engineering proceedings

MDPI

Proceeding Paper

Research of a Virtual Infrastructure Network with Hybrid Software-Defined Switching †

Yuri Ushakov *, Margarita Ushakova and Leonid Legashev

Mathematical and Information Technology Faculty, Orenburg State University, Pobedy 13, Orenburg 460018, Russia; m.v.ushakova@mail.ru (M.U.); silentgir@gmail.com (L.L.)

* Correspondence: ushakov@unpk.osu.ru; Tel.: +7-353-291-2195

† Presented at the 15th International Conference "Intelligent Systems" (INTELS'22), Moscow, Russia, 14–16 December 2022.

Abstract: Modern trends in the information technology have led to the fact that entire systems of infrastructure are becoming software-defined. Modern hyper-converged solutions use software-defined networking and soft switches for the hypervisor networking subsystem. The paper goal is to study traffic processing in hyperconverged structures with software switching based on OpenFlow versus traditional approaches. The features of the hyperconverged solutions network infrastructure are considered, approaches to the study of software-defined environments are described. A model of the processing traffic internal structure of a converged node, combining the functions of a hypervisor, a storage system and a switch, is proposed. Interface models reproduced traffic switching with the traditional approach and with higher-level OpenFlow processing have been developed. The approaches to the implementation of the developed models based on experimental studies of network equipment are described. The results of an experimental study of a network node and a synthesized model are presented. The possibility of implementing the proposed approaches within the specified accuracy are described.

Keywords: modeling; OpenFlow; switch; traffic; QoS

Citation: Ushakov, Y.; Ushakova, M.; Legashev, L. Research of a Virtual Infrastructure Network with Hybrid Software-Defined Switching. *Eng. Proc.* **2023**, *33*, 52. https://doi.org/10.3390/engproc2023033052

Academic Editors: Askhat Diveev, Ivan Zelinka, Arutun Avetisyan and Alexander Ilin

Published: 19 July 2023

1. Introduction

Today, cloud infrastructures require ever more flexible approaches and technologies that involve the transition to more distributed applications, microservice architecture, and flexible IT models. Modern trends in the development of multi-cloud services lead to the widespread use of software-defined infrastructure, storage, networks, as well as the introduction of convergent and hyper-converged services and equipment [1]. Convergence in networks combines the traffic of heterogeneous services on a single technology stack with a hardware component (traditional switches supporting prioritization, segmentation at levels 2–4, combining several layer 2 protocols), while hyperconvergence is based on a software-oriented architecture. Hyperconvergence does not exclude the use of converged technologies in the network stack, but uniform network management across hypervisors, storage systems, virtual switches and routers, and equipment can greatly improve the quality of flow control, logical segmentation, security and prioritization. The use of virtual dedicated channels from source to destination with guaranteed marginal characteristics based on packet-switched software-defined networks makes it possible to provide traffic quality of service and security with much lower organizational and configuration costs than on the basis of traditional converged networks [2]. The main feature of hyper-converged solutions from different manufacturers is the combination of management of all network devices into a unified software system, and most manufacturers add support for virtual overlay transport technologies (Overlay Transport), such as VXLAN networks (Virtual Extensible LAN), Cisco OTV (Overlay Transport Virtualization), as well as various

overlay networks of virtual infrastructure hypervisors (VMWare NSX-T, Microsoft HNV, OVN, Cisco ACI), and container orchestrators (swarm overlay network, flannel, k-vswitch, OVN) [3]. Open cloud solutions (OpenStack, OpenNebula) and container solutions (Docker Swarm, Rancher, Kubernetes, OpenShift) can use overlay networks through many technologies through a plug-in system, but the most common is the use of VXLAN solutions and transmission control through Open vSwitch. The use of Open vSwitch in conjunction with software-defined networking technologies (both via the OpenFlow protocol and using ovs-ofctl, ovs-vsctl) allows you to combine a virtual switch and equipment into a single managed network, as well as make universal traffic management virtual machines and services running directly on the hardware (bare-metal), for example, software-defined storage systems (CEPH, drbd9, gluster, NFS, VMWare vSAN). At the same time, studying the operation of such networks in conjunction with hyperconverged infrastructure is an open problem, too many factors can affect each element of the infrastructure and, thereby, performance parameters. One of the ways to comprehensively cover all factors for the study of networks can be modeling, but it is complicated by a high level of nesting and mutual dependence of components. The authors of the paper [4] describe the creation of a test bed for experiments with software-defined networking and the cloud using hyperconverged SmartX Boxes distributed over several sites. Each SmartX Box consists of several virtualized functions, which are divided into SDN and cloud functions with the ability to track resources on a distributed cloud platform. The paper [5] discusses an approach to automation for the efficient implementation of a variety of services through distributed hyper-converged blocks towards a converged software-defined infrastructure using the SmartX automation software environment. Within the framework of the paper [6], the principle of integrating software-defined networks and hyperconverged architectures is demonstrated using OpenFlow as a communication protocol and Opendaylight as a controller in the control plane. The authors of [7] note that the hyperconverged architecture has higher availability and performance when evaluating the provision of services in a cloud computing environment. The author of the study [8] deals with the issues of improving the performance of applications in a hyper-converged infrastructure.

2. Theoretical Part

Let us consider one node of hyperconverged infrastructure based on the most common open solutions, such as KVM hypervisor, virtual switching Open vSwitch and Linux Bridge, software defined storage (SDS) (block, object, or file SDS), based on the resources of the same servers on which virtualization is performed (CEPH, Gluster, ScaleIO, SwiftStack, HDFS and the like). Each hyperconverged infrastructure server contains a computing node (processor, memory, disks), a Linux operating system, a network part (network cards, an Open vSwitch, Linux Bridges, veth virtual network cards), a virtualization node controlled by an external orchestrator, a server node SDS with storage on local disks integrated with a common storage subsystem (see Figure 1a). Since there are many options for internal connections (see Figure 1b), it is necessary to foresee and explore all possible options when creating a model [9].

The main problem of modeling networks with a large number of traffic processing technologies and ensuring quality of service is a huge number of possible combinations of various parameters, types and settings of queues, traffic processing policies. Consider the option of creating a network interface with enabled VLAN functionality, a classifier in both directions, incoming and outgoing rate limiting through queues, and traffic filtering in both directions. In this case, the classifier and queues use DSCP; WRED (Weighted Random Early Detection) is selected as the congestion management policy, and prioritization occurs through WRR (Weighted Round Robin) for classes other than ef, for which LLQ (Low Latency Queue) is used. Limiting the amount of incoming traffic also occurs through WRED removal of packets from the queue with different weights of different classes (Class-based policing) [9]. The classifier is configured to mark some packets with the classes af11, af13, af21, af23, ef, and be. The input from the network occurs through a bidirectional mac

module, the output and input of the upper levels through filters (ACL). This sequence of actions at the input from the network (decapsulation, classification, bandwidth handling, labeling, filtering) and at the output to the network (filtering, labeling, bandwidth handling, queues, encapsulation) was created based on the description of the sequence of actions Cisco IOS [10] (see Figure 2).

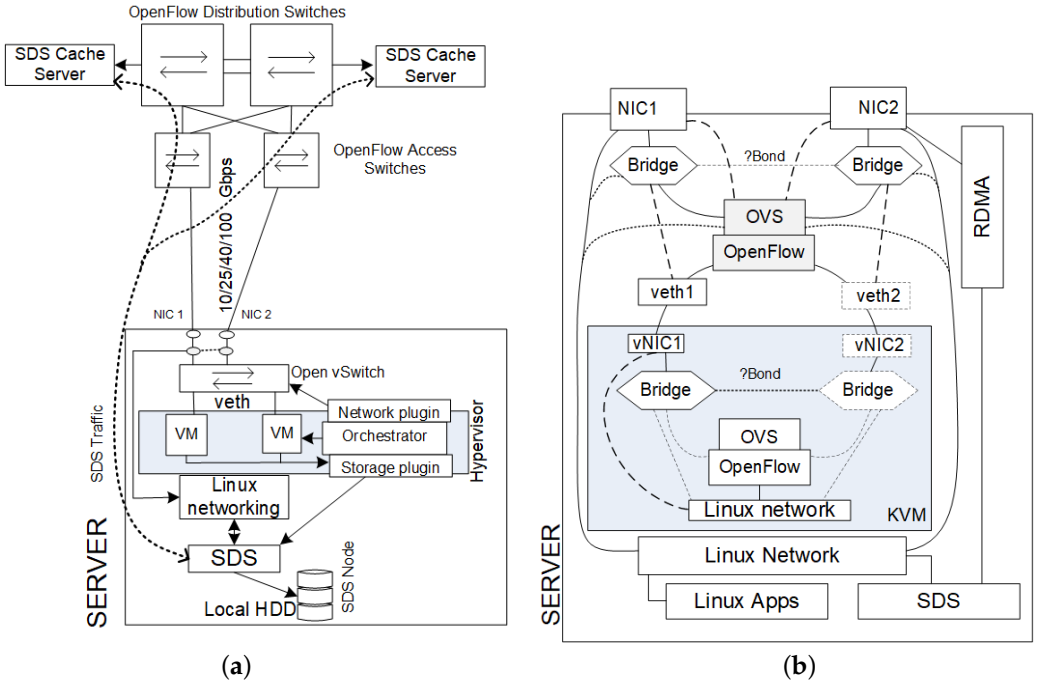

Figure 1. Server internal organization diagram: (**a**) component connection diagram; (**b**) diagram of possible internal network connections.

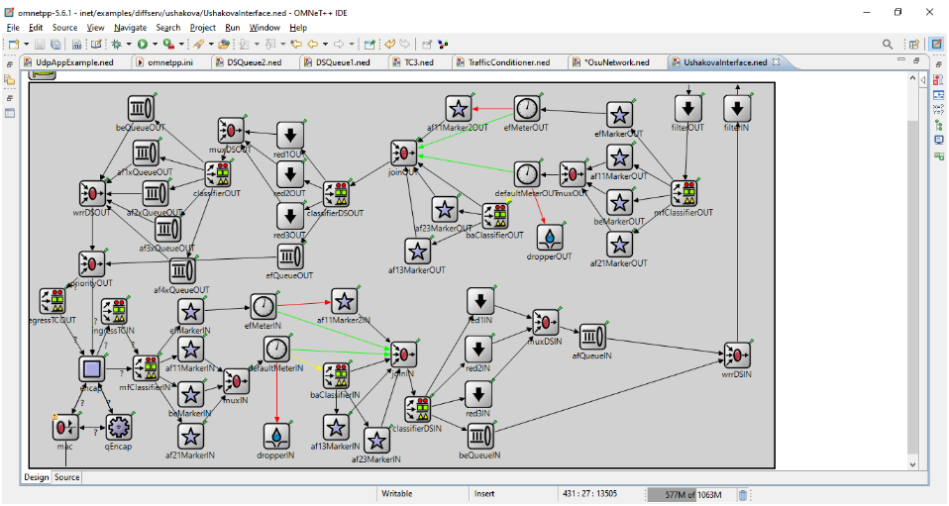

Figure 2. The scheme of the packet passing through the network interface.

To model a realistic delay, the created module of the Traffic-Conditioner type was used to put the ingressTCIN and egressTCOUT components in place. It is used to introduce a delay in the incoming direction, corresponding to the processing time of the packet by the switch on the real network. At the same time, the use of other combinations of queue settings, their marking and connection logic (for example, nested WRR queues) can significantly complicate the interface model. Using OpenFlow for traffic processing generally does not affect the interface scheme in terms of operating with queues, but removes some of the classification and marking, which is now performed at the OpenFlow Data

Plane level. In the general case, QoS is implemented in OpenFlow only as a classification, labeling of packets, and measurement of current flow performance [11]. The management of queue parameters, traffic limiters, and prioritization policies in most implementations is left for traditional configuration through the mechanisms of the switch operating system (or the control program, as is the case with Open vSwitch). Some controllers (for example, OpenDaylight, ONOS, Floodlight) have mechanisms for setting up queues on a certain set of switches via NETCONF or directly by commands (for example, for OVSDB). However, this does not solve the problem of creating a model of such equipment. Figure 3 shows the scheme of the interface in OpenFlow mode, all marking and rate limiting takes place in the OpenFlow core, but queues and work with them, including prioritization, remain the same.

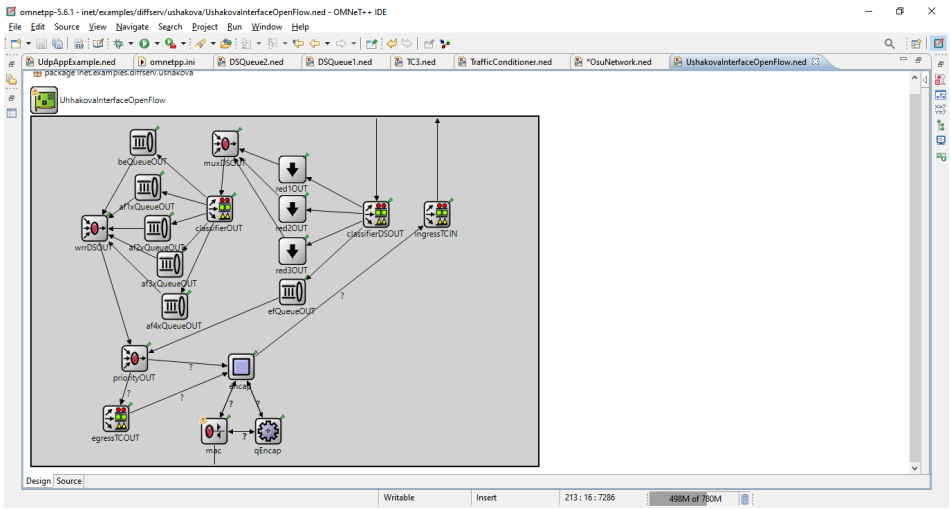

Figure 3. The scheme of the packet passing through the OpenFlow network interface with QoS support.

To study the operation parameters of a software-defined device over a period of time T, we denote the given input parameters that describe incoming traffic and current device settings as $Z(t) = \{Traffic(t), Device(t)\}, \forall t \in T$, and the measured output parameters, which include timing, input and output queue parameters, queue statistics, and a snapshot of device metrics, as $Y(t) = \{Latency(t), Qout_{ij}(t), Qin_{ij}(t), S_{ij}(t), State'_i(t)\}, \forall t \in T$ Then the function of mapping input parameters to output parameters will be defined as:

$$F(t) : Z(t) \rightarrow Y(t), \forall t \in T \quad (1)$$

Let us represent the tree of input parameters Z(t) as a linear sweep

$$Z(t) = \{z_1(t), z_2(t)...z_n(t)\}, \forall t \in T,$$

where is the number of input parameters in the sweep. Similarly, we represent the tree of output parameters $Y(t)$ as a linear sweep

$$Y(t) = \{y_1(t), y_2(t)...y_m(t)\}, \forall t \in T,$$

where $m \in N$ is the number of output parameters in the sweep. The input set of parameters $Z(t_i) at \forall t_i \in [T_{i-1}..T_i) \in T$ should not change, and the duration of the time interval $[T_{i-1}..T_i)$ should be sufficient for the required number of measurements of the output parameters and to achieve stationarity of the probabilistic patterns of the output

parameters $y_1(t_i), y_2(t_i)...y_m(t_i)$. In such case mapping (1) can be reduced to the form $F(t) = \{f_1(t), f_2(t)...f_m(t)\}$ in this way:

$$\begin{cases} f_1(t_i) : \{z_1(t_i), z_2(t_i)...zn(t_i)\} \rightarrow y_1(t_i) \\ f_2(t_i) : \{z_1(t_i), z_2(t_i)...zn(t_i)\} \rightarrow y_2(t_i) \\ ... \\ f_k(t_i) : \{z_1(t_i), z_2(t_i)...zn(t_i)\} \rightarrow y_k(t_i) \end{cases}$$

When modeling, it is necessary to ensure the correspondence of the obtained output parameters of the device under study $Y(t_i)$ and the obtained output parameters of the model $Y^*(t)$ with the same sets of input parameters. To do this, we find a mapping $F(t) : Z(t) \rightarrow Y^*(t), \forall t \in T$, such that

$$\left| \frac{y_j(t_i) - y_j^*(t_i)}{y_j(t_i)} \right| < \varepsilon, \forall j \in [1..m],$$

where $y_j^*(t_i) \in Y^*(t_i)$ are the measured output parameters of the model; ε is the denoted accuracy. The study of a device with all combinations of input parameters is hampered by the fact that the number of all possible combinations of parameters depends on the number and ranges of parameters of specific devices with specific versions of system software, and, in general, is an NP-complete problem [12]. Therefore, to build a model, the method of determining indicators for boundary values and a certain number of intermediate values of each parameter separately will be used, and the dependencies between the parameters for the maximum possible values will also be studied. This will make it possible to obtain an interpolation estimate of the values of indicators for the model with an accuracy determined by the number of intermediate measurements of parameters and the shape of the dependency curves.

3. Experimental Research

The study of the operation of model devices connected to a network was carried out on the example of a new experimental segment of the data processing center (DPC) of Orenburg State University. Virtual machines in the data center are divided into groups, between which network access must be excluded. There can be more than a hundred such groups, depending on the time and needs, and the groups can have their own routers, DHCP servers, repeating VLANs and VXLAN networks. Therefore, instead of creating overlay networks using the capabilities built into OpenNebula, it was decided to isolate traffic through OpenFlow at the physical and data link levels. The OMNeT++ simulation tool with the INET framework was used as a modeling environment, which is the most suitable tool for modeling networks with SDN support. A feature of OMNeT++ is the decomposition of the model to level 1 of the OSI model with maximum detail of all technologies, protocols, and data formats used. The segment was implemented to provide high-performance data processing and create a private cloud for educational and production tasks, including recording and processing video streams, saving Wi-Fi client traffic dumps, creating training virtual machines, and big data processing tasks. The data center segment includes two switches that play the role of a distribution layer. These switches connect 10 GbE channels to each of the seven access switches. Each server connects to two access switches with two 10 GbE links (storage segment and switching segment) and two 1 GbE links (management). Since the storage system is built on the basis of CEPH, based on the disks of the same servers that are used to run all virtual machines, the requirements for the storage segment lead to the need to provide a peak node speed of at least 10–15 Gb/s when exchanging at the link level and delay. When writing to the storage, duplication occurs on several servers, in addition, regular data rebalancing and sharding occurs, which significantly increases traffic. The switching segment is used for inter-network interaction of virtual machines with external consumers and among themselves. When using distributed neural networks

and accessing Hadoop-based storages that physically store data on separate drives of the same servers, the performance requirements for the switching segment also have lower limits. Figure 4 shows the network topology.

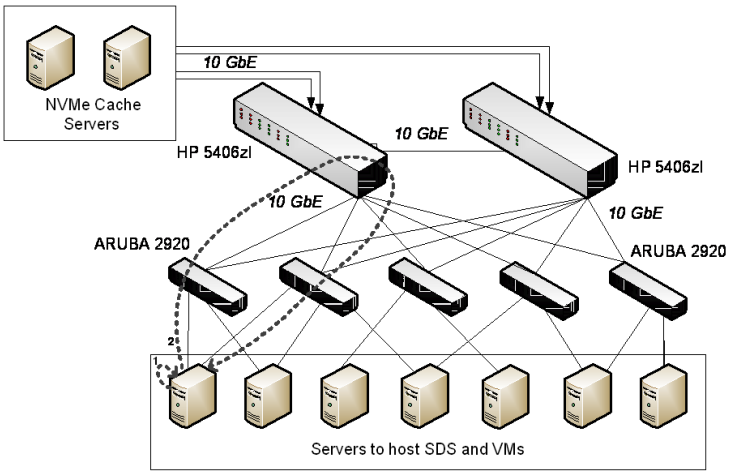

Figure 4. Experimental segment topology.

At the time of research, the experimental segment included seven servers (Xeon Gold, 128 GB of RAM, NVMe SSD, four 10 Gb/s network cards with hardware packet processing) to host storage and virtual machines, two servers to work only as systems storage, five access switches (HPE ARUBA 2920), and two distribution switches (HP ProCurve Switch 5406zl with five HP J9309A ProCurve 4-Port 10GbE SFP+ zl modules). Virtual machines are controlled by the OpenNebula orchestration system, Open vSwitch is used as virtual switches in hybrid switching mode with proactive OpenFlow mode (rules are installed on the local server switch by the orchestrator when performing actions with virtual machines or networks). Using scripts that automate the connection of network components (Figure 1a), delays in the experimental segment of the data center were investigated at traffic generation intensity values Traffic(t) at various time intervals t. To do this, after launching all the necessary components, the script starts intercepting tcpdump traffic with writing to the tmpfs disk, as well as generating traffic with the trafgen utility at various intensities. During the experiment, device state snapshots were taken using the SNMP protocol and OpenFlow metrics. Based on the topology shown in Figure 4, a model of this network was formed in the OMNeT ++ using standard tools and a study of delays in the model network was carried out (see Figure 5).

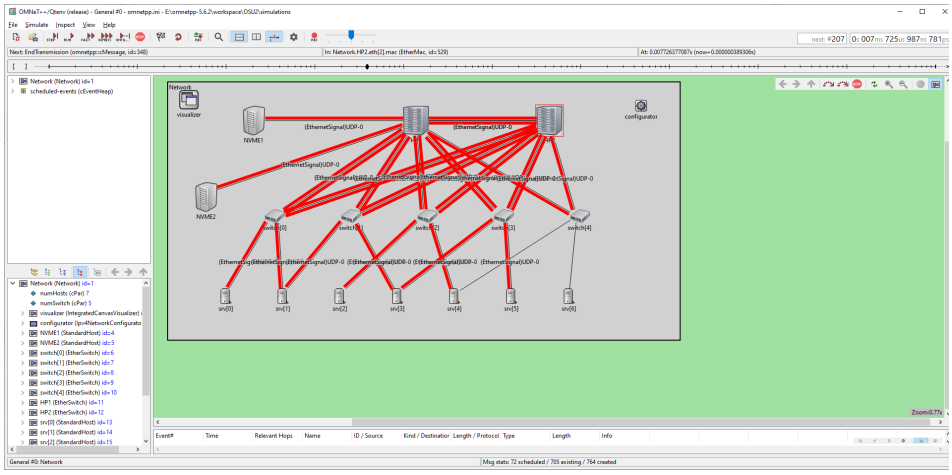

Figure 5. Network model in OMNeT ++.

With the help of the developed scripts for the automation of the experiment, a study of all models of network devices and servers that are part of the network under study was carried out, and models of these devices were synthesized. To study delays on a network model, a traffic generation plan was synthesized, similar to experiments on equipment. Figure 6 shows a graphical interpretation of the results of studying network delay without using queues (on the equipment, queues were minimized to one packet per queue, multiplexing into one channel was excluded), its model obtained using standard tools of the OMNeT ++, and also a model obtained by means of the developed software and hardware complex. The study was carried out under various work scenarios (different paths shown in Figure 1a) for various traffic generation intensities.

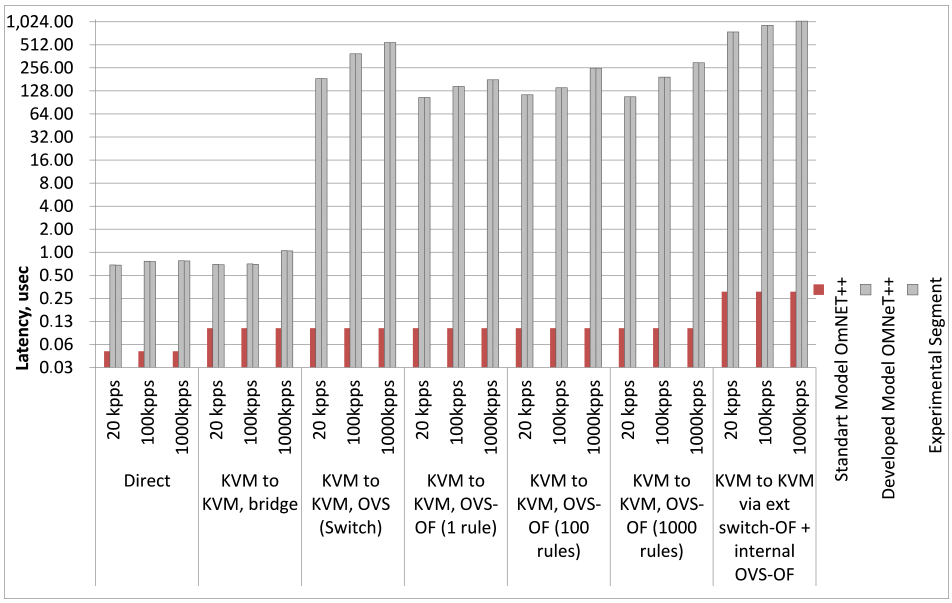

Figure 6. Results of the study of latency in the network and its models.

The delays were introduced by the ingressTCIN module, which interpolated the received packet processing times during an experimental study to the current state of the switch and the available enabled settings (VLANs, queues, classifiers, markers). Figure 6 shows the summary results of measuring delays in a real network, a network model obtained by standard means of the simulation system, and a network model. According to the results obtained, when using the switch models included in the OMNeT++, it is not possible to investigate the influence of the frame processing time by the switch on the total packet transmission delay in the network. Comparison of the results of the study of the model obtained using the tools of the developed software–hardware complex with the delays of the experimental segment of the data center showed a coincidence at similar rates of frame generation within 5%.

4. Conclusions

In this paper, we propose a methodology for studying the processes of traffic flow in hyper-converged structures with software switching based on traditional approaches and OpenFlow tables. Interface models were created for traffic switching in the usual way and with higher-level OpenFlow processing. To obtain the probabilistic-temporal characteristics of packet processing by equipment, a method for the initial synthesis of the equipment model with the developed interface model is proposed, and the possibility of traffic passing along various virtual paths within the virtual infrastructure is also taken into account. The obtained results of research and modeling showed a significant influence of the settings and sizes of the OpenFlow tables of virtual infrastructure elements on the package processing time, including the non-linear form of dependence. The use of interpolation of the results

of an experimental study in the model to provide the required form of the dependence of the packet processing time showed high modeling accuracy under similar conditions and traffic generation intensities within 5%.

Author Contributions: Conceptualization, Y.U. and M.U.; methodology, Y.U.; software, Y.U.; validation, M.U. and L.L.; writing—original draft preparation, Y.U.; writing—review and editing, Y.U. and M.U.; visualization, L.L.; supervision, L.L.; All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the Russian Science Foundation (project No. 22-71-10124).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable due to privacy restrictions.

Acknowledgments: The data for building the models were collected at the Mathematical and Information Technology Faculty of Orenburg State University under the supervision of Canditate of Technical Sciences, D.I. Parfenov.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Melo, C.; Dantas, J.; Maciel, P.; Oliveira, D.M.; Araujo, J.; Matos, R.; Fé, I. Models for hyper-converged cloud computing infrastructures planning. *Int. J. Grid Util. Comput.* **2020**, *11*, 196–208. [CrossRef]
2. Bhaumik, S.; Chanda, K.; Chakraborty, S. Poster: Controlling Quality of Service of Container Networks in a Hyperconverged Platform. In Proceedings of the IFIP Networking Conference (Networking), Paris, France, 22–26 June 2020; pp. 661–663.
3. Meneses-Narvaez, S.E.; Dominguez-Limaico, P.; Rosero-Montalvo, D.; Narvaez-Pupiales, S.; Vizuete, M.Z.; Peluffo-Ordónez, D.H. Design and Tests to Implement Hyperconvergence into a DataCenter: Preliminary Results. *Adv. Emerg. Trends Technol.* **2019**, *1066*, 54–67.
4. Risdianto, A.C.; Usman, M.; Kim, J.W. SmartX box: Virtualized hyper-converged resources for build-ing an affordable playground. *Electronics* **2019**, *8*, 1242. [CrossRef]
5. Kim, J.W. Realizing Diverse Services Over Hyper-converged Boxes with SmartX Automation Frame-work. In *Conference on Complex, Intelligent, and Software Intensive Systems*; Springer: Cham, Switzerland, 2017; pp. 817–822.
6. Meneses, S.; Maya, E.; Vasquez, C. Network Design Defined by Software on a Hyper-converged Infra-structure. Case Study: Northern Technical University FICA Data Center. In *International Conference on Systems and Information Sciences*; Springer: Cham, Switzerland, 2020; pp. 272–280.
7. Melo, C.; Dantas, J.; Maciel, R.; Silva, P.; Maciel, P. Models to evaluate service provisioning over cloud computing environments-A block-chain-as-A-service case study. *Revista de Informática Teórica e Aplicada* **2019**, *26*, 65–74. [CrossRef]
8. Wen, H. Improving Application Performance in the Emerging Hyper-Converged Infrastructure. Ph.D. Thesis, University of Minnesota, Minneapolis, MN, USA, 2019.
9. Bök, P.B.; Noack, A.; Müller, M.; Behnke, D. *Quality of Service. Computernetze und Internet of Things*; Springer Vieweg: Wiesbaden, Germany, 2020; pp. 251–291.
10. Szigeti, T.; Zacks, D.; Falkner, M.; Arena, S. *Cisco Digital Network Architecture: Intent-based Net-working for the Enterprise*; Cisco Press: Indianapolis, IN, USA, 2018.
11. Karakus, M.; Durresi, A. Quality of service (QoS) in software defined networking (SDN): A survey. *J. Netw. Comput. Appl.* **2017**, *80*, 200–218. [CrossRef]
12. Lei, Y.; Tai, K.C. In-parameter-order: A test generation strategy for pairwise testing. In Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium, Washington, DC, USA, 13–14 November 1998; pp. 254–261.