Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 5, No. 4, November 2020, Pp. 291-298

As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST API and Django Framework

Adian Fatchur Rochim*[1], Abda Rafi Hamaminata[2], Adnan Fauzi[3], Kurniawan Teguh Martono[4]
Department of Computer Engineering, Faculty of Engineering, Diponegoro University, Tembalang, Semarang, 50275, Indonesia[1,2,3,4]

Article Info
Keywords:
Network Automation, REST API, Django, Software Defined Network

Article history:
Received 16 June 2020
Accepted 12 September 2020
Published 30 November 2020

Cite:
Rochim, A., Rafi, A., Fauzi, A., & Martono, K. (2020). As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST-API and Django Framework. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 5(4). doi:https://doi.org/10.22219/kinetik.v5i4.1093

*Corresponding author.
Adian Fatchur Rochim
E-mail address:
adian@ce.undip.ac.id

Abstract
Information technology is very high because of COVID-19 pandemic. Organizations from business through education tend to use this technology most of the time. Information technology uses computer networks for integration and management data. A manageable network configuration for networked devices will be easier to maintain and reduce communication problems. Traditionally, network administrators must configure each network device manually. This process takes time and inefficient. Automated network configuration can overcome the repetitive process, but it is relatively slow. In this research, we propose an alternative model of a network automation system. The model system was implemented with a controller application that used REST API (Representational State Transfer Application Programming Interface) architecture and built by the Django framework with Python programming language to increase the performance of network automation. The design model, called the As-RaD System, uses a web-based application for maintenance and automates networking tasks with easy GUI. The network devices used in this research include the Cisco CSR1000V because it supports REST API communication to manage its network configuration and could be placed on the server either. The As-RaD System provides 75% faster performance than Paramiko and 92% than Network Automation and Programmability Abstraction Layer with Multivendor.

1. Introduction

In this era, computer networks have become dynamic and complex [1]. The availability and reliability of network devices then become a challenge for computer network providers. To configure network devices, network engineers use a well-known tool as a secured shell (SSH). However, manual configuration is time-consuming because repetitive tasks, i.e., login and logout, entry user, and passwords, are done for every device.

Network automation with an application programming interface (API) can reduce the time and repetition of network maintenance [2]. The tasks include monitoring the network to prevent vulnerability [3]. The automation network can modify static and dynamic routing; it can also be used to configure users [4]. Therefore, we can say that network automation uses programming logic to manage network devices so that network administrators can configure network devices automatically [5].

Network automation (NA) uses the Python programming language [6]. Paramiko and NAPAL implement the NA concept that was coded by the Python language. Paramiko is a Python implementation library of SSH protocol and could provide NA [7]. The Network Automation and Programmability Abstraction Layer with Multivendor (NAPALM) support is a Python library that implements a set of functions to interact with different router vendor devices using a unified API [6]. The REST API (Representational State Transfer Application Programming Interface) recently became popular in network protocol design [8]. REST in the development of computer networks is an architecture that allows applications to send configurations to other applications, which in this case, are virtual computer network devices [9]. A Python script is still needed to enable NA; thus, to improve network administration, it is necessary to develop web-based applications that have a display or GUI and can be accessed centrally [10].

Rheza et al., in 2014, explained how an application could be designed using the Python programming language to automate network device administration such as routing and backup restore device configuration. This approach could reduce the complicated and repetitive tasks of a network administrator [11]. Zhou et al., in 2014, explained various issues regarding the RESTful protocol for computer network design needed with programming approaches and how the HTTP protocol can be used to control a computer network device with the advantages of RESTful [8]. Mihaila, in 2017, made comparative automation comparisons using several methods between the NAPALM, Netmiko, and Paramiko methods [12]. He demonstrated each method of configuring network devices. However, a comparison of the time to implement each method into a framework is unclear.

The research questions are as follows: 1) how to create a framework for NA; 2) how to compare the performance of the proposed framework with existing methods, i.e., NAPALM and Paramiko. Both methods take a long time for access to network devices [13][14]. The focus of the research question is time-consuming to access, configure, control, etc. the network devices based on NA. This research question is important because the longer the time required executing one command on a network device, the greater the impact on the time required for subsequent commands for other devices in a network infrastructure. Thus, there is an overall negative impact on the whole institution that is currently using the network infrastructure.

Through this paper, the authors proposed: 1) How to make a system to decrease time-consuming to configuring network devices, 2) building an application to control network configuration with the Django framework, and 3) comparison of performance testing NA framework based on time requirements.

There are four sections in this paper. The first section describes the background and purpose of NA using REST API and previous research. The second section describes the methodology used. The third section describes the results and discusses system implementation. And finally, the conclusion is presented in the fourth section.

2. Research Method

The research starts with a review of the literature from several journals and books. Developing NA relies on the Design Science Research methodology [15], which consists of system definition, system specifications, system configuration, evaluation, and results.

The first step is defining the system. This step includes identification of system requirements, the goal and benefit of the system, how the system works, and the network topology used. The next step identifies the requirement specifications to match the system definition. The third step is system configuration. This step aims to design specifications for predetermined requirements. These requirements cover the network topology or design that is applied as a whole system or system unit that makes the system run.

The final step is to evaluate the system implementation and makes performance comparison with other methods to get the percentage of effectiveness of the REST API method. The proposed design model is the NA system based on the REST API and Django framework (As-RaD System). Figure 1 illustrates each step of the research methodology.

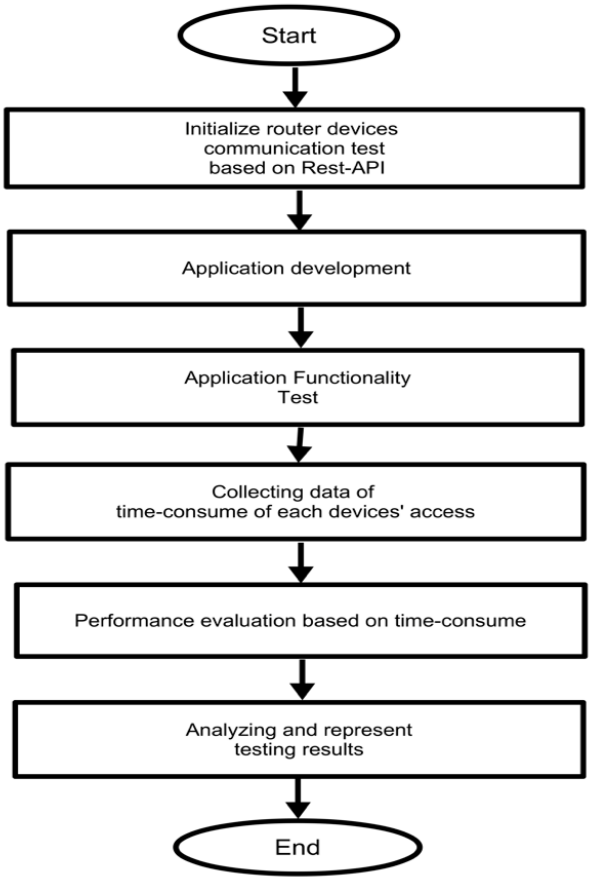

Figure 1. Flowchart System

2.1 Functional and Non-functional Requirements

The system's functional requirements include a controller that could automate common network management tasks, such as adding Internet Protocol (IP) Address, static route, dynamic route (OSPF and BGP), custom configuration, exporting the syslog, and validating the configuration. While the non-functional requirement of the NA system is a controller in the form of an application system that has a user-friendly and responsive interface, it can be run on several types of browsers and operating systems either.

2.2 Hardware Requirements

The NA controller is implemented on a server by a cloud provider, such as the specifications shown in Table 1.

Table 1. Instance Specification

Component	Server
Instance Type	Amazon EC2 T2 Micro
CPU	1 x Intel Xeon vCPU 2.20 GHz
RAM	1024MB DDR4
Storage	1 GB Amazon EBS

2.3 Software Requirements

The As-RaD System requires several software tools to build an application controller and other components. IOS XE as the operating system of the Router used to support REST API architecture communication [16]. A network automation controller application is made to implement As-RaD System in a tested model. The application is developed using the Python programming language and the Django framework [17]. Django framework uses the Model View Template to build a web application faster because it has several built-in templates [18].

The other software used was Ubuntu as a server, Python 3.7.5 for application development, Django 3.0.3 as a web framework, AWS Cloud [19][20] as the cloud system, and Pop OS 19.10, based on Debian Linux, as the operating system used in system development.

Three units of Cisco CSR1000V Routers [16] as the devices connect to the NA system. One laptop was used to configure the system and create the application.

2.4 Topology Design

The network device works with cloud-based virtual computing that supports the scenario for testing the model. The test includes automatic configuration features and validation from the topology created. Figure 2 shows the topology used for this research.

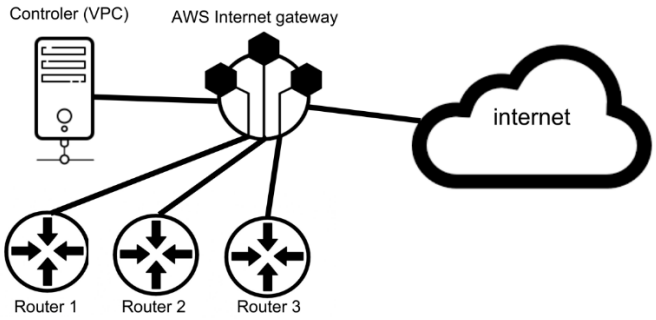

Figure 2. Network Topology

Virtual Private Cloud [21] is used as a local network area in the cloud system to put the components to be tested. These components include the NA controller, three Cisco CSR1000V Routers, and an Internet Gateway. Figure 3 illustrates the As-RaD System design.

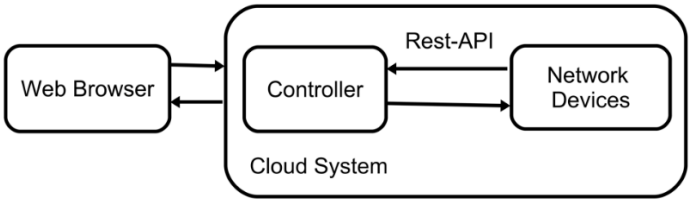

Figure 3. Network Automation System Design

3. Results and Discussion

This chapter discusses system evaluation, such as communication testing with the device, implementing the controller, testing the controller's application system, comparing the As-RaD System with other methods, and discussing the application development of the controller.

Details of the As-RaD NA process to communicate and configure the devices are presented as a flowchart in Figure 4.

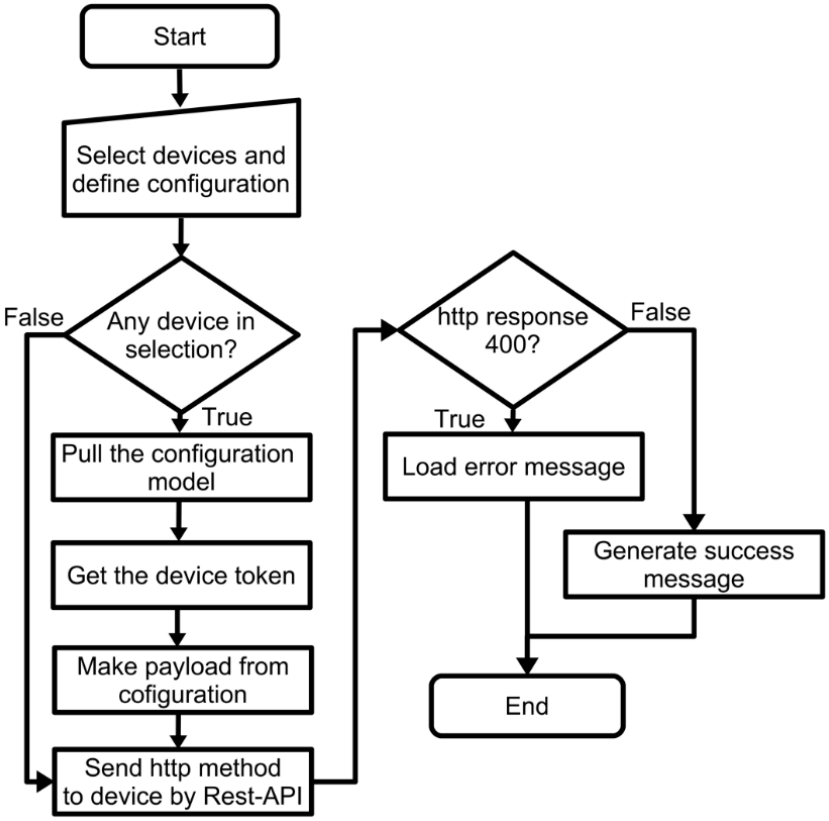

Figure 4. Network Automation Flow Chart

The following is the pseudocode of the As-RaD System NA process:
Input: configuration parameter
Output: configured device
1: select devices to configure
2: input configuration parameter
3: for each device in selected devices do
4: pull the configuration model
5: get token of the device
6: make payload from the configuration model
7: send payload to devices through http methods
8: if devices http response > 400
9: load error message
10: else
11: generate a success message
12: endif
13: endfor

3.1 REST API Evaluation

Using cURL [22] An HTTP Method request is tested for each API endpoint [23] to make sure the established connection using the REST API architecture with a Cisco CSR1000V Router. Table 2 shows the communication of each request made. Each HTTP method is sent with JSON Properties, following client-requested parameters [24].

Table 2. REST API Requests

Configuration	HTTP Method	Endpoint	Response
Add IP Address	PUT	https://ip_address:55443/api/v1/interfaces/	204 No Content
Static Route	POST	https://ip_address:55443/api/v1/routing-svc/static-routes	201 Created
OSPF Route	POST	https://ip_address:55443/api/v1/routing-svc/ospf/ospd_id/networks	201 Created
BGP Route	POST	https://ip_address:55443/api/v1/routing-svc/bgp/bgp_id/networks	201 Created
Validate Configuration	PUT	https://ip_address:55443/api/v1/global/cli	200 OK
Export Syslog	GET	https://ip_address:55443/api/v1/global/syslog	200 OK
Custom Configuration	PUT	https://ip_address:55443/api/v1/global/cli	200 OK

3.2 Home Dashboard Evaluation

The Home Dashboard displays devices registered to the controller. This dashboard also displays the last event that occurred in the controller application, such as the time of the automation, the target device's IP address, automation status, and the user who did the automation.

Figure 5 shows the results of the Home Dashboard test, which shows the controller doing an activity/event in the form of an experimental configuration of the registered device. The information obtained shows NA success.

3.3 Configuration Menu Evaluation

The Configuration Menu offers options for performing NA actions on network devices such as Add or Update IP Addresses, Static Routes, Dynamic Routes, and Custom Configuration. Users are given a form to fill in the parameters of each configuration they need. Each configuration has its own parameters (i.e., updating IP address on the registered devices could be done simultaneously applies to all devices with each of their own configuration contained in this menu).

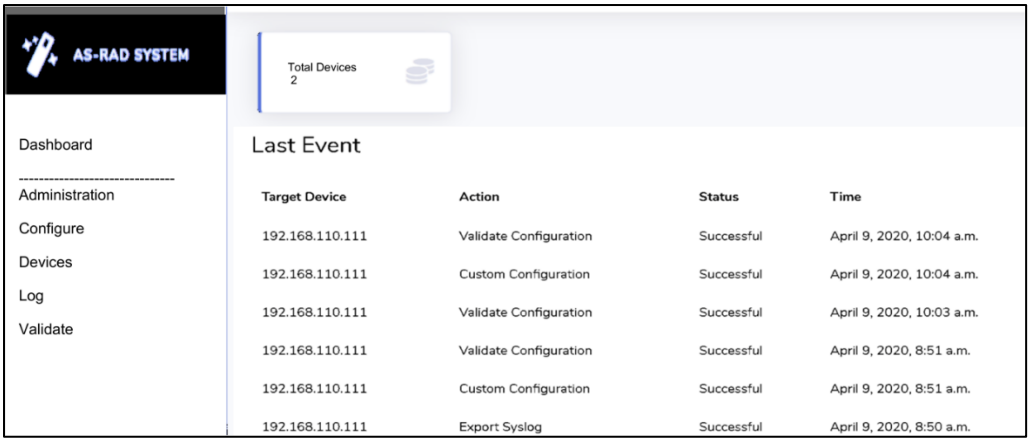

Figure 5. Home Dashboard

3.4 Log Menu Evaluation

This Log Menu offers users to view the activities of the controller application, which is configuration information. Besides, there is a menu for exporting syslog generated by network devices. The detailed log shown information related to the configuration made by the user so that the configuration has occurred.

3.5 Application Evaluation

Application evaluation explains the results and workflow of the controller application that has been developed. The initial work process contained in the application is the login page where the user is authenticated to enter the system. After authentication, users will be redirected to the main page that contains the Dashboard Home. On the Home

Dashboard, there is the last event that occurs in the application. This page also has a navbar that presents options to configure, view devices, log, and configure validation.

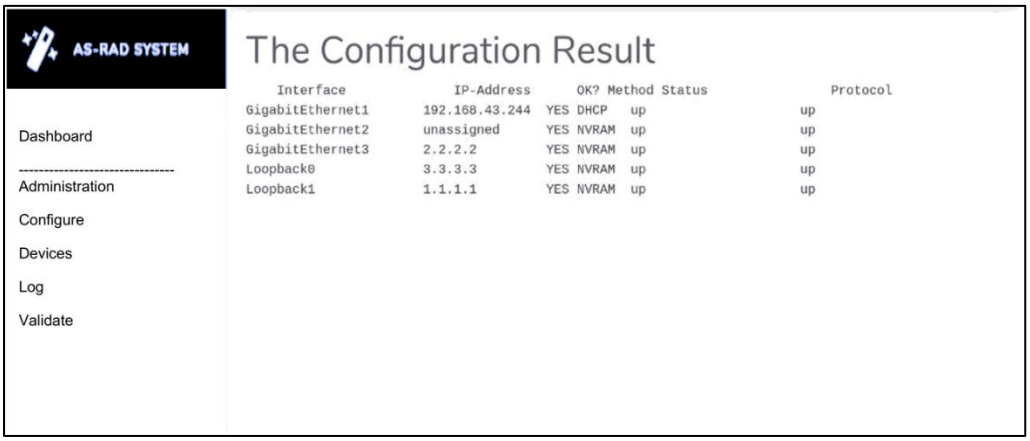

Figure 6. Validation Menu

The configuration validation menu displays the results of the configuration applied to network devices by entering command-line commands. Figure 6 shows the configuration results in the configuration validation menu, showing the device has been configured successfully on the network device.

3.6 Performance Evaluation

To perform a performance evaluation test of the As-RaD System, a comparison of the execution time scale taken for each method in implementing NA. The two methods applied in the previous research (Paramiko and NAPALM) were taken to provide comparative variables with the methods applied. Data collected from the execution time to complete network automation in the time scale of the millisecond using the As-RaD System, Paramiko, and NAPALM. Retrieval of data is to determine the performance of the methods applied.

The test scenario is to add an IP address to the loopback interface and be applied to the same topology. From testing 100 times, the dataset obtained was filtered using the Peirce classification to eliminate data outliers that affect the logging of actual performance [25].

After having datasets, the next step is to compare the performance of each method. The data visualization shown in Figure 7 shows the comparison of the three methods tested. The As-RaD System can complete execution time faster than NAPALM and Paramiko.

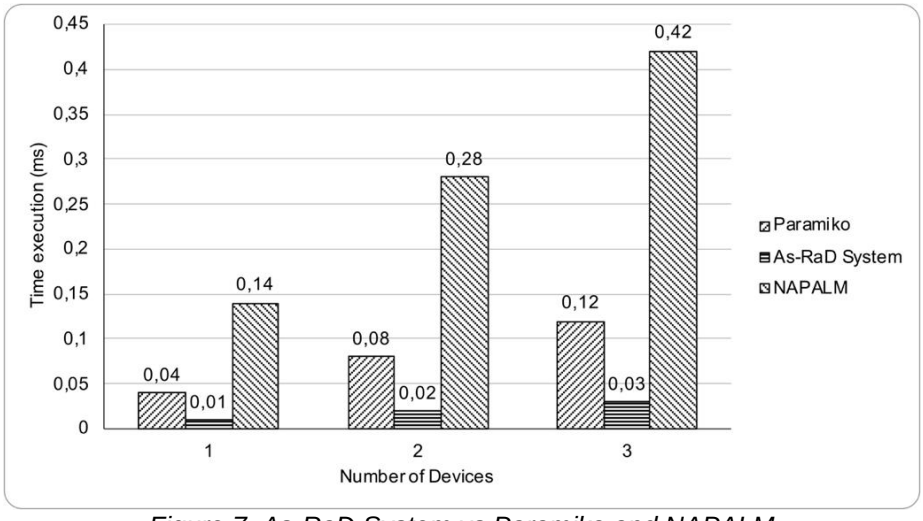

Figure 7. As-RaD System vs Paramiko and NAPALM

From the results obtained in the graph shown by Figure 7, a calculation is made to find the percentage of the results of the comparison of the As-RaD System with the Paramiko and NAPALM methods. The Equation 1 approach is as follows.

$$Result = \frac{Time\ Method - As\_RaD\ System}{Time\ Method} \times 100\ \% \quad (1)$$

The result is that the As-RaD System is 75% faster than Paramiko and 92% faster than NAPALM. From the result, we conclude that As-RaD has better performance to apply network automation based on time execution compare to the previous methods applied (Paramiko and NAPALM).

4. Conclusion

As-RaD System has the web-based application that is used to automate Cisco CSR1000V devices. This application applies the REST API method that can communicate with computer network devices, especially those that already have REST API architecture support. When communicating using the REST API, which in this case is network device management, this method will return data in the JSON (Javascript Object Notation) [26] format. This is different from the command-line interface, which returns information that is intended for humans [27] but not for an application. In this work, we obtained results of network automation tests using the As-RaD System has a faster performance of 75% compared to the Paramiko method and 92% faster than the NAPALM method in completing network management tasks.

Acknowledgment

This research was financially supported by The Faculty of Engineering, Diponegoro University, Semarang, Indonesia, through Strategic Research Grant 2020 number: 2496/STK05/UN7.5.3.2/PP/2020.

References

[1] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, 2013. https://doi.org/10.1109/MCOM.2013.6461195
[2] J. Thielens, "Why APIs are Central to a BYOD Security Strategy," *Netw. Secur.*, vol. 2013, no. 8, pp. 5–6, 2013. http://dx.doi.org/10.1016/S1353-4858(13)70091-6
[3] A. F. Rochim, M. A. Aziz, and A. Fauzi, "Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack," *ICECOS 2019 - 3rd Int. Conf. Electr. Eng. Comput. Sci. Proceeding*, pp. 338–342, 2019. https://doi.org/10.1109/ICECOS47637.2019.8984494
[4] A. P. Paramitha, A. F. Rochim, and A. Fauzi, "Design and Implementation Network Administrators Account Management System Based on Authentication, Authorization, and Accounting Based on TACACS and LDAP," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 803, no. 1, 2020. https://doi.org/10.1088/1757-899X/803/1/012040
[5] A. Ratan, *Practical Network Automation*, Second. Mumbai: Packt Publishing Ltd, 2018.
[6] E. Chou, *Mastering Python Networking*. Packt Publishing, 2018.
[7] B. Rhodes and J. Goerzen, *Foundations of Python Network Programming, Third Edition*, Apress, 2014.
[8] W. Zhou, L. Li, M. Luo, and W. Chou, "REST API Design Patterns for SDN Northbound API," *Proc. - 2014 IEEE 28th Int. Conf. Adv. Inf. Netw. Appl. Work. IEEE WAINA 2014*, pp. 358–365, 2014. https://doi.org/10.1109/WAINA.2014.153
[9] F. Mehmood, I. Ullah, S. Ahmad, and D. H. Kim, "A Novel Approach Towards the Design and Implementation of Virtual Network Based on Controller in Future IoT Applications," *Electron.*, vol. 9, no. 4, 2020. https://doi.org/10.3390/electronics9040604
[10] A. Rohyans, *et al.*, "Cisco SD-WAN Cloud Scale Architecture," Cisco, 2019.
[11] R. A. Wiryawan and N. R. Rosyid, "Website-Based Network Administration Automation Application Development Using the Python Programming Language," *Simetris J. Tek. Mesin, Elektro dan Ilmu Komput.*, vol. 10, no. 2, pp. 741–752, 2019. https://doi.org/10.24176/simet.v10i2.3589
[12] P. Mihaila, T. Balan, R. Curpen, and F. Sandu, "Network Automation and Abstraction using Python Programming Methods," in *MACRo 2015*, 2017, vol. 2, no. 1, pp. 95–103. https://doi.org/10.1515/macro-2017-0011
[13] Snatch, "NAPALM's Initial Connection is to Slow," *Github*.
[14] Sturmf, "Performance of Paramiko is Pretty Slow," 2013.
[15] R. Hevner Alan, "A Three Cycle View of Design Science Research," *Scand. J. Inf. Syst.*, vol. 19, no. 2, pp. 87–92, 2007.
[16] S. Lota and M. Markowski, "Performance Analysis of Virtual Computer Network Based on Cisco Cloud Services Router 1000V in a Private Cloud Environment," *J. Appl. Comput. Sci. Methods*, vol. 7, no. 2, pp. 117–132, 2016. http://doi.org/10.1515/jacsm-2015-0013
[17] O. Karnalim and M. Ayub, "The Use of Python Tutor on Programming Laboratory Session: Student Perspectives," *Kinetik*, vol. 2, no. 4, pp. 327−336, 2017. https://doi.org/10.22219/kinetik.v2i4.442
[18] Django Software Foundation, "Design Philosophies," 2017.
[19] K. R. Jackson, *et al.*, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*, pp. 159–168, 2010. https://doi.org/10.1109/CloudCom.2010.69
[20] S. I. Abrita, M. Sarker, Faheem Abrar, and M. A. Adnan, "Benchmarking, VM Startup Time in the Cloud," in *Lecture Notes in Computer Science*, pp. 53–64, 2018. https://doi.org/10.1007/978-3-030-32813-9_6
[21] O. Hrebicek and L. Chung, "Virtual Private Cloud that Provides Enterprise Grade Functionality and Compliance." Google Patents, 2014.
[22] M. T. Jones, "Conversing Through the Internet with cURL and libcurl," pp. 1–10.
[23] D. Jacobson, D. Woods, and G. Brail, *APIs: A Strategy Guide*. O'Reilly Media, 2011.
[24] M. Amundsen, *RESTful Web Clients: Enabling Reuse Through Hypermedia*. O'Reilly Media, 2017.
[25] S. M. Ross, "Peirce's Criterion for the Elimination of Suspect Experimental Data," no. September 2003, 2016.
[26] C. Severance, "Discovering JavaScript Object Notation," *Computer (Long. Beach. Calif).*, vol. 45, no. 4, pp. 6–8, Apr. 2012. https://doi.org/10.1109/MC.2012.132
[27] B. Claise, J. Clarke, and J. Lindblad, *Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI*. Pearson Education, 2019.