ICRRD Journal

INTERNATIONAL CENTER FOR RESEARCH AND RESOURCES DEVELOPMENT article

ISSN Number: 2773-5958, SSM Number: (1347468-T), doi.org/10.53272/icrrd, www.icrrd.com

Network Automation: Enhancing Operational Efficiency Across the Network Environment

Anirban Datta[1*], A. T. M. Asif Imran[2], Chinmay Biswas[3]

[123]*Department of Information and Communication Technology, Bangladesh University of Professionals, Dhaka- 1216.*

**Corresponding author; Email: engr.anirban@gmail.com*

Received: 18 November 2022 **Available Online**: 20 March 2023

Revision: 03 December 2022 **Published:** 20 March 2023

Accepted: 09 January 2023 **Volume-4**, **Issue-1**

Cite This: *ICRRD Journal,* 2023, 4(1), 101-111

ABSTRACT: Network automation has evolved into a solution that emphasizes efficiency in all areas. Furthermore, communication and computer networks rely on a platform that provides the necessary technological infrastructure for packet transfer through the Internet using routing protocols. Border Gateway Protocol (BGP) is a standardized gateway protocol that exchanges routing information across autonomous systems (AS) on the Internet. The traditional technique to configure BGP is inefficient compared to the network automation concept. Network automation helps to assist network administrators in automating and verifying the BGP configuration using scripting. This paper implemented network automation using Ansible to configure BGP routing and advanced configuration in the live network environment. This study is focused on automated scripting to configure IP Addresses to the interfaces, BGP routing protocol, configuration backup. Ansible ran the scripting on Network Automation Docker and pushed the configurations to the routers. The network automation controller communicated with other routers via SSH. The findings show that Ansible has successfully deployed the configuration to the routers with no errors. Ansible can help network administrators minimized human mistakes, reduce time-consuming and enable device visibility across the network environment. This study can help network administrators minimized human mistakes, reduce time-consuming and enable device visibility across the network environment. Implementing different types of authentications and hardening process can enhance the network security level for future study.

Keywords: Network Automation, Ansible, BGP, Automation Docker

Introduction

Network automation has become a solution provided today that prioritizes effectiveness in every area. The concept of network automation focuses on automating the management, deployment, testing, and activity of physical devices or virtual networks in a control node with a running program. It assists in performing several tasks, reducing time consumption, and eliminating potential mistakes since all configurations are written and served in a script [1]. Networks of ISPs are now working in ad-hoc basis as it is known as emergency service in all aspects. This is the reason for which lack of standardization

in the process to run a network occurs. Automation will create a standardized process for any network in ISP industry. It will spin-up the configuration of devices. If any a dashboard can be introduced in a common platform, it will decrease the steps for maintenance as well. huge number of mistakes happened due to manual input. As we know manual configuration in time consuming, engineer got stressed. Operational barrier occurs due to higher time consumption. High value tasks can be optimized as well as help to create a good business case. The network automation is implemented using a tool called Ansible. Ansible is agent-less which does not require additional software installed and communication via SSH [2].

Errors in network configuration are rather prevalent, and it might be difficult to diagnose and correct them. Because the configuration of a network is done in a dispersed fashion, faults in the settings of one device have the potential to influence other nodes in the network [3]. In addition, many errors in the initial setup are concealed until the appropriate conditions are met. When the condition of the network changes or when the policies are altered, it is quite simple for mistakes to occur. Problems with the settings, the program, or the hardware can all lead to failures of this kind [4]. As more companies turn to providing their services online, the cost of troubleshooting and repairing network faults rises. When troubleshooting configuration problems, network administrators frequently rely on ad hoc approaches and tools such as ping and traceroute. These problems might arise anywhere among the tens of thousands of lines of code that are spread across several devices. The language used to configure the router is not particularly sophisticated; it lacks high-level primitives and other capabilities that may be helpful in more complicated configurations. The capability of the Border Gateway Protocol (BGP) to accept a broad range of policy implementations leads, in addition, to the potential of wrong settings being applied. Because there is no universally accepted language for setting routers, it is possible that network managers may need to become fluent in many languages in order to keep things working smoothly while utilizing routers produced by a variety of companies. When a user's expectations and the router's actual behavior are at odds with one another, erroneous router settings may result. This is because configuration languages are notoriously complicated. There are many different types of routing faults, some of which include loops, black holes, leaky or unfiltered routes, unstable or hijacked routes, and inadequately filtered routes [5].

Related Works

Network Automation improves the efficiency of configuration network devices using automated scripting rather than the traditional method. Mazin et al. [6] described the performance differences in time between manual configuration and automation using scripting. There were 36 Cisco network devices used in the research with various types of IOS image versions. The study found that implementing automation to configure the devices improved efficiency and reduced the time needed. The results show that the time required is faster and efficient when using automation (120 seconds) than the manual configuration (5797 seconds).

Islami et al. (2020) studied the implementation using network automation in Raspberry Pi to configure network devices. The study used the Ansible tool to complete the tasks. The result concluded that network automation could reduce equipment configuration and maintenance time and reduce human error in configuration syntaxes. Their study indicated that Raspberry Pi has a restriction with the use of Ansible version [1].

Based on the research conducted by Wijaya and his team, the study showed scripting effectiveness in implementing network devices. The method used in this study is by using Ansible as an automation tool to configure the network device, the Ubuntu environment and the CISCO IOS image [7]. However, Ansible supports only Linux and not Windows environments. The study concluded that the network administrator must create a proper infrastructure and implement automation scripting using Ansible without configuring each device.

Ortiz-Garces, Echeverria, and Andrade (2021) proposed a network automation model to harden the campus network. The model consists of three phases focused on the communication protocols, hardening configurations and playbook deployment. The researchers implemented a network automation model using Ansible and Open Shortest Path First (OPSF) for the routing protocol [8]. The hardening process has two levels covered on the rules: network AAA rules, access rules, routing rules and OSPF authentication. Their study found an increment in the percentage of hardening of the campus network.

Routing Protocols are the set of defined rules used by the routers to communicate between source & destination. They do not move the information to the source to a destination, but only update the routing table that contains the information. Network Router protocols helps to specify way routers communicate with each other. It allows the network to select routes between any two nodes on a computer network [9].

BGP is the last routing protocol of the Internet, which is classified as a DPVP (distance path vector protocol). The full form of BGP is the Border Gateway Protocol. This type of routing protocol sends updated router table data when changes are made. Therefore, there is no auto-discovery of topology changes, which means that the user needs to configure BGP manually [10].

Research that has been done in the past has focused on both the testing of data planes in networks and the verification of control planes. Using control plane analysis and verification tools such as RCC, Batfish, ARC, ERA, and Minesweeper, one is able to make assertions concerning network properties such as reachability and consistency [11]. RCC, or router configuration checker, uses static analysis to assess configuration limitations in accordance with a high-level accuracy standard. The goal of this process is to locate mistakes in BGP settings so that they may be corrected. The normalized representation of BGP configurations is accomplished through the utilization of the MySQL database. RCC does not develop a model of the network as other tools do in order to do analysis on the control plane. Batfish uses a subset of Datalog to encode a logical model of the control plane so that the configuration data and semantics of the control plane can be described as logical truths. This is done so that the control plane can be understood. It is possible for Batfish to investigate the data plane that was created from the logical model and the environment that was supplied (i.e. connection status and route announcements from neighbors represented as logical facts). Because Batfish can only reason about one data plane at a time and there are so many different possible settings, it is not possible to use it to build and analyze all of the data planes for a certain control plane [12]. This is because there are so many different combinations of settings. ARC develops an abstract representation for control planes by turning network settings into a set of weighted digraphs that characterize the real forwarding behavior of the network. The ability to reason about the control plane is made possible by computing the graph properties of ARC. Despite this, ARC is merely an abbreviation for a limited portion of the control plane protocols and capabilities. In ERA, a control plane message is described as a 128-bit vector that encapsulates the attributes of the route announcement. Sets of route

announcements are represented as binary decision diagrams (BDDs), which makes it easier to do rapid set operations. The control plane may be conceptualized as a function that accepts a set of routes as input and returns the same set of routes as output. There are some conditions in which it is impossible for an ERA to examine the configurations. Administrators of networks have the ability to define BGP restrictions using Bagpipe. Network traces are used to represent the configurations that are displayed. Bagpipe initially does a network reduction by making use of an SMT-based symbolic execution solver. This is done in order for the system to test the control plane rules for a single AS. Bagpipe only simulates BGP because its primary concentration is on that protocol; hence, it does not model any IGPs [13]. Minesweeper expresses the many potential network configurations through the use of the logical formula N. In a similar manner, network operators are able to express P-properties through the use of logical formulations. Minesweeper makes use of a solver for SMT problems in order to determine whether or not there is a solution to N P, and if there is, it ensures that the aforementioned criteria cannot be met under any circumstances. Because it uses an SMT solver, Minesweeper only ever produces a single solution that breaks the rules of the game's formula. With the assistance of Automatic Test Packet Generation (ATPG), tests for a particular data plane can be created automatically. A network's forwarding capabilities as well as its connection may be tested with the help of this configuration. This tool's objective is to provide the network administrator with assistance in determining whether or not the data plane adheres to the configuration that has been specified [14]. In a similar manner, our tool provides assistance to network operators in determining whether or not the control plane matches to the configuration that is required. When performing control plane testing, rather than using data plane test packets like ATPG does, we instead observe the movement and transformation of routing announcements. OpenConfig (abbreviated as ope) is a community-driven initiative with the goal of standardizing vendor-neutral data representations in YANG [15]. OpenConfig and Batfish have a significant amount of information that is comparable to that which is found in the other application. The standardization of the OpenConfig data models is a goal that lies further out in the future. We were unable to locate a method to build OpenConfig data models using the router parameters that were supplied to us. Because there is presently no standard for OpenConfig's data models, we were forced to construct our IR using the Batfish data model [16].

Methodology

Design and Development

The controller machine can be accessed from anywhere of the world by a secure vpn tunnel. This machine will talk with the network devices via ssh(secure shell) protocol. The implementation starts with an IOS-XR image in live network of any ISP. Figure-1 describes the network topology of this project.

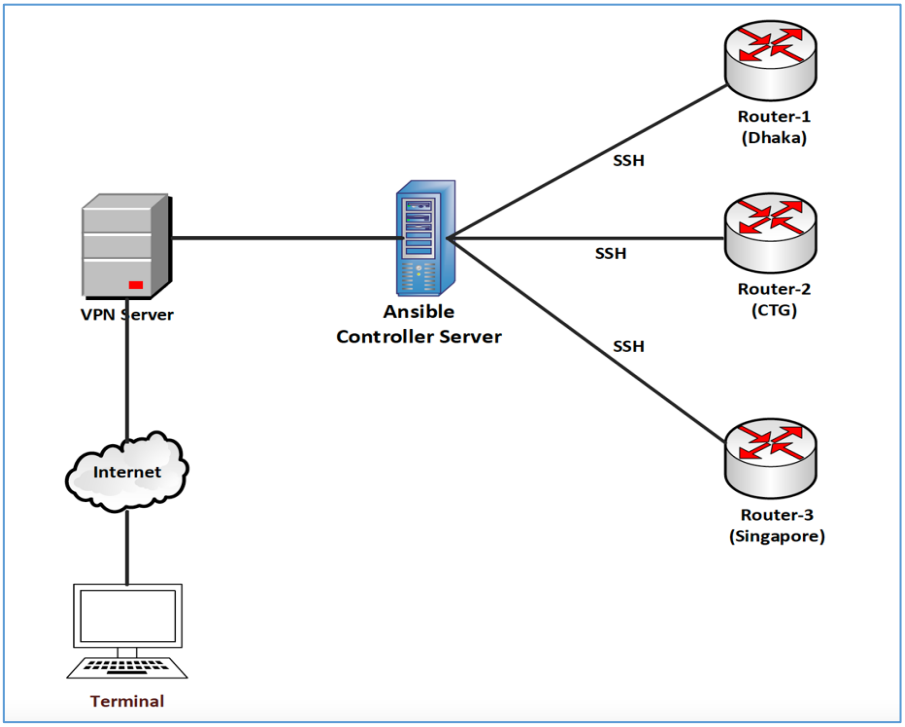

Figure 1: Network Topology

YAML Files

YAML is a human-readable data serialization standard that can be used in combination with all programming languages and is often used to create configuration files.

Preparation of YAML Files

A few YAML records must be made some time recently pushing the scripting on the routers: organize interface setup, ansible hosts and ansible actions. Figure-2 shows the interface configuration of server where ansible is running.

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses: [ 103.170.204.51/24 ]
      gateway4: 103.170.204.49
      nameservers:
          addresses:
              - "8.8.8.8"
```

Figure 2: Server Interface Configuration

Ansible host file contains the inventory list of the routers where Ansible used to determine where the tasks need to perform and communicated through host name by add the IP address which belongs to the router interface. Figure-3 show the host file.

```
[IOS-XR]
router_singapore ansible_host=192.168.1.248 ansible_network_os=iosxr
```

Figure 3: Host File

Ansible Playbook Configuration:

Ansible Playbooks are lists of tasks that automatically execute against hosts which already specified in the host file. One or more Ansible tasks can be combined to form a play. Two or more plays can be combined to create an Ansible Playbook. playbook pull nodes/hosts from inventory, write code there and push to the nodes/hosts through SSH(secured shell). Figure-4 shows work-flow of ansible work-station.

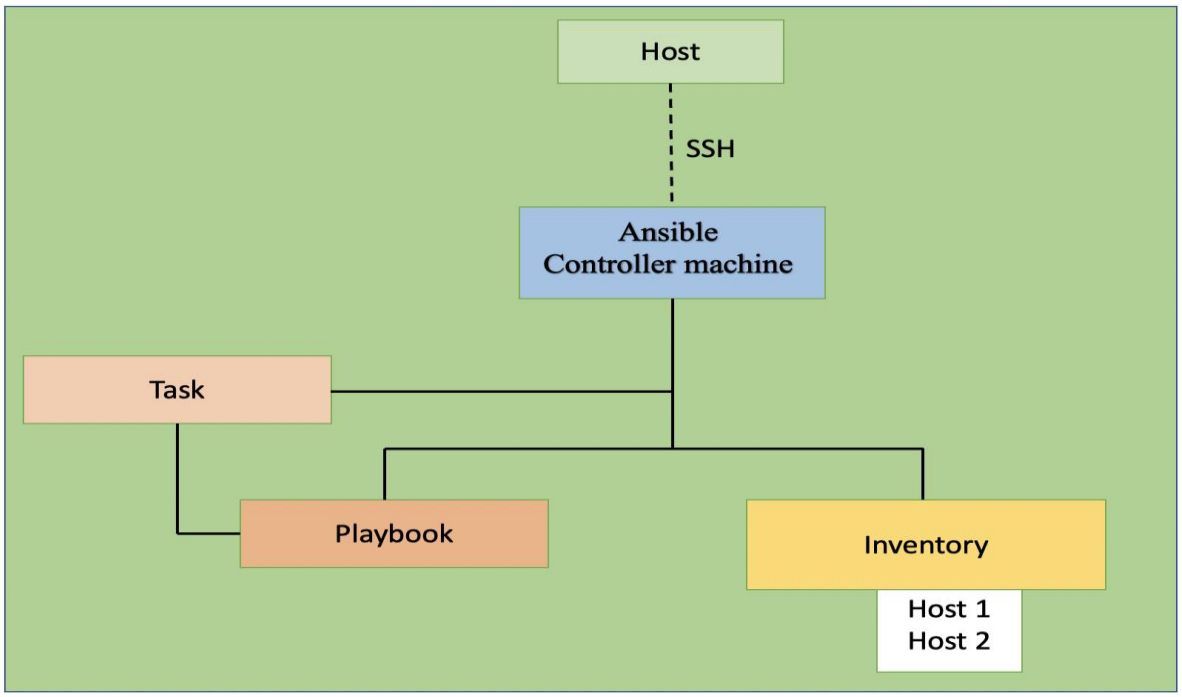

Figure 4: Workflow of ansible workstation

Assessment and Study

We have done three types of assessments in this project which are IP address configuration, BGP configuration and router full configuration backup. At first for ip address configuration in interface loopback101, we have created the playbook script. Figure-5 shows playbook script for interface configuration where ansible will run this playbook by given information. Here the task playbook will run the task in the targeted ios-xr image of Cisco router.

```
- name: CONFIGURE INTERFACE SETTINGS
  iosxr_config:
    provider: "{{ provider }}"
    parents: interface Loopback101
    lines:
      - description test-ansible interface
      - ipv4 address 172.31.1.1 255.255.255.0
      - no shutdown
```

Figure 5: Playbook script for Interface Configuration

Now, for BGP configuration with all the information have inserted in the playbook script. Here we are showing eBGP but both eBGP and iBGP scripts are same.

Figure 6 describes the playbook script of BGP configuration where neighbor peering ip is 172.31.1.2 with peer ASN(Autonomous system number) 58587 also own asn is 139009

```
- name: BGP CONFIGURATION
  iosxr_config:
    provider: "{{ provider }}"
    parents: router bgp 139009
    lines:
      - neighbor 172.31.1.2
      - remote-as 58587
      - description "TEST BGP"
```

Figure 6: Playbook script for BGP Configuration

At the third assessment, we have created the script for keeping backup of full configuration of router.

```
- name: configurable backup path
  iosxr_config:
    provider: "{{ provider }}"
    commands:
      - show run
    backup: yes
    backup_options:
      filename:
      dir_path: /etc/ansible/inventory/backups
```

Figure 7: Playbook script for Router Configuration-Backup

Here we have given the directory path where backup files will be stored. Filenames need to give an input as per requirement. If no name is given in the filename, it will automatically create the filename in the format of hostname/date/time.

Result and Discussion:

Server End:

The playbook which we have created have three tasks those are executed. Tasks are login to the router, define the provider and configure (the interface with given ip address, BGP and configuration backup) of the router. Figure-8 shows the outputs of three playbook outcomes.

Here we can see that in ipconfig.yml file, playbook have done 3 tasks one by one. At first it logged into the router with provided credential. In 2nd task, it found the provider's name which we have given and last task have done by the configuration implemented in the router interface. Moreover, at the end of the recap of playbook, three tasks are showing **ok** and 1 changed have been done which means configuration of interface is been done.

In the bgp.yml playbook, the first 2 tasks are same as before and have been executed in the router. The third task where we have configured the bgp configuration is been is been done. In the third task of BGP configuration, as per our inputs of neighbor ip and ASN, command have been executed in the router which we will find in the router end output result. And of course, three of the tasks are ok and 1 changed was made. There are no failed operation. In the playbook of configuration backup, we have given the command line to execute in the router as well as the directory path where the backup files should be stored. Here, we are observing the full running configuration file of the router is stored the provided directory with file name with date and exact time.

```
root@automation:/etc/ansible# ansible-playbook bgp.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [router_singapore] ********************************************************************************************

TASK [OBTAIN LOGIN INFORMATION] ************************************************************************************
ok: [router_singapore]

TASK [DEFINE PROVIDER] *********************************************************************************************
ok: [router_singapore]

TASK [BGP CONFIGURATION] *******************************************************************************************
changed: [router_singapore]

PLAY RECAP *********************************************************************************************************
router_singapore           : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
root@automation:/etc/ansible# ansible-playbook ipconfig.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [router_singapore] ********************************************************************************************

TASK [OBTAIN LOGIN INFORMATION] ************************************************************************************
ok: [router_singapore]

TASK [DEFINE PROVIDER] *********************************************************************************************
ok: [router_singapore]

TASK [CONFIGURE INTERFACE SETTINGS] ********************************************************************************
changed: [router_singapore]

PLAY RECAP *********************************************************************************************************
router_singapore           : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
root@automation:/etc/ansible/inventory/backups# ls
router_singapore_config.2022-10-25@13:48:16  router_singapore_config.2022-10-29@09:02:10
router_singapore_config.2022-10-25@14:23:13  router_singapore_config.2022-10-29@10:02:11
router_singapore_config.2022-10-25@15:02:10  router_singapore_config.2022-10-29@11:02:11
router_singapore_config.2022-10-25@16:02:11  router_singapore_config.2022-10-29@12:02:10
router_singapore_config.2022-10-25@17:02:11  router_singapore_config.2022-10-29@13:02:10
router_singapore_config.2022-10-27@01:02:10  router_singapore_config.2022-10-29@14:02:10
router_singapore_config.2022-10-27@02:02:10  router_singapore_config.2022-10-29@15:02:13
router_singapore_config.2022-10-27@03:02:11  router_singapore_config.2022-10-29@16:02:11
router_singapore_config.2022-10-27@04:02:10  router_singapore_config.2022-10-29@17:02:11
router_singapore_config.2022-10-27@05:02:11  router_singapore_config.2022-10-29@18:02:11
```

Figure 8: Outcomes of Ansible Playbook

Router End:

Figure 9 and Figure 10 shows the output results of ansible playbook which we ran in the server end. In the Figure 9, the interface configuration have been done successfully with description and the ipv4 address on the interface with subnet of /24. In Figure 10, we can see the BGP configuration with neighbor ip and ASN has been formed with description of the BGP configuration.

```
RP/0/RP0/CPU0:WCL-SNG-NCS#sh run int lo101
Mon Nov 14 17:42:31.860 BDT
interface Loopback101
 description test-ansible interface
 ipv4 address 172.31.1.1 255.255.255.0
!
```

Figure 9: Interface configuration in Router

```
RP/0/RP0/CPU0:WCL-SNG-NCS#sh run router bgp | b 172.31.1.2
Mon Nov 14 17:43:06.688 BDT
 neighbor 172.31.1.2
  remote-as 58587
  description "TEST BGP"
  address-family ipv4 unicast
```

Figure 10: BGP configuration in Router

Security

As ansible is installed and configured in Linux environment, it is needed to be secured. We have used IPTABLES as a firewall in our linux box. In the script of iptables we had to apply these ports for ansible to work with the routers.

-A INPUT -i eth0 -p tcp -m tcp --dport 11371 -m state --state NEW, ESTABLISHED -j

ACCEPT

-A INPUT -i eth0 -p tcp -m tcp --sport 11371 -m state --state NEW, ESTABLISHED -j ACCEPT

Conclusion

Network automation by ansible creates a standardized process for any network in ISP industry. It will spin-up the configuration of devices. As we know ansible is agentless for which it is very simple and human readable YAML templates so users can program repetitive tasks to happen automatically without having to learn an advanced programming language. The purpose of this project was to implement automated network configuration which was deployed and gave us the output successfully. There are some proposals for future research: Implement with other vendors like Juniper, Mikrotik etc. If any a dashboard can be introduced in a common platform, it will decrease the steps for maintenance as well.

In the future, one of our goals is to broaden the scope of our IR so that it can incorporate additional BGP capabilities (such as route reflectors, route aggregation, and so on) as well as additional networking protocols (such as OSPF, ISIS, and so on). This will allow us to create a more complete model of the network. In the future, the study of BGP configurations and rules hopes to achieve the capacity to infer the function that a router plays inside a network. The configurations of comparable routers might then be analyzed thanks to this possibility. Before checking for symmetric behavior, we might be able to further enhance our equivalence testing by performing surgeries on router configurations. These would be performed before checking for symmetric behavior.

CONFLICTS OF INTEREST

There are no conflicts to declare.

REFERENCES

[1] M. Islami, P. Musa, M. L.-J. I. KOMPUTASI, and undefined 2020, "Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI," *ejournal.jak-stik.ac.id*, Accessed: Jan. 25, 2023.

[2] M. Faris, M. Fuzi, K. Abdullah, I. Hazwam, A. Halim, and R. Ruslan, "Network automation using ansible for EIGRP network," *ir.uitm.edu.my*, vol. 6, no. 4, 2021, Accessed: Jan. 25, 2023.

[3] T. Xu and Y. Zhou, "Systems Approaches to Tackling Configuration Errors," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, Jul. 2015, doi: 10.1145/2791577.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM," p. 126, Nov. 2003, doi: 10.1145/958491.958506.

[5] M. Handley, E. Kohler, A. Ghosh, … O. H.-… on S. on, and undefined 2005, "Designing extensible IP router software," *usenix.org*, Accessed: Jan. 25, 2023.

[6] A. M. Mazin, R. A. Rahman, M. Kassim, and A. R. Mahmud, "Performance analysis on network automation interaction with network devices using python," *ISCAIE 2021 - IEEE 11th Symposium on Computer Applications and Industrial Electronics*, pp. 360–366, Apr. 2021, doi: 10.1109/ISCAIE51753.2021.9431823.

[7] J. W.-T. E. dan Informatika. I. Teknologi and undefined 2018, "Network Automation using Ansible for Cisco Router Basic Configuration," *osf.io*, Accessed: Jan. 25, 2023.

[8] I. Ortiz-Garcés, … A. E.-2021 F. W., and undefined 2021, "Automation Tasks Model for Improving Hardening Levels on Campus Networks," *ieeexplore.ieee.org*, Accessed: Jan. 25, 2023.

[9] A. Manzoor, M. Hussain, and S. Mehrban, "Performance Analysis and Route Optimization: Redistribution between EIGRP, OSPF & BGP Routing Protocols," *Comput Stand Interfaces*, vol. 68, p. 103391, Feb. 2020, doi: 10.1016/J.CSI.2019.103391.

[10] D. Tsumak, "Securing BGP using blockchain technology," 2018, Accessed: Jan. 25, 2023.

[11] R. Beckett, A. Gupta, R. Mahajan, … D. W.-S. I. G. on, and undefined 2017, "A general approach to network configuration verification," *dl.acm.org*, vol. 14, pp. 155–168, Aug. 2017, doi: 10.1145/3098822.3098834.

[12] Y. Li *et al.*, "A survey on network verification and testing with formal methods: Approaches and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 940–969, Jan. 2019, doi: 10.1109/COMST.2018.2868050.

[13] K. Weitz, S. Lyubomirsky, S. Heule, E. Torlak, M. D. Ernst, and Z. Tatlock, "SpaceSearch: A library for building and verifying solver-aided tools," *dl.acm.org*, vol. 25, no. ICFP, p. 25, Sep. 2017, doi: 10.1145/3110269.

[14] Y. Tang, G. Cheng, Z. Xu, F. Chen, K. Elmansor, and Y. Wu, "Automatic belief network modeling via policy inference for SDN fault localization," *Journal of Internet Services and Applications*, vol. 7, no. 1, pp. 1–13, Dec. 2016, doi: 10.1186/S13174-016-0043-Y/FIGURES/6.

[15] Y. Zhao, H. Wang, X. Lin, … T. Y.-2017 I. 37th, and undefined 2017, "Pronto: Efficient test packet generation for dynamic network data planes," *ieeexplore.ieee.org*, Accessed: Jan. 25, 2023.

[16] T. Bui, *Understanding Border Gateway Protocol Configurations and Policies*. 2018. Accessed: Jan. 25, 2023.