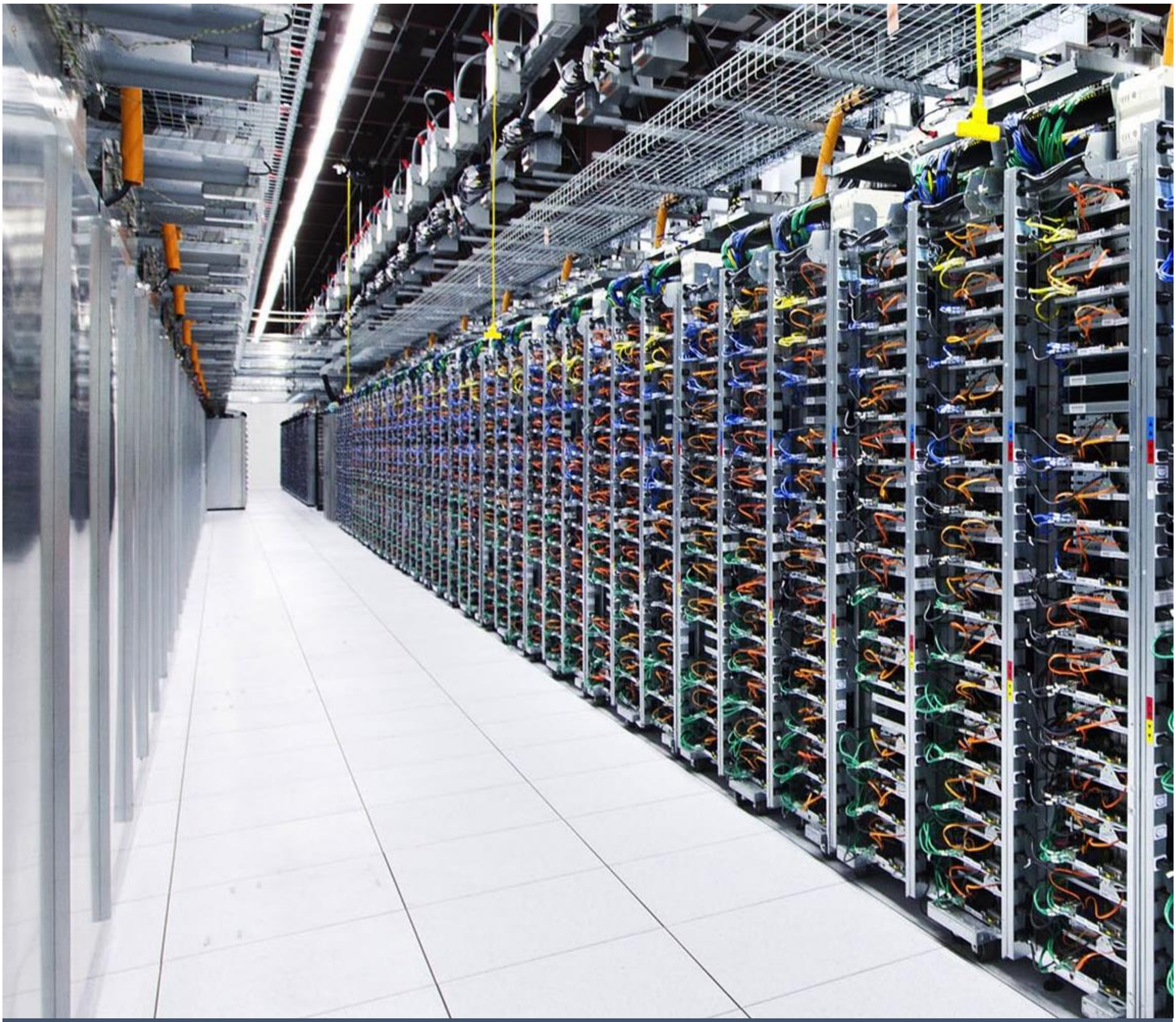


ASDN – Automated Software Defined Networking

Technical Specification – CA400

FILIP NIKOLIC – 14470852

20/05/2018

Supervisor: Brian Stone

Table of Contents

1. Introduction

1.1. Overview

This document will outline the system architecture, high-level design overview, the problems encountered while developing the application as well as project testing documentation.

Automated Software Defined Networking (ASDN) been developed to automate the daily maintenance of a typical enterprise network. This has been achieved with the use of a variety of complex tools and systems, including some machine learning algorithms and the latest network automation libraries.

It is also dependent on highly specialised hardware/software to be used to its full potential. In addition to carrying out changes autonomously it allows the user (owner of the network or a technical staff member) to be able to see any changes this system has made, as well as an overall network status.

The typical users of this system include highly specialised network engineers, who wish to automate tedious network maintenance work. Additionally, companies that wish to reduce their dependencies on 3rd party networking contractors will benefit greatly from using this platform.

1.2. Glossary

FTP Server – Hardware/software using the File Transfer Protocol to store, receive and share files over a network.

VPN – Virtual Private Network is a private network that is built over a public infrastructure. Security mechanisms, such as encryption, allow users to access network from different locations via the Internet.

Database Server - Hardware/software using languages such as MySQL to store and access data from a local/remote location for tasks such as analysis, storage, manipulation and archiving.

Network Switch – Connects various devices together on a computer network. It uses packet switching to receive, process and forward data to the destination device, operating on the data link layer of the OSI Model.

Network Router – Forwards data packets between different networks. Packets are usually forwarded between routers, until it reaches its destination node. They operate on the Network Layer of the OSI Model.

Firewall – A network security device that monitors incoming and outgoing traffic, making decisions in real-time whether to allow or block traffic based on a defined set of security rules.

ICMP – Network layer protocol that provides troubleshooting, control and error message services. Allows for the usage of a network utility called Ping.

SSH – A cryptographic network protocol for operating network services securely over an unsecured network. The best-known example application is for remote login to computer systems by users.

XML – A markup language that defines rules for encoding data in a human readable and machine-readable code.

ARP – Address Resolution Protocol is used by the Internet Protocol (IPv4), to map IP network addresses to the hardware addresses (MAC). It operates below the network layer as a part of the interface between the OSI network and link layer.

WAN – Wide Area Network is geographically distributed private telecommunications network that interconnects multiple local area networks.

LAN – Local Area Network is a group of computers and associated devices that share a common communications line or wireless link to a server.

Enterprise Network – An enterprise's communications backbone that interconnects networking capable devices located in different physical locations. It reduces WAN based protocols, facilitating quicker inter-device communication, as well as improved internal and external enterprise data management.

IKE – Internet Key Exchange is a protocol used to set up a security association in the IPsec protocol suite. Most commonly used as the first step for establishing a VPN connection.

IPSec – It is a secure network protocol suite, that authenticates and encrypts data packets sent over the network. Used most often as the second step of establishing a VPN connection, while being able to protect data flows between a pair of hosts.

VLAN -Virtual Local Area Network abstracts the idea of the LAN and might comprise of a subset of ports on a single switch or multiple switches.

YAML – A is a human-readable data serialization language with Python-style indentation, most commonly used for configuration files.

MVC – Model View Controller Framework is commonly used developing methodology that divides an application into three connected parts. It decouples these major components allowing for efficient code reuse and parallel development.

ORM – Object Relational Mapping is a technique that lets the user query and manipulate data from a database using an object-oriented paradigm.

Ansible – Ansible is an open-source automation engine that allows for software provisioning, configuration management, and application deployment.

SNMP – Simple Network Management Protocol is used for collecting information from network devices such as switches, routers, firewalls, etc.

2. Architecture

2.1. High-Level Design

To achieve the goals outlined above, five main high-level components have been developed, which make up the ASDN platform. The modules communicate with one another using multiple different channels and are described in further detail below:

- **Network Communication and Mapping –** Using a variety of protocols including SSH, this module logs into the networking devices to execute commands which are generated by the Network Automation Component. It is also responsible for connecting to devices located on customer's networks. They are most commonly in a different geographical location and to facilitate a secure and reliable connection it uses a site-to-site VPN.

- **Network Automation –** It's main purpose is to carry out automated changes to infrastructure critical networking devices, such as firewalls routers and switches. It is dependent on other modules to be able to make appropriate changes that achieve the desired goal, without affecting the network availability or negatively impacting the user's experience.

- **Statistical Machine Learning –** As the platform is fully automated, requiring almost no interference from the end-user, this is one of the most important modules. It is used in conjunction with the Network Automation component to help evaluate whether the changes made to the network are positive or negative. It can evolve over time to be able to more accurately assess the overall state of the network as well as each individual device.

- **Web Backend –** It's main purpose is to connect the end-user to the backend of the application. Additionally, it handles marshalling and unmarshalling of data between the frontend and the database. It also deals with user accounts and allows them to set up the VPN connection between their network and the ASDN network.

- **Web Frontend –** It is a dynamic web component and apart from allowing for the above mentioned backed functionality, it also gives the user the ability to see network related statistics such as how many of their devices are online, a graph of traffic flow through the network and many other.

2.2. Network Communication and Mapping

All network communication and mapping functionality was developed using Python 3.6. With the three main modules.

- Paramiko – Used to facilitate communication with remote devices via SSHv2.
- XmlToDict – The data flowing between the ASDN platform and the network equipment must be marshalled and unmarshalled. This library facilitates this functionality.
- scapy and netifaces – Used for packet manipulation and network scanning. These two libraries are also used to allow the Linux server to determine its uplink interface, local

subnet and scan the local network for devices. It uses ICMP protocols such as ping, the server's ARP cache to find all IP and MAC addresses on the LAN.

- urllib and bs4 – Used to get the information relating to the latest recommended software suitable for each type of device from the manufacturer's official support website.

2.3. Network Automation

To achieve the most streamlined automation process, I am using a combination of two powerful lightweight libraries. Most of the automation modules are developed in Python 2.7, as some key libraries used do not provide support for newer python versions.

- Ansible – Juniper Networks devices (and many other vendors) provide support for Ansible to manage devices running Junos OS. The modules required to do this are Ansible core and Ansible Galaxy. For every operational or configuration command to be carried out on the device, a "playbook" is created, containing a list of desired functions to be executed.
- Jinja2 – This is a templating engine used to generate the Ansible automation "playbooks" using the supplied parameters to create a YAML configuration file.
- Ftplib – In addition to the above, to eliminate the possibility of the program writing directly to the local file structure, FTP is used to upload the generated YAML file with minimal privileges to a protected directory on the server. This file is later accessed by a privileged Python 2.7 script whose only purpose is to execute that "playbook".

2.4. Statistical Machine Learning

This module was developed using Python libraries from the SciPy Stack Projects, that include Matplotlib, Pandas and sklearn. It is used to guide the Automation Module in making decisions as to what configurations to apply to the network and whether they had any negative impacts.

2.4.1. Metrics Used

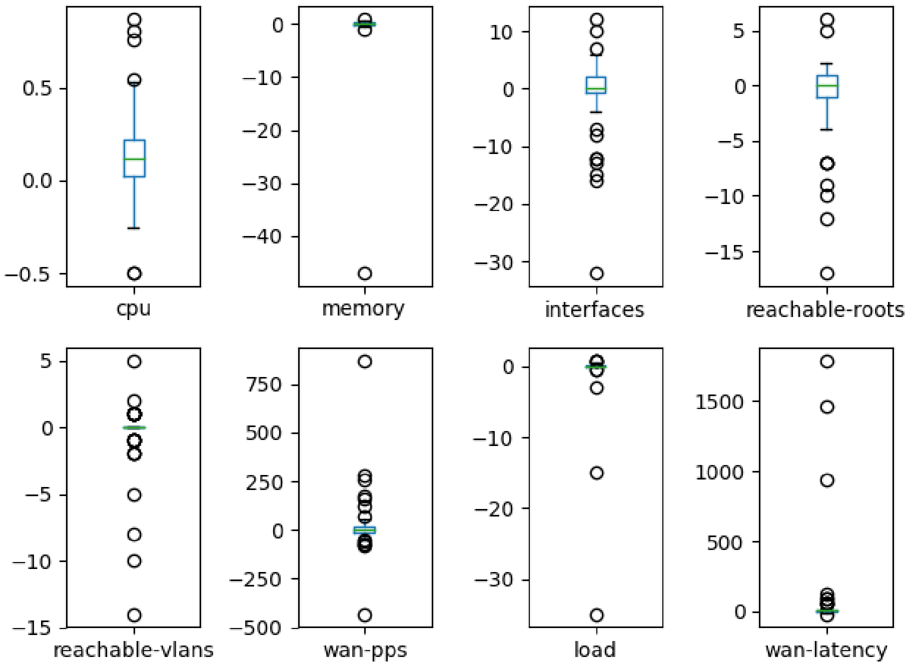
Many metrics were considered when making these decisions. Each is a delta of two values, the first taken before the configuration has been committed and the second after the commit. They must all be considered by the system when arriving at the result.

- **CPU Utilisation**
  - Indicating the potential overhead increase on the device's processor.
  - In many cases, a dramatic change of this value (both positive and negative) suggests a negative network state change.
- **Memory Utilisation**
  - This metric is used in conjunction with CPU utilisation and can further help determine whether an error on the network occurred.
  - It should be an even more stable metric then CPU utilisation, where extreme fluctuation almost guarantees unwanted behaviour.

- **Number of Interfaces**
    - By considering the number of physical and virtual interfaces that came online or went offline, we can determine whether the configuration has added or removed devices from the network.
    - When multiple interfaces go offline, this is most commonly a negative state change, alternatively a single added interface in conjunction with other attributes such as increase in system load, latency, etc may imply a loop introduced on the network.
- **Reachable Roots**
    - Using this metric, we can determine how many more/less Layer 3 devices we can reach after the configuration change.
    - In most situations a decrease is negative and increase negative.
- **VLAN Availability**
    - This metric is used to show how many more/less Virtual Local Area Networks are reachable by the device.
    - Akin to the reachable roots metric, a decrease is in most cases negative and increase positive.
- **Wide Area Network packets per second (pps)**
    - This distinct metric may help determine whether a loop has been introduced or indicate a significant decrease in user traffic.
    - Most commonly, a drastic positive or negative change in this value hints negative impacts of the new configuration.
- **System Load Average**
    - Unlike CPU and memory utilisation metrics, the system load average takes into consideration many more points, such as number of processes running, system response times, processes on hold, etc.
    - A sharp change of this value is ordinarily negative indication.
- **WAN Latency measured in milliseconds (ms)**
    - To measure WAN latency in milliseconds, we take the transit delays of packets to the default device route.
    - A significant increase of this value is undesirable.
- **Result Class**
    - The two classes are used to indicate the dataset result. The “valid” class implies that the changes made to the network had no negative impacts. The “invalid” class indicates potential unwanted behaviour, whereby one or all network clients are experiencing reduced performance or have lost connection completely.
    - They are output results of running an evaluation process on the set of above mentioned metrics.

2.4.2. Data Visualisation

There were three datasets with 50 entries each and a validation algorithm is used to determine the most accurate one. To better understand the data and decide on the best suited dataset, we use a number of visualisation techniques. The one I found most helpful is a box plot graph:



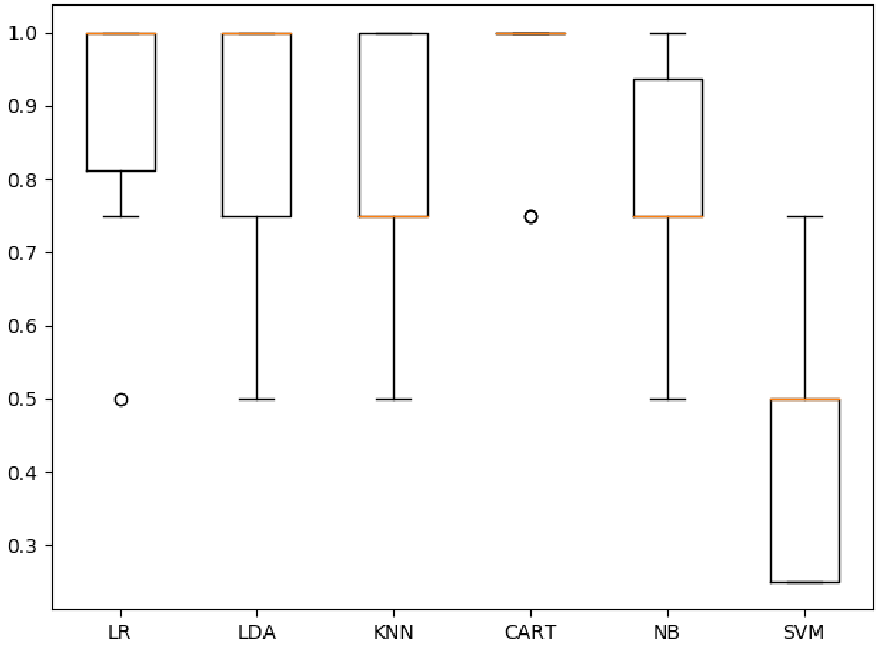
2.4.3. Algorithm Evaluation and Building the Best Model

The data must be separated out to form a validation dataset, 80% of which will be used to train our models and 20% that held back as a validation dataset. This will then be used to run the 10-fold cross validation.

Six algorithms were evaluated to choose the best one that will work with this dataset. Their accuracy results tell us in how many cases the evaluation of the algorithm was correct and in how many incorrect. Listed below are the results of those algorithms ran on the "best" dataset:

1. Logistic Regression (LR) is on average correct 90% of the time, with a mean deviation of score of 0.165831.
2. Linear Discriminant Analysis (LDA) is on average correct 87.5% with a mean deviation of score of 0.167705.
3. K-Nearest Neighbours (KNN) is on average correct 80% of the time, with a mean deviation of score of 0.187083.
4. Classification and Regression Trees (CART) is on average correct 92.5% of the time, with a mean deviation of score of 0.114564.
5. Gaussian Naive Bayes (NB) is on average correct 80% of the time, with a mean deviation of score of 0.150000.
6. Support Vector Machines (SVM) is on average correct 45% of the time, with a mean deviation of score of 0.187083.

Above is a good combination of simple linear (LR and LDA), nonlinear (KNN, CART, NB and SVM) algorithms. To ensure the evaluation of each is done using the same data splits, we reset the random number seed before every run. This makes the results directly comparable.



Using the graph above, we can see that the most suitable algorithm for this dataset is the Classification and Regression Trees Algorithm. Therefore, we will use it to further build and improve upon the model accuracy and performance.

2.4.4. Making Predictions using the CART Algorithm

After running the algorithm on a set of 10 data points, an accuracy of 100% was observed. Whereby, CART correctly classified six invalid and four valid configuration changes correctly. The following metrics were observed:

- **Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.
- **Recall** is the ratio of correctly predicted positive observations to the all observations.
- **F1-Score** is the weighted average of Precision and Recall.

	precision	*recall*	*f1-score*	*support*
Invalid	1.0	1.00	1.00	6
Valid	1.00	1.00	1.00	4
avg / total	1.00	1.00	1.00	10

2.5. Web Backend

A Tomcat Server is used to host the application and a Spring MVC backend to process web based tasks and pass data between the database and the front end. Below are some of the main dependencies used and their functionality:

- Hibernate – Implemented as a ORM (Object Relational Mapping) utility, it allowed me to streamline relational database communication via JDBC. By creating classes such as *User* with appropriate parameters, the database fields are populated and Java instead of SLQ syntax is used to interact with it.
- Spring Framework Modules – The main libraries used from this framework are:
    - **Security** to keep a user session alive by tracking a session token and handling login, registration and logout operations.
    - **Mail Service,** used to send the user the network device username and encrypted password, to be

2.6. Web Frontend

The web front end is developed in Angular 6 with all dependencies handled by a package manager called npm. Main frontend dependencies:

- Bootstrap – Used for rapid frontend prototyping, allowing for responsive web design that is compatible with most modern browsers.
- Google Charts – Useful when displaying complex network information such as traffic flow histograms and device availability pie charts.
- JavaScript Vis – A network map visualisation tool, allowing me to show all devices on a single network to the user.

2.7. Security

This is one of the most important considerations when designing the project, as any compromises of the application may result is severe damages both to property, as well as loss of private customer data. The following is a list of technologies, tools and practises used to reduce the likelihood of a successful attack:

- **Site-to-site VPN** – It is automatically deployed between the customer and ASDN network. The platform allows the user to choose between several different VPN algorithms to best suit their security needs.
    - The IKE Authentication and Encryption Algorithms supported include, but are not limited to SHA (all modern versions supported), MD5 and HMAC.
    - The IPSec Authentication and Encryption Algorithms supported are 3DES, DES and AES (all modern versions supported).

- **SSL Website Encryption –** The web application is secured using an SSL certificate which is applied to the publicly accessible domain name asdn.ie. It keeps the user’s connection to the website secure safeguards any sensitive data transmitted between two systems.

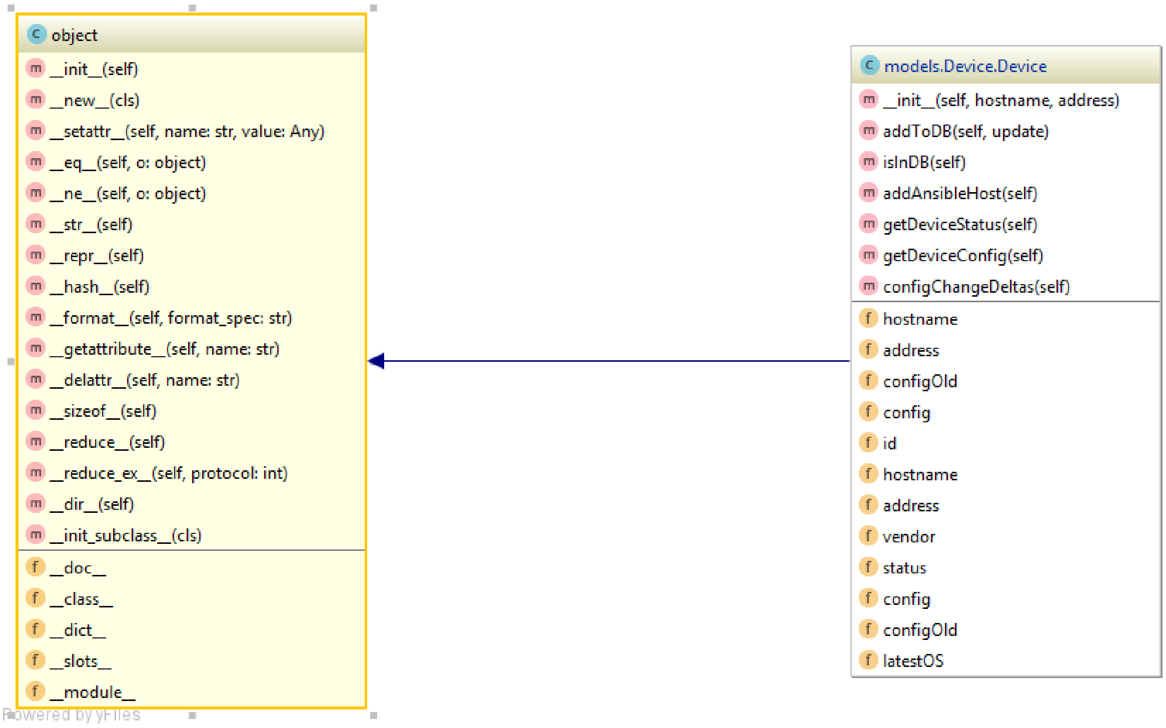
- **Two physical Firewalls –** The main network firewall positioned between the ASDN application and the user is a Juniper Networks SRX300. It can provide up to 1 Gb/s of Firewall Traffic Filtering as well as 100 Mb/s of site-to-site VPN Traffic. In addition to it, the main server (Ubuntu Server OS) has a deployed Uncomplicated Firewall that allow an extremely limited traffic flow between it and the outside network.

- **Stored Data Encryption –** The ubuntu application server utilises an Encrypting File System in addition to all sensitive information such as passwords being stored encrypted in the database. The sensitive data is done using SHA1 and an 8-byte salt.

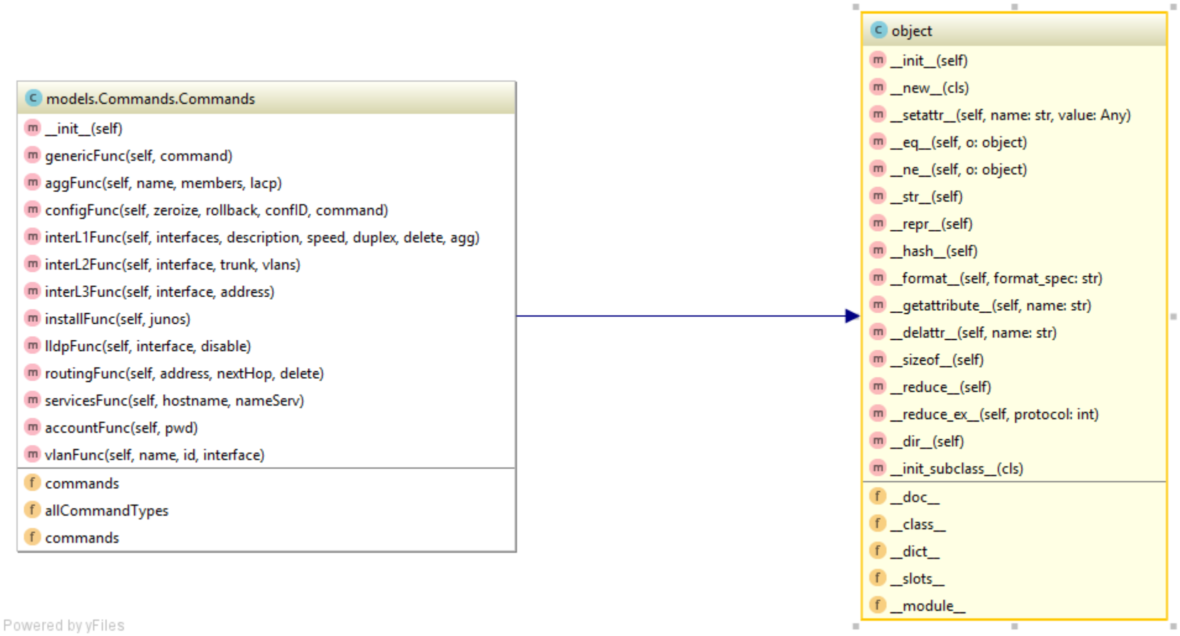
- **POLP –** The Principle of Least Privilege is followed, where the system implements a strict permissions policy. Only the root user can run and execute as sudo, all other users have non-escalated privileges including the one executing the application code.

2.8. Class Diagrams

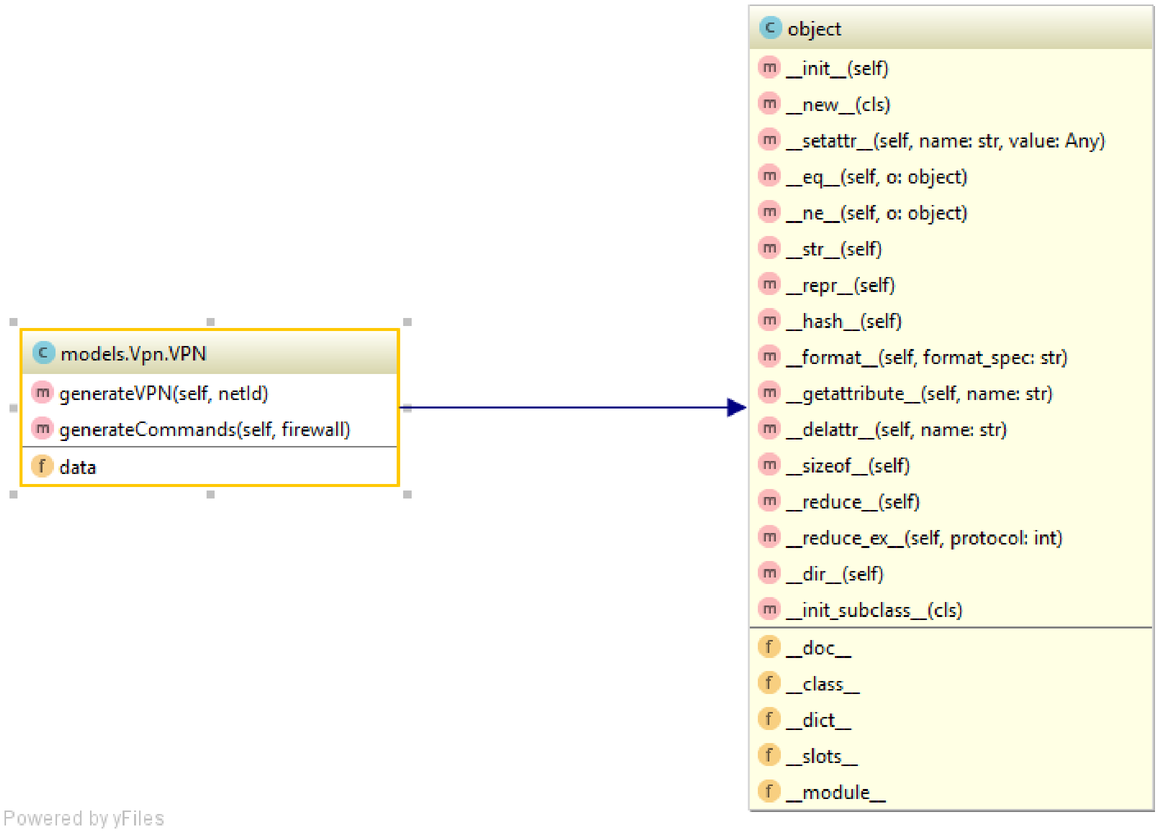
Python



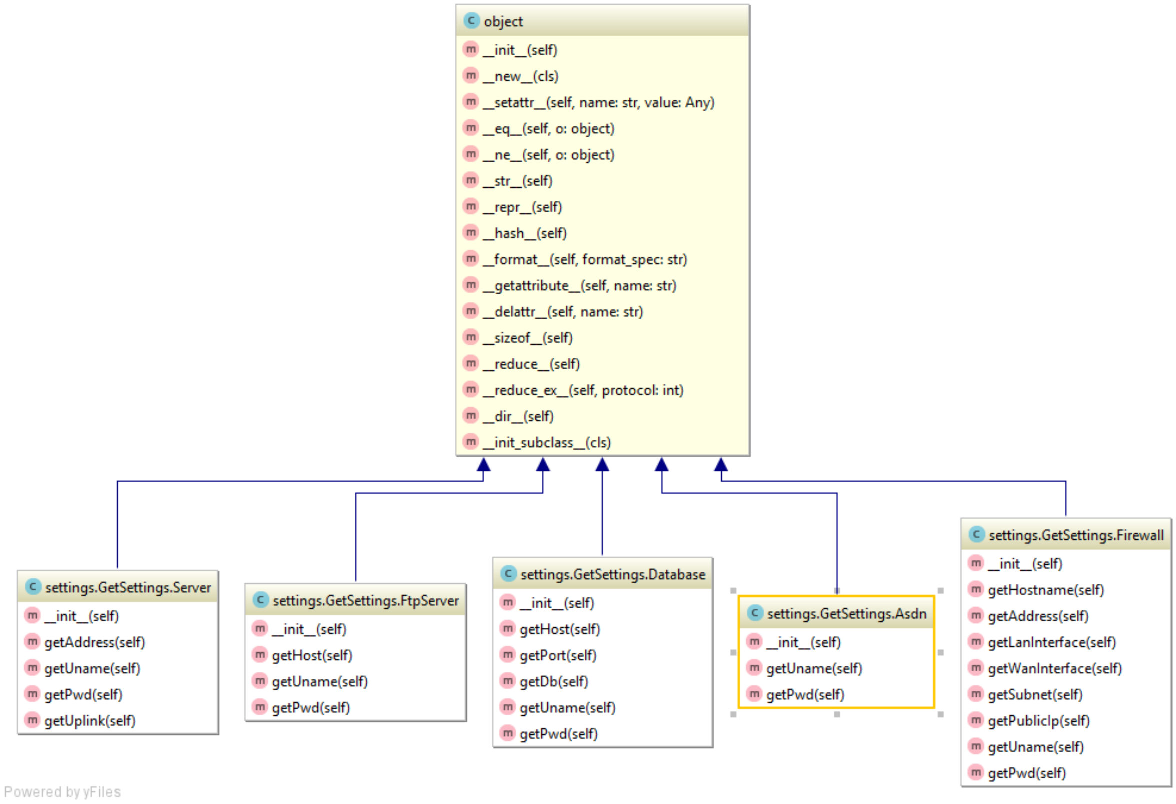
Device Class


models.Commands.Commands
__init__(self)
genericFunc(self, command)
aggFunc(self, name, members, lacp)
configFunc(self, zeroize, rollback, confID, command)
interL1Func(self, interfaces, description, speed, duplex, delete, agg)
interL2Func(self, interface, trunk, vlans)
interL3Func(self, interface, address)
installFunc(self, junos)
lldpFunc(self, interface, disable)
routingFunc(self, address, nextHop, delete)
servicesFunc(self, hostname, nameServ)
accountFunc(self, pwd)
vlanFunc(self, name, id, interface)
commands
allCommandTypes
commands
object
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__class__
__dict__
__slots__
__module__
Powered by yFiles

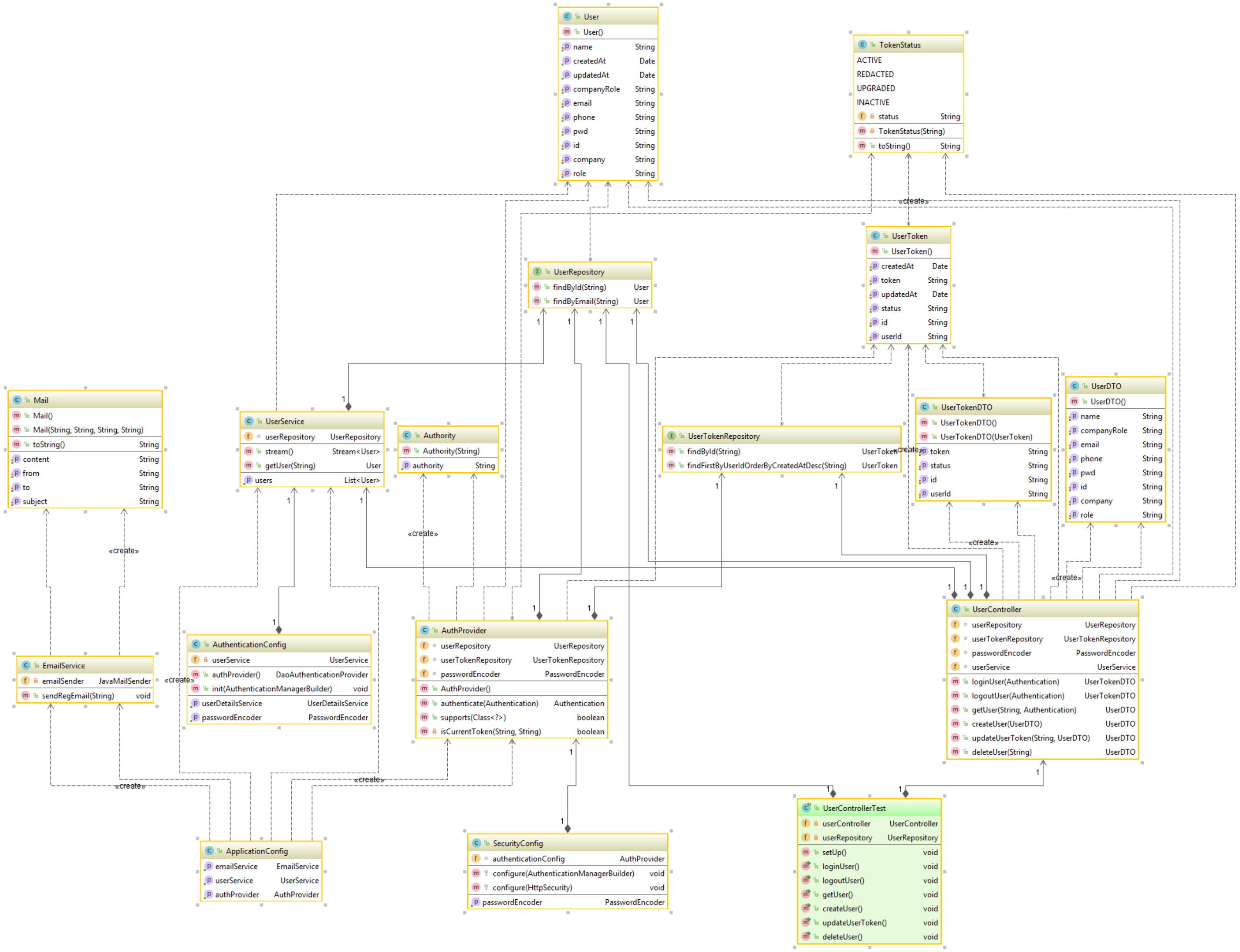
Commands Class



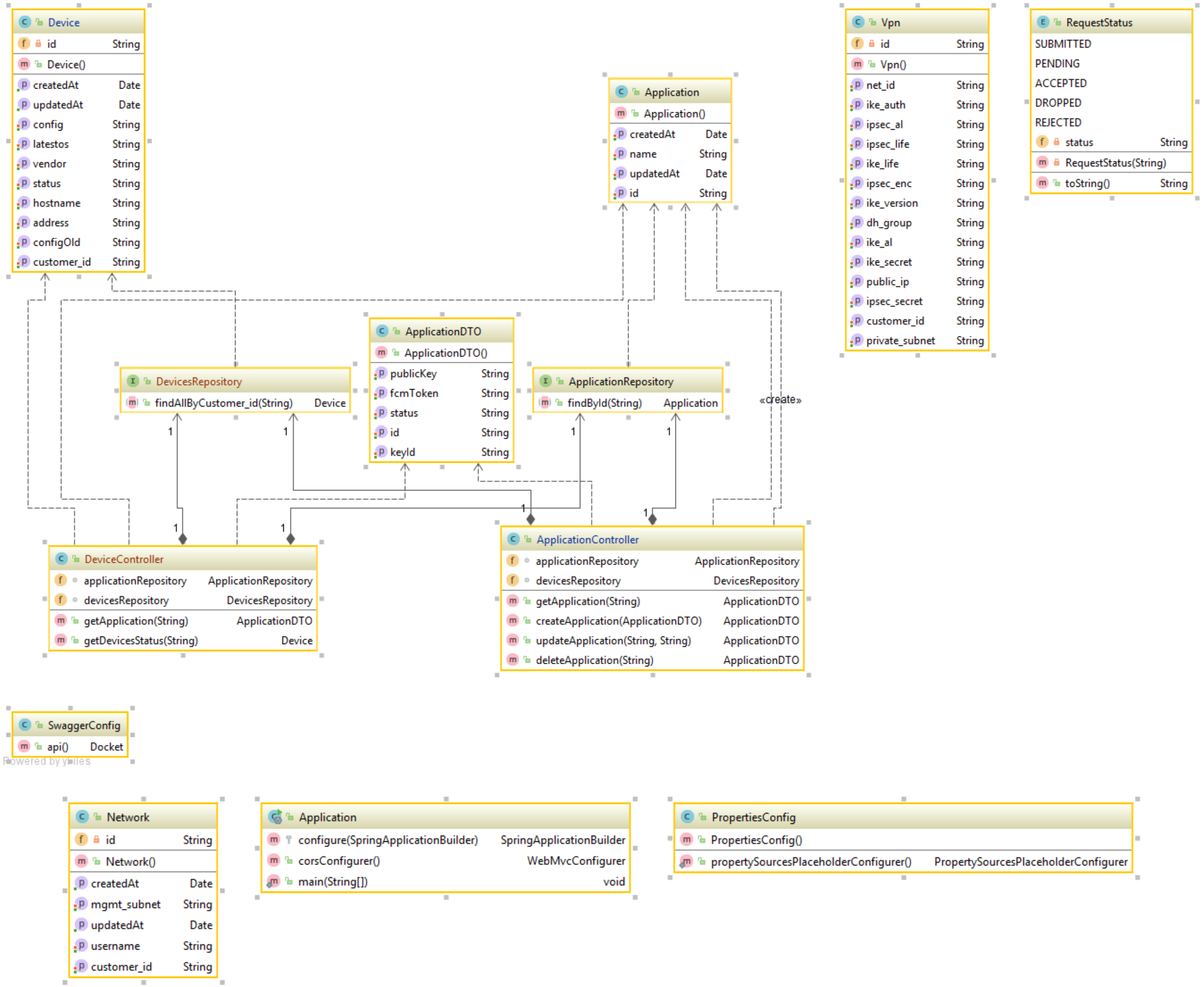
VPN Class


object
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
settings.GetSettings.Server
__init__(self)
getAddress(self)
getUname(self)
getPwd(self)
getUplink(self)
settings.GetSettings.FtpServer
__init__(self)
getHost(self)
getUname(self)
getPwd(self)
settings.GetSettings.Database
__init__(self)
getHost(self)
getPort(self)
getDb(self)
getUname(self)
getPwd(self)
settings.GetSettings.Asdn
__init__(self)
getUname(self)
getPwd(self)
settings.GetSettings.Firewall
__init__(self)
getHostname(self)
getAddress(self)
getLanInterface(self)
getWanInterface(self)
getSubnet(self)
getPublicIp(self)
getUname(self)
getPwd(self)
Powered by yFiles

VPN Class



Main Spring Models, Controllers and Configurations



Application DTOs and Repositories

2.9. Design Constraints

Network automation projects naturally present many challenges due to their dynamic nature and complexity. During the planning phase of the project, the following constraints were identified:

- **Remote Networking Devices –** As none of the customer's firewalls, routers and switches were initially accessible by my application, I had to find a way to securely monitor and maintain the offsite LAN.
- **Correctly Mapping the Network –** To generate an accurate map of a devices are located on a network, many potential network topologies must be taken into consideration. For example, ICMP might be blocked on a network, LLDP disabled on most devices, as well as SNMP. Therefore, the ability to contact other devices is very limited.
- **Different Device Vendors –** This was one of the most challenging constraints, whereby I had to decide which vendors to support, as the interaction process vastly differs between networking device manufacturers.
- **Access to Hardware and Testing –** To be able to demonstrate, build and test the application, access to high-end, enterprise-grade networking devices was critical. This is where the company I work for, Agile Networks provided me with devices as well as a lab environment for testing ASDN.
- **Limited Functionality Support** – A typical networking device supports thousands of possible operational and configuration commands and I had to choose which are to most important that can be successfully automated.

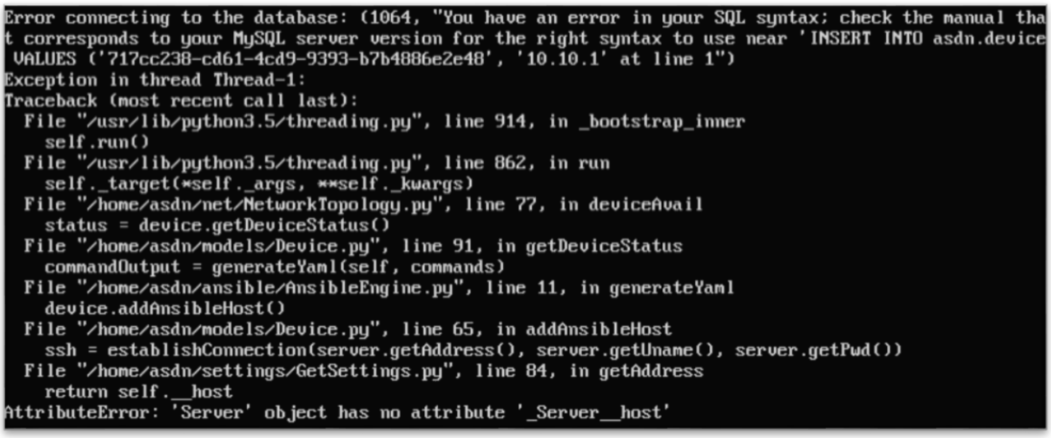
3. Problems and Solutions

Below is a list of areas where I was faced with the most challenging problems and a quick explanation as to how they were solved.

3.1. Networking, Infrastructure and OS

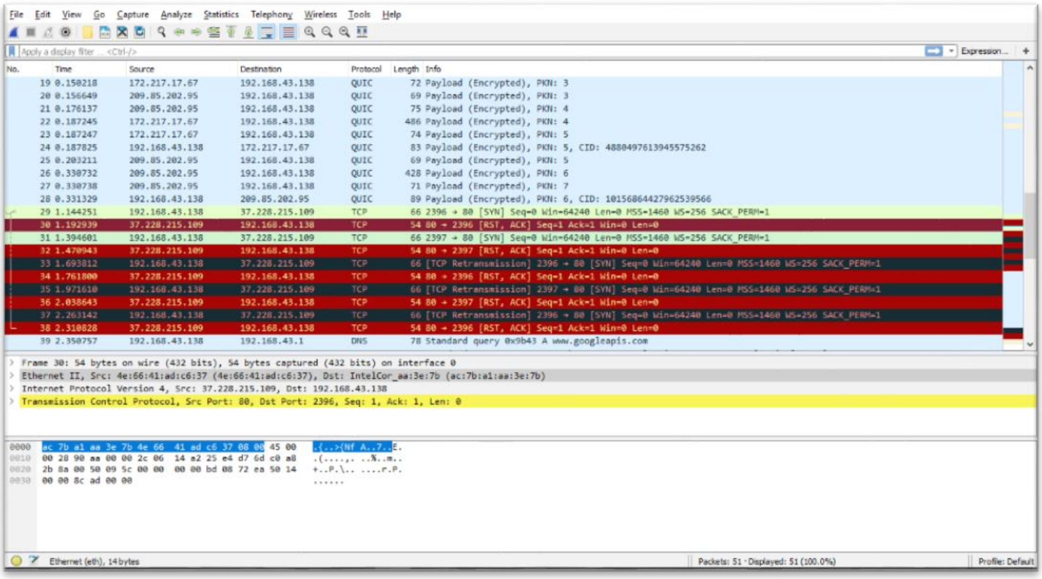
Most of the challenges during the development of this application I faced fall within this scope.

- **Pymysql** unable to execute multiple commands on Linux, but the same code works on Windows. After researching the possible solutions for this issue, this was the most helpful solution I found, that after further modification helped me fix this issue. It was solved by creating a temporary non-sudo SSH user and exporting path variables, to allow it to execute the desired commands.
  I have listed the output of the errors thrown:



```
Error connecting to the database: (1064, "You have an error in your SQL syntax; check the manual tha
t corresponds to your MySQL server version for the right syntax to use near 'INSERT INTO asdn.device
 VALUES ('717cc238-cd61-4cd9-9393-b7b4886e2e48', '10.10.1' at line 1")
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib/python3.5/threading.py", line 914, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.5/threading.py", line 862, in run
    self._target(*self._args, **self._kwargs)
  File "/home/asdn/net/NetworkTopology.py", line 77, in deviceAvail
    status = device.getDeviceStatus()
  File "/home/asdn/models/Device.py", line 91, in getDeviceStatus
    commandOutput = generateYaml(self, commands)
  File "/home/asdn/ansible/AnsibleEngine.py", line 11, in generateYaml
    device.addAnsibleHost()
  File "/home/asdn/models/Device.py", line 65, in addAnsibleHost
    ssh = establishConnection(server.getAddress(), server.getUname(), server.getPwd())
  File "/home/asdn/settings/GetSettings.py", line 84, in getAddress
    return self.__host
AttributeError: 'Server' object has no attribute '_Server__host'
```

- **VPN Generation Tool** was unable to automate the deployment of VPNs correctly. After extensive troubleshooting and research where I looked for similar applications, but was unable to find any, I discovered bug in the then current release of Juniper OS, whereby some the UDP port 500 needed to establish the VPN was not being allowed by the firewall. Below is a Wireshark capture of the level of troubleshooting required to fix this issue.



```
No.   Time      Source           Destination      Protocol Length Info
   19 0.150218  172.217.17.67    192.168.43.138   QUIC       72 Payload (Encrypted), PKN: 3
   20 0.156649  209.85.202.95    192.168.43.138   QUIC       69 Payload (Encrypted), PKN: 3
   21 0.176137  209.85.202.95    192.168.43.138   QUIC       75 Payload (Encrypted), PKN: 4
   22 0.187245  172.217.17.67    192.168.43.138   QUIC      486 Payload (Encrypted), PKN: 4
   23 0.187247  172.217.17.67    192.168.43.138   QUIC       74 Payload (Encrypted), PKN: 5
   24 0.187825  192.168.43.138   172.217.17.67    QUIC       83 Payload (Encrypted), PKN: 5, CID: 4880497613945575262
   25 0.203211  209.85.202.95    192.168.43.138   QUIC       69 Payload (Encrypted), PKN: 5
   26 0.330732  209.85.202.95    192.168.43.138   QUIC      428 Payload (Encrypted), PKN: 6
   27 0.330738  209.85.202.95    192.168.43.138   QUIC       71 Payload (Encrypted), PKN: 7
   28 0.331329  192.168.43.138   209.85.202.95    QUIC       89 Payload (Encrypted), PKN: 6, CID: 10156864427962539566
   29 1.144251  192.168.43.138   37.228.215.109   TCP        66 2396 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   30 1.192939  37.228.215.109   192.168.43.138   TCP        54 80 → 2396 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   31 1.394601  192.168.43.138   37.228.215.109   TCP        66 2397 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   32 1.470943  37.228.215.109   192.168.43.138   TCP        54 80 → 2397 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   33 1.693812  192.168.43.138   37.228.215.109   TCP        66 [TCP Retransmission] 2396 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   34 1.761800  37.228.215.109   192.168.43.138   TCP        54 80 → 2396 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   35 1.971610  192.168.43.138   37.228.215.109   TCP        66 [TCP Retransmission] 2397 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   36 2.038643  37.228.215.109   192.168.43.138   TCP        54 80 → 2397 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   37 2.263142  192.168.43.138   37.228.215.109   TCP        66 [TCP Retransmission] 2396 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   38 2.310828  37.228.215.109   192.168.43.138   TCP        54 80 → 2396 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   39 2.350757  192.168.43.138   192.168.43.1     DNS        78 Standard query 0x9b43 A www.googleapis.com

> Frame 30: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: 4e:66:41:ad:c6:37 (4e:66:41:ad:c6:37), Dst: IntelCor_aa:3e:7b (ac:7b:a1:aa:3e:7b)
> Internet Protocol Version 4, Src: 37.228.215.109, Dst: 192.168.43.138
> Transmission Control Protocol, Src Port: 80, Dst Port: 2396, Seq: 1, Ack: 1, Len: 0

0000  ac 7b a1 aa 3e 7b 4e 66  41 ad c6 37 08 00 45 00   .{..>{Nf A..7..E.
0010  00 28 90 aa 00 00 2c 06  14 a2 25 e4 d7 6d c0 a8   .(.....,. ..%..m..
0020  2b 8a 00 50 09 5c 00 00  00 00 bd 08 72 ea 50 14   +..P.\.. ....r.P.
0030  00 00 8c ad 00 00                                  ......

Ethernet (eth), 14 bytes          Packets: 51 · Displayed: 51 (100.0%)          Profile: Default
```

3.2. Ansible Automation

The two main challenges when deploying ansible were:

- How to safely execute the Python 2.7 script that applies the configuration to the device. The solution for this problem was a library called execnet, enabling me to spin off a separate process from Python 3.6 that safely executes a Python 2.7 script. Despite being very difficult to implement, requiring OS level changes, I was eventually able to get it working.
- How to dynamically generate configuration (YAML) files. After using three different libraries and unable to achieve the results needed, I was told by a colleague at work to try Jinja2 and got it working very quickly.

3.3. Machine Learning

The most challenging two problems when designing my machine learning module were

- Which network parameters to take into consideration when training the prediction model.
- What are the most suitable Machine Learning algorithms that can be applied to most effectively solve the problem.

After careful research of many scholar publications that cover how machine learning can be applied to computer networks (some in the references section below), I have chosen the parameters and algorithms explained in the Machine Learning Section.

4. Deployment Guide

This goal of this guide is to enable the user to deploy the ASDN application efficiently. As it uses a client-server architecture, this process is not to be carried out by a standard user, but is an aid in how to migrate this application to a different environment (business, datacentre, etc.)

4.1. Dependencies and Development Platforms

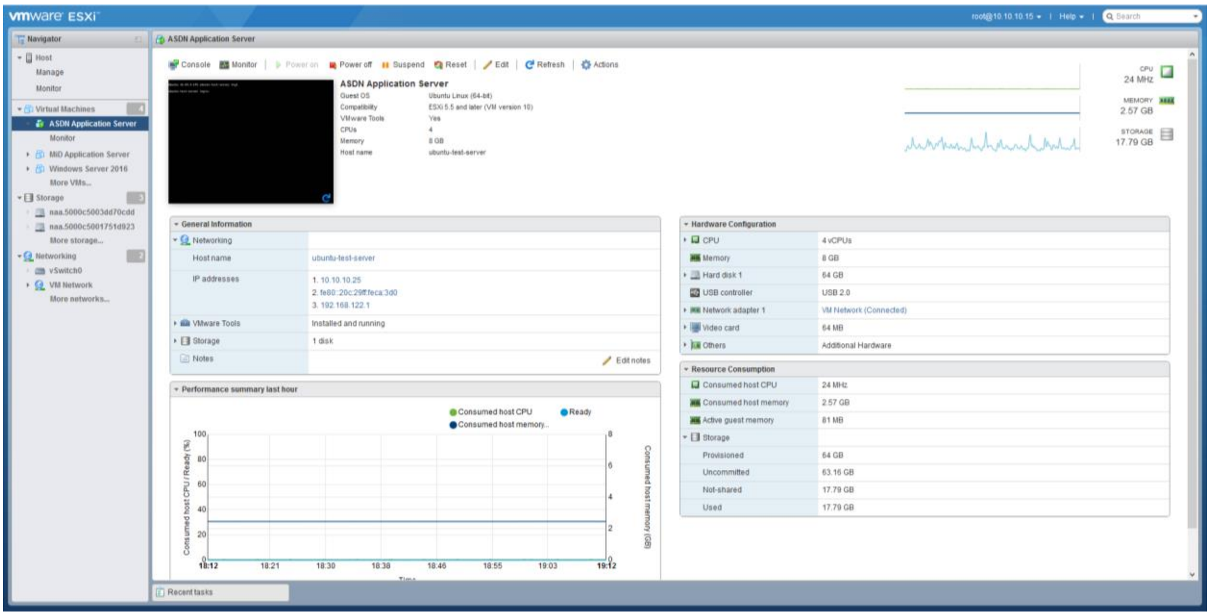
Both Python 3.6 and python 2.7 need to be installed. To manage dependencies, pip is used, and a list of main all dependencies can be found here. Tomcat Server 9.0.4 is required for the spring application.

4.2. Local Network Requirements

The key networking device required for the application to operate is a Juniper Firewall. The current configuration of the firewall can be found here and should be used as a template when deploying the module to a server on the same network.

4.3. Server Architecture

The OS is deployed on an ESXi 6.5 server with the following setup:



ASDN Application Server

Guest OS	Ubuntu Linux (64-bit)
Compatibility	ESXi 5.5 and later (VM version 10)
VMware Tools	Yes
CPUs	4
Memory	8 GB
Host name	ubuntu-test-server

General Information

Networking	
Host name	ubuntu-test-server
IP addresses	1. 10.10.10.25 2. fe80::20c:29ff:feca:3d0 3. 192.168.122.1
VMware Tools	Installed and running
Storage	1 disk
Notes	Edit notes

Performance summary last hour

Hardware Configuration

CPU	4 vCPUs
Memory	8 GB
Hard disk 1	64 GB
USB controller	USB 2.0
Network adapter 1	VM Network (Connected)
Video card	64 MB
Others	Additional Hardware

Resource Consumption

Consumed host CPU	24 MHz
Consumed host memory	2.57 GB
Active guest memory	81 MB
Storage	
Provisioned	64 GB
Uncommitted	63.16 GB
Not-shared	17.79 GB
Used	17.79 GB

5. Testing Documentation

The main objective of the tests is to verify the functionality of the ASDN Platform, with the main emphasis on dynamic network testing and the use of automated testing tools.

5.1. Unit Testing

Unit testing has been carried out on most Python functions with multiple possible arguments supplied to account for boundary values. To simulate external dependencies mock is used to assert dummy values. All Python unit tests can be found here and Java Spring here.

5.2. Integration Testing

To test how different components, work together I have created a set of automated tests that are ran, where functions call one another with different parameters and the results returned are compared to an expected set of values. All integration tests can be found here.

5.3. User Testing

User testing has been done in Agile Networks by qualified network engineers, as well as some non-technical staff. All users have been given different tasks and instructed to rate each based in the different criteria. The rating system for each task and criteria is 0 (worst) to 10 (best).

Tasks:

1. Create an account on the ASDN Website and log in.
2. Set up the VPN connection between their network and ASDN using the popup configuration wizard.
3. Log out of the account.
4. Find the percentage of unsupported devices on the network.
5. Identify the average network activity in Mb/s in March.
6. Find a device's configuration and identify when it was committed.
7. Identify which ports on a device are *unused*.
8. Find the percentage of *static routes* on a device, *VoIP VLANS* and rules in the zone called *vpn.*

Criteria:

- **Discovery** – how intuitive the system is to use without requiring any prior knowledge of how application operates.
- **Design Validation** – what the user's initial impressions are and whether they understand it.
- **Navigation/Findability** – can the users navigate the system correctly and find the item they were looking for by using the menus.
- **Task Completion** – were they able to correctly carry out the tasks assigned to them.
- **Value and Purpose** – do they find the application useful and see themselves using it in the future.

User 1 – Network Engineer

Task No	Discovery	Design Validation	Navigation/Findability	Task Completion	Value and Purpose
1	10	9	10	10	8
2	7	8	9	10	10
3	10	9	10	10	4
4	8	9	8	10	8
5	7	9	10	10	9
6	9	4	9	9	3
7	6	10	8	10	4
8	10	9	10	10	6

User 2 – Sales Consultant

Task No	Discovery	Design Validation	Navigation/Findability	Task Completion	Value and Purpose
1	8	10	10	10	6
2	1	3	7	0	5
3	10	10	10	10	6
4	5	8	7	6	7
5	8	6	4	7	7
6	3	2	4	3	4
7	3	7	6	7	8
8	6	6	9	8	9

User 3 – Intern (Network Engineering Position)

Task No	Discovery	Design Validation	Navigation/Findability	Task Completion	Value and Purpose
1	10	9	9	10	8
2	6	7	8	8	7
3	9	9	10	10	8
4	9	7	6	9	7
5	8	10	9	9	10
6	7	6	7	8	9
7	5	6	7	8	3
8	9	10	9	9	7

5.4. Functional Testing

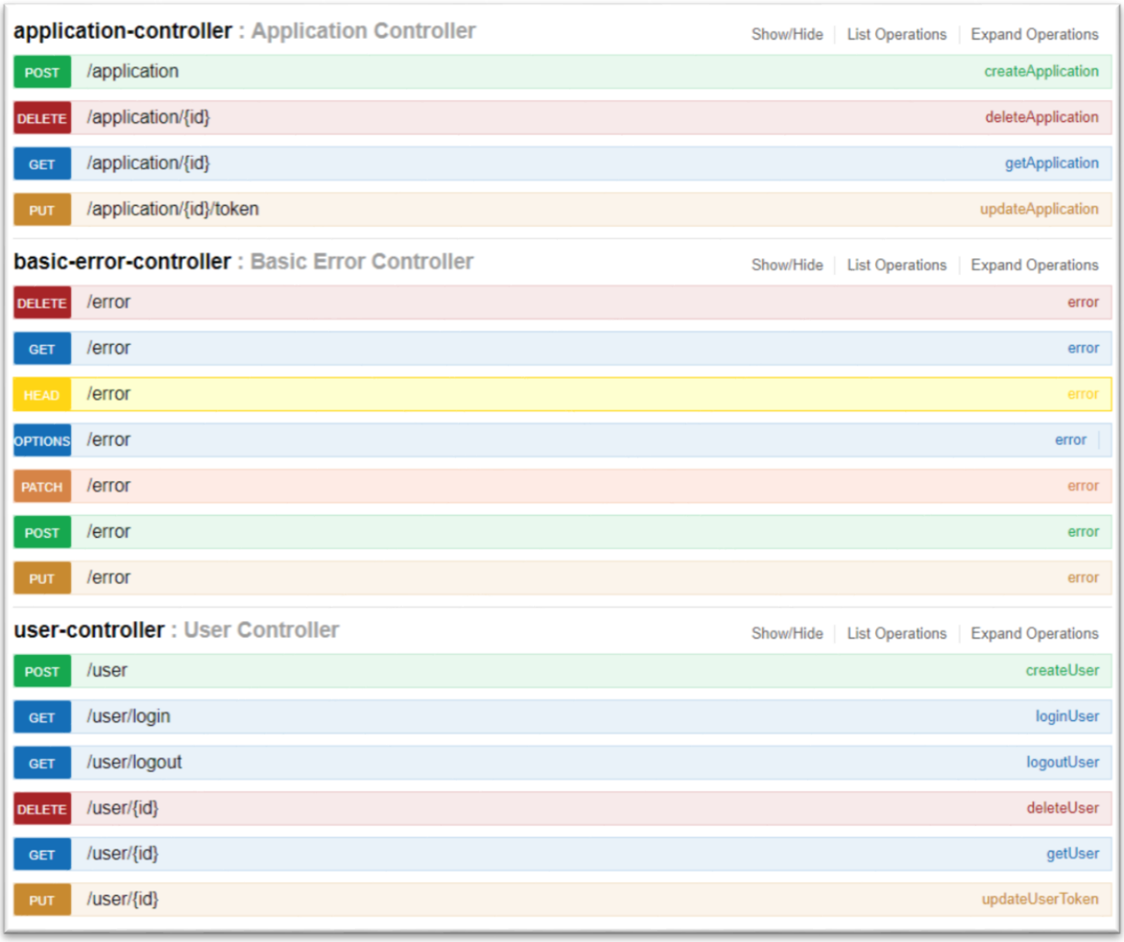
Workflow No	Description	Importance	Difficulty	Input	Result
1	Create User Account	Hight	Low	First Name, Last Name Email, Password, Confirmed Password	Profile information is propagated to the database and the password stored encrypted.
2	Login User	High	Low	Email, Password	Check the input parameters against the database entry and redirect the user to the dashboard page if they match.
3	Generate a site-to-site VPN	High	High	*Large input list containing all VPN parameters	Apply the configuration to the local Juniper Networks firewall.
4	Validate the VPN Connection	High	High	LAN Subnet gathered from the Database (provided by the user during VPN configuration)	Able to ping or connect to the network's default gateway (firewall)
5	Scan the Remote Network	High	High	Local IP Address returned by the previous function	A list of devices on the network and their status and detection of ones that went offline
6	Update All Frontend UI	Medium	Medium	User id	Return all device parameters that belong to the user
7	Make automated network changes	High	High	Network Activity (e.g. new device plugged in)	The device is added to the proper VLAN.
8	Using ML Evaluate Network Changes	High	High	Device status before and after the applied configuration	Evaluate whether the change is Correct or Incorrect

Workflow No.	Input	Expected Result	Actual Result	Comment
1	*Elon, Musk, colony@mars.com. spaceman123, spaceman123*	Account Created	Account Created	Working as intended.
1	*James, Webb, telescopes@nasa.com. space123, space321*	Error	Error	The two passwords supplied must match.
2	*colony@mars.com, spaceman123*	Login User	Login User	Working as intended.
3	*Large number of data matching user's firewall VPN setup	Establish VPN Connection	Establish VPN Connection	Working as intended.
3	*Large number of data not matching user's firewall VPN setup	Error	Error	The information on the VPN capable device and provided connection details must match.
4	Correct LAN information given by the user	Connection Validated	Connection Validated	Working as intended.
4	Incorrect LAN information given by the user	Connection reported as not reachable	Connection reported as not reachable	The user must provide the correct LAN subnet as well as terminating the VPN connection on that address range.
4	Correct LAN information given by the user	Connection Validated	Connection reported as not reachable	If the internet connection between the two sites drops the network will be reported as not reachable.
5	LAN subnet range	A list of network devices	A list of network devices	Working as intended.
5	LAN subnet range	A list of network devices	Error	One of the following protocols must be allowed on the network: ICMP, SNMP, ARP
6	User id	Updated Frontend UI	Updated Frontend UI	Working as intended
6	User id	Updated Frontend UI	Error	The user must firs setup the VPN connection to allow for this functionality

Workflow No.	Input	Expected Result	Actual Result	Comment
7	Network Action, Device Details	Apply the automated network changes	Apply the automated network changes	Working as intended
7	Network Action, Device Details	Apply the automated network changes	Error	The device must allow ASDN SSH access, by containing the credentials provided to the user.
7	Network Action, Device Details	Apply the automated network changes	Error	The device is powered off, disconnected or the VPN connection drops.
8	Device, Pre-config Status, Post-config Status	Predict Outcome of the configuration	Predict Outcome of the configuration	Working as intended
8	Device, Pre-config Status, Post-config Status (device defective, e.g. excessive CPU utilisation)	Predict Outcome of the configuration	Error	If a device has an issue such as a memory leak, it may cause CPU reading to be abnormal, skewing the ML system’s ability to correctly predict the output.

5.5. Automated Testing Tools

- To test the Spring RESTful API functionality, I am using a platform called Swagger. Each Controller has a set of URL endpoints which deal with GET, POST, PUT and DELETE requests. All Swagger API functionality test results are here.



application-controller : Application Controller — Show/Hide | List Operations | Expand Operations

POST	/application	createApplication
DELETE	/application/{id}	deleteApplication
GET	/application/{id}	getApplication
PUT	/application/{id}/token	updateApplication

basic-error-controller : Basic Error Controller — Show/Hide | List Operations | Expand Operations

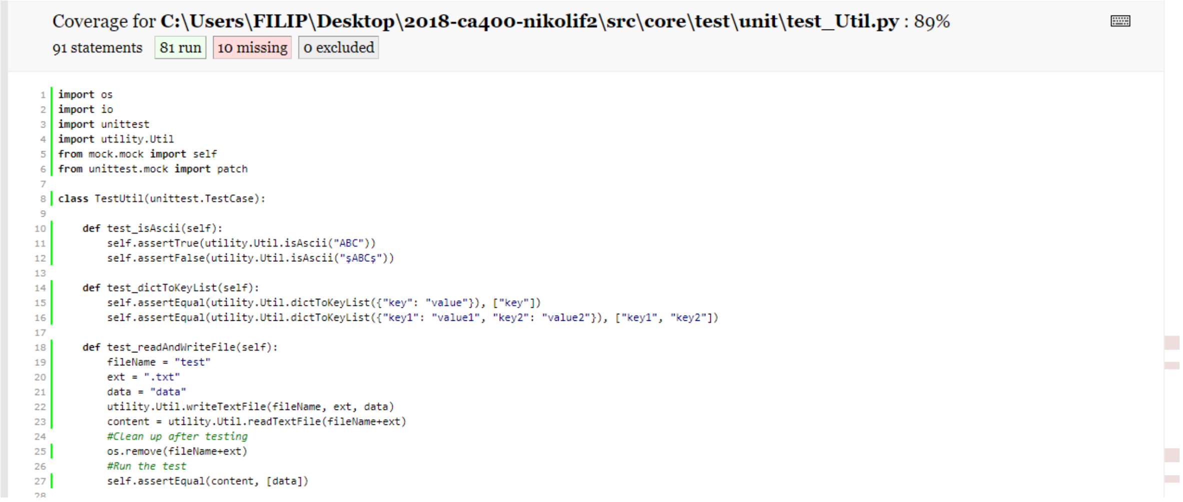
DELETE	/error	error
GET	/error	error
HEAD	/error	error
OPTIONS	/error	error
PATCH	/error	error
POST	/error	error
PUT	/error	error

user-controller : User Controller — Show/Hide | List Operations | Expand Operations

POST	/user	createUser
GET	/user/login	loginUser
GET	/user/logout	logoutUser
DELETE	/user/{id}	deleteUser
GET	/user/{id}	getUser
PUT	/user/{id}	updateUserToken

List of some Controller API Endpoints

- All code coverage and other automated tests ran can be found here.



Coverage for **C:\Users\FILIP\Desktop\2018-ca400-nikolif2\src\core\test\unit\test_Util.py** : 89%

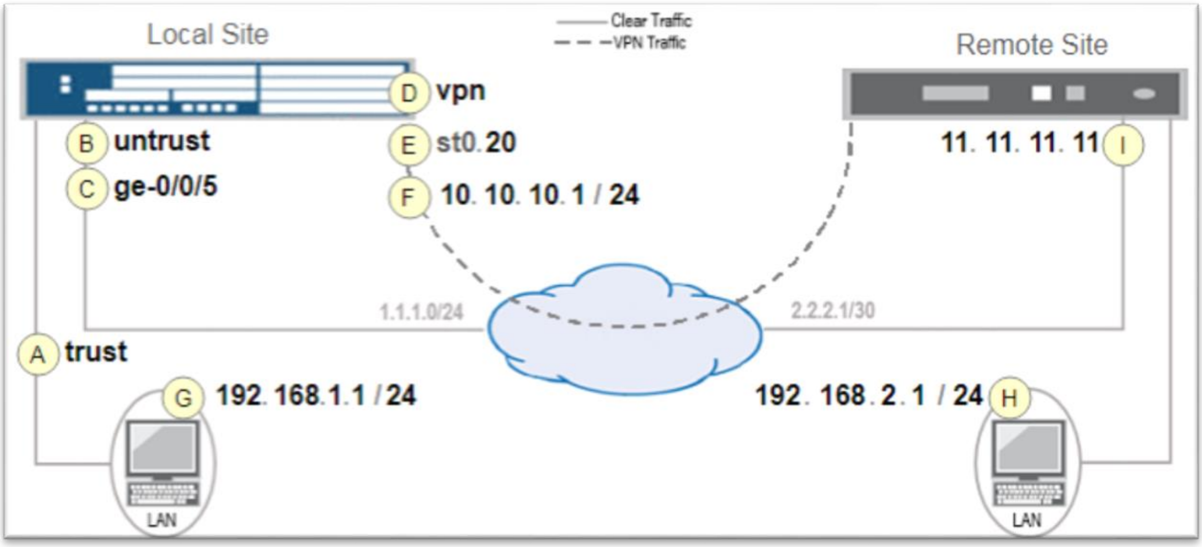
91 statements 81 run 10 missing 0 excluded

```
import os
import io
import unittest
import utility.Util
from mock.mock import self
from unittest.mock import patch

class TestUtil(unittest.TestCase):

    def test_isAscii(self):
        self.assertTrue(utility.Util.isAscii("ABC"))
        self.assertFalse(utility.Util.isAscii("şABCş"))

    def test_dictToKeyList(self):
        self.assertEqual(utility.Util.dictToKeyList({"key": "value"}), ["key"])
        self.assertEqual(utility.Util.dictToKeyList({"key1": "value1", "key2": "value2"}), ["key1", "key2"])

    def test_readAndWriteFile(self):
        fileName = "test"
        ext = ".txt"
        data = "data"
        utility.Util.writeTextFile(fileName, ext, data)
        content = utility.Util.readTextFile(fileName+ext)
        #Clean up after testing
        os.remove(fileName+ext)
        #Run the test
        self.assertEqual(content, [data])
```

5.6. Dynamic Network Testing

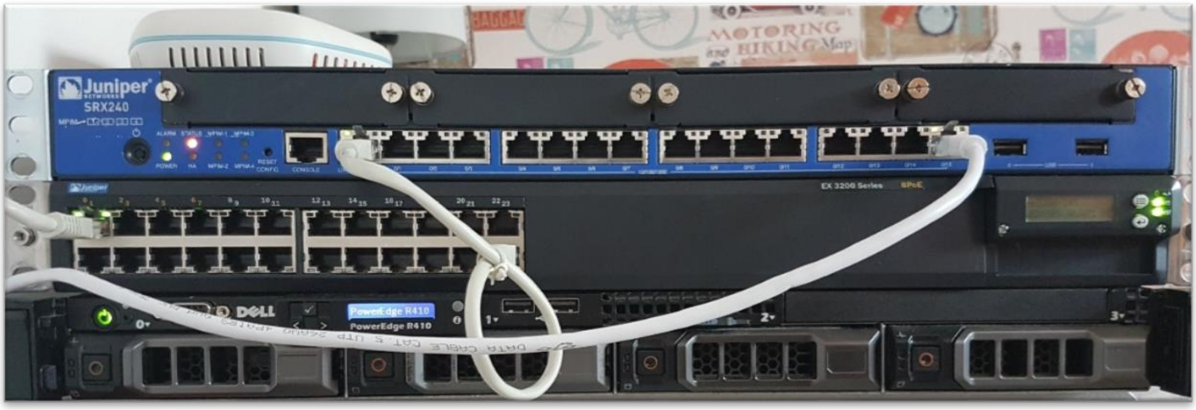
Many different combinations of networking devices, OS versions and Network topologies needed to be tested before completing each large component of the project.

- Below is a sample network configuration used to test the Automated VPN Deployment functionality. By changig different parameters such as remote site public addresses, firewall rules on both the local and remote sites and many other parameters, I was able to determine the network topologies that may potentially break the application.



VPN Test Network Diagram

- This is the network setup I have deployed at home, connected via a site-to-site VPN to the remote customer's network.
- I kept the device models the same as the datacentre network to ensure guaranteed compatibility in the early stages of development. However since, I have added more different devices to the remote network and tested its behaviour.
- Images and details of the remote customer network can be found in my blog.



Local ASDN Network

6. References

Cover Image, (2018), Datacentre [ONLINE]. Available at: http://ccscleaning.com/wp-content/uploads/2017/04/DataCentre.jpg [Accessed 17 April 2018].

Machine Learning Mastery. 2016. Machine Learning Project in Python. [ONLINE] Available at: https://machinelearningmastery.com/machine-learning-in-python-step-by-step/. [Accessed 2 April 2018].

Christine Heckart. 2017. Machine learning: Are we there yet?. [ONLINE] Available at: https://www.networkworld.com/article/3197119/lan-wan/machine-learning-are-we-there-yet.html. [Accessed 23 January 2018].

KDDI R&D Laboratories Inc. 2016. World's first successful AI-assisted automated network operation system PoC. [ONLINE] Available at: http://www.kddi-research.jp/english/newsrelease/2016/022201.html. [Accessed 13 February 2018].

Kenneth Reitz. 2017. XML parsing¶. [ONLINE] Available at: http://docs.python-guide.org/en/latest/scenarios/xml/. [Accessed 14 October 2017].

Hochstein, L., 2015. Ansible Up & Running. 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc.

Smith, S., 2016. Automating Junos Administration. 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc.

Sawtell, S., 2018. Day One: Automating Junos® with Ansible. 1st ed. 1730 Varsity Drive Suite 210 Raleigh, NC 27606: Juniper Networks Inc.

Mauro, A., 2017. Mastering VMware vSphere. 1st ed. Livery Place, 35 Livery Street, Birmingham B3 2PB, United Kingdom: Packt Publishing.