RunBook - Technical Specification

Network Deployment Automation Maintenance Tool

Contents

1.Introduction

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

The Network Deployment Automation & Maintenance Tool has been developed to automate the most common/repetitive task carried out by network engineers.

1.1 Overview

The application is dependent on highly specialised hardware/software to be used to its full potential. The three main areas the program automates are device *deployment* (Juniper Networks Routers, Switches and Firewalls), *maintenance* (version control of configuration files - GitHub like feature) and *troubleshooting* - a network mapping tool.

It communicates with other systems on the Local Area Network using various protocols, such as SSH, FTP, TCP, etc. In order to accomplish this, there are various requirements needed to be satisfied by the user (Network Engineer) in order to use some program features, mainly the deployment module.

1.2 Glossary

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

Console Server – a device containing one or multiple serial ports, allowing it to

interface with other devices using various networking technologies. Mainly deployed as a management device, as it enables the user to monitor and control devices plugged in from a local or remote network.

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

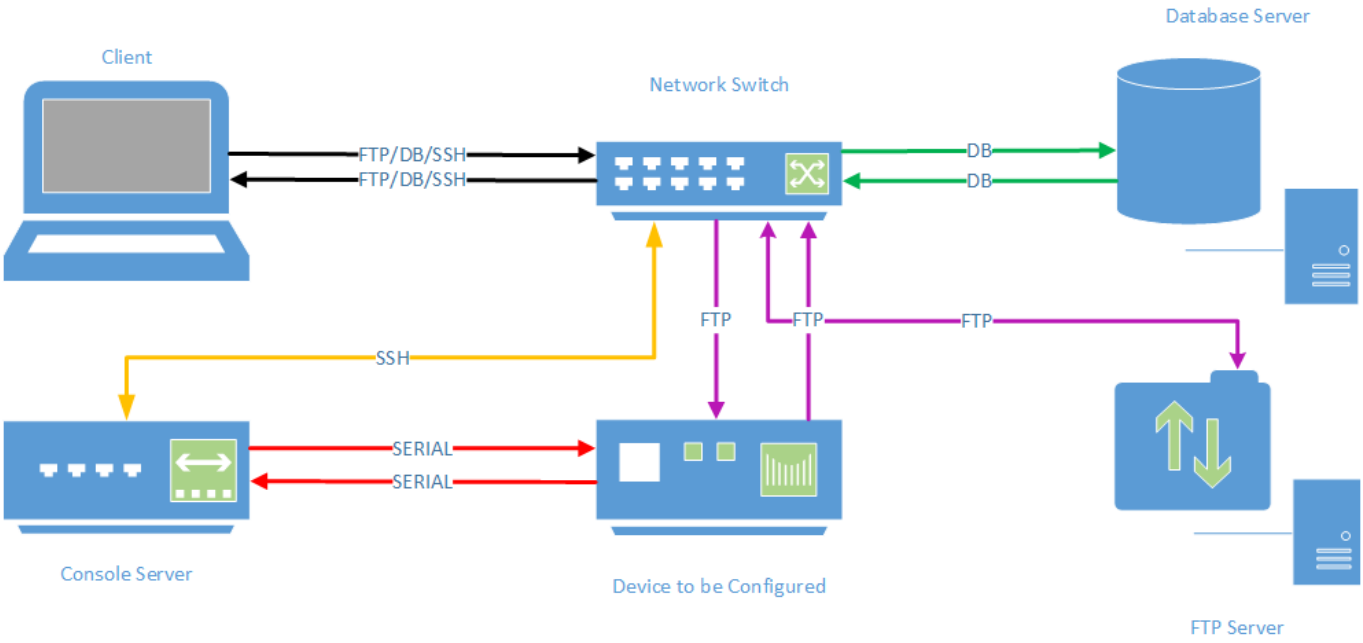
to computer systems by users.

XML – is 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.

2.System Architecture

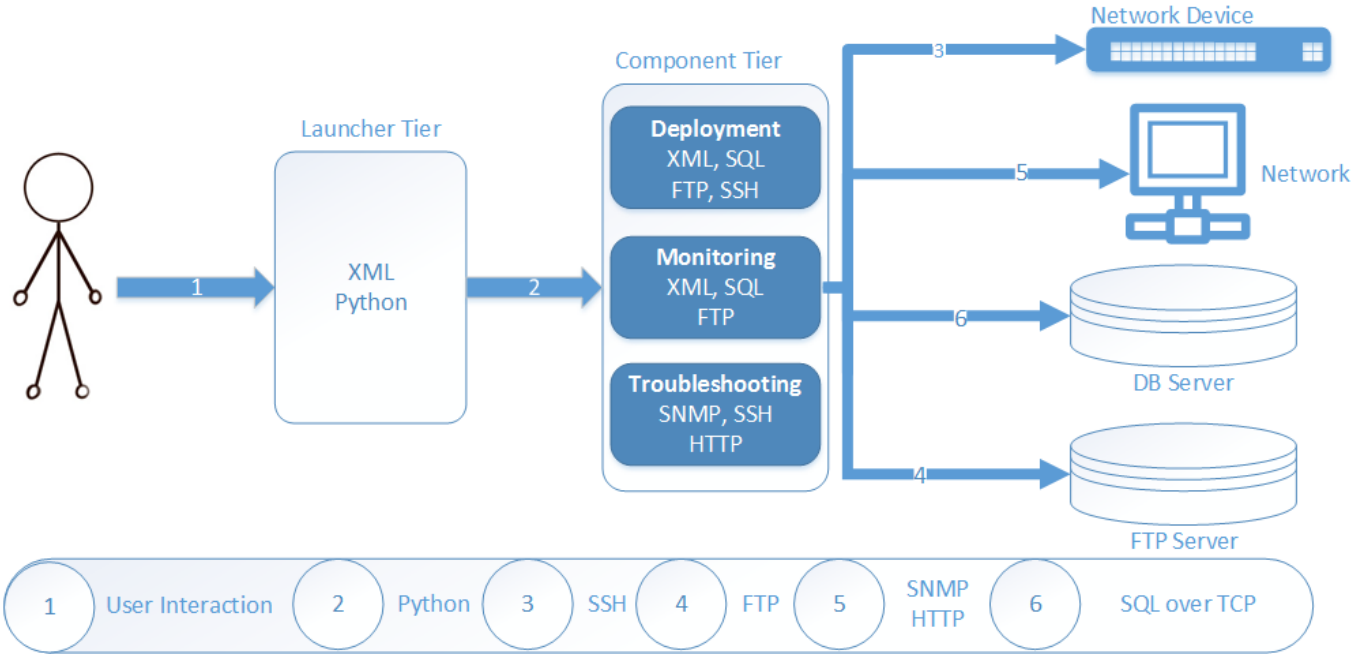
Network Architecture Required:



The system has been split up into **three** main modules:

1. **Deployment**This module is used to automate the deployment of Juniper Networking devices, by automatically updating the device's operating system as well as applying an initial configuration file.
2. **Maintenance**
   This module is responsible for a version control of configuration files applied to devices. It shows the user the lines added and removed from the application file, as well as allowing for traceability of who has applied the changes to the device configuration.
3. **Troubleshooting**
   The main feature of this module is to create a map of a local area network, by representing each device as a node on a graph. It also gives the user the ability to easily check device details such as IPv4 address, MAC address and manufacturer of device.
   Finally, if there is a Juniper device on the network, additional information will be shown such as CPU temperature, RAM usage, device uptime and active alarms on a device.

System Architecture Diagram:



3rd party dependencies used:

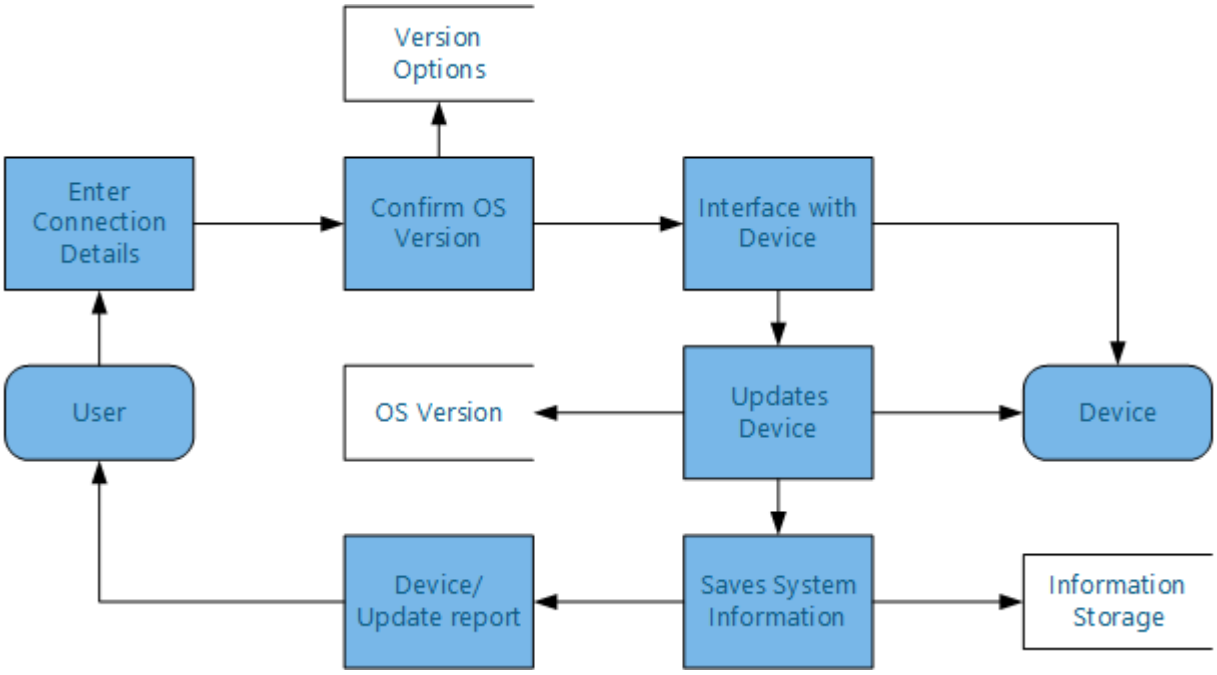
- *xmltodict* – Used to encode and decode xml files while parsing output from a Juniper device.
- *paramiko* – Used to communicate with various devices on the network via the SSH protocol.
- *PyQt5* – A library used to develop the main application user interface.
- *NetworkX* & *Matplotlib* - Used to create and interactive graph in the troubleshooting module, after completing a network scan.
- *Pymysql* – Used to communicate with the database server over the network.

3.High-Level Design

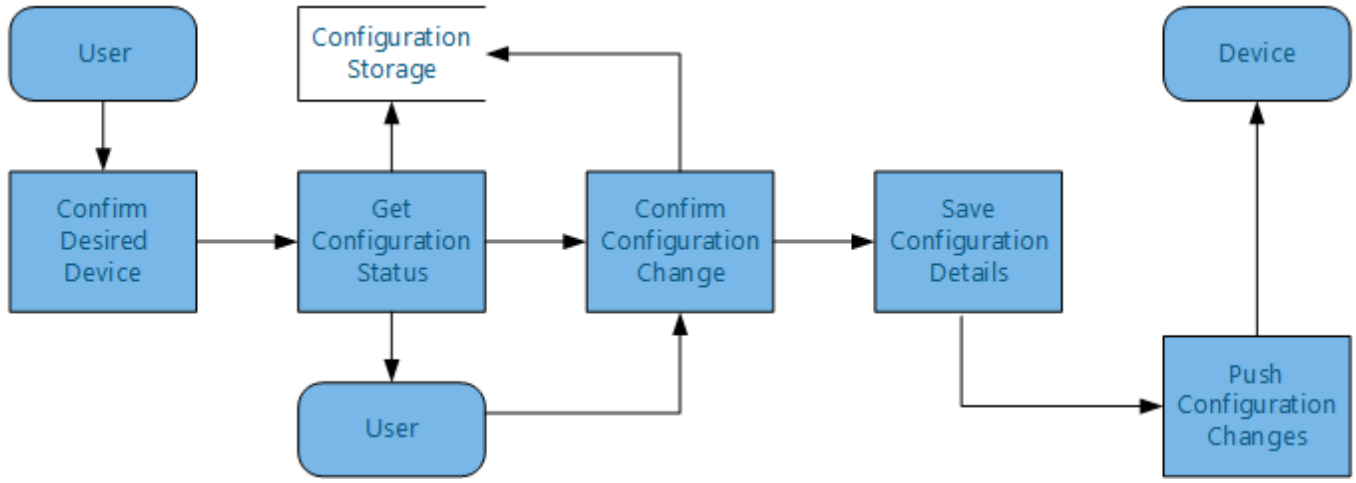
The three modules outlined previously in the document have no direct communication channels, all communication is carried out via the main GUI thread *launcher.py*.

Data Flow Diagrams:

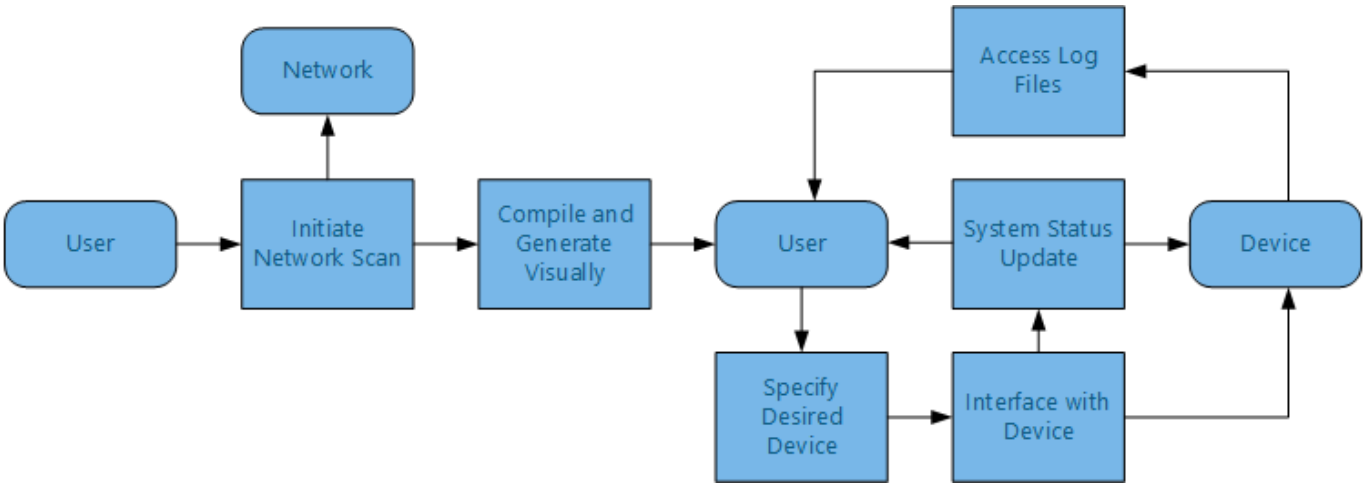
Deployment



Maintenance



Troubleshooting



Constraints:

- JUNOS - while interacting with a device to be configured from the deployment module, all commands sent and recieved are compatible with JUNOS (Juniper Networks Operating System).
- Network Infrastructure - as stated previously in the document, a combination of specialised network hardware/software is required in order to utilise the application to its full potential.
- Security - keeping the application dependency for an internet connection down to a minimum, as one of the main reasons for not using the GitHub platform for version control is the feedback gathered by the companies in the industry with security concerns.

All decisions during the design phase have been to accomodate the constraints stated above.

4.Problems and Resolution

This section should include a description of any major problems encountered during the design and implementation of the system and the actions that were taken to resolve them.

Problems:

1. Ensuring the application has as much cross platform client support as possible (Windows, Linux, MacOS).
2. Taking a list of devices located on a local area network scanned and construct an interactive visual representation from it.
3. Gathering device hardware information (basic/advanced).
4. PyQt5 user interface was a challenge due to the multithreaded nature of the application.
5. The security profile of the application, as it handles very sensitive and valuable data.
6. An efficient way of scanning files for differences, used by the maintenance (version control) module.
7. Creating a simple application deployment and setup process.

Solutions:

1. We have achieved this by using commands and string parsing processes which work correctly for each OS. In addition, we check which operating system the software is running on. Commands and output layouts were always different, making it quite a challenge.
2. This problem required multiple minor problems to be solved.
    1. We have used a plotting library for Python which interacts well with PyQt5 called *Matplotlib.* Allowing us to construct a node graph specified in the Functional Specification.
    2. Making a static graph interactive. As the graph is an image file, overlaid it with a mouse event listener with dynamically determined coordinates. This is connected to a window area, meaning the 0,0 coordinate is in the graph's window. We save the coordinates of each node. Allowing for a click on each node to trigger an event.
    3. The calculations had to be done on a separate thread, therefore, an additional one was needed to compute graph data at the same time. This meant that we had to implement a producer-consumer architecture, whereby a global variable is set, in order to notify the sleeping thread that data is available.

3. Gathering hardware information came in two steps, basic and more advanced which was applicable only to Juniper devices.
    1. The basic information is produced by using the ARP table in order to gather device MAC address live on the network.
       We use this information to extrapolate the device vendor.
       When online we do it by interfacing with an API which is connected to the IEEE database.
       If the user does not have access to the internet, we consult a static page of 26,000+ entries for a device vendor.
    2. The advanced information required Juniper OS detailed knowledge, as well as how to parse the for the Information gathered from a device.
       We had to use XML parsing to achieve this.
4. GUI updates must be done on the main thread, however any heavy calculation/connection related activity needs its own thread to improve efficiency, as well as to avoid application crashes. This also meant that data processed during the time of execution had to be handed back to the main thread.
   This came in two forms, firstly the parent class was passed and then the updated local or global variable.
   This approach required the main thread to be aware of the one carrying out the calculation.
   Mutual exclusion is not an issue, as there was never more than one thread working on a variable.
5. To achieve desired functionality, logging into various devices is required. This means that usernames and passwords are being handled by the application.
   We have minimised our attack surface by prompting the user for a password only at runtime, therefore not storing it anywhere.
   In addition, we have followed best security practises, by requiring password protected accounts to be set up.
   The latest network protocols for communication, such as SSH were used.
6. The maintenance module is used to compare a local configuration file with ones located on the FTP server.
   This required another multithreaded component to be developed. Once both files are available, a compare function will check their contents line by line and add what was deleted and added to two separate arrays.
   IO operations are only done on one file during the comparing process, as the second file has already been pulled down and saved in memory.
7. We have developed an installation process that requires the user to run two install files (Windows) and a single file (Linux).
   This was the only solution we came across, as many of the dependencies we have used do not work correctly when ran as an executable file.

5.Installation Guide

- Please refer to the user manual for a detailed installation guide on how to install our application