Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 6, No. 5, October, 2022

Development of Algorithm for Calculating Data Packet Transmission Delay in Software-Defined Networks

Islam Alexandrov [1], Aslan Tatarkanov [1*], Vladimir Kuklin [1], Maxim Mikhailov [1]

[1] *IDTI RAS Institute for Design - Technological Informatics of RAS, Moscow, Russian Federation.*

Abstract

The relevance of this type of network is associated with the development and improvement of protocols, methods, and tools to verify routing policies and algorithmic models describing various aspects of SDN, which determined the purpose of this study. The main purpose of this work is to develop specialized methods to estimate the maximum end-to-end delay during packet transmission using SDN infrastructure. The methods of network calculus theory are used to build a model for estimating the maximum transmission delay of a data packet. The basis for this theory is obtaining deterministic evaluations by analyzing the best and worst-case scenarios for individual parts of the network and then optimally combining the best ones. It was found that the developed method of theoretical evaluation demonstrates high accuracy. Consequently, it is shown that the developed algorithm can estimate SND performance. It is possible to conclude the configuration optimality of elements in the network by comparing the different possible configurations. Furthermore, the proposed algorithm for calculating the upper estimate for packet transmission delay can reduce network maintenance costs by detecting inconsistencies between network equipment settings and requirements. The scientific novelty of these results is that it became possible to calculate the achievable upper data delay in polynomial time even in the case of arbitrary tree topologies, but not only when the network handlers are located in tandem.

Keywords:

Software-Defined Networks;
Data Transmission Delay;
Routing;
Network Topology;
Network Model;
Network Functions Virtualization.

Article History:

Received:	16	May	2022
Revised:	11	July	2022
Accepted:	23	July	2022
Available online:	16	August	2022

1- Introduction

Digital technology has changed organizations, especially teachers, in an irreversible way. Digitization is transforming organizations, work environments, and processes, creating new challenges that must be addressed by leaders. The latest findings of a Eurobarometer survey show that most respondents believe that digitization has a positive impact on the economy (75%), quality of life (67%), and society (64%). Indeed, in the last decade, the daily lives of people and the operations of businesses have been greatly transformed by digital technologies [1]. The digital transformation of an organization mainly refers to the adoption of a portfolio of technologies such as the Internet of Things (IoT), digital platforms, social media, artificial intelligence (AI), machine learning (ML), and Big Data. At the macro level, the transition to modern technologies defines new competition mechanisms, structures, new work systems, and interactions that may emerge. At the micro-level, digitization affects the dynamics of organizations, their processes, and the required new skills of their employees from the highest to the lowest levels [2]. Based on the above, it seems that leaders are key pillars in the development of the digital culture of a modern organization [3].

The Internet, the current architecture of which was developed more than 40 years ago, is gradually suffering from numerous disadvantages. Its scale has increased significantly, and the current load and traffic on the network have nearly reached peak levels. The universal architecture of the telecommunication network is currently quite demanding to ensure

* **CONTACT**: as.tatarkanov@yandex.ru

DOI: http://dx.doi.org/10.28991/ESJ-2022-06-05-010

speedy access to information. In addition, modern technological advancements in wireless communication have caused a sharp increase in intelligent terminals and the demand for mobile Internet access. Furthermore, transferring Internet application support from traditional web browsers to local platforms for accessing content [1], such as social networks and online games, is rapidly growing. As a result, mobile Internet has become an integral part of the Internet, posing the following problems: each user can freely access the network at any time and from any location, making the network vulnerable to malicious attacks, and data collected by the provider can be used for any purpose without repercussions. Furthermore, many network resources are not incorporated into the Internet architecture, resulting in a decrease in its efficiency; the average use of communication channels in the backbone network is only about 30%–40%, whereas the average use of the access network is less than 10% [2]. Information-centric networking (ICN), network functions virtualization (NFV), and software-defined networking (SDN) [3-5] are some of the current technologies that can tackle some of the abovementioned problems caused by the evolution of the Internet.

More focus should be placed on SDN technology. The network management approach provides an efficient dynamic configuration for improving network performance and monitoring, making it more similar to cloud computing than traditional network management.

Traditional networks have a decentralized and complex static architecture, while modern networks require more flexibility and easy troubleshooting. SDN can centralize everything in one segment, separating the network packet forwarding process (the data plane) from the routing process (the control plane). This makes it possible to directly program network management and basic infrastructure by abstracting applications and network services. The most developed SDN standard is the OpenFlow protocol [6-8]. Thus, the main advantage of SDN is that the administrator can control the behavior of the entire network by changing the packet switching rules in the table of any switch. Therefore, SDN is particularly convenient for developing and modifying network applications.

The verification of SDN configuration is relevant now that some software tools, such as FlowChecker [9], Anteater [10], Veriflow [11], NetPlumber [12], and the atomic predicates (AP) Verifier [13], have already been developed. These tools are designed to check the various properties of a network. Finite-labeled transition systems were developed for the static analysis of various SDN configurations. They were tested for properties of node reachability, absence of cyclic routes, and other parameters using various software algorithms, such as Binary Decision Diagram (BDD) based calculations (FlowChecker [14]) and satisfiability (SAT) problem-solving procedures (Anteater [15]). The following systems were developed for operational verification: VeriFlow for the verification of network invariants, NetPlumber for packet header analysis, and the APVerifier system for describing the totality of all patterns of packet switching rules through a limited set of atomic formulas, which can significantly reduce the data size and accelerate the verification of reachability requirements.

The disadvantage of existing verification systems is that almost all use formal description methods (regular expressions or temporal logics), not designed to describe static system objects (SDN configurations).

In addition, the current network configuration verification tools are applicable only for checking a narrow predetermined set of properties, such as the absence of routing loops and black holes. In this case, the possibility of checking many other properties, such as reachability, is absent or requires modification of the original algorithms and program code of their implementation. One of such problems of connection characteristics prediction is estimating the maximum end-to-end transmission time of data packets. The reason is that existing methods and tools for constructing such estimates are intended for use in highly specialized networks or have relatively low accuracy, limiting their practical application.

Therefore, to solve the urgent problem of providing a high rate and high throughput of information networks, it is necessary to optimize routing policies and develop a mathematical algorithm for calculating the packet transmission delay through a sequence of switches, taking into account the increase in traffic volumes of the global network. Therefore, this work aimed to develop an algorithm for calculating data packet transmission delays in SDN.

1-1- Literature Review and Analysis

Each network technology must meet specific functional requirements that are implemented in its components, including components' interrelationships and the principles that govern the structure and development of the system. Network traffic is the most critical factor in determining the number of resources, audience behavioral characteristics, pricing strategies, and network architecture. For ensuring efficient and stable network operation, a modification strategy must be developed to enable the network to maintain the necessary level of functionality under increasing loads. The network infrastructure should have high performance, reliability, energy efficiency, and scalability potential as critical characteristics.

Implementing a new network service requires introducing entirely new technological solutions in the existing infrastructure and network, and the implemented technology must not affect the operation of other components. In this case, according to the concept of NFV, it is proposed to abandon the use of hardware devices for each function in favor

of some virtual machines, resulting in the virtualization of different classes of network node functions, which dramatically simplifies network operation, reduces the number of required resources, and simplifies network management. To date, the following architectural systems have been identified as systems that provide solutions to specific network traffic problems:

- DiffServ architecture, which was designed to solve problems related to the management of different types of traffic and the creation of classifications;
- IntServ architecture, which provides a practical solution for resource reservation and traffic volume control tasks;
- Information-oriented network (ION) architecture, which makes it possible to meet the ever-growing and changing requirements of users, such as changes in the principles of obtaining information and confidential data about a user; and
- SDN network architecture, which uses open-interface software. SDN uses centralized intelligent management for network monitoring, providing additional data flow protection.

Let us consider some relevant studies in the subject area under review. In a previous study [16], the authors analyzed the advantages and disadvantages of SDN. The advantages include ease of management and configuration. However, routing management in such networks addresses various problems, tasks, and parameters. Iqbal et al. (2022) [17] considered forming flow rules using machine learning in SDN-based edge computing. Simultaneous processing of multiple streams by substitution rules is problematic, which dramatically reduces the network performance. The authors propose using a programmed controller with a built-in rule formation mechanism. The study results show the performance achievements of this method.

The authors in Swaminathan et al. (2021) [18] developed a routing algorithm based on a graphical neural network. This method uses generalized information from the SDN controller to predict the optimal path with the minimum average delay between source and destination nodes in SDN. Experiments have been conducted to verify the robustness of routing algorithms to changes in network structure and the influence of various hyperparameters. The results show the algorithm's robustness to the changing graph structure of the network.

Liu et al. (2022) [19] developed a traffic anomaly detection scheme for non-directional attacks in optical SDN. The scheme included a functional design and an extension of the OpenFlow protocol, using an adaptive threshold detection algorithm based on a time sliding window (TSW-ATD) and a repeated flow detection algorithm (RFD) for completion. The result was a template-based measurement method. Furthermore, the authors improved the network parameters based on the experiments performed.

Scaranti et al. (2022) [20] devoted to unsupervised online anomaly detection in SDN. The developed system is based on online clustering to detect attacks in the evolving SDN network using the entropy of source and target IP addresses and ports. This technique allows for a comprehensive analysis that provides insight into the intensity, seasonality, and type of attack. Hari Krishna & Sharma (2021) [21] presented an overview of applications' programming interfaces in SDN and the virtualization of network functions is given. The authors propose a list of effective and efficient orchestration approaches differing in design for microservices in SDN and NFV domains

Yang et al. (2021) [22] highlight flow routing as one of the most critical problems in SDN. There was proposed optimization of the throughput with the constraint under which the probability of forwarding reliability of each pair of switches should exceed a threshold value. Experimental results and results of large-scale network simulation reveal that the developed algorithms improve the network throughput by almost twice and reduce the maximum number of required flow inputs by about 53.1% compared to the existing solutions according to reliability requirements. Cheng et al. (2021) [23] presented a malicious payload based on machine learning is considered. The authors propose a new OpenFlow-enabled deep packet inspection (OFDPI) approach based on the SDN paradigm to provide adaptive and efficient packet inspection. Numerical experimental results show that this method improves detection accuracy with acceptable overhead costs.

Mohamed et al. (2021) [24] presented an overview of SDN for resource allocation in cloud computing is provided. The authors studied resource allocation in cloud computing based on SDN. The work resulted in a classification of resource allocation mechanisms, including both cloud computing and SDN domains. In Cui et al. (2021) [25], the focus is on denying SDN service. The authors review DDoS (Distributed Denial of Service) attack detection mechanisms and give their classification. As a result, it was found that DDoS detection mechanisms based on machine learning and based on thresholds are the two most popular technologies used to detect DDoS attacks. The paper also discusses the challenges and future directions of detection.

Nguyen et al. (2022) [26] evaluated the performance of switching Moving Target Defenses (MTD) mechanisms. In addition, the SRN performance model for different MTD switching strategies is proposed. We may conclude from this research that the MTD strategy impacts the system's performance.

In Dou et al. (2021) [27], the maintenance of network programmability for software-defined wide area networks (SD-WAN) in case of multiple controller failures were described. The authors proposed a solution for restoring autonomous flows. To adjust the cost of managing autonomous switches based on a given control capability of active controllers, Matchmaker changes the paths of some autonomous flows. Simulation results show that Matchmaker outperforms existing solutions by increasing the number of recovered autonomous flows up to 45% in the ATT topology and up to 77% in the Belnet topology.

Hu et al. (2021) [28] developed a method for the reliable placement of controllers in case of channel failures in SDN. For this purpose, the CPP for multi-channel link failures (CPP-MLF) was studied. Experimental results show that the proposed heuristic algorithm performs well with the number of controllers, worst-case delay, and reliability while providing acceptable run-time overhead costs. In Salman & Wang (2021) [29], path selection algorithms and target rate adaptation functions. The authors discovered that the most appropriate system for SDN is one in which paths are computed using a routing model without attention. The RACKE+AD matching model maximizes throughput, provides better utilization of network resources and minimizes delay.

Thus, SDN is characterized by a three-tier architecture. The user, information, and control planes are delimited in a logical controller, which has a global network representation and can manage the routing of traffic flows. Using the structure of SDN makes it possible to achieve centralized data management without proceeding from the used network technologies, which are the basis for connecting the devices. Therefore, it is the controller, the centralized control point, the way to implement the functionality required for the entire operation of applications in the network.

In the process of network design, it is increasingly necessary to solve problems associated with the choice of a rule set for the service of some package, considering the settings of network nodes, when the network will meet specific requirements (called the packet routing policy), such as:

- The network must provide subscribers with the services for which it was created;
- The packet processing rules to be developed must consider the administrative requirements of the particular;
- It is necessary to use the computing resources allocated to the network as efficiently as possible.

It is important to note that each network forms its routing policy, which can be implemented differently. Many tools work in automatic mode, but they do not cover the full range of possible policies. As a result, it is not always possible to identify and resolve policy inconsistencies so that the network can function as efficiently as possible, which leads to the need to use high-level routing policies in the design and prescribe them in the settings of nodes. In doing so, service protocols are adjusted to the needs of each particular network. Consequently, when designing complex systems, the following problems must be solved: automating the processes that correlate routing policies and developing tools that can calculate most errors quickly and accurately. The system's main requirement is to detect such problems before the configuration is performed on the whole network.

To obtain an objective and reasonable design result, which makes it possible to understand how the network parameters meet the customer's requirements, creating a model with similar characteristics and analyzing its properties is necessary. According to the analysis results, the conclusion is that the network meets the requirements put forward. However, the existing network architecture for a long time did not allow such analytical studies to be fully developed, which is associated with some problems.

- The problem of assessing the state of the entire network in a situation where any of its nodes at any time can change (asynchronously, without coordination with other nodes) the rules of processing data packets;
- The problem of describing the state of the network based on the allocated from a large number of service protocols, which interact in many different ways, the rules of service data packets.

The opportunity to solve such problems appeared after introducing the SDN concept. In that construction, the emphasis is not on the network's reliability (the fundamental principle taken as a basis for the Internet) but on its ease of management. Furthermore, the development of the SDN concept allowed new and developing methods to analyze and verify the network parameters based on their configuration.

2- Research Methodology

The procedure of modeling the controller's behavior, which organizes and participates in the work of the switching network, may involve the creation of numerous formal SDN models, each of which will meet a specific set of requirements. In this case, it becomes possible to solve the main analysis tasks and verify the routing policies. Each selected model must undergo a verification procedure, where it will be confirmed that the model in question fully meets the pre-declared properties and features.

The principle of SDN behavior is based on the specific behavior of each component. Consequently, the construction of an SDN model requires defining an exact list of component models, which must be consistent with each other. Moreover, they must meet the following set of requirements:

- The property of expressivity is one of the key factors without which it is impossible to form a model. Therefore, it is important to determine the key features of the system's functioning and structure through a specialized OpenFlow network protocol.
- The presence of a description of all the compositional characteristics of the models. It is necessary to consider that the models included in SDN can consist of even simpler structural elements; hence, it is necessary to create a description of these models.
- Full consistency with specification languages.
- Another important component is the use of various levels of model abstraction. For example, when it is necessary to model the individual elements of SDN, whose operation is determined by the routing policy, the most appropriate in such a situation is to use two models, concrete and abstract.
- Compactness. The model description under formation should contain all the necessary information but be brief at the same time.
- The efficiency of solving analytical problems for models. It compromises the model expressiveness, the variety of operations allowed in modeling with their economic performance, and the complexity of the analytical problems solved based on the model. For example, high values of computational complexity indicate that the system lacks efficient algorithms to solve verification problems.
- Availability of tool support. It is often much easier to use ready-made software tools for analyzing network behavior than to create new verification algorithms. It becomes possible to achieve significant savings in time and resources in this case.

Property classes can classify the behavioral features of SDNs into local, globally static, globally dynamic, and real-time. For formally representing these properties, it is advisable to use different specification languages, which helps to reflect their diversity in the best possible way. In this case, the critical role is to observe the hierarchy. Each language necessary to create a description of complex parameters should include a language that describes the primary and elementary parameters.

The main task of the verifier is to perform instrumental verification of SDN for compliance of system elements with a specific routing policy defined by various input and output specifications. The main functional parts include the following:

- Model builder. Its task is to accept the incoming file, which contains a description of the SDN topology and the rules of the OpenFlow protocol: Then, the model builder analyzes the provided information, based on which it builds a tree of binary decision diagrams, which will be entirely consistent with the semantics of the static model. As a result, their behavior in SDN will be fully prescribed.
- Specification analyzer: The necessity of this element is explained by conducting a detailed parsing of the specification language expressions, for which the corresponding routing policy is used. The analyzer's input is a text file containing many specification definitions.
- SDN verifier: It is needed to verify the parameters of a given configuration according to its many requirements. As a result, this tool will detect all existing errors.

Based on the information in the specification file and the corresponding grammar, the result of such parsing is an abstract syntax tree with leaf nodes. The relationships of these nodes to each other and constants will modulate all other SDN components.

In this paper, to calculate the maximum delay of data packet transmission through the network, we applied the methods of network calculus based on the Min-Plus system theory. Their distinctive feature is the ability to construct two-way estimates of information packet transmission time for the sequence of handlers with high accuracy. Furthermore, the methods of network calculus theory based on Min-Plus system theory allow investigating the connection between the performance of network elements and the quality of data processing service. The theory is based on the principle of calculating deterministic estimates based on the best and worst scenarios of functioning isolated network fragments and the following optimal combination of possible scenarios [30, 31].

The methods of the network calculus theory allow us to evaluate the data flow at the network output and analyze and study in detail the relationship between the quality of service and the parameters of particular elements of the network. It is worth remembering that such models have a wide range of possibilities; for example, they make it possible to

consider boundary scenarios for individual components. As a result, by collecting different estimates and comparing their results with each other, it is possible:

– To determine the worst quality service when transmitting the information flow through the network with a specific set of parameters;

– To formulate a list of minimum requirements for the particular network components. Such requirements should provide the necessary level of quality in servicing the flows.

A schematic representation of the stages and sequence of the study in this paper is illustrated in Figure 1.

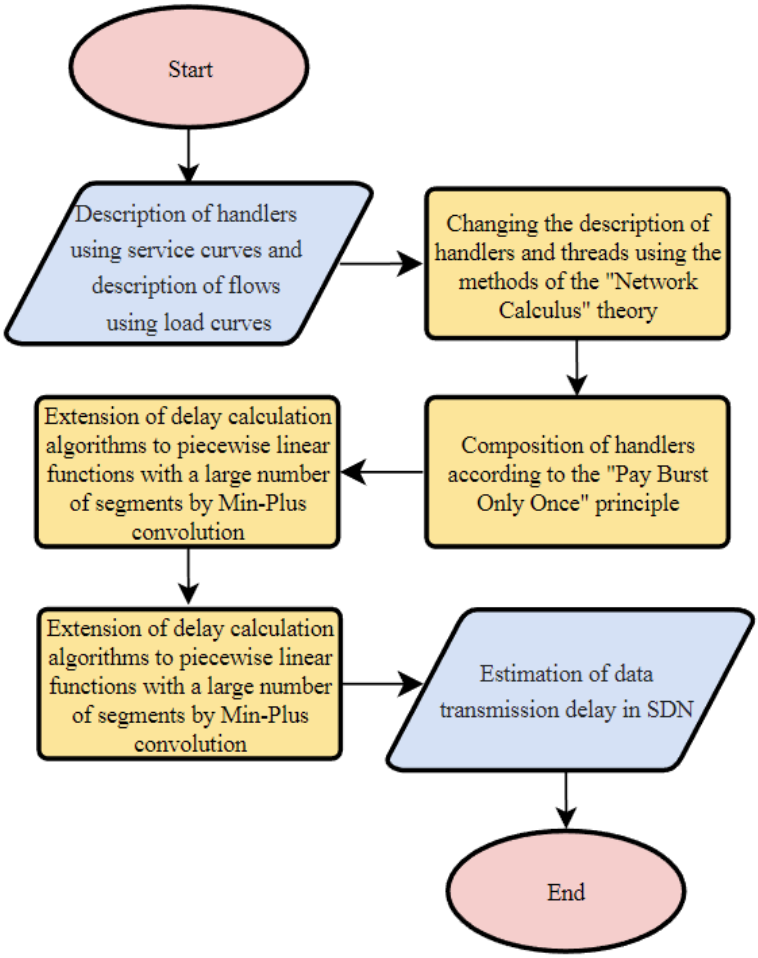

Figure 1. Schematic representation of the stages of data delay estimation in SDN using generalized algorithms based on network calculus theory and Min-Plus convolution methods

To determine the worst quality service when transmitting the information flow through the network with a specific set of parameters.

To formulate a list of minimum requirements for the particular network components. Such requirements should provide the necessary level of quality in servicing the flows.

The main procedures required for building models of SDN data flows and handlers using network calculus methods are:

- Partitioning the network into specific handlers, each of which can model differently sized elements, e.g., physical lines for switching, individual switches, subnets;
- Creating a description of the data flows that may be required for sequencing data transfers using handlers;
- Analyzing the task, using parameters by type of load curve or constraints in terms of rate of arrival of information to the network;
- Assigning, using a parameter such as the service curve, each handler's behavior, allowing the proper allocation of resources between the flows coming through the handlers.

A mandatory part of the model is a number of cumulative time functions F:

$$\begin{aligned} &F = \{ f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \cup \{+\infty\} \,|\, LeftContinuous(f) \wedge NonDecresing(f) \wedge Causal(f) \}; \\ &NonDecresing(f) = \forall t_1 \leq t_2 : f(t_1) \leq f(t_2); \\ &LeftContinuous(f) = \forall t_0 \in \mathbb{R}: \lim_{t \to t_0 - 0} f(t) = f(t_0); \\ &Causal(f) = \forall t = 0 \to f(t) = 0 \end{aligned} \quad (1)$$

They reflect the required time to transfer the given information. Set F is a set of non-decreasing and left-continuous functions, the argument of which always takes non-negative values. One of such functions, A ∈ F, is used to model the arrival of information to a specific handler S, making it possible to describe the dependence of the total amount of data in a flow that came to this handler on the time parameter. The particular function, D ∈ F, makes it possible to describe a specific dependence between the total amount of data in the flow and the transmitter that transmits it. Figure 2 shows a graphical representation of these functions.

Each handler S not only transmits the received information but also performs a detailed comparison of the arrival function and the departure function. It is worth noting that the departure function, in this case, depends on the A function and parameters of S. Thus, it is possible to conclude that to create a description of handler model S, it is necessary to list all pairs of functions that correspond to this handler and also to use various additional indicators graphically interpreted in Figure 2: lag b(t), period [SBP(t); EBP(t)] of lag and delay d(t) of handler S while serving a flow with a pair of cumulative functions A, D∈F, ⟨A, D⟩∈S.

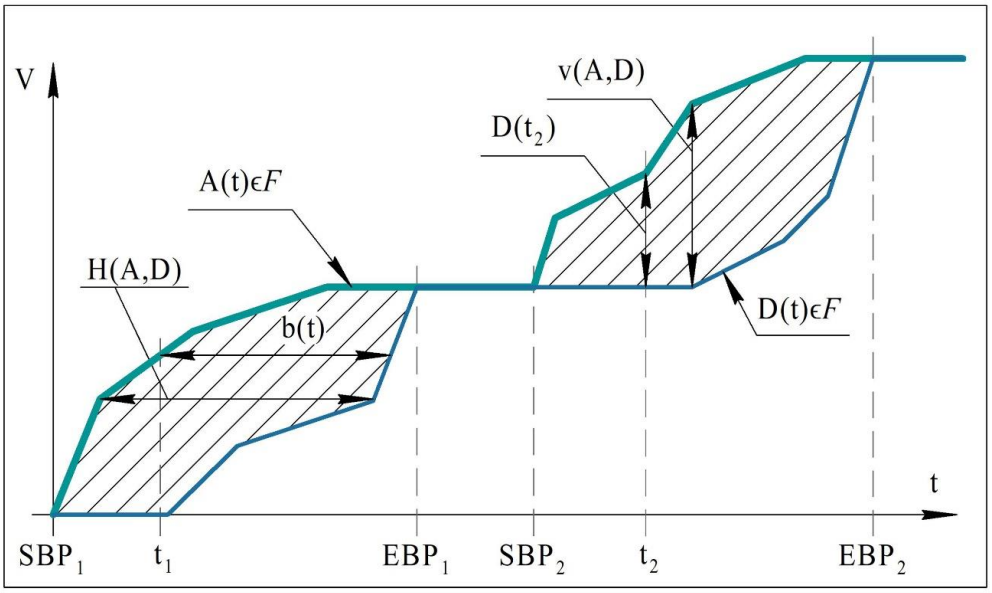

Figure 2. Graphical representation of pair of cumulative functions A, D∈F, ⟨A, D⟩∈S, simulating the processes of data arrival to handler S and their further departure

Lag is a function of quantifying the amount of data stored in the handler and the time in which it occurs:

$$b(t) = A(t) - D(t) \quad (2)$$

The time during which a certain portion of the information received by the handler will be in it is a delay during the operation of handler S:

$$d(t) \leq \inf \{ \tau \geq 0 \,|\, A(t+\tau) = D(t+\tau) \} \quad (3)$$

The time during which a positive numerical value will characterize the lag indicator b(t) is the lag period. If the fixed moment t∈ℝ is within the lag period [SBP(t); EBP(t)], then the boundaries of the lag period will be described by the following set of formulas:

$$SBP(t) = \sup \{ u \leq t \,|\, A(u) = D(u) \} \quad (4)$$

$$EBP(t) = \inf \{ u \geq t \,|\, A(u) = D(u) \} \quad (5)$$

The above approach to modeling the work of the handler S using the full list of arrival and departure functions is not practical. A simpler handler model can be represented as a relationship between incoming and outgoing data using

service functions βs and productivity βP. The strict service curve of handler S is function βs∈F reflecting the service process of flows with arrival function A and departure function D, ⟨A,D⟩∈S,, during each lag period [s; t], when it processes at least βs(t - s) of data:

$$D(t)-D(s)\geq \beta_s(t-s) \tag{6}$$

The performance function P∈F of handler S expresses the amount of data that handler S transfers in a given time, assuming that it is fully loaded. Handler S performance curve is function βP∈F, reflecting the fact that for each time interval of length τ≥0, the value of βP(τ) must not exceed the amount of data that the handler can handle during this interval:

$$\forall t:\forall \tau: P(t+\tau)-P(t)\geq \beta_p(\tau) \tag{7}$$

Since the performance P makes it possible to calculate departure function D∈F for arbitrary arrival function A∈F, S can be specified without explicitly listing all pairs of form ⟨A, D⟩ that satisfy it:

$$D(t)=A(t)-\sup_{s\leq t}\left\{\left[\left(A(t)-A(s)\right)-\left(P(t)-P(s)\right)\right]^{+}\right\} \tag{8}$$

$$D(t)=\inf_{s\leq t}\left\{A(t)-\left[\left(A(t)-A(s)\right)-\left(P(t)-P(s)\right)\right]^{+}\right\} \tag{9}$$

$$D(t)=\inf_{s\leq t}\left\{A(s)+\left(P(t)-P(s)\right)\right\} \tag{10}$$

Performance curve βP makes it possible to plot the worst service (lower) estimate of the departure parameter D in handler S considering function A:

$$D(t)\geq \inf_{s\leq t}\left\{A(s)+\beta_p(t-s)\right\} \tag{11}$$

3- Results

Simulating handler features is a rather complicated process, especially if there is a need to build an exact arrival function. The problems arise from the need to implement dynamic changes in the information transfer rate because only it becomes possible to prevent potential overloading of the transmitted data flow. An effective technique for this is to replace the arrival function A with a load curve, which gives an upper estimate of the data flow rate limitation at all possible intervals of a given length. The load curve is understood as such function A∈F for which each time interval of length τ and the amount of information transmitted during this interval does not exceed the value α(τ) determined by the formula:

$$\forall t:\forall \tau: A(t+\tau)-A(t)\leq \alpha(\tau) \tag{12}$$

In any lag period, when the shaper buffer contains data whose transfer rate is not inferior to the rate ρ of the arrival of tokens into the container. Thus, the shaper can be described by a strict service curve $\beta_s(t)=[\rho t]^{+}\epsilon F$.

$$\forall t: D(t)-D(SBP(t))\geq \beta_s(t-SBP|(t)) \tag{13}$$

As a standard service curve, the function $\beta=[\rho t+\sigma]^{+}$ is suitable here:

$$D(t)=\inf_{s\leq t}\left\{A(s)+\beta(t-s)\right\} \tag{14}$$

Note that for each value of s, the function's graph $f_s(t)=A(s)+\beta(t-s)$ under *inf* is plotted by shifting the service curve graph β by s units to the right and by A(s) units up. It is possible to duplicate the graph of function β in all coordinate systems to construct a set of functions *fs*. To obtain such a graph it is required to shift the zero of the initial system to an arbitrary point of graph A. As a result, some departure function D will be given, which will show that this graph coincides completely with the bottom edge of the constructed set of points.

Next, it is necessary to pay more attention to Figure 2. It shows the graph of function A∈F characterizing the information arrival process to a special shaper, as well as the service curve, which corresponds to it. The elements c) and d) here show the estimations of transfer function based on the strict and standard service curves $\beta_s(t)=[\rho t]^{+}$ and $\beta=[\rho t+\sigma]^{+}$.

Let us consider element c) of Figure 3 in more detail. According to function A, the first amount of data arrives in the shaper at a constant rate during the period [t_1; t_3], and by time t_1, the container with tokens is entirely complete. As the packet arrival rate exceeds ρ, the number of tokens in the container gradually decreases until the container becomes empty at time t_2. The information transfer rate will decline to ρ, resulting in a certain part of the received information being stored within a special shaper buffer. By final period t_4, this buffer is emptied, increasing the total number of tokens in the container. The increment is carried out until the entire volume of the container is filled.

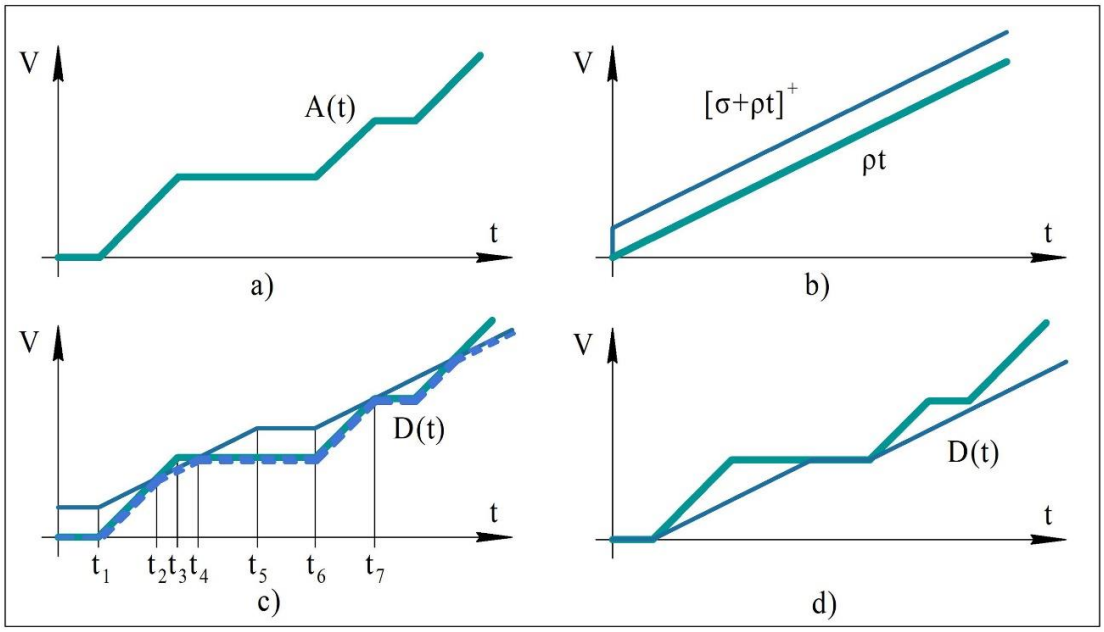

Figure 3. Building the departure function D∈F under correspondence with the arrival function A∈F by the shaper, working according to the flow algorithm

The number of tokens passed through the container corresponds to the function M∈F, whose graph is shown as a thin solid line. A regulator is such handler S for which an envelope function E∈F can be built. Regardless of the function A used, the amount of information transmitted cannot exceed α(u) in total.

It is further proposed to consider the operation of the regulator with E∈F and the handler in a little more detail. In this configuration; the data flow A coming to handler S and coming out of the regulator is limited by the envelope function $A(t) - A(s) \leq E(t-s)$. Thus, the envelope function E∈F of the regulator is also the load curve for the data stream A that enters the handler connected to its output. As a delay rate handler RL, a special category of handler S is considered, whose use makes it possible to transmit input information with constant rate R and response T. This handler is shown in Figure 3. It is worth noting that each of these RL handlers is characterized by its service curve $\beta = R(t-T)^+$.

The calculation of the general service curve for a system consisting of an arbitrary number of tandem RL handlers is performed by the following formula:

$$\beta = \beta_1 \otimes \beta_2 \otimes ... \otimes \beta_n = \min(R_1, R_2, ..., R_n)t \otimes \delta_{T_1+T_2+...+T_n} \quad (15)$$

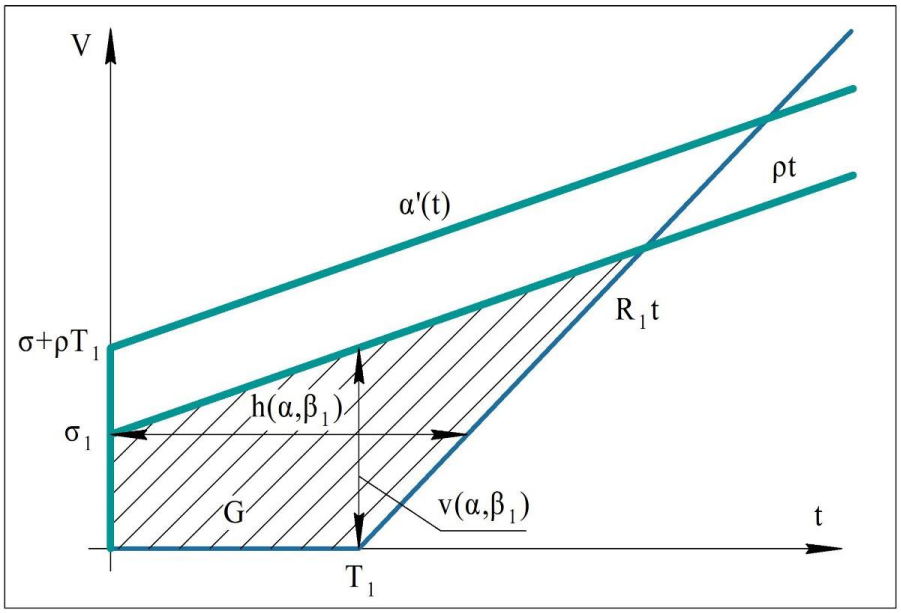

Figure 4. Calculation of upper estimates for lag b(t), delay d(t), and arrival curve α'(t) of the output flow bounded by function a(t)=pt+σ and RL handler with service curve β(t)=R(t-T)+

A test program was developed to experimentally estimate the delay of data transmission through the network based on the following assumptions: switches implement output buffering without forming cyclic dependencies of data flows; throughputs of flows are limited at the peripheral switches of the network by shaping according to the flow algorithm. Next, a network simulation model was created based on the Network Simulator 3 (NS-3) computer network simulation library [32]. Finally, to test the developed tools, two parameterized families of test scenarios were built, which use a linear network topology (Figure 5) and a neck topology (Figure 6).

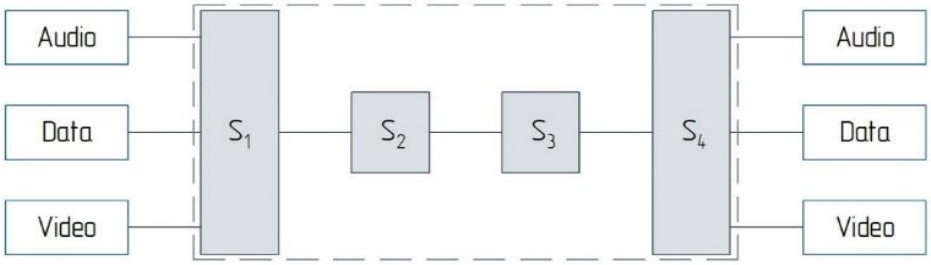

Figure 5. Example of network topology for sequential test scenario

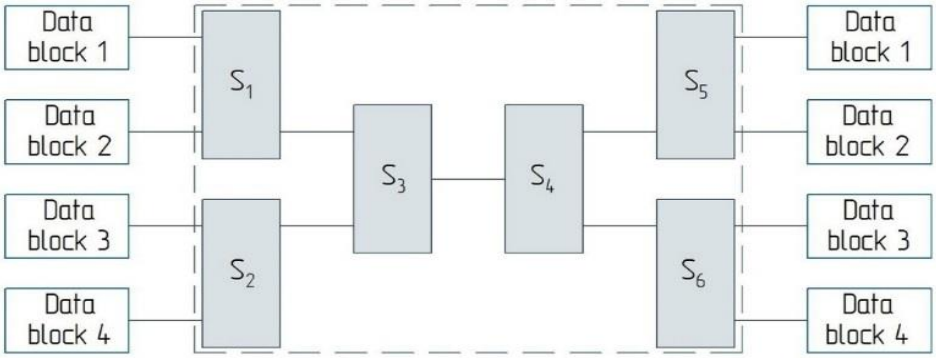

Figure 6. Example of network topology for neck test scenario

During the experiments, the flow rate changed depending on the configuration of the initial topology so that the total rate of the flows transmitted through the topology was equal to the channel throughput. The experimental results show that the estimation of the data transmission delay increases almost linearly for the linear topology and the number of switches. Figure 7 shows the theoretically calculated upper delay estimates and the maximum delay values derived from the simulation for the neck network topology. The test results demonstrate sufficiently high accuracy of theoretical estimation methods when using a small number of flows for the neck topology scenario.

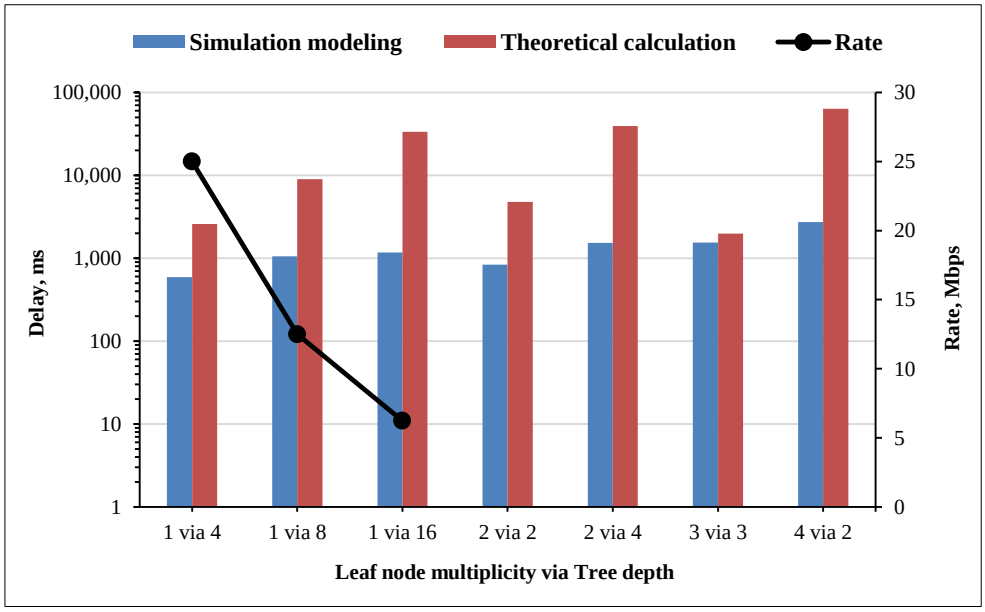

Figure 7. Dependence of upper theoretical estimates and maximum simulated data transmission delays for different combinations of topology and number of switches

4- Discussion

Current traffic management concepts are based on Quality of Service (QoS) mechanisms determined by assessments of some network indicators like throughput, information transmission delay, and packet loss. The Quality of Experience (QoE) becomes critical for users, whose key indicators are response time and disconnection time. These parameters are properties of network performance. In order to determine the properties of this type, it is necessary to use interval arithmetic operations that are fundamentally time-dependent.

Quite a large number of researchers suggest using an approach requiring the calculation of upper estimates to determine the end-to-end packet delay and then summarizing these estimates. However, it should be understood that if some packet experiences the maximum delay in one network element, the delay in another element may not be the maximum. So the total estimate as the sum of the maximum delays will be overestimated.

This paper considers a method to analyze the performance properties of computer networks by their configurations based on calculating data packet transmission delays. A polynomial algorithm for estimating the maximum end-to-end delay of data packet transmission has been developed for networks with a tree-like topology. Based on the algorithm for computing the deconvolution, the presented algorithm for building upper delay estimates for direct-coupled networks allows us to extend the application area of the previously proposed methods for building upper delay estimates to a broader class of functions – piece linear functions with an arbitrary number of segments. Besides, it can solve the mentioned problem and provide the most accurate delay estimates, which can be calculated within the assumptions used by the network calculus theory. It must be understood that the current algorithms can only be used for networks whose data flows are strictly defined by piecewise linear curves of one or two segments [33-35]. The main advantage of our proposed approach is that it becomes possible to overcome such limitations and consider arbitrary tree topologies.

Consideration should also be given to the limitations of the results obtained in this paper; the calculations given in this paper are suitable for use only in those SDNs where data flows do not intersect and do not compete for handler resources. The integrated services model fulfills this assumption, but many modern networks do not meet these assumptions.

5- Conclusion

The current load and traffic have peaked because there is currently a need to provide fast access to the Internet world's network information. Therefore, there is a need to develop a strategy for modifying this network to maintain the necessary level of functioning with increasing user needs. This work found that SDN technologies are the most convenient for developing and modifying network applications.

This paper considers a method for determining the timeliness of data transmitted through the network based on calculating estimates of data transmission delay. In this research work, the upper estimate for the end-to-end delay value was obtained using network calculus methods, which allowed us to extend the scope of existing algorithms for estimating the transmission time of data packets for the case of an arbitrary number of segments.

The developed algorithm for calculating data packet transmission delays in SDN has been analyzed using a network simulation model based on the computer network simulation library. Experimental results of the computational experiment to estimate the data packet transmission delay through the network showed that the developed verification tool could evaluate the configuration efficiency of the SDN elements using the model of integrated services. Thus, the proposed algorithm for calculating the upper estimate for packet transmission delay can reduce network maintenance costs by detecting inconsistencies between network equipment settings and requirements. The results obtained in this research work can later be used to design and modernize real-time computing systems.

6- Declarations

6-1- Author Contributions

Conceptualization, I.A. and A.T.; methodology, V.K.; software, M.M.; validation, I.A., A.T. and V.K.; formal analysis, M.M.; investigation, I.A.; resources, A.T.; data curation, V.K.; writing—original draft preparation, M.M.; writing—review and editing, I.A.; visualization, A.T.; supervision, V.K.; project administration, M.M.; funding acquisition, I.A. All authors have read and agreed to the published version of the manuscript.

6-2- Data Availability Statement

Data sharing is not applicable to this article.

6-3- Funding

Selected findings of this work were obtained under the Grant Agreement in the form of subsidies from the federal budget of the Russian Federation for state support for the establishment and development of world-class scientific centers performing R&D on scientific and technological development priorities dated April 20, 2022, No. 075-15-2022-307.

6-4- Institutional Review Board Statement

Not applicable.

6-5- Informed Consent Statement

Not applicable.

6-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

7- References

[1] Gunatilaka, D. (2013). Recent Information-Centric Networking Approaches. Recent Information-Centric Netw. Approaches, 1-16.

[2] Fisher, W., Suchara, M., & Rexford, J. (2010). Greening backbone networks. Proceedings of the First ACM SIGCOMM Workshop on Green Networking - Green Networking'10. doi:10.1145/1851290.1851297.

[3] Yu, K., Eum, S., Kurita, T., Hua, Q., Sato, T., Nakazato, H., Asami, T., & Kafle, V. P. (2019). Information-Centric Networking: Research and Standardization Status. IEEE Access, 7, 126164–126176. doi:10.1109/ACCESS.2019.2938586.

[4] Kim, D., Kim, Y. H., Park, C., & Kim, K. Il. (2018). KREONET-S: Software-defined wide area network design and deployment on KREONET. IAENG International Journal of Computer Science, 45(1), 27–33.

[5] Mijumbi, R., Serrat, J., Gorricho, J. L., Latre, S., Charalambides, M., & Lopez, D. (2016). Management and orchestration challenges in network functions virtualization. IEEE Communications Magazine, 54(1), 98–105. doi:10.1109/MCOM.2016.7378433.

[6] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J. (2008). OpenFlow. ACM SIGCOMM computer communication review, 38(2), 69-74. doi:10.1145/1355734.1355746.

[7] McGeer, R. (2012). A safe, efficient update protocol for openflow networks. Proceedings of the First Workshop on Hot Topics in Software Defined Networks - HotSDN '12. doi:10.1145/2342441.2342454.

[8] Azzouni, A., Trang, N. T. M., Boutaba, R., & Pujolle, G. (2017). Limitations of OpenFlow topology discovery protocol. 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net). doi:10.1109/MedHocNet.2017.8001642.

[9] Al-Shaer, E., & Al-Haj, S. (2010, October). FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures. Proceedings of the 3rd ACM workshop on Assurable and usable security configuration - SafeConfig '10. doi:10.1145/1866898.1866905.

[10] Mai, H., Khurshid, A., Agarwal, R., Caesar, M., Godfrey, P. B., & King, S. T. (2011). Debugging the data plane with anteater. Computer Communication Review, 41(4), 290–301. doi:10.1145/2043164.2018470.

[11] Khurshid, A., Zou, X., Zhou, W., Caesar, M., & Godfrey, P. B. (2013). VeriFlow: Verifying Network-Wide Invariants in Real Time. 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2-5 April, 2013, Lombard, United States.

[12] Kazemian, P., Chang, M., Zeng, H., Varghese, G., McKeown, N., & Whyte, S. (2013). Real time network policy checking using header space analysis. 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2-5 April, 2013, Lombard, United States.

[13] Yang, H., & Lam, S. S. (2016). Real-time verification of network properties using atomic predicates. IEEE/ACM Transactions on Networking, 24(2), 887–900. doi:10.1109/TNET.2015.2398197.

[14] Henzinger, T. A., Nicollin, X., Sifakis, J., & Yovine, S. (1994). Symbolic model checking for real-time systems. Information and computation, 111(2), 193-244. doi:10.1006/inco.1994.1045

[15] Bouillard, A., Boyer, M., & Le Corronc, E. (2018). Deterministic network calculus: From theory to practical implementation. John Wiley & Sons, Hoboken, United States. doi:10.1002/9781119440284.

[16] Nayyer, A., Sharma, A. K., & Awasthi, L. K. (2021). Learning-based hybrid routing for scalability in software defined networks. Computer Networks, 198, 108362. doi:10.1016/j.comnet.2021.108362.

[17] Iqbal, S., Maryam, H., Qureshi, K. N., Javed, I. T., & Crespi, N. (2022). Atomized flow rule formation by using machine learning in software defined networks based edge computing. Egyptian Informatics Journal, 23(1), 149–157. doi:10.1016/j.eij.2021.10.001.

[18] Swaminathan, A., Chaba, M., Sharma, D. K., & Ghosh, U. (2021). GraphNET: Graph Neural Networks for routing optimization in Software Defined Networks. Computer Communications, 178, 169–182. doi:10.1016/j.comcom.2021.07.025.

[19] Liu, T., Wang, H., & Zhang, Y. (2022). A traffic anomaly detection scheme for non-directional denial of service attacks in software-defined optical network. Computers and Security, 112, 102467. doi:10.1016/j.cose.2021.102467.

[20] Scaranti, G. F., Carvalho, L. F., Barbon, S., Lloret, J., & Proença, M. L. (2022). Unsupervised online anomaly detection in Software Defined Network environments. Expert Systems with Applications, 191. doi:10.1016/j.eswa.2021.116225.

[21] Hari Krishna, S. M., & Sharma, R. (2021). Survey on application programming interfaces in software defined networks and network function virtualization. Global Transitions Proceedings, 2(2), 199–204. doi:10.1016/j.gltp.2021.08.018.

[22] Yang, X., Xu, H., Liu, J., Qian, C., Fan, X., Huang, H., & Wang, H. (2021). Achieving high reliability and throughput in software defined networks. Computer Networks, 197, 108271. doi:10.1016/j.comnet.2021.108271.

[23] Cheng, Q., Wu, C., Zhou, H., Kong, D., Zhang, D., Xing, J., & Ruan, W. (2021). Machine learning based malicious payload identification in software-defined networking. Journal of Network and Computer Applications, 192, 103186. doi:10.1016/j.jnca.2021.103186.

[24] Mohamed, A., Hamdan, M., Khan, S., Abdelaziz, A., Babiker, S. F., Imran, M., & Marsono, M. N. (2021). Software-defined networks for resource allocation in cloud computing: A survey. Computer Networks, 195, 108151. doi:10.1016/j.comnet.2021.108151.

[25] Cui, Y., Qian, Q., Guo, C., Shen, G., Tian, Y., Xing, H., & Yan, L. (2021). Towards DDoS detection mechanisms in Software-Defined Networking. Journal of Network and Computer Applications, 190, 103156. doi:10.1016/j.jnca.2021.103156.

[26] Nguyen, T. A., Kim, M., Lee, J., Min, D., Lee, J. W., & Kim, D. (2022). Performability evaluation of switch-over Moving Target Defence mechanisms in a Software Defined Networking using stochastic reward nets. Journal of Network and Computer Applications, 199. doi:10.1016/j.jnca.2021.103267.

[27] Dou, S., Miao, G., Guo, Z., Yao, C., Wu, W., & Xia, Y. (2021). Matchmaker: Maintaining network programmability for Software-Defined WANs under multiple controller failures. Computer Networks, 192, 108045. doi:10.1016/j.comnet.2021.108045.

[28] Hu, T., Ren, Q., Yi, P., Li, Z., Lan, J., Hu, Y., & Li, Q. (2021). An efficient approach to robust controller placement for link failures in Software-Defined Networks. Future Generation Computer Systems, 124, 187–205. doi:10.1016/j.future.2021.05.022.

[29] Salman, M. I., & Wang, B. (2021). Boosting performance for software defined networks from traffic engineering perspective. Computer Communications, 167, 55–62. doi:10.1016/j.comcom.2020.12.018.

[30] Le Boudec, J. Y., Thiran, P. (1999). Network Calculus Using Min-Plus System Theory. High-Performance Networks for Multimedia Applications. Springer, Boston, United States. doi:10.1007/978-1-4615-5541-4_9.

[31] Le Boudec, J. Y., & Thiran, P. (2001). Network calculus: a theory of deterministic queuing systems for the internet. Springer, Berlin, Germany. doi:10.1007/3-540-45318-0_1.

[32] Saino, L., Cocora, C., & Pavlou, G. (2013). A Toolchain for Simplifying Network Simulation Setup. Proceedings of the 6th International Conference on Simulation Tools and Techniques. doi:10.4108/simutools.2013.251735.

[33] Sinha, D., Haribabu, K., & Balasubramaniam, S. (2016). Real-time monitoring of network latency in Software Defined Networks. 2015 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). doi:10.1109/ANTS.2015.7413664.

[34] Tian, Y. Traffic and Performance Measurements in SDNs. Hong Kong University of Science and Technology. PhD Thesis, department of Electeronic and Computer Engineering, Hong Kong University of Science and technology, Clear water Bay, Hong Kong. doi:10.14711/thesis-991012730463103412.

[35] Chefrour, D. (2022). One-Way Delay Measurement from Traditional Networks to SDN: A Survey. ACM Computing Surveys, 54(7), 1–35. doi:10.1145/3466167.