Automation of Network Device Configuration Using Zero-Touch Provisioning - A Case Study

Ivan Šimunić and Ivan Grgurević(✉)

Faculty of Transport and Traffic Sciences, University of Zagreb, Vukelićeva 4, 10000 Zagreb, Croatia
ivan.simunic@combis.hr, igrgurevic@fpz.unizg.hr

Abstract. The installation of new network devices in the production environment represents a process of several steps. First of all, it is necessary to connect the hardware and then define all the necessary global and local parameters by using the Command Line Interface (CLI) and so for each device in the implementation separately. With the development of automation models within the concept of Software Defined Networks (SDN), the Zero Touch Provisioning (ZTP) occurs as a potential solution to reduce the complexity of installation of a new network infrastructure. The ZTP model can be used to reduce time and costs of installing new devices in the network. This paper presents a case study of implementing the ZTP model on the example of Cisco CSR1000v network device in the local area network. The initial step was the creation of the network environment in the emulation software Emulated Virtual Environment Next Generation. In the network environment the necessary servers were defined and the device that supports the ZTP model was assigned. Then it was necessary to create a script in the Python programming language and upload it to the appropriate server. At startup the device downloads the script from the server. The script is then executed, and as the result of this process, the appropriate configuration on the device is applied. After the implementation of the ZTP model and review of the results, the potential that enables such a solution in networks of much larger scales is evident. The application of this model would certainly simplify the process of installing new devices on the network. The applied ZTP model and the created one's own script have proven significant saving in the installation time (over 95%), at the same time achieving also a saving in the costs of employees, and affecting a reduction in human error. The results of this paper can contribute to better understanding of the process of implementing the ZTP model with the aim of its more widespread application.

Keywords: Zero-touch provisioning · Software defined networking · Cisco IOS XE

1 Introduction

Computer networks today support significantly different IT environment compared to several years ago. The network is becoming the critical component in the overall infrastructure of the company, and the end users expect it to follow the technological evolution

D. Perakovic and L. Knapcikova (Eds.): FABULOUS 2021, LNICST 382, pp. 105–119, 2021.
https://doi.org/10.1007/978-3-030-78459-1_8

and support the constant growth of the network requirements [1]. The configuration of such networks is usually a complex and time-consuming process, and automation models and response mechanisms are virtually non-existent [2]. Although the network experts have managed to adapt the existing networks to new challenges by continuous optimization, implementing new approaches in network design and introducing new features, computer networks are still based on traditional and inflexible infrastructure [3].

The idea of Software Defined Networks (SDN) is being developed as a potential solution to these problems. SDN provides a programmable network infrastructure with centralized approach which would enable automation of end-to-end services in the network, such as customer segmentation, quality of service, and analytics [4].

Furthermore SDN provides the possibility of automation of the user policy for appropriate access control and of the user experience quality when using applications [5]. Also, within the SDN concept, the model of configuration automation is being developed when installing new devices on the network (Day-Zero Automation). By introducing automation and enabling multipoint auto configuration of network devices, the Zero Touch Provisioning (ZTP) concept emerges as a possible alleviation of the complex network provisioning and infrastructure services deployment process.

The aim of this research is to automate the configuration process of Cisco IOS XE network device when implementing the device into the network according to the ZTP model. The device automation will be based on downloading the configuration script from the appropriate servers which will then be executed on the device and with the help of the configuration file, also located on a certain server, perform the configuration. The implementation of this research will require the establishment of a laboratory with appropriate network devices and servers and the development of a script in the *Python* programming language. The purpose of this research is to use the process of ZTP model implementation in an environment founded on the existing solutions, to present all the benefits whose application would significantly reduce the network installation time for the network experts (research focus), reduce the financial costs and the likelihood of human error.

The remainder of the paper is organized as follows. Section 2 deals with the related research. Section 3 shows the Zero-Touch Provisioning overview. Section 4 describes the environmental development for the Zero-Touch Provisioning model, such as network topology and Python script development. Section 5 shows the results of implementing the Zero-Touch Provisioning model. The paper concludes with the remarks on the future development in Sect. 6.

2 Related Research

The configuration automation has the potential of significantly reducing the device implementation time, but also the costs that occur during such a process. In the world of technology today, the method of automating the provisioning of devices is becoming more and more common. Thus, the term ZTP has been increasingly appearing. ZTP is a term that includes an automation solution developed to reduce the likelihood of error and save administrators time in implementing new devices. This term has been present for some time in the networking world in the domain of the service provider. The author

[6] in his paper investigates and proposes multi-service procedures for activating the ZTP procedure for telecom operating operators using network virtualization.

Often this process has been used precisely in cases of providing a large device when maintained efficiency is required, not referring to a single network. Thus, the authors [7] described the examples of ZTP methodology in the context of providing IoT devices. The author of this paper states that ZTP is an appropriate solution to reduce the probability of error and reduce the time required to train new devices and the proposed conclusion about the advantages.

Paper [8] explores the options for the tactical deployment of network automation with NetZTP to provide maximum return on investment as organizations explore strategic approaches to network automation.

In paper [9] the authors present the results of the ongoing development of the Cloud Services Delivery Infrastructure (CSDI) that provides a basis for infrastructure centric cloud service provisioning, operation and management in multi-cloud multi-provider environment defined as a Zero Touch Provisioning, Operation and Management (ZTP/ZTPOM) model.

The authors in [10] in defining a new Meto-Haul architecture based on the SDN concept and network virtualization incorporate ZTP functionality as an appropriate solution to enable the installation and provisioning of new hardware automatically. In addition, this functionality is cited as an integral part of their demo.

Research like the one from Cisco suggests that Day-Zero Automation can reduce the implementation costs by as much as up to 79% [11]. Also, due to lower probability of configuration failure it is possible to reduce the time spent in finding the problem by 50%. Besides, 70% of all network errors and 35% of total time during which the network was not functional were caused by human inconsistency in configuration which could be significantly reduced by introducing automation models for the implementation of network devices [11].

ZTP is a model developed within the SDN concept and represents a solution for automating the configuration of new network devices. Several authors have evaluated the possibilities of the ZTP model in various environments and highlighted its advantages. In research [12] the authors state the advantages of the ZTP model in configuration automation, where the authors emphasize as a special advantage the possibility of configuring a large number of devices simultaneously. Thus, it is possible to centralize an otherwise distributed network model using a single point of integration, which opens up additional opportunities for the implementation of complex network services throughout the network domain.

Other authors also mention the characteristics of the ZTP model and its advantages in large-scale environments and in the use of cloud-based applications. The authors in paper [13] state the orientation of network vendors such as Cisco and Juniper towards the development of this concept and they state the possibilities of implementing the ZTP model by individual vendors. Also, the possibility of implementing this model with the already existing network solutions and technologies is stated.

Under the influence of these trends it can be concluded that it is possible to achieve various advantages by implementing the ZTP model when installing new devices in the

network. Also, in a review of the existing literature it has been found that there are documents that have evaluated the ZTP model and its characteristics in cloud computing [12, 13], but there is still a lack of research dealing with the development of the environment and the application of the ZTP model in access networks.

Therefore, the purpose of this case study, in addition to demonstrating the functionality, is to use the process of ZTP model implementation in an environment founded on the existing solutions, to present all the benefits whose application would significantly reduce the network installation time for the network experts in the production environment, reduce the financial costs and the likelihood of human error.

3 Zero-Touch Provisioning Overview

ZTP provides open bootstrap interfaces to automate network device provisioning in heterogeneous network environments [14]. The holistic approach to the definition of the ZTP model suggests that the network devices that are connected to the network should be configured automatically. In more detail, ZTP could be described as the automation solution developed in order to reduce the probability of the occurrence of error and time saving for the network experts in the implementation of the new network infrastructure [15]. The ZTP model makes it possible for the network device to be automatically provisioned by obtaining the necessary information from the network, eliminating most of human interventions [16].

The ZTP process has its roots in the automated server setup process, which is a typical process that has been part of the IT world since the first Linux server [13]. However, in the networking world, the idea of the ZTP process appeared only a few years ago with the development of the SDN concept. When using the ZTP approach for network device configuration, the network administrator receives the device and then installs it physically into the network cabinet, conducts appropriate cabling and connects the device to the network. After connecting the device to the network, the device is started and with the help of standard network protocols the necessary information is obtained in order to download the configuration script. Then the script is downloaded and executed on the device, after which the device is ready to be used on the network [17]. A detailed process of the ZTP model operation usually includes the following steps:

1. Connecting the device to the network and starting it.
2. The device locates the DHCP server and sends a query and in response gets the IP address, network mask, default gateway and if necessary DNS server address, and
3. DHCP server will also provide the device with the TFTP server address or Uniform Resource Locator (URL) to access the HTTP server from which the system image and the configuration script are downloaded.

Before starting the ZTP process it is necessary to set the script to an available TFTP or HTTP server on the network. Besides, it is necessary to configure the DHCP server to assign the appropriate information.

When the device is connected to the network it receives the IP address of the TFTP server or the URL for access to the HTTP server and the file path to the configuration file, which requires two options to be configured on the DHCP server [14]:

- Option 150 – contains a list of IP addresses that point to the TFTP or HTTP server, and
- Option 67 – contains the file path to the configuration file or script on the TFTP or HTTP server.

Consequently, in this paper one's own script has been developed in the Python programming language whose application will serve to test its advantage, in compliance with the observation range – number of device installations within the ZTP model.

3.1 Guest Shell

Guest Shell is a virtualized Linux-based environment developed with the purpose of running custom Linux applications, including Python for automated control and management of Cisco devices. Using the Guest Shell environment, it is possible to install, upgrade and manage third-party applications. It is primarily intended for certain device management tools and applications [14].

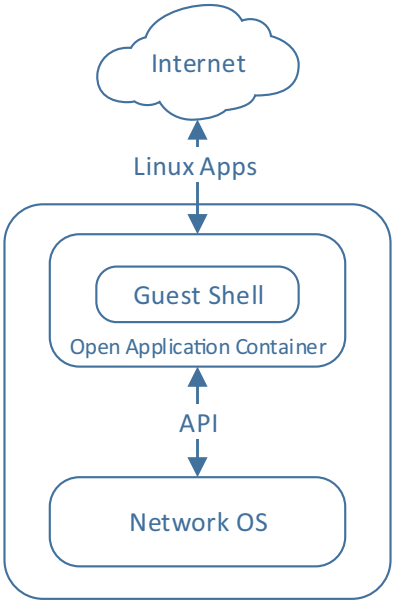

Fig. 1. Overview of IOx platforms [18]

Guest Shell shares the kernel with the network device operating system; in this case IOS XE, version 16.6.7. The users are given access to the Guest Shell environment and are allowed to edit scripts and to upgrade the software packages. Guest Shell container is managed using IOx (Fig. 1). IOx represents an end-to-end application framework that provides application-hosting capabilities for different types of applications on Cisco network platforms, and Guest Shell represents one such application [18].

Within the Guest Shell environment Cisco devices support Python version 2.7 (newer versions support only version 3.6). The possibility of Python scripting provides programmatic access to the CLI of the device to perform various tasks, but also the ability to implement the ZTP model [14].

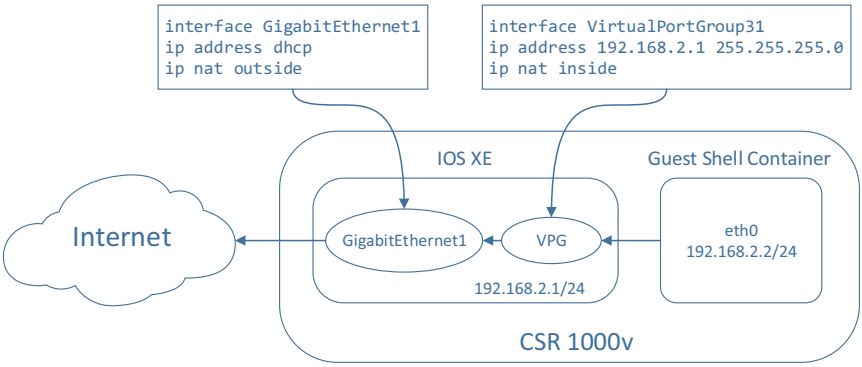

Fig. 2. Guest shell environment configuration for application hosting [19]

When running the ZTP process IOx is first enabled, which manages various applications, including the Guest Shell environment. In the case of routing platform which uses the VirtualPortGroup for front panel networking, as in the case of this paper, the configuration of GigabitEthernet and VirtualPortGroup interfaces is performed first. Guest Shell uses VirtualPortGroup as the source interface to connect to the outside network through NAT (Network Address Translation) [14]. The interface configuration is shown in Fig. 2.

3.2 Python CLI Module

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for the use as a scripting or glue language to connect the existing components together. Python supports modules and packages which encourages modularity and code reuse [20].

Cisco provides Python module that allows the execution of CLI commands in the Guest Shell Python environment. There are six functions available in the Python CLI module for executing various CLI commands and an additional help() function that displays features of the Cisco CLI module. To use these functions, it is necessary to first import the module with the import cli module command. The argument of these functions is the string of particular CLI commands. The following functions are available [14]:

- cli.cli(command) – The function takes any IOS command as an argument, parsing is performed and the command is executed, and after that the resulting text is returned. If an invalid command is issued, Python exception occurs.

- cli.clip(command) – This function behaves the same as the previous one, except that the resulting text is printed. The functions cli and clip provide the possibility of forwarding multiple commands.
- cli.execute(command) – This function individually executes the user and privileged EXEC level commands and returns the output.
- cli.executep(command) – Like the previous command, this function executes individual EXEC level commands, but the result of executing the function is printed unlike the previous case where the result is returned.
- cli.configure(command) – This function configures the network device according to the passed commands in the function argument. The function returns a list containing the submitted command, the success of the execution of the command, and information about error if it exists. It is possible to pass a series of commands separated by commas.
- cli.configurep(command) – This function behaves identically as the previous one, except that it prints the output of executing the function.

4 Environment Development for Zero-Touch Provisioning Model

The process of developing the environment in which the case study will be presented requires a two-step approach. First, it will be necessary to define the network topology in which the ZTP process will be performed and to configure the network devices and servers in it. Second, the development of the script in the Python programming language which will be executed on a Cisco CSR1000v router. This is followed by starting the device and applying the configuration.

4.1 Development of Network Topology

The ZTP model will be tested in the emulated network by using the Emulated Virtual Environment Next Generation (EVE-NG) software, version 2.0.3-110. Thus Router1, Switch1, Linux server and Cisco CSR1000v router on which the ZTP model is tested, are connected to the local network (Fig. 3).

Linux server with Ubuntu 20.04. operating system hosts DHCP, TFTP and HTTP servers.

The configuration of the aforementioned servers on Linux required the Internet access to install the necessary upgrades. Thus, Router1 is configured to allow access to external networks. The configuration of DHCP server on Ubuntu OS is presented in Fig. 4.

4.2 Python Script Development

The Guest Shell environment within the device comes with the support for Python 2.7 that allows access to the device CLI to perform various processes, including also the ZTP process. The idea when developing the script was to create a single script that could be executed on multiple devices on the network, and a configuration file would be available on a specific server to be downloaded by the script, containing variable values for each device. The configuration file would be written in an easy-to-understand JavaScript Object Notation (JSON) format (Fig. 5) whereby the values could be entered

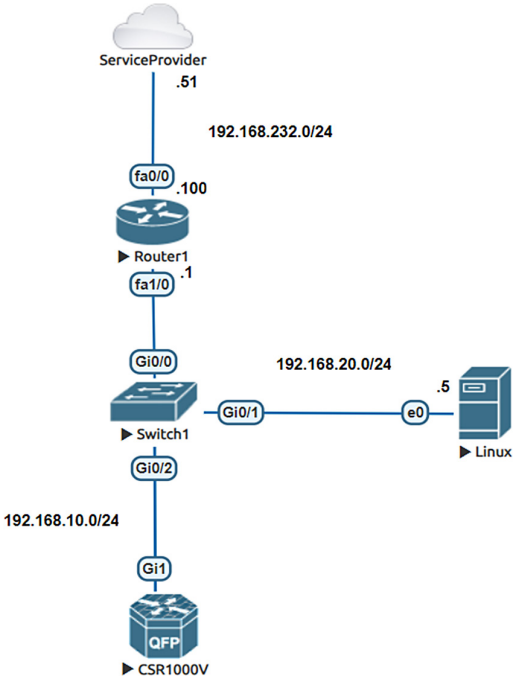

Fig. 3. Topology of test network environment

```
subnet 192.168.10.0 netmask 255.255.255.0 {
range 192.168.10.10 192.168.10.254;
option bootfile-name "ZTP_python.py";
option tftp-server-name "192.168.20.5";
option subnet-mask 255.255.255.0;
option routers 192.168.10.1;
option broadcast-address 192.168.10.255;
default-lease-time 600;
max-lease-time 7200;
```

Fig. 4. Configuration of DHCP server on Ubuntu OS

by anyone, and not just the developer. In this way, it is possible to configure a large number of devices in a very efficient way.

The script imports three Python modules, and these are re, cli and json. Module re or Regular Expression represent a string of characters which create a pattern to search a particular string. Then cli module which, as already mentioned, is used for passing commands to the network device from the Python environment. The script defines three functions for every step of the device configuration, which are called at the end of the script.

```
[
    {
          "SN": "9KZE846DMPS",
          "hostname": "ZTPDevice1",
          "ipaddress": "192.168.10.10",
          "netmask": "255.255.255.0",
          "secret": "admin"
     }
]
```

Fig. 5. Configuration file in JSON format

```
def get_serials():
    print '\nSerialNo parsing...\n'
    inventory = cli('show inventory')
    invetoryList = inventory.split('\n\n')
    for chassis in invetoryList:
         if (re.search('NAME: "Chassis"',chassis)):
             match = re.findall('SN:\s(.*)', chassis)
    serialNo = match[0]
    print '\nSerial number: {}\n'.format(serialNo)
    return serialNo
```

Fig. 6. Function within the script to obtain a serial number of the network device

The first function, i.e. get_serials(), has the task to use re module to parse the serial number of the network device that is obtained on it by submitting an adequate command via cli module. This function eventually returns the variable which contains the requested serial number (Fig. 6).

```
def get_config():
    url = 'http://192.168.20.5/device_config.json'
    print '\nDownloading configuration from URL: {}'.for-
mat(url)
    clip('copy http://192.168.20.5/device_config.json
flash')
    config = cli('more flash:device_config.json')
    json_config = json.loads(config)
    print '\nConfiguration commands: Collected!\n'
    return json_config
```

Fig. 7. Function within the script to download the configuration file from HTTP server

Furthermore, the function get_config() has been defined for the purpose of downloading a configuration file in JSON format from an HTTP server. A unique URL is

first defined, which is then passed along with the rest of the corresponding file download command via the cli module. After the download, the JSON form is converted into a dictionary data type, i.e. in the list of dictionary data which is then returned by the function (Fig. 7).

```
def configure_device(config, serial):
    print '\nConfiguring device...\n'
    for line in config:
        if line["SN"] == serial:
            device_config = line

            configurep(['hostname {}'.format(device_con-
fig['hostname'])])
            configurep(['int GigabitEthernet1', 'ip add
{} {}'.format(device_config['ipaddress'],
                        device_config['netmask']), 'no
shut', 'end'])
            configurep(['line con 0', 'logging synchro-
nous', 'no exec-timeout', 'end'])
            configurep(['ip domain-name domain.com'])
            configurep([crypto key generate rsa modulus
1024'])
            configurep(['username user password admin'])
            configurep(['enable secret {}'.format(de-
vice_config['secret'])])
            configurep(['line vty 0 4', 'transport input
ssh', 'login local'])
```

Fig. 8. Function that performs device configuration

Finally, the function configure_device() has been defined, which retrieves the corresponding values of variables from the list of dictionaries and passes them through the cli device configuration module. The configuration in this case is for the purpose of displaying the ZTP model and the arbitrary commands are selected (Fig. 8).

5 Results of Implementing Zero-Touch Provisioning Model

After developing the network environment that allows the implementation of the ZTP process and the creation of a script and the necessary configuration file, it is possible to begin with the implementation of the specified process. Once the Cisco CSR1000v router with the IOS XE operating system is connected to the network, it can be started.

The Cisco device begins the startup process by checking certain components, such as the processor, Random Access Memory (RAM) and Non-Volatile Random Access Memory (NV-RAM). After the tests have been performed, the bootstrap program is copied from Read-Only Memory (ROM) into RAM memory. After copying, the processor executes the instructions from the bootstrap program. The main task of the bootstrap

program is to locate the Cisco IOS and to load it into RAM. The last step in booting the device is to check the existence of the startup configuration in the NVRAM memory.

When the device starts and checks for the absence of a startup configuration in the NVRAM memory, it enters the ZTP operating mode. The device then finds the DHCP server and then receives a DHCP offer which includes the option 150 and 67, respectively. The device receives information on the location of the TFTP server and the file path to the script which is located on that server and downloads the script and stores it in the flash memory. The enabling of the Guest Shell environment starts then, preceded by enabling of the IOx platform.

After the process of starting the Guest Shell environment is completed, the script can be executed in the Python environment. The device in this process obtains the configuration file from the HTTP server, parses the values of the variables and commands are passed by means of the cli module. With the submission of commands, the ZTP process ends and the device is configured and ready for use online. By entering the show running-configuration command it is possible to see the applied configuration on the CSR1000v router. Figure 9 shows the hostname configuration on the device. Further in the same display it is possible to see the encrypted configured password which is used to enter the privileged EXEC level. Finally, there is a display of configured domain name for the purpose of allowing access via the SSH protocol. The hostname and the password values were parsed from the JSON file.

Fig. 9. Configured hostname, password and the domain name

Figure 10 shows the configuration of the VirtualPortGroup interface, whose configuration is automated when enabling the Guest Shell environment. In addition, the script configures the GigabitEthernet1 interface to which an IP address is associated which is parsed from a JSON file.

Figure 11 shows the configuration of IOx application framework. The configuration of the IOx framework is followed by the configuration of NAT where the role of the inside interface is played by the VirtualPortGroup, and the outside by GigabitEthernet1. NAT is configured automatically when enabling the Guest Shell environment.

```
!
interface VirtualPortGroup31
 ip address 192.168.2.1 255.255.255.0
 ip nat inside
 no mop enabled
 no mop sysid
!
interface GigabitEthernet1
 ip address 192.168.10.10 255.255.255.0
 ip nat outside
 negotiation auto
 no mop enabled
 no mop sysid
!
```

Fig. 10. Configured Virtual Port Group and interface

```
!
iox
ip nat inside source list NAT_ACL interface GigabitEthernet1 overload
ip forward-protocol nd
ip http server
no ip http secure-server
ip http client source-interface GigabitEthernet1
!
!
!
ip access-list standard NAT_ACL
 permit 192.168.0.0 0.0.255.255
!
```

Fig. 11. Configuration of IOx application framework

Finally, Fig. 12 shows the configuration of the console port and the configuration of the Virtual Terminal lines, thus enabling the remote access by means of the SSH protocol.

```
!
line con 0
 logging synchronous
 stopbits 1
line vty 0 4
 login local
 transport input ssh
!
```

Fig. 12. Configuration of console port and Virtual Terminal lines

This type of automation results in the saving of time, costs of employees – reflected in the saving of engineer-hours and reduction of human error. The result of savings depends on the number of installations, i.e. the increase in the number of devices causes an increase in the efficiency in relation to the traditional method. Also, the difference in the saving is reflected in the dependence on the kind and type of the device, implementation environment and end user requirements. With the done research, for instance, the time savings for the selected 100 arbitrary installations amounts to more than 95% time, which represents a much greater saving in relation to the analyzed research [11]. Additionally,

the time saving depends directly on the reduction of engineer-hours, which results in financial savings.

Below is the conclusion about the carried out case study and the proposal for further research.

6 Conclusion

Today's computer networks encounter challenges of rapid development and implementation of new services in order to satisfy the ever stricter user requirements. In order to satisfy the user expectations for fast, elastic and scalable networking, fundamental changes in the method in which heterogeneous networks support a wide spectrum of applications are needed. New paradigm of Software Defined Network (SDN) with programmable infrastructure appears as a promising solution.

Until now, in traditional networks, the process of implementing new devices into the network represented a time-consuming and financially demanding process with the always present possibility of human error. As consequence of the increasing need for automation of the network processes and the development of the SDN concept, the Zero-Touch Provisioning model has been presented.

ZTP is an automation solution that allows reduction of time and costs of implementing new devices into the network. Although there are various variations of implementing this model, the case study in this paper presents the possibility of use in broadband access on the example of Cisco CSR1000v router with IOS XE operating system. The realization of the ZTP process in the emulated network has confirmed the assumption about its efficiency, which is primarily reflected in the saving of time, in our case of over 95% (depending on the number of device installations). Although ZTP automates most part of the configuration process of new devices in the network, its execution requires certain preparation, i.e. training of the necessary servers, creation of the script and configuration file and certain tests. Consequently, such a model has special advantages in somewhat larger network implementations where the savings of time, costs and reduced probability of error especially come to the fore.

In discussing the ZTP model it should be emphasized that all the major network equipment manufacturers are oriented to device configuration automation, but the way of implementing the ZTP model differs somewhat from manufacturer to manufacturer. Therefore, the implementation of the ZTP model in various scenarios of heterogeneous networks with several different manufacturers will be a little less efficient. Using the method of correlation and regression as well as the method of comparison (Checklist method, Before-and-after method, etc.) it is possible to define the deviations in the implementation of the ZTP model in various scenarios of heterogeneous networks with several different manufacturers. In order to solve this problem it will be necessary to perform additional research and develop an appropriate type of joint platform and the necessary components that would allow the performance of the ZTP model in the networks with devices from several manufacturers, which is at the same time the direction of the planned future research.

References

1. Bakshi, K.: Considerations for software defined networking (SDN): approaches and use cases. In: 2013 IEEE Aerospace Conference, pp. 1–4. IEEE, Big Sky (2013).
2. Prajapati, A., Sakadasariya, A., Patel, J.: Software defined network: Future of networking. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp. 1351–1354. IEEE, Coimbatore, India (2018)
3. Cisco Software-Defined Access. https://www.cisco.com/c/dam/en/us/products/se/2018/1/Collateral/nb-06-software-defined-access-ebook-en.pdf. Accessed 14 Oct 2020
4. Ahmed, K., Nafi, N.S., Blech, J.O., Gregory, M.A., Schmidt, H.: Software defined industry automation networks. In: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–3. IEEE, Melbourne, Australia (2017)
5. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
6. Yoshino, M., Astawa, I.G., Trinh, T., Suzuki, H., Koswara, M., Nguyen, B.: Zero-touch multi-service provisioning with pluggable module-type olt on access network virtualization testbed. In: 25th Opto-Electronics and Communications Conference OECC 2020, pp. 2020–2022 (2020)
7. Boskov, I., Yetgin, H., Vucnik, M., Fortuna, C., Mohorcic, M.: Time-to-provision evaluation of IoT devices using automated zero-touch provisioning. In: IEEE Global Communications Conference GLOBECOM 2020 - Proceedings, vol. 2020-Jan (2020)
8. Brockelsby, W., Dilda, S.: Tactical network automation with NetZTP and one shot. In: IEEE 40th Sarnoff Symposium, Newark, NJ, USA, pp. 1–3 (2019)
9. Demchenko, Y., et al.: ZeroTouch provisioning (ZTP) model and infrastructure components for multi-provider cloud services provisioning. In: IEEE International Conference on Cloud Engineering Workshop (IC2EW), Berlin, Germany, pp. 184–189 (2016)
10. Andrus, B.M., et al.: Zero-touch provisioning of distributed video analytics in a software-defined metro-haul network with P4 processing. Optics InfoBase Conference Paper, vol. Part F160, pp. 2019–2021 (2019)
11. Szigeti, T., Zacks, D., Falkner, M., Simone, A.: Cisco Digital Network Architecture: Intent-based Networking for the Enterprise. Cisco Press, Hoboken (2018)
12. Filiposka, S., et al.: Enabling high performance cloud computing using zero touch provisioning. In: 23rd Telecommun Forum (TELFOR), pp. 67–70. IEEE, Belgrade (2015)
13. Demchenko, Y., et al.: Enabling Automated Network Services Provisioning for Cloud Based Applications Using Zero Touch Provisioning. In: 8th International Conference on Utility and Cloud Computing (UCC), pp. 458–464. IEEE, Limassol (2015)
14. Programmability Configuration Guide, Cisco IOS XE Everest 16.6.x. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/zero_touch_provisioning.html. Accessed 20 Oct 2020
15. Mishra, R., Gijare, V., Malik, S.: Zero touch network: a comprehensive network design approach. Int. J. Eng. Res. Technol. (IJERT) **8**(9), 792–794 (2019)
16. Zero-Touch Provisioning. https://infocenter.nokia.com/public/7750SR1910R1A/index.jsp?topic=%2Fcom.sr.basic%2Fhtml%2Fztp.html. Accessed 22 Oct 2020
17. Deliverable D13.3 Proposed Network Architectures – White Paper. https://www.geant.org/Projects/GEANT_Project_GN4-1/Documents/D13-3_Proposed-Network-Architectures_White-Paper.pdf. Accessed 22 Oct 2020
18. Introduction to GuestShell. https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2018/pdf/DEVNET-1695.pdf. Accessed 23 Oct 2020

19. Programmability Configuration Guide, Cisco IOS XE Amsterdam 17.1.x. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/171/b_171_programmability_cg.html. Accessed 23 Oct 2020
20. Kumar, A., Panda, S.P.: A survey: how python pitches in IT-world. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), pp. 248–251. IEEE, Faridabad, India (2019)