”. To run the file “python” keyword needs to be used before the name of the script.

To update and install the latest packages/libraries we should use the following command in the terminal window:

Figure 14. Ubuntu docker configuration file.

apt-get update && apt-get upgrade

Now we can install libraries that we are going to use together with Python to execute the scripts:

apt-get install python -y

apt-get install build-essential libssl-dev libffi-dev -y

apt-get install python-pip -y

pip install cryptography pip install paramiko

Two libraries had been considered while doing this project: Telnetlib and Paramiko. Since Telnetlib uses Telnet protocol which is unsecure, because the data is sent in plain text, the second library was chosen.

## **4.4 Text editor**

Every programmer has his favourite text editor to write the actual code for a certain

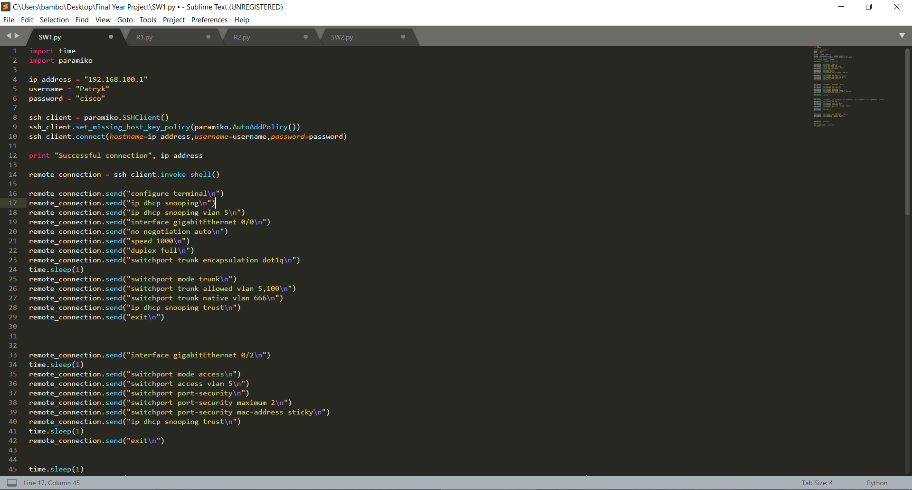
project. I have decided to use Sublime Text 3 (see figure 15).

Figure 15. Screenshot of Sublime Text 3.

It is well known for Python programmers. Known for its speed and ease of use. The hidden power of this editor doesn’t come by default. We can compare to google chrome because it allows to install plugins and edit many things. This makes writing the code so much easier due to the flexibility of the options we have.

## **4.5 Start-up configurations**

Each Cisco device needs to have start-up configuration to connect to it via SSH to perform further configuration. If we don’t have it, we would have to configure each device separately using crossover cable (serial connection) connected to the device which is being configured. This project is focused on network programmability hence, we need to have basic configuration. Start-up configuration between the

router and the switch on each side will be different due to the fact that we need to implement router on the stick concept. It finds its use when routing between two (or more) different VLANs is needed. In this type of configuration, a single physical interface on the router manages traffic between multiple VLANs on the network using 802.1Q (often called dot1q). It is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. It is based on tagging (see figure 16). The code for this configuration can be found in Appendix 1.

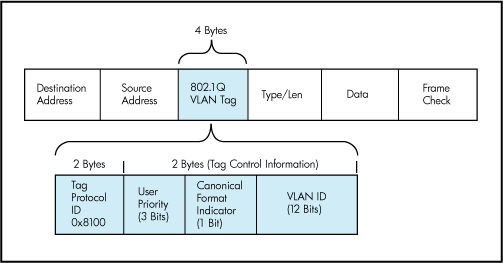


Figure 16. IEEE 802.1Q Tag, 2018.

## **4.6 Remote configurations**

To connect remotely using SSH we must have IP address, hostname, domain-name, enable vty lines and generate rsa keys. The scripts will be saved inside the docker container using simple text editor called nano. It is well known from the systems that use command line interface (CLI). To open the editor from the CLI simply type nano and the name of the script. To save it use the combinations of the keys: ctr + x and then to save confirm by the enter. To run the script the “python” keyword needs to be put before the name of the script. In the remote configuration I have added print statements after every section of the code. Using them I could see where the problem was when I had an error. Many problems occurred due to the limitation of the CPU. I had to add delays between some of the statements, so I have spent quite a lot of time on debugging the code.

## **4.7 Problems and errors**

There is countless number of problems when it comes to creating networks in GNS3 even if it is the best network emulator available on the market. First, the emulator has many errors and it is very difficult to set it up. Since it is virtual environment sometimes we might get different results than we would get in the real world. Errors, delays occur very often. In my project I will try to avoid most of them by using the latest releases of GNS3 and virtual IOS images dedicated for GNS3. They are less prone to errors than the other versions of images available. During development of this project I have encountered many problems. Starting from setting up the application GNS3 through common networking errors and ending up on problem with the drivers that are needed for communication between virtual hosts and GNS3 environment. The first problem was linked to docker container. I could not drag and drop the container to the topology from the list of available devices (see figure 17).

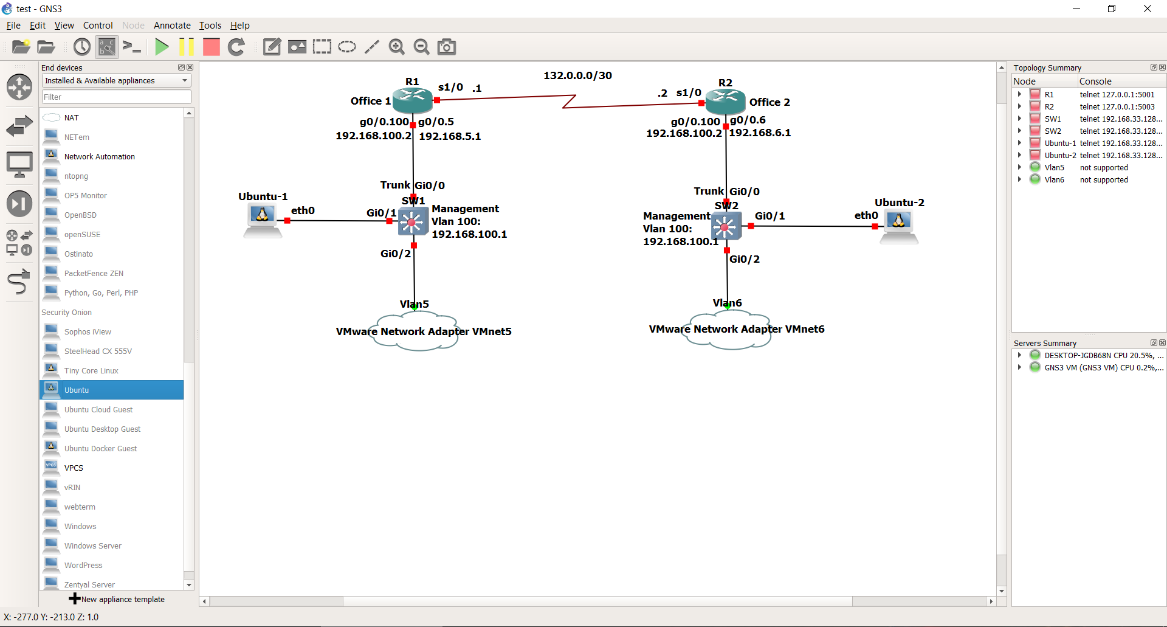


Figure 17. Screenshot of available network devices in GNS3

I was getting the following error:

Docker has returned an error: 500 Get <https://registry-1.docker.io/v2/>: net/http: request cancelled while waiting for connection (Client.Timeout exceeded while awaiting headers).

Since it is an issue with the software I posted a discussion on GNS3 website:

<https://www.gns3.com/qa/windows-ubuntu-docker>

I got a lot of help but still I could not solve this problem. After many attempts I have decided to fully format my laptop and reinstall the Windows. After the clean installation a new error appeared. Now I was able to drag and drop the docker, but I could not connect both Clouds (VMnet5 and VMnet6) to the switches. I was getting the following error:

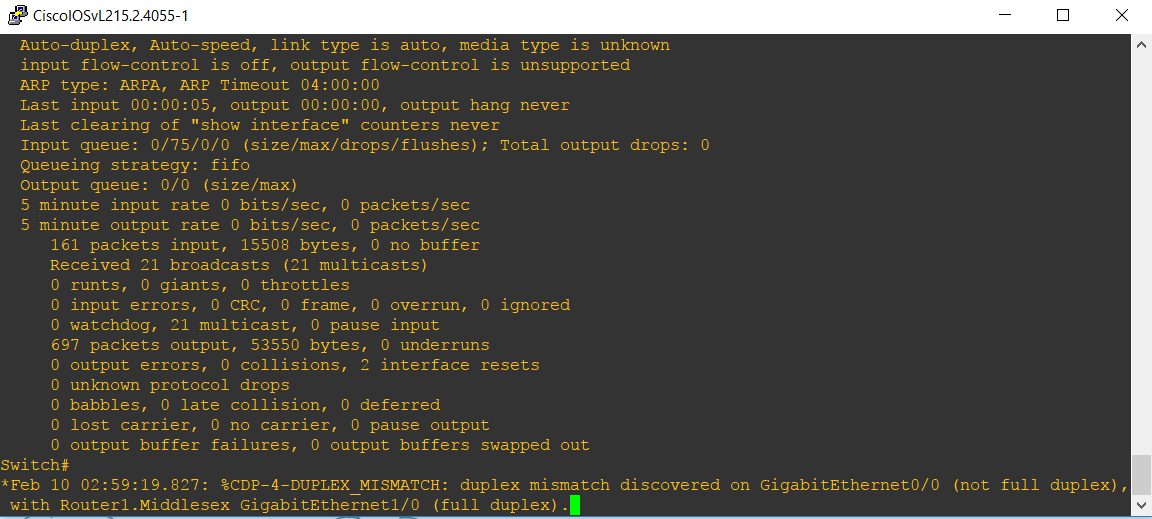
Server error from http://127.0.0.1:8000: vios\_l2-adventerprisek9-m.vmdk.SSA.152-4.0.55.E: unable to create NIO Ethernet for bridge 'ethernet0.vnet'

The discussion about this error can be found here:

https://github.com/GNS3/gns3-gui/issues/840

I started cmd.exe as an administrator and typed sc config npf start= auto and rebooted the laptop per instructions posted. That did not help me. I knew the problem was with Winpcap. It is an industry-standard tool for link-layer network access in Windows environments. Simply we need those drivers to send the packets across our virtual hosts. I was installing and uninstalling different versions of it. That did the job because the old version: “WinPcap\_4\_1\_3” finally worked for me.

After that, I got the real issue which is very common in enterprise networks. Since I use new images (VIRL) together with old ones (Dynamips), the second problem was duplex mismatch (see figure 18).

Figure 18. Screenshot of duplex mismatch error between router and the switch.

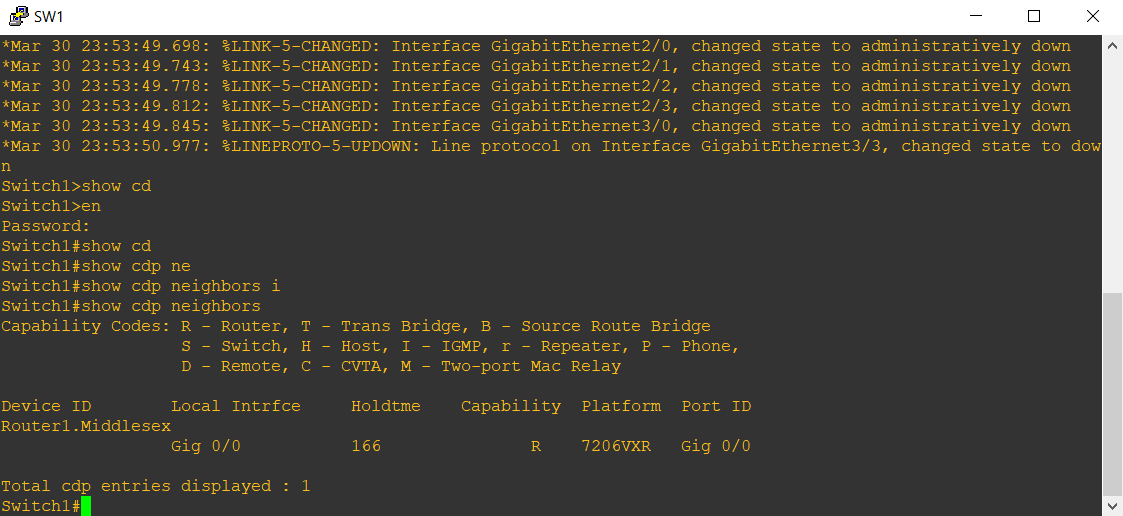
CDP stands for Cisco Discovery Protocol. Since it is Layer 2 protocol it can work using MAC addresses. It provides information about adjacent devices only between cisco devices. It is enabled by default on the most of cisco devices. It uses multicast address of: 01-00-0c-cc-cc-cc to which all the cisco devices belong. On Switch 1, from enabled mode we can issue a command: “show cdp neighbors” (see figure 19).

Figure 19. Screenshot of CDP neighbours issued from Switch 1.

The output refers only to directly connected devices and gives the information such as:

* domain name
* platform
* kind of device
* platform

There is a second command: “show cdp neighbors detail” which gives much more information about adjacent devices (see figure 20).

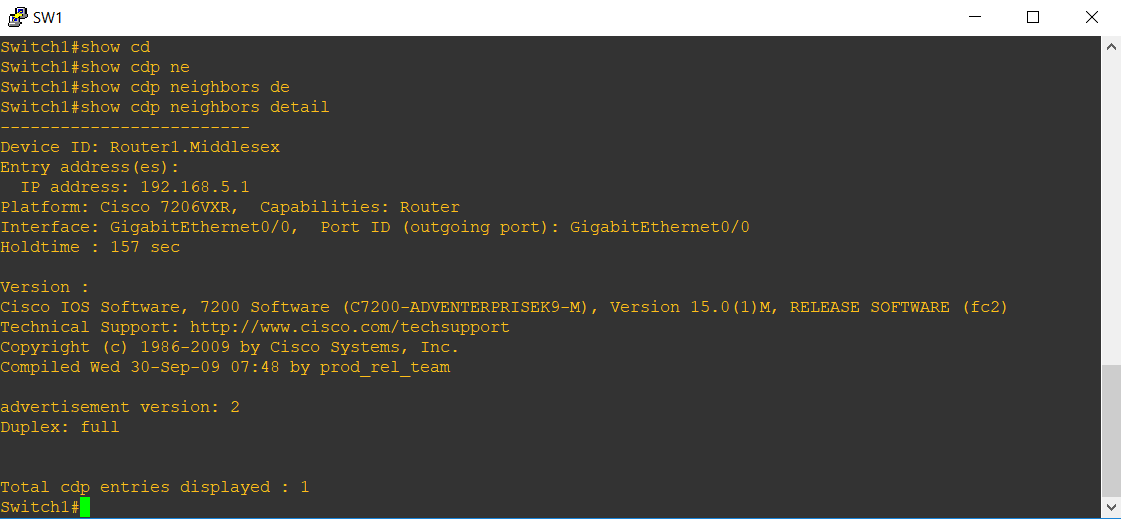


Figure 20. Screenshot of CDP neighbours detail command issued from Switch 1.

This command gives IP address of the directly connected device which is unsecure. Using this feature, an attacker can map the network, so we should remember to disable this option on untrusted interfaces. CDP can be disabled in global configuration mode by issuing the command: “no cdp run” or under interface configuration using: “no cdp enable” command.

I struggled with DHCP Snooping feature because from theory it should be applied only on the interface that goes to the router which provides DHCP server. After reading threads on cisco.com I applied the command to both interfaces and then it worked. I could not find the answer why, but it also happens in the real world.

# Chapter 5. Evaluation

The aims of this project have been achieved. Topology has been created and configured in GNS3 emulator. Script establishes a VoIP session between 2 windows hosts so they can call each other and everything is equipped with security features.. However, in the real environment everything would be run from the single script for a Local Area Network. Since network devices are faster than virtual environment, hence why there are so many delays in the code written in Python. This is because routers and switches are aimed for high speed. Those devices are very expensive, and it is unreachable to get the same results running everything in the emulator on the laptop which is worth less than one thousand pounds.

The project focuses on security, voice over internet protocol and network automation however further upgrades can be implemented to the project but that would require a better PC. Further improvements and functionality to this project could include:

- creating more VLANs, subnets, hosts.

- configurations for multiple hosts from the script using loops, nested loops.

- adding more network devices such as L2 (Switches) and L3 (Routers and Multi-Layer Switches) devices.

-adding more security devices such as ASAv (Adaptive Security Appliance Virtual), IDS (Intrusion Detection System), IPS (Intrusion Prevention System)

We can automate every set of commands that need to be implemented on more than one device so the list of further improvements is infinite. For instance, we would like to automate creating VLANs 2 to 10 and assigning them appropriate names. This can be achieved of 3 lines of codes. In comparison in the CLI we would have use commands VLAN, name and exit in total 24 times which is not time efficient. (see figure 21).

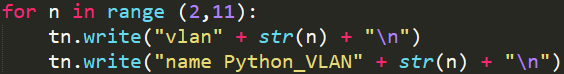


Figure 21. Creating 2-10 VLANs on a switch using Python and paramiko library

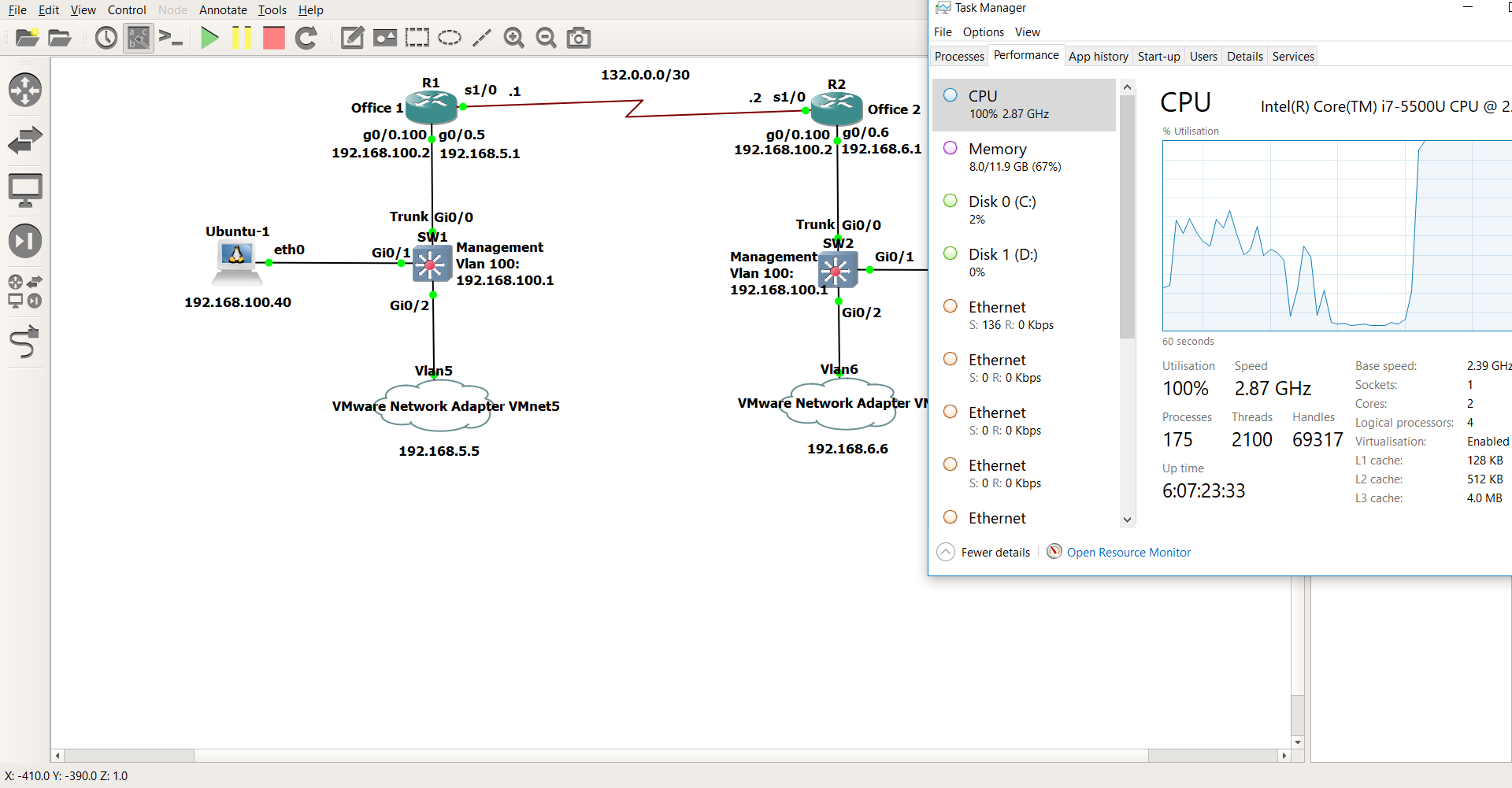
 Even though not all the concepts of SDN are available in GNS3 we can still add a few more such as OpenFlow Protocol and OpenFlow Manager. This was not added to the project due to the limitation of resources such as CPU and RAM. GNS3 is a great emulation tool and every network can be built inside of it. That is why big companies test the code within virtual environment first, but they need high performance PC’s for it. In my project only, a few network devices have been used and my CPU reached 100% of utilisation (see figure 22).

Figure 22. Screenshot of CPU utilisation

Nevertheless, the project can be improved further if physical equipment is added. This is possible, because GNS3 supports a big range of network devices. Script and network topology can be expanded within enterprise network for testing purposes which can reduce number of errors or delays while implementing new configurations for the existing networks. This way of testing new solutions is very useful, because if network engineer has any error in the code he will first see it in the virtual environment which is not a problem because virtual networks are not connected to the company’s networks.

# Chapter 6. Conclusion

This project raises a subject of Software Defined Networking. It focuses on network automation which is one of the concepts of SDN technology. It shows its use which will have a big impact in the future networks because it is time efficient and previously used code shouldn’t have any errors. We can compare it to home automation in Internet of Things (IoT) technology. Both network automation and home automation have been developed to make people’s lives easier. In second case, programmer’s life. Therefore, almost every network engineer has already started his journey with Python or has it in his plans for 2018 like me. I can admit that I have learnt a lot while doing this project. Especially my troubleshooting skills moved to the next level.

All objectives of this project have been achieved. Two hosts can call each other using Cisco IP Communicators running from Windows which are run inside Hypervisor. They are connected to network devices in GNS3 emulator where all the configurations occur. Finally, network programmers have less work using the script that has been written during this project. If I do this project again I would try to use real equipment together with GNS3 emulator running on a better high-performance PC to achieve the best results.

# References

[1]["The Pulsion Telephone"](http://paperspast.natlib.govt.nz/cgi-bin/paperspast?a=d&d=HBH18900130.2.17&e=-------10--1----2--), New Zealand: Hawke's Bay Herald, Vol. XXV, Iss. 8583, January 30, 1890, p. 3.

[2] Cioara, J. and Valentine, M. (2012). CCNA voice 640-461. Indianapolis, Ind.: Cisco.

[3] Flood, J. (1987). Alexander Graham Bell and the invention of the telephone. [online] pp.1387-1388. Available at: https://ieeexplore-ieee-org.ezproxy.mdx.ac.uk/stamp/stamp.jsp?tp=&arnumber=5252617&tag=1 [Accessed 18 Nov. 2017].

[4] Gupta, A. and Tong, H. (1984). The First Decade of Personal Computers. [online] 72(3), pp.246-250. Available at: https://ieeexplore-ieee-org.ezproxy.mdx.ac.uk/stamp/stamp.jsp?tp=&arnumber=1457125&tag=1 [Accessed 25 Nov. 2017].

[5] Python.org. (2018). Welcome to Python.org. [online] Available at: [6] https://www.python.org/ [Accessed 10 Apr. 2018].

[7] Mininet.org. (2018). Mininet Overview - Mininet. [online] Available at: http://mininet.org/overview/ [Accessed 10 Apr. 2018].

[8] Kirstein, P. (1999). Early experiences with the Arpanet and Internet in the United Kingdom. IEEE Annals of the History of Computing, [online] 21(1), pp.38-41. Available at: https://ieeexplore.ieee.org/document/759368/ [Accessed 31 Nov. 2017].

[9] Arroyo, L. (2010). THE FIRST WORLDWIDE PUBLIC PACKET SWITCHING NETWOR. [online] pp.1-3. Available at: https://ieeexplore-ieee-org.ezproxy.mdx.ac.uk/stamp/stamp.jsp?tp=&arnumber=5735296 [Accessed 4 Dec. 2017].

[10] Minoli, D. and Minoli, E. (2002). Delivering voice over IP networks. New York: Wiley.

[11] Johnston, A. (2005). SIP. 2nd ed. Norwood Mass.: Books24x7.com.

[12] Gns3.com. (2018). GNS3 | The software that empowers network professionals. [online] Available at: https://www.gns3.com/ [Accessed 19 Dec. 2017].

[13] Docker. (2018). Docker. [online] Available at: https://www.docker.com/ [Accessed 19 Dec. 2017].

[14] Barrett, D. and Silverman, R. (2001). SSH, the secure shell. Beijing: O'Reilly.

[15] Lammle, T. (2011). Ccna. Indianapolis, Ind.: Sybex.

[16] Containers vs. VMs. (2018). [image] Available at: https://www.sdxcentral.com/cloud/containers/definitions/containers-vs-vms/ [Accessed 27 Feb. 2018].

[17] An Introduction to Software Defined Networking (SDN). (2017). [DVD] Directed by T. Slattery. Live Lessons.

[18] CCNA Security 210-260. (2016). [DVD] Directed by O. Santos, A. Woland and M. Harris. Live Lessons.

[19] Packet Tracer Application. (2018). [image] Available at: https://ccnav6.com/7-1-1-4-packet-tracer-acl-demonstration-instructions-answers.html [Accessed 27 Feb. 2018].

[20] Layer 2 Switch. (2018). [image] Available at: https://www.elive.co.nz/cisco-catalyst-48port-poe-manageable-switch-wsc2960s48lpdl.php [Accessed 20 Mar. 2018].

[21] Cisco (2018). Cisco ISR Routers. [image] Available at: http://ciscorouterswitch.over-blog.com/article-cisco-isr-g2-the-best-a-branch-can-get-any-service-any-branch-anywhere-100881958.html [Accessed 21 Mar. 2018].

[22] Cisco IP Phone. (2018). [image] Available at: https://superuser.com/questions/963952/whats-the-minimum-hardware-for-cisco-ip-phones [Accessed 25 Mar. 2018].

[23] Public and Private IP Addresses. (2018). [image] Available at: https://vlsm-subnetting.blogspot.co.uk/2014/09/public-and-private-ip-address-classes.html [Accessed 26 Mar. 2018].

[24] IEEE 802.1Q Tag. (2018). [image] Available at: http://www.ebrahma.com/2013/05/difference-between-isl-802-1q/ [Accessed 27 Mar. 2018].

[25] CCNA Security Training Videos (640-554). (n.d.). [DVD] Directed by H. Frisbee. Udemy.

# Appendix A – Start-up configurations with explanation

Switch1:

en

conf t

interface gigabitEthernet 0/1

switchport mode access

switchport access vlan 100

exit

interface vlan 100

ip address 192.168.100.1 255.255.255.0

no shutdown

exit

hostname Switch1

ip domain-name Middlesex

username Patryk secret cisco

enable secret cisco

username Patryk privilege 15

line vty 0 4

transport input ssh

login local

exit

crypto key generate rsa

1024

exit

In lines 1 to 6 interface gigabit ethernet 0/1 which goes to the docker container has been configured as an access port and assigned to management vlan 100.

In lines 7 to 10 interface vlan 100 has been created and ip address of 192.168.100.1 with the subnet mask of 255.255.255.0 has been assigned. IP address allows the ubuntu container to connect via SSH.

In lines 11 to 22 all the steps have been made to allow SSH connectivity using local username and password.

Router1:

conf t

interface gigabitEthernet 0/0

no shutdown

interface gigabitEthernet 0/0.100

encapsulation dot1q 100

ip address 192.168.100.2 255.255.255.0

no shutdown

interface gigabitEthernet 0/0.5

encapsulation dot1q 5

ip address 192.168.5.1 255.255.255.0

no shutdown

exit

hostname Router1

ip domain-name Middlesex

username Patryk secret cisco

enable secret cisco

username Patryk privilege 15

line vty 0 4

transport input ssh

login local

exit

crypto key generate rsa

1024

exit

In lines 1 to 12 concept of a router on a stick has been configured. In line 9 encapsulation dot1q has been applied, because the trunk which is configured on the switch side uses it.

Lines 13 to 24 are equal to the steps for the Switch to allow SSH connectivity.

Switch2:

en

conf t

interface gigabitEthernet 0/1

switchport mode access

switchport access vlan 100

exit

interface vlan 100

ip address 192.168.100.1 255.255.255.0

no shutdown

exit

hostname Switch2

ip domain-name Middlesex

username Dominika secret cisco

enable secret cisco

username Dominika privilege 15

line vty 0 4

transport input ssh

login local

exit

crypto key generate rsa

1024

exit

Router2:

conf t

interface gigabitEthernet 0/0

no shutdown

interface gigabitEthernet 0/0.100

encapsulation dot1q 100

ip address 192.168.100.2 255.255.255.0

no shutdown

interface gigabitEthernet 0/0.6

encapsulation dot1q 6

ip address 192.168.6.1 255.255.255.0

no shutdown

exit

hostname Router2

ip domain-name Middlesex

username Dominika secret cisco

enable secret cisco

username Dominika privilege 15

line vty 0 4

transport input ssh

login local

exit

crypto key generate rsa

1024

exit

# Appendix B – Remote configuration with explanation

Switch1:

import time

import paramiko

ip\_address = "192.168.100.1"

username = "Patryk"

password = "cisco"

ssh\_client = paramiko.SSHClient()

ssh\_client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh\_client.connect(hostname=ip\_address,username=username,password=password)

print "Successful connection", ip\_address

remote\_connection = ssh\_client.invoke\_shell()

remote\_connection.send("configure terminal\n")

remote\_connection.send("ip dhcp snooping\n")

remote\_connection.send("ip dhcp snooping vlan 5\n")

remote\_connection.send("interface gigabitEthernet 0/0\n")

remote\_connection.send("no negotiation auto\n")

remote\_connection.send("speed 1000\n")

remote\_connection.send("duplex full\n")

remote\_connection.send("switchport trunk encapsulation dot1q\n")

time.sleep(1)

remote\_connection.send("switchport mode trunk\n")

remote\_connection.send("switchport trunk allowed vlan 5,100\n")

remote\_connection.send("switchport trunk native vlan 666\n")

remote\_connection.send("ip dhcp snooping trust\n")

remote\_connection.send("exit\n")

remote\_connection.send("interface gigabitEthernet 0/2\n")

time.sleep(1)

remote\_connection.send("switchport mode access\n")

remote\_connection.send("switchport access vlan 5\n")

remote\_connection.send("switchport port-security\n")

remote\_connection.send("switchport port-security maximum 2\n")

remote\_connection.send("switchport port-security mac-address sticky\n")

remote\_connection.send("ip dhcp snooping trust\n")

time.sleep(1)

remote\_connection.send("exit\n")

time.sleep(1)

remote\_connection.send("interface range gigabitEthernet 0/3, gigabitEthernet 1/0-3, gigabitEthernet 2/0-3, gigabitEthernet 3/0-3\n")

remote\_connection.send("switchport mode access\n")

time.sleep(1)

remote\_connection.send("switchport access vlan 999\n")

remote\_connection.send("switchport port-security\n")

remote\_connection.send("switchport port-security mac-address sticky\n")

remote\_connection.send("shutdown\n")

time.sleep(1)

remote\_connection.send("exit\n")

time.sleep(1)

remote\_connection.send("interface range gigabitEthernet 0/1-2\n")

remote\_connection.send("spanning-tree portfast edge\n")

remote\_connection.send("spanning-tree bpduguard enable\n")

remote\_connection.send("end\n")

time.sleep(1)

output = remote\_connection.recv(65535)

print output

In the first two lines libraries time and paramiko have been added to the project. First one is used only to add some delays between commands due to the delay caused by virtualization or CPU limitation.

In lines 3 to 5 three variables have been defined that are used later in the code.

Line 6 states that paramiko client (not the server) will be used for SSH connection.

Line 7 has been added to reject unknown SSH keys and accept public keys from the switch in this example. This is not recommended in the real environment.

In line 8 a connection takes place and all the variables that have been previously defined are used for SSH connection.

Line 9 has been added only to check if the connection has been established.

In line 10 shell needs to be invoked, because shell commands will be sent remotely.

In lines 11 to 13 ip dhcp snooping feature has been enabled.

In lines 14 to 17 configuration to eliminate duplex mismatch have been applied.

In lines 20 to 24 security features have been applied.

In lines 25 to 35 more features have been applied such as maximum number of allowed MAC addresses from one IP address to increase security.

In lines 36 to 40 all the unused ports have been assigned not default vlan and place in a shutdown state to again increase the security.

In lines 46 to 48 portfast and bpdguard features have been enabled which dealt with faster convergence time in spanning tree protocol. The rest of the lines simply end the connection with the remote host.

Router1:

import time

import paramiko

ip\_address = "192.168.100.2"

username = "Patryk"

password = "cisco"

ssh\_client = paramiko.SSHClient()

ssh\_client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh\_client.connect(hostname=ip\_address,username=username,password=password)

print "Successful connection", ip\_address

remote\_connection = ssh\_client.invoke\_shell()

remote\_connection.send("configure terminal\n")

remote\_connection.send("ip access-list extended VOIP\n")

remote\_connection.send("permit ip host 192.168.5.5 host 192.168.6.6\n")

remote\_connection.send("exit\n")

remote\_connection.send("username R2 password cisco\n")

remote\_connection.send("ip route 0.0.0.0 0.0.0.0 132.0.0.2\n")

remote\_connection.send("interface serial 1/0\n")

remote\_connection.send("encapsulation ppp\n")

remote\_connection.send("ppp authentication chap\n")

remote\_connection.send("ppp chap hostname R1\n")

remote\_connection.send("ip address 132.0.0.1 255.255.255.0\n")

remote\_connection.send("no shutdown\n")

remote\_connection.send("exit\n")

print "Interface Serial 1/0 has been configured"

remote\_connection.send("ip dhcp excluded-address 192.168.5.1\n")

remote\_connection.send("ip dhcp pool Vlan5\n")

remote\_connection.send("network 192.168.5.0 255.255.255.0\n")

remote\_connection.send("domain-name Patryk.com\n")

remote\_connection.send("default-router 192.168.5.1\n")

remote\_connection.send("lease infinite\n")

print "DHCP has been configured"

remote\_connection.send("dial-peer voice 1 voip\n")

time.sleep(1)

remote\_connection.send("destination-pattern 666.\n")

remote\_connection.send("session target ipv4:132.0.0.2\n")

remote\_connection.send("exit\n")

remote\_connection.send("telephony-service\n")

remote\_connection.send("max-ephones 1\n")

remote\_connection.send("max-dn 1\n")

remote\_connection.send("ip source-address 192.168.5.1 port 2000\n")

remote\_connection.send("auto assign 1 to 1\n")

remote\_connection.send("transfer-system full-consult\n")

remote\_connection.send("exit\n")

remote\_connection.send("ephone-dn 1\n")

time.sleep(2)

remote\_connection.send("number 5555\n")

remote\_connection.send("description Patryk\_VLAN5\n")

remote\_connection.send("name Patryk\_Vlan5\n")

remote\_connection.send("label Patryk\_Vlan5\n")

remote\_connection.send("exit\n")

remote\_connection.send("ephone 1\n")

remote\_connection.send("keepalive 10\n")

remote\_connection.send("exit\n")

print "VoIP has been configured"

remote\_connection.send("end\n")

time.sleep(1)

output = remote\_connection.recv(65535)

print output

Lines 1 to 10 are the same as for the Switch to connect and establish an SSH session between ubuntu docker and the network device.

Lines 12 and 13 creates an extended access list to permit the host 192.168.6.6 to access the network. Every ACL at the end has implicit deny statement so every other connection will be dropped.

In line 16 the default route has been specified which means if the route doesn’t know the route to the specific network he needs to pass it to the router router R2.

Lines 17 to 23 are responsible for authentication process and assigning IP address for the WAN interface.

Lines 25 to 30 are creating DHCP server on a router to allocate addresses to the hosts.

For the project I have setup lease time to infinite. By default, it is normally 7 days.

From line 32 up to the end VoIP has been configured.

Switch2:

import time

import paramiko

ip\_address = "192.168.100.1"

username = "Dominika"

password = "cisco"

ssh\_client = paramiko.SSHClient()

ssh\_client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh\_client.connect(hostname=ip\_address,username=username,password=password)

print "Successful connection", ip\_address

remote\_connection = ssh\_client.invoke\_shell()

remote\_connection.send("configure terminal\n")

remote\_connection.send("ip dhcp snooping\n")

remote\_connection.send("ip dhcp snooping vlan 6\n")

remote\_connection.send("interface gigabitEthernet 0/0\n")

remote\_connection.send("no negotiation auto\n")

remote\_connection.send("speed 1000\n")

remote\_connection.send("duplex full\n")

remote\_connection.send("switchport trunk encapsulation dot1q\n")

time.sleep(1)

remote\_connection.send("switchport mode trunk\n")

remote\_connection.send("switchport trunk allowed vlan 6,100\n")

remote\_connection.send("switchport trunk native vlan 666\n")

remote\_connection.send("ip dhcp snooping trust\n")

remote\_connection.send("exit\n")

remote\_connection.send("interface gigabitEthernet 0/2\n")

time.sleep(1)

remote\_connection.send("switchport mode access\n")

remote\_connection.send("switchport access vlan 6\n")

remote\_connection.send("switchport port-security\n")

remote\_connection.send("switchport port-security maximum 2\n")

remote\_connection.send("switchport port-security mac-address sticky\n")

remote\_connection.send("ip dhcp snooping trust\n")

time.sleep(1)

remote\_connection.send("exit\n")

time.sleep(1)

remote\_connection.send("interface range gigabitEthernet 0/3, gigabitEthernet 1/0-3, gigabitEthernet 2/0-3, gigabitEthernet 3/0-3\n")

remote\_connection.send("switchport mode access\n")

time.sleep(1)

remote\_connection.send("switchport access vlan 999\n")

remote\_connection.send("switchport port-security\n")

remote\_connection.send("switchport port-security mac-address sticky\n")

remote\_connection.send("shutdown\n")

time.sleep(1)

remote\_connection.send("exit\n")

time.sleep(1)

remote\_connection.send("interface range gigabitEthernet 0/1-2\n")

remote\_connection.send("spanning-tree portfast edge\n")

remote\_connection.send("spanning-tree bpduguard enable\n")

remote\_connection.send("end\n")

time.sleep(1)

output = remote\_connection.recv(65535)

print output

Router2:

import time

import paramiko

ip\_address = "192.168.100.2"

username = "Dominika"

password = "cisco"

ssh\_client = paramiko.SSHClient()

ssh\_client.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh\_client.connect(hostname=ip\_address,username=username,password=password)

print "Successful connection", ip\_address

remote\_connection = ssh\_client.invoke\_shell()

remote\_connection.send("configure terminal\n")

remote\_connection.send("ip access-list extended VOIP\n")

remote\_connection.send("permit ip host 192.168.6.6 host 192.168.5.5\n")

remote\_connection.send("exit\n")

remote\_connection.send("username R1 password cisco\n")

remote\_connection.send("ip route 0.0.0.0 0.0.0.0 132.0.0.1\n")

remote\_connection.send("interface serial 1/0\n")

remote\_connection.send("encapsulation ppp\n")

remote\_connection.send("ppp authentication chap\n")

remote\_connection.send("ppp chap hostname R2\n")

remote\_connection.send("ip address 132.0.0.2 255.255.255.0\n")

remote\_connection.send("no shutdown\n")

remote\_connection.send("exit\n")

print "Interface Serial 1/0 has been configured"

remote\_connection.send("ip dhcp excluded-address 192.168.6.1\n")

remote\_connection.send("ip dhcp pool Vlan6\n")

remote\_connection.send("network 192.168.6.0 255.255.255.0\n")

remote\_connection.send("domain-name Dominika.com\n")

remote\_connection.send("default-router 192.168.6.1\n")

remote\_connection.send("lease infinite\n")

print "DHCP has been configured"

remote\_connection.send("dial-peer voice 1 voip\n")

time.sleep(1)

remote\_connection.send("destination-pattern 555.\n")

remote\_connection.send("session target ipv4:132.0.0.1\n")

remote\_connection.send("exit\n")

remote\_connection.send("telephony-service\n")

remote\_connection.send("max-ephones 1\n")

remote\_connection.send("max-dn 1\n")

remote\_connection.send("ip source-address 192.168.6.1 port 2000\n")

remote\_connection.send("auto assign 1 to 1\n")

remote\_connection.send("transfer-system full-consult\n")

remote\_connection.send("exit\n")

remote\_connection.send("ephone-dn 1\n")

time.sleep(2)

remote\_connection.send("number 6666\n")

remote\_connection.send("description Dominika\_VLAN6\n")

remote\_connection.send("name Dominika\_Vlan6\n")

remote\_connection.send("label Dominika\_Vlan6\n")

remote\_connection.send("exit\n")

remote\_connection.send("ephone 1\n")

remote\_connection.send("keepalive 10\n")

remote\_connection.send("exit\n")

print "VoIP has been configured"

remote\_connection.send("end\n")

time.sleep(1)

output = remote\_connection.recv(65535)

print output