Note: *This document describes a customized approach to achieve ZTP-like functionality on Arista cEOS devices. Due to limitations in cEOS, this solution may not align with conventional ZTP methods. Specific issues encountered and troubleshooting steps are documented in the* ***Troubleshooting Attempts*** *section.*

Overview

The purpose of this setup is to automate IP address assignment and device configuration using DHCP and custom scripts. Here, devices obtain IP addresses dynamically via DHCP, and once accessible, a configuration script applies specific settings to each device.

Topology

- **R2**: DHCP server, providing IPs and network information.
- **R4**: Initial DHCP relay agent, forwarding requests from R6 to R2.
- **R6**: DHCP client initially, later configured as a relay agent for R7 after configuration.
- **R7**: DHCP client obtaining IPs relayed by R6.

Objectives

1. Automatically assign IP addresses to R6 and R7.
2. Apply custom configurations to R6 and R7 based on their IP availability.
3. Run the configuration script as a Linux systemd service for persistent automation.

Step-by-Step Configuration

Step 1: Configure the DHCP Server (R2)

1. **Define DHCP Subnets and Address Ranges**: Configure IP address ranges and gateways for each subnet, along with setting options for TFTP and boot files.

```
dhcp server
```

```
   subnet 192.51.0.0/30
      range 192.51.0.2 192.51.0.6
      name VLAN10
      default-gateway 192.51.0.1
```

Step 2: Configure the Relay Agent (R4)

1. **Enable IP Forwarding** on R4 to allow packet relay.
2. **Add the IP Helper Address** to the interface facing R6:

```
interface Ethernet2
   ip helper-address 192.168.100.3  # DHCP server (R2) IP address
```

Step 3: Initial Configuration for R6 as DHCP Client and Relay

1. **Create R6 Configuration File (*r6_config.cfg*)** with initial setup on the NMAS:

```
ip routing

interface Ethernet1
   no switchport
   ip address 192.51.0.2/30

router ospf 1
   network 192.51.0.0/30 area 10
   network 192.51.0.4/30 area 10

ip dhcp relay information option
ip dhcp relay always-on
ip dhcp relay all-subnets default

interface Ethernet2
   no switchport
   ip address 192.51.0.5/30
   ip helper-address 192.168.100.3  # Points to R2
```

```
router rip
   network 192.51.0.4/30
   no shutdown
```

Step 4: R7 Configuration

1. **Create R7 Configuration File (*r7_config.cfg*)** for its setup on the NMAS:

```
ip routing

interface Ethernet2
   no switchport
   ip address 192.51.0.6/30

router rip
   network 192.51.0.4/30
   no shutdown
```

Step 5: Python Automation Script (*ztp.py*)

This script uses the netmiko and loguru libraries to automate device configuration once IPs are reachable.

Script (ztp.py)

```
import time
import os
import threading
from netmiko import ConnectHandler
from loguru import logger

devices = {
      "R6": {
      "device_type": "arista_eos",
      "host": "192.51.0.2",
      "username": "admin",
      "password": "admin",
      "config_file": "r6_config.cfg"
```

```
      },
      "R7": {
      "device_type": "arista_eos",
      "host": "192.51.0.6",
      "username": "admin",
      "password": "admin",
      "config_file": "r7_config.cfg"
      }
}

def ping_until_reachable(device_name, device_info):
      ip_address = device_info["host"]
      config_file = device_info["config_file"]
      with open(config_file) as config_file_obj:
      config_commands = config_file_obj.read().splitlines()
      while True:
      if os.system(f"ping -c 1 {ip_address}") == 0:
            logger.info(f"{ip_address} ({device_name}) is reachable.")
            push_config(device_info, config_commands)
            break
      else:
            logger.info(f"{ip_address} ({device_name}) is not reachable.
Retrying in 10 seconds...")
      time.sleep(3)

def push_config(device_info, config_commands):
      device_connection_info = {k: v for k, v in device_info.items() if k
!= "config_file"}
      connection = ConnectHandler(**device_connection_info)
      connection.enable()
      try:
      connection.send_config_set(config_commands)
      connection.save_config()
      logger.success(f"Configuration applied successfully to
{device_info['host']}.")
      except Exception as e:
      logger.error(f"Error on {device_info['host']}: {e}")
      finally:
      connection.disconnect()
      logger.info(f"Disconnected from {device_info['host']}.")

threads = []
for device_name, device_info in devices.items():
```

```
      thread = threading.Thread(target=ping_until_reachable,
args=(device_name, device_info))
      thread.start()
      threads.append(thread)
for thread in threads:
      thread.join()

logger.success("Configuration process completed for all devices.")
```

Step 6: Set Up the Script as a Systemd Service

To enable the script to run as a service, set up a systemd unit file.

1. **Create the Service File**:

```
sudo vim /etc/systemd/system/ztp.service
```

2. **Service Configuration**:

```
[Unit]
Description=Zero Touch Provisioning for Network Devices with Logging
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/python3 /path/to/ztp.py
WorkingDirectory=/path/to/
Restart=on-failure
User=student
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Reload systemd and Enable the Service:

```
sudo systemctl daemon-reload
sudo systemctl enable ztp.service
sudo systemctl start ztp.service
```

Troubleshooting Attempts

During the initial setup, several issues arose while configuring ZTP on Arista cEOS, especially due to the limitations of ZTP support on this platform and TFTP constraints. Here's a breakdown of these troubleshooting steps:

1. **TFTP Server Configuration Issues**:
    - Attempted setting up a TFTP server on Ubuntu to serve configuration files.
    - Configured DHCP Option 66 to provide TFTP server details and Option 67 to specify the boot file, but cEOS devices did not initiate the expected TFTP request.
    - Verified the TFTP server's port (69) and allowed traffic, yet devices did not retrieve the file.
    - Attempted to manually copy a file from the TFTP server to Arista cEOS. While the file was created, its content did not transfer. Tried all ways to remediate this (*link*), but in vain, nothing worked.
2. **Unsupported ZTP on Arista cEOS**:
    - Attempted to enable ZTP on cEOS using the zerotouch enable command, but received an error indicating that ZTP cannot be enabled interactively.
    - Research confirmed that ZTP is not supported natively on cEOS, requiring alternative methods like DHCP relay and custom automation scripts for configuration. Source: link
3. **Automated Configuration with netmiko**:
    - As a workaround for the lack of ZTP, implemented an automation script (ztp.py) that pings devices until reachable and pushes configuration files once they obtain an IP.
    - This script, coupled with the DHCP relay configuration on intermediate devices (R4 and R6), created a functioning alternative to traditional ZTP.