IT Technology

Assignment 33 – PARAMIKO

Lillebaelt Academy of
Professional Higher Education

Authors:
Milan Kristof Vince: mila1025@edu.eal.dk

Dilshad Khalaf: dils0008@edu.eal.dk

Michal Skórczewski: mich74p0@edu.eal.dk

Kevin Herhold Larsen:kevi2901@edu.eal.dk

2017-04-19

Table of Contents

Introduction

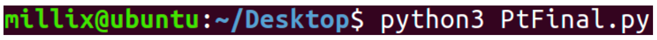
This report is about a paramiko exercise that we in our group decided to work on. This report will show the main goal of the exercise and that is to show, how one can connect and login to a SRX router and save the configuration file to a local device.
By using Python libraries we have managed to create a simple GUI application to the program. Within the program, we have used Paramiko which is the implementation of SSH protocol. This is used for both providing client and server functionality. Besides describing how one can establish connection from the client to the router, we have also added some diagrams in the report to give a better understanding of how the program itself is built in Python. Description comments also added to the code for better understanding.

User Manual

This user manual will show how to run the Python program from a Ubuntu client.
Before you can start, make sure you have SSH installed on your Ubuntu client.

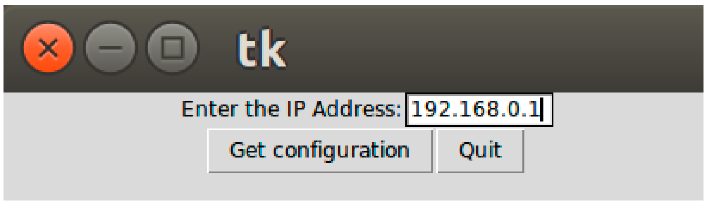
Step 1.
After installing SSH on your Ubuntu client, open terminal, run the Python paramiko program using following command:

"**python3 PtFinal.py**" and hit enter (PtFinal.py is the file name of our program!)

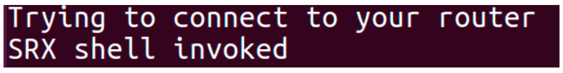


```
millix@ubuntu:~/Desktop$ python3 PtFinal.py
```

Step 2.
When the program is running you will see the following GUI interface.
-Type in the IP-Address of the interface to the router that you are trying to connect.

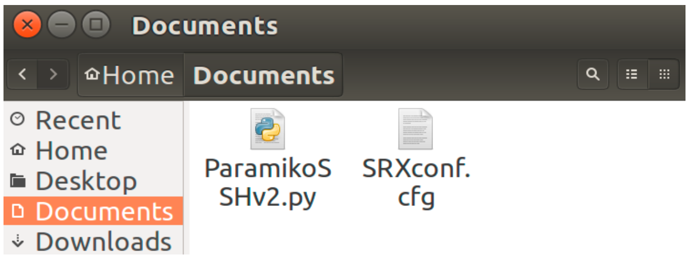


Step 3.
Once you click "Get configuration", you will see this message in your Ubuntu clients terminal. Wait, and the file will be saved on your local device.



```
Trying to connect to your router
SRX shell invoked
```

Step 4.
Check where file is saved on your local device… In the screenshot below, we have successfully managed to save the configuration file of the router.

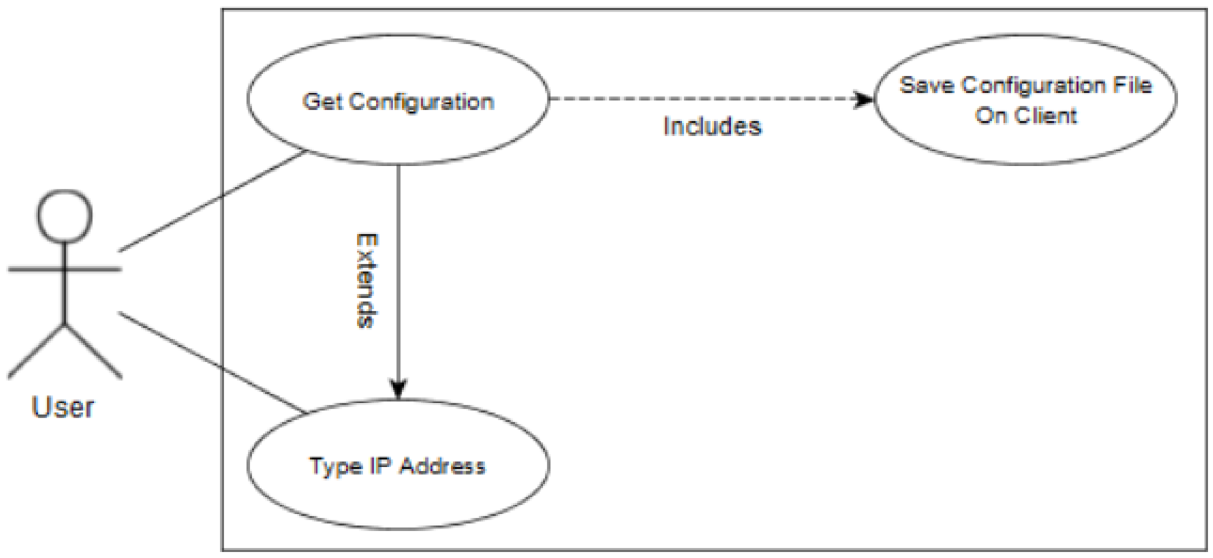


Features

Some of the features in our program:

- Possibility of IP change
- Secure Shell (SHH) establishment
- Fetching configuration files of remote routers

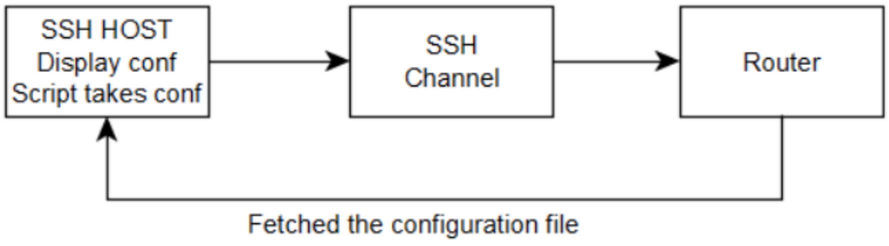
Use-Case Diagram



Explanation:

This use case diagram displays the steps the user must take in our system.
First, the user must specify an IP Address of the router he wants to fetch the configuration file from.
The user must then click the “Get Configuration” button. When this button is clicked, an SSH channel is created behind to scenes to fetch the SRX configuration file. When the file is fetched from the router, it is then saved on our client.

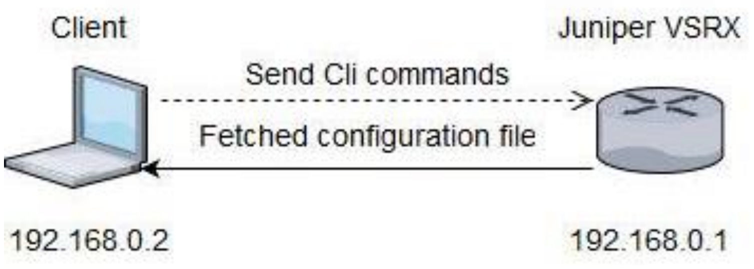
Block Diagram



Explanation:

This block diagram presents how our system works.
First we create an SSH channel via our SSH Host. We then connect our host through the established SSH channel. When this is done, we are then able to send cli commands to the router to fetch the SRX configuration file and save it on our client.

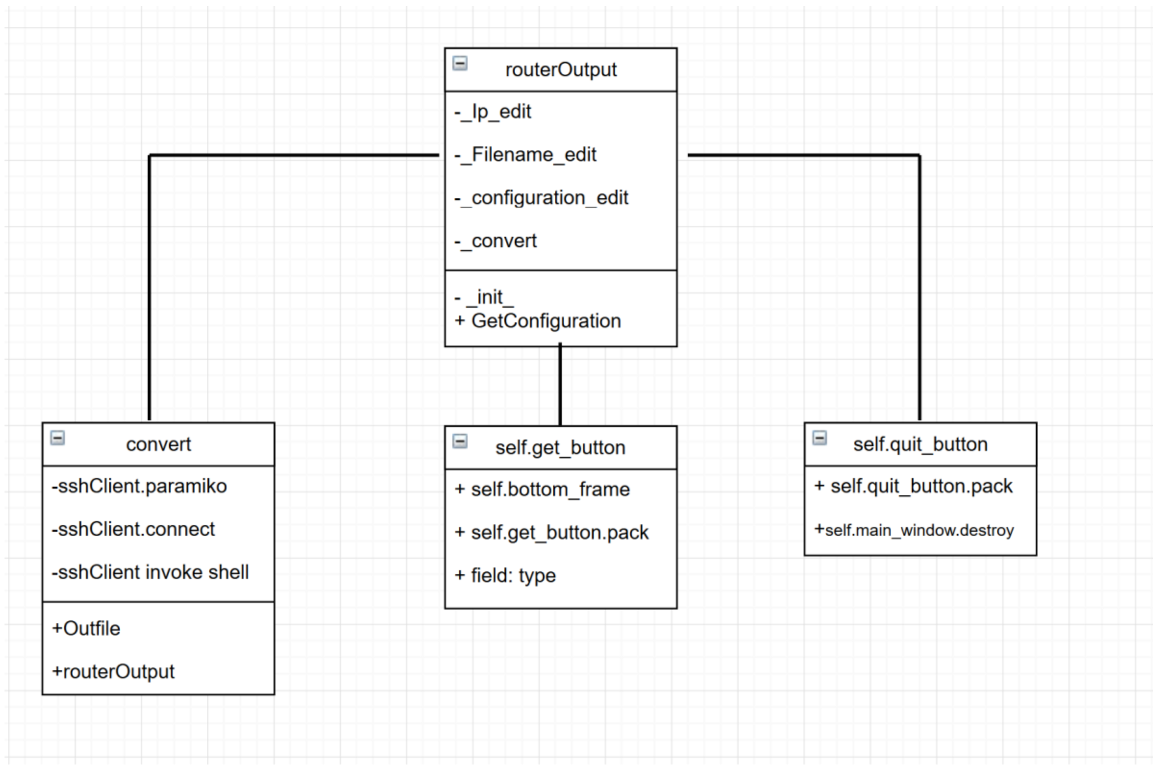
Network Diagram



Explanation:

Layer 3 Network Diagram maps the structure of our network with one client and one router. The user starts the paramiko program from the client which creates an SSH channel to get the configuration file from the router.

Class Diagram



Explanation:

class: routerOutput
The program starts from the class "*routerOutput*" with an __init__ operation method. The class is containing three compartments. There is also one operation, "*GetConfiguration*" which is connected with the "*self.get_button*" class. This function allows the user to fetch and save the configuration file.

Class: convert
Once the user clicks the "*get configuration*" button, the "*Convert"* class creates a SSH connection between the client and the router. The convert class has two operations, Outfile (which is the file name we use to open the configuration) and routerOutput.

Class: self.get_button
Important class where the user can run the program by clicking on the "get" button. While the "self.get_button.pack" packs the button, the "button_frame" creates group widgets in the tkinter.

Class: self.quit_button
The class is used if the user decides to quit from the program by pressing on the "Quit" button. "self.quit_button.pack" pack the "Quit" button, "self_main.destroy" gets activated when "Quit" button is clicked and eliminates the window aswell as closes the program.

Project Management

Conclusion

In relation to the program itself, we have in agreement concluded that this exercise was great because we learned much about not only SSH, but also what Python programming is capable of. Working with this project we have learned to manage our work better for the near future as we were lacking team management.
We have come to agreement to divide the tasks into subtasks, verify each of our individual work as a group, and become better in communication overall.

Here is what we all individually have learned from this exercise.

Dilshad: I have learned to create SSH channels and make connectivity between a client and a router using the Paramiko module in Python. I have learned to send specific commands to the router, and fetch the configuration file. I have become better to create use case diagrams, network diagrams, block diagrams and class diagrams.
In relation to project management, I have learned that we need a lot more communication in the group. We must help each other when things get complicated, and when we divide tasks, we must take the time to verify other team members work.

Milan: I have learned about how to create SSH channels and make connectivity between the router and one client by using paramiko. I got also better understanding about creating use-case, block and network diagrams after we had a chat with the lecturer. It was a good exercise which gave me useful experience.

Michal: From assignment 33 about Paramiko I have learnt a lot of things. At first I improved my python programming skills mainly in SSH scripting and object oriented programming. I also refreshed my memory about diagrams I've learnt on Morten's System Design/Development. Very important was also ability to write nice-looking document.

Kevin: I have improved my diagram skills. I have learned to create and automatize a backup sytem for the SRX,
using python. I have also learned to use Paramiko which is a SSH tool. (which we implemented in our system)

Appendix

Python source code:

```
import tkinter
import tkinter.messagebox
import paramiko, time, os
class routerOutput:
    def __init__(self):

        # Create the main window.
        self.main_window = tkinter.Tk()

        # Create two frames to group widgets.
        self.top_frame = tkinter.Frame()
        self.bottom_frame = tkinter.Frame()

        # Create the widgets for the top frame.
        self.prompt_label = tkinter.Label(self.top_frame, \
                    text='Enter the IP Address:')
        self.conf_entry = tkinter.Entry(self.top_frame, \
                                        width=10)

        # Pack the top frame's widgets.
        self.prompt_label.pack(side='left')
        self.conf_entry.pack(side='left')

        # Create the button widgets for the bottom frame.
        self.get_button = tkinter.Button(self.bottom_frame, \
                                        text='Get configuration', \
                                        command=self.convert)
        self.quit_button = tkinter.Button(self.bottom_frame, \
                                    text='Quit', \
                                    command=self.main_window.destroy)
        # Pack the buttons.
        self.get_button.pack(side='left')
        self.quit_button.pack(side='left')

        # Pack the frames.
        self.top_frame.pack()
        self.bottom_frame.pack()

        # Enter the tkinter main loop.
        tkinter.mainloop()

    def convert(self):
        sshClient = paramiko.SSHClient()

        sshClient.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        print ('Trying to connect to your router')

        sshClient.connect('192.168.0.1 / ge-0/0/0', username='root',
password='test12')
        channel = sshClient.invoke_shell()
        print ('SRX shell invoked')
        time.sleep(2)
        os.system('clear')
        channel.send('cli\n')
        #Wait 1 second for the router to enter CLI
        time.sleep(1)

        #Read router. Output/replies not used
        routerOutput = channel.recv(1000)
        time.sleep(3)

        #Show the whole configuration
        channel.send('show configuration | no-more\n')

        #Wait 1 second for the router to list config
        time.sleep(1)

        output = channel.recv(10000) # Read router configuration output

        #List the configuration in the terminal window
        print (output)
        channel.close()
        #Apparently this has to be wb, or it wont work! wb means write in
binary!
        outFile = open('SRXconf.cfg','wb')

        # Save the configuration in a file
        outFile.write(output)

        #Close the file
        outFile.close()


conf_conv = routerOutput()
```

Some important snippets of the source code

import paramiko ## Imports the Pythons paramiko module[1]. ## Paramiko is a Python 2.6+, 3.3+ implementation of ##SSHv2 protocol that provides ##client and server functionality.
sshClient.connect('192.168.0.1 / ge-0/0/0', username='root', password='test12') ## Specifies IP address, the username and password of the router, and then establishes an ## SSH channel to our SRX router.
channel = sshClient.invoke_shell() ## Starts an interactive shell session. A new channel is opened and connected to a pseudo- ## terminal. The user can specify the terminal type and size ## Some parameters are; term (terminal type to emulate), width, height of the terminal ## and a command environment
channel.send('') ## This command allows users to send specific commands to the SRX router. ## -To fetch the SRX file, we have send different commands to the router.(See the code)
time.sleep(1) ## The **sleep()** method[2] suspends execution for the given number of seconds. For example, ## you can tell your program to wait a couple of second before sending the next command ## to the router. Here we have just added 1 second before the next execution.

[1] http://www.paramiko.org/
[2] https://www.tutorialspoint.com/python/time_sleep.htm