*Annals of Emerging Technologies in Computing (AETiC)*
*Vol. 6, No. 1, 2022*

*Research Article*

# A Comparative and Analytical Study for Choosing the Best Suited SDN Network Operating System for Cloud Data Center

**Maiass Zaher* and Sándor Molnár**

Budapest University of Technology and Economics, Hungary
zaher@tmit.bme.hu; molnar@tmit.bme.hu
*Correspondence: zaher@tmit.bme.hu

Received: 6th December 2021; Accepted: 28th December 2021; Published: 1st January 2022

**Abstract: The growing deployment of Software Defined Network (SDN) paradigm in the academic and commercial sectors resulted in many different Network Operating Systems (NOS). As a result, adopting the right NOS requires an analytical study of the available alternatives according to the target use case. This study aims to determine the best NOS according to the requirements of Cloud Data Center (CDC). This paper evaluates the specifications of the most common open-source NOSs. The studied features have been classified into two groups, i.e., non-functional features such as availability, scalability, ease of use, maturity, security and interoperability, and functional features, such as virtualization, fault verification and troubleshooting, packet forwarding techniques and traffic protection solutions. A Decision support system, Analytical Hierarchy Process (AHP) has been applied for assessing specifications of the inspected NOSs, namely, ONOS, Opendaylight (ODL), Floodlight, Ryu, POX and Tungsten. Our investigation revealed that ODL is the most suitable NOS for CDC compared to the rest studied NOSs. However, ODL and ONOS have almost similar scores compared to the rest NOSs.**

**Keywords:** ***Analytical Hierarchy Process; Cloud Data Center; Data Center Network; Network Operating System; Software Defined Network***

## 1. Introduction

SDN paradigm introduces flexibility in data networks as network devices comply with NOS instructions by separating the control and data planes. In particular, switches in the data plane perform packet forwarding related functions as determined by the control plane. The so-called NOS runs on computer hardware [1], and the control plane represents the business logic in SDN paradigm as it is the controller of network functions. Specifically, it determines how to handle incoming data packets using protocols such as OpenFlow [2]. On the other hand, SDN paradigm also provides a network abstraction for applications at the management plane so that the operator can efficiently perform different network tasks. Figure 1 shows the layers of the SDN paradigm. The layers separation provides flexibility for introducing new solutions for problems of the traditional network paradigm. The layered architecture provided by SDN simplifies the deployment of network services and functions such as virtualization, packet forwarding, troubleshooting, etc. SDN layers communicate with each other using dedicated Application Programming Interface (API) as depicted in Figure 1. Therefore, adopting the suitable NOS has significant impacts.

CDC motivated to develop SDN paradigm as it relies on virtualization technologies to provide services within the cloud environment. In this context, the traditional network paradigm does not provide the required flexibility to apply these technologies. In addition, data centers have unique traffic patterns, which are different from those in the traditional networks [3]. As a result, CDC becomes one of the most implemented use case of SDN paradigm [4]. Therefore, many commercial companies, organizations and

Maiass Zaher and Sándor Molnár, "A Comparative and Analytical Study for Choosing the Best Suited SDN Network Operating System for Cloud Data Center", Annals of Emerging Technologies in Computing (AETiC), Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 43-59, Vol. 6, No. 1, 1st January 2022, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2022.01.003, Available: http://aetic.theiaer.org/archive/v6/v6n1/p3.html.

research centers have developed NOSs, but no specific NOS that can be used for all use cases because there are several alternatives according to the requirements. Hence, the research problem was embodied in answering the following questions:

(i) Which NOS is the best suited for CDC?

(ii) What are the functional and non-functional specifications to be verified for this use case?

(iii) How important is each of the specifications to answer the first question?

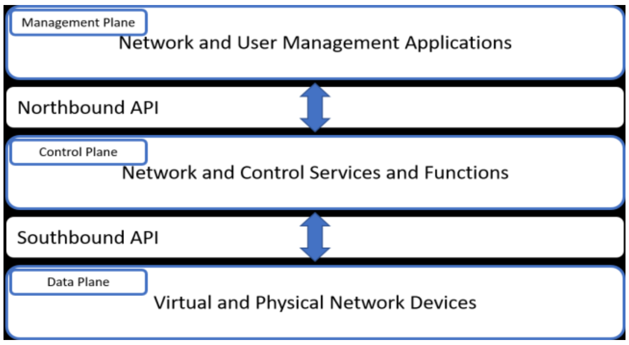
Management Plane
Network and User Management Applications
Northbound API
Control Plane
Network and Control Services and Functions
Southbound API
Data Plane
Virtual and Physical Network Devices

**Figure 1.** SDN paradigm

The importance of this research lies in the following aspects:

1) There are many NOSs, and their development frequency is rapid. Therefore, selection of the appropriate NOS for CDC requires an analytically comparative study.

2) The selection of the appropriate NOS for the aimed use case profoundly impacts the offered services as it is the control plane in SDN paradigm.

3) This study is expected to help researchers and operators in this field to transform from traditional network to SDN paradigm, identifying the most common open-source NOSs and determine the most appropriate NOS for CDC.

This paper has the following contributions:

a) Framework to determine the best suited NOS to be used in CDC.

b) Provide an analytical study of the functional and non-functional specifications of six open-source NOSs.

The paper consists of the following sections. In Related works section, we present the literature review. In Research Methodology section, we overview the used research methodology. We introduce the problem and explain the employed decision-making method in the problem statement and AHP analysis section. We discuss the results in Results and discussion section. Finally, we present the conclusion in Conclusion section.

## 2. Related Works

Several research works compared NOSs. Khondoker *et al.* [5] conducted a comparison of specifications for each of the following NOSs, (ODL, Ryu, POX, Ryu, Floodlight, Trema). Ryu was found to be the best NOS according to the adopted criteria for the evaluation. Khondoker *et al.* did not consider so common NOSs such as ONOS and this study considered no specific use case. The study in [6] compared southbound API, Openflow version, programming language and round-trip time (RTT) delay of POX, ODL, ONOS, Ryu. They found that Ryu has the best score based on TOPSIS method. The work in [6] considered a small set of criteria which are OpenFlow version, NOS programming language, RTT, interfaces and documentation. However, we consider more criteria in this study to evaluate different NOS specifications. The work in [7] presented a feature AHP-based comparison of ODL, NOX, Beacon, Trema, POX, OpenMUL, Ryu, Floodlight, OpenContrail and ONOS in terms of traffic classification as the targeted use case. They found that ODL provides the best specifications for the use case. However, the study did not consider criteria related to cloud based data center similar to what we consider in Table 3. The work in [8] presented an ANP-based comparative study of features and performance of Floodlight, ONOS, ODL, POX, Ryu and Trema. The authors found ODL has the best score. However, this study considered many quantitative measurements such delay, throughput and CPU utilization to find the optimal NOS whereas we consider only the qualitative criteria in this study. The comparison conducted in [9] presented the optimum NOS for deploying SDN-WAN. The authors compared Ryu, ODL, POX, Trema, Floodlight.

Using AHP, the authors tackled the problem of finding the best NOS might be used for real use case which is connecting university campuses using SDN instead of Multiprotocol Label Switching (MPLS) Border Gateway Protocol (BGP) technology. Amiri *et al.* [10] compared features and performance of many open-source NOSs by using Best-Worst Method (BWM) decision-making method and Cbench to assess the qualitative and quantitative criteria of NOSs. The work in [10] employs Cbench to measure throughput and latency as quantitative criteria and ODL has the highest rank in both. However, the authors did not aim a specific use case. Ali at el. [11] proposed a prioritized features method used for selecting the best NOS. The authors employed ANP to implement the proposal. The investigated NOSs are ODL, ONOS, POX, RYU and Trema. The aim of this study was reducing the computational complexity of the selection process by selecting high weight features and ignoring low weight features. However, the study did not consider a wide spectrum of criteria and it did not aim CDC as a use case. The work in [12] investigated many features of ODL, ONOS, Floodlight, POX, RYU and Trema to find the NOS which has the best feature set. In particular, the authors' final target was creating a hierarchical cluster of the best NOS to improve performance such as delay, throughput, fault tolerance and scalability. This work is not tailored for specific use case but it aimed to improve the cluster performance, and it did not consider CDC related features. On the other hand, many studies reviewed SDN state of the art to investigate different aspects of SDN solutions and NOSs. In this context, our study introduces different aspects of NOSs as well but for identifying the best NOS for CDC. In this context, Abuarqoub [13] reviewed SDN NOSs in terms of scalability. On the other hand, Sarmiento *et al.* [14] surveyed SDN NOSs in terms of the challenges yielded from inter-site networking services. Studies in [15-16] observed SDN based load balancing techniques and [18] reviewed SDN NOSs according to NOS placement problem.

However, to the best of our knowledge, we consider a use case that has not been considered in the literature. In this study, we aimed to find the best suited NOS for CDC. In addition, our study covers the support of different features such as fault verification and troubleshooting, traffic security and packet forwarding techniques. We conducted a qualitative comparison only since there are many previously published papers on performance comparisons, such as in [6-8, 18].

## 3. Research Methodology

We experimentally verified the NOS specifications to decide on criteria and alternatives weights. We installed NOSs, and many of their features and services were investigated. Furthermore, we investigated information provided on their official websites. Several criteria were imposed according to the aimed use case. Initially, the specifications required to be provided by the suitable NOS were identified, as shown in Table 1. Then AHP [19] was applied by which the studied specifications of each NOS were evaluated.

**Table 1.** Version of the considered NOS

| NOS | Version |
|---|---|
| **POX** | dart[1] |
| **Ryu** | 4.23[2] |
| **Floodlight** | v1.2[3] |
| **ODL** | Sodium[4] |
| **ONOS** | Sparrow[5] |
| **Tungsten** | Tungsten 5.1[6] |

### 3.1. The Studied NOS

The most common open-source NOSs presented at the time of this study were studied, which are ONOS, POX, RYU, ODL,Floodlight and Tungsten. The reason for choosing these NOSs is that they are considered the most widespread open-source NOSs. Most of the other open-source NOSs are either

[1] "Pox-doc", McCauley, last modified 2019, https://noxrepo.github.io/pox-doc/html/
[2] "ryu the network operating system(nos)", RYU Community, last modified 2019, https://ryu.readthedocs.io/en/latest/index.html
[3] "Floodlight controller", Floodlight Community, last modified 2019, https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/
[4] "Start guide", Opendaylight, last modified 2019, https://docs.opendaylight.org/en/stable-sodium/getting-started-guide/index.html
[5] "Onos", ONOSProject, last modified 2019, https://wiki.onosproject.org/display/ONOS/ONOS
[6] "Tungsten fabric", Linux Foundation, last modified 2019, https://tungstenfabric.github.io/website/

designed for a specific purpose or no longer being developed. However, all unconsidered NOSs in this study are used less according to the frequency of their appearance in research works [20] [21]. ONOS is the most recent NOS produced. It is developed by ON-Lab, which is a non-profit organization. One of its main aspects is that it supports legacy networks. POX was developed to be the successor to NOX. an open research community developed POX. It is suitable for conducting SDN-based researches . According to the component-based model, Ryu has been developed by NTT company to facilitate the process of adding and modifying its components in any programming language. Big Switch developed Floodlight, and its main goal was to provide a NOS capable of dealing with open network hardware (i.e., hardware that does not contain software and is called "white box"). As a result of the effort of a consortium of several international companies, under Linux Foundation, ODL was introduced to obtain the acceptance of companies and users and to receive improvements continuously. Finally, some NOSs are considered as cloud oriented SDN NOSs where they aim at managing the network infrastructure of CDC. In this context, we consider Tungsten which is the open source version of Juniper Contrail and now it is a project of Linux Foundation. Table 1 lists versions of NOSs involved in this study.

### 3.2. Characteristics of Cloud Data Center

Nowadays, cloud computing drives most businesses and shapes a new era of technology delivery. Cloud computing provides different services such as software, storage, and virtual resources, so it abstracts the technical complexities, and it eliminates the cost and risk associated with hardware maintaining and acquisition [4]. In recent years, SDN has been employed in CDCs since SDN provides central control of the network. SDN-based Data Center Network (DCN) is preferred to the traditional DCN since it can improve DCN efficiency [22]. Furthermore, hypervisors are used in CDC to enable the deployment of virtual resources, so functions such as traffic control and isolation are crucial for providing efficient services. In this context, SDN and Network Function Virtualization (NFV) are seen as enablers of network slicing to create multiple virtual networks over a shared infrastructure. Each virtual network can be logically isolated from other virtual networks and assigned to serve specific requirements. Since SDN NOS has a global view of the network, traffic routing decisions can satisfy service demands, and each network slice can be provisioned with network and cloud resources based on QoS and Service Level Agreement (SLA). Due to the programmable model of SDN and the separation of the control plane and data plane, SDN facilitates the deployment of network functions that are essential for managing virtual resources where Virtualized Network Function (VNF) can be deployed in networks dynamically by SDN's configuration and automation functionalities.

In this context, many protocols are used as a southbound interface so that NOS can programmatically contact the switches to exchange information and send instructions. Besides, many NOSs expose northbound interfaces that can be used by the management plane and cloud orchestrator to implement different solutions (e.g., Quality of Service (QoS) management, traffic engineering, fault recovery, network statistics, topology discovery, etc.). Furthermore, the dynamically changing status of the network should be considered by the cloud systems in terms of the management of computing and storage resources, Virtual Machine (VM) provisioning, and addressing virtual resources requirements [23, 24]. The massive scale of a CDC, which consists of thousands of compute nodes and network devices, imposes the necessity for network management, performance improvement and dynamic provisioning of computing and network resources [25]. Therefore, the architecture of the control plane should be taken into account whether it should be centralized or distributed. The centralized control plane represents a single point of failure. Although centralized architecture might be an efficient solution for small scale CDCs, it may not be efficient for large scale CDCs. Therefore, a distributed control plane consisted of many NOSs should be deployed. However, such architecture yields other challenges such as synchronization and heterogeneity of the underlying data plane [26]. On the other hand, SDN paradigm triggers additional motivations for the attackers. SDN NOSs and OpenFlow are considered potential security vulnerabilities which require efficient security solutions to improve the safety [27]. In particular, cloud orchestrator and SDN NOS integrate each other. Cloud orchestrator manages virtual resources like VMs provisioning, while SDN NOS manages physical and virtual network resources by southbound API (e.g. Openflow) and VM traffic. Besides, they communicate by north-bound API (e.g., Rest API) [28]. Hence, since SDN NOS and CDC orchestrator are integrated, maintaining the security is vital. Therefore, security solutions should utilize SDN capabilities and protect SDN components as well.

**Table 2.** Non-functional features of studied NOS

| Feature | | | Description |
|---|---|---|---|
| Easy to use | Usability | GUI | Display information about network components and metrics |
| | | Documentation | Information required for utilizing and developing NOS |
| | Development | North API | The standard used for communications between the control and management planes |
| | | Modularity | NOS architecture based which new modifications may be introduced |
| | | Community | The contributions by developers and operators to improve open source NOSs |
| Maturity | Update Frequency | | The rate of developments the NOS receives |
| | Application Availability | | Applications provided to be used based on NOS |
| Interoperability | Control Protocols | | Communication protocols used to control network hardware by NOS |
| | Management Protocols | | The protocols used to manage the settings of network hardware by NOS |
| Security | | | The security features provided by NOS to perform its functions safely |
| Scalability | | | The ability of running multiple instances of NOS within a distributed and consistent cluster |
| Availability | | | The ability of providing services by NOS in case of failures |

**Table 3.** Functional features of studied NOS

| Feature | | Description |
|---|---|---|
| Virtualization | Overlay Networks | Creation of virtual networks based on physical network hardware |
| | Isolation | Maintaining the isolation of the shared virtual resources |
| Packet Forwarding Techniques | Load Balancing | Improve network utilization according to the network situation |
| | Quality of Service | Mechanisms used to ensure that the required level of service is achieved |
| Traffic Protection Solutions | | Detect security threats by monitoring network situation and applying pre-defined security policies |
| Fault Verification and Troubleshooting | | Mechanisms used for faults recovery and avoidance |

In this context, SDN can be employed to prevent a distributed denial-of-service (DDoS) attack by utilizing SDN functions such as traffic analysis. On the other hand, the SDN NOS must be secured itself as it represents the central control layer in SDN paradigm [29]. As Big Data and Internet of Things (IoT) applications exploit CDC, utilizing SDN-based CDC can enhance the performance of these applications by applying load balancing and QoS solutions [30]. For this sake, SDN can provide different QoS mechanisms for different applications, such as bandwidth slicing. In addition, scalability is an essential concern [31]. Since SDN NOS gathers information from data plane devices, it is crucial to avoid deploying a single SDN NOS in large scale CDC. Single SDN NOS represents a single point of failure, which could be problematic because there is a big number of devices in the data plane of CDC. Therefore, the distributed architecture of many SDN NOSs, to improve the availability, can be employed using westbound and eastbound APIs to maintain the synchronization [32][33]. Furthermore, the faults represent a critical challenge to meet SLAs. In CDC, fault tolerance techniques provide the ability to maintain QoS. In this context, improving the reliability in CDC can be proactively achieved to avoid fault occurrence. For example, system load can be identified as an indication of potential future failures so that new resources might be added to avoid failures proactively [34] [35]. Although there are many open-source NOSs, and they are differently matured in terms of CDC requirements. It is challenging to find NOS that can provide CDC requirements, and it can integrate with the CDC orchestrator. As a result, based on the former characteristics of CDC, we summarize in Tables 1 and 2 the functional and non-functional features, respectively, which are required to be provided by the best suited NOS.

## 3.3. Decision support system

In this study, AHP [19] is used to make a decision according to multiple criteria. It is the most appropriate method according to the nature of the problem since it is a mathematical method for decision-making, which generates criteria weights through pair-wise comparisons. This comparison is also used to evaluate alternatives against each criterion. According to AHP, criteria and alternatives are treated separately. In addition, AHP presents problems in a hierarchical structure, which helps classify the criteria into levels according to the nature of the studied problem. Therefore, AHP reduces the size of the comparison matrices, thus maintaining the accuracy and consistency of results. Furthermore, AHP checks the consistency of the evaluation as long as the dimensions of the comparison matrices are under ten. On the other hand, other decision support systems like Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [36] is suitable for problems with negative criteria, while all criteria in this study have positive values only. Analytical Network Process (ANP) [37] is appropriate in case of correlated criteria and alternatives, while all the criteria and alternatives in this study are independent. On the other hand, Best-Worst Method (BWM) [38] identifies the worst and the best criteria in advanced so that the rest criteria are compared with them. However, in our case, we have many criteria, in different levels of the hierarchy, with similar importance what makes AHP the prefer method for this problem as well [39][40].

## 4. The Problem Statement and AHP Analysis

This section presents the problem as a decision making problem applying AHP. The studied alternatives are $A_n$where $n = 6$, and the adopted criteria are $C_m$ where $m \in Z^+$. We aim to find the optimum NOS among the investigated NOSs using AHP according to the features listed in Table 2 and Table 3. Therefore, the hierarchical presentation of the studied criteria is depicted in Figure 2-4. AHP requires to assign weights for the evaluation criteria. For this sake, we created $m \times m$ matrices for the criteria on the same levels of the hierarchy, and their elements are the priorities for each pair of the criteria to signify the importance of a single criterion to another, as shown in Eq. 1 where $C_{ij}$ represents the importance of criterion $C_i$ in comparison to $C_j$. In order to assign priority, AHP defines a scale between 1 and 9 to present the prioritization, as shown in Table 4 [19].

**Table 4.** Priority scale

| Priority Value | Indication |
|---|---|
| 1 | equally important |
| 3 | moderately more important |
| 5 | strongly more important |
| 7 | very strongly more important |
| 9 | extremely more important |

$$\begin{bmatrix} \mathbf{1} & \cdots & \boldsymbol{C_{1m}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{C_{m1}} & \cdots & \mathbf{1} \end{bmatrix}, \boldsymbol{C_{ij}} > \mathbf{0}, \boldsymbol{C_{ij}} = \frac{\mathbf{1}}{\boldsymbol{C_{ij}}} \qquad (1)$$

Then, the columns are normalized to find the relative weights by applying Eq. 2

$$\overline{\boldsymbol{C}}_{\boldsymbol{ij}} = \frac{\boldsymbol{C_{ij}}}{\sum_{i=1}^{m} \boldsymbol{C_{ij}}} \ \boldsymbol{where} \ \sum_{i=1}^{m} \overline{\boldsymbol{C}}_{\boldsymbol{ij}} = \mathbf{1} \qquad (2)$$

Next, vectors of adding the elements of each row are created as shown in Eq. 3

$$\boldsymbol{v_i} = \sum_{i=1}^{m} \overline{\boldsymbol{C}}_{\boldsymbol{ij}} \qquad (3)$$

By calculating the average of the previous values, we obtain a vector of weights Wm×i, which shows the weights of the criteria according to their pair-wise priorities, as shown in Eq. 4-5:

$$\boldsymbol{W_i} = \frac{\boldsymbol{v_i}}{\boldsymbol{m}} \ \boldsymbol{where} \ \sum_{i=1}^{m} \boldsymbol{w_i} = \mathbf{1} \qquad (4)$$

$$\boldsymbol{W_{i\times j}} = \begin{bmatrix} \boldsymbol{W_{1\times j}} \\ \vdots \\ \boldsymbol{W_{m\times j}} \end{bmatrix} \qquad (5)$$

The final weights of the leaf criteria are calculated by finding the resulted values in Eq. 5 multiplied with their parent criteria. Figure 2-4 present the final global weights, which indicate the significance of each criterion to achieve the goal. Then, the priority vector consistency is verified by employing the notion of Eigen-value to compute Consistency Index (CI) as in Eq. 6:

$$CI = \frac{\lambda_{max} - m}{m-1} \tag{6}$$

$\lambda_{max}$ is the summation of multiply weight vectors with the summation vectors of columns of the pair wise comparison matrix. Consistency Ratio (CR) can be computed as in Eq. (7) using Random Index (RI).

$$CR = \frac{CI}{RI} \tag{7}$$

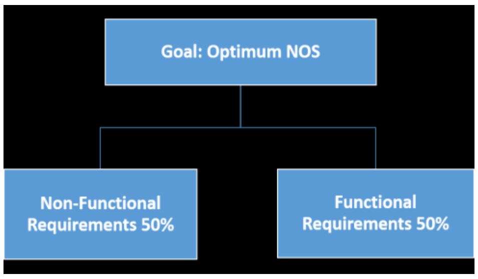

**Figure 2.** Criteria hierarchy

The values of RI are presented in Table 5. According to [19], *RI* is the pre-calculated average consistency index computed by [19] of randomly generated comparison matrices with different scales, denoted as scale in Table 5. *RI* can be used as a reference value to check how *CI*, which is computed in Eq. 6, of our comparison matrices is far or close to the matched-scale RI. In particular, if *CR*, in Eq. 7, is 1, this means that we have completely random priorities. Therefore, we have no meaningful comparisons, and we have to revise our comparisons. The opposite is true in case Eq. 7 (i.e., *CR*) equals zero. In case *CR* is below 10%, the priority values are supposed to be consistent; otherwise, pair-wise priorities should be modified, and the pairwise comparisons should be repeated. Similarly, the alternatives are pair-wise compared against each criterion as in Eq. 8 based on the convention in Table 4, and weight vectors of the alternatives are computed, and their consistency is inspected as in Eq. 7, as well. Finally, the final value of each alternative is computed as in Eq. 9 where $W_{m\times1}$is the weight vector of child criteria, $\acute{W}_{m\times1}$ is the alternative's weights against all criteria, and $X_{n\times1}$ is a vector contains the final result of each alternative.

$$\begin{bmatrix} 1 & \cdots & A_{1n}^{m} \\ \vdots & \ddots & \vdots \\ A_{n1}^{m} & \cdots & 1 \end{bmatrix}, A_{ij} > 0, A_{ij} = \frac{1}{A_{ji}} \tag{8}$$

$$X_{n\times1} = \acute{W}_{n\times m} \times W_{m\times1} \tag{9}$$

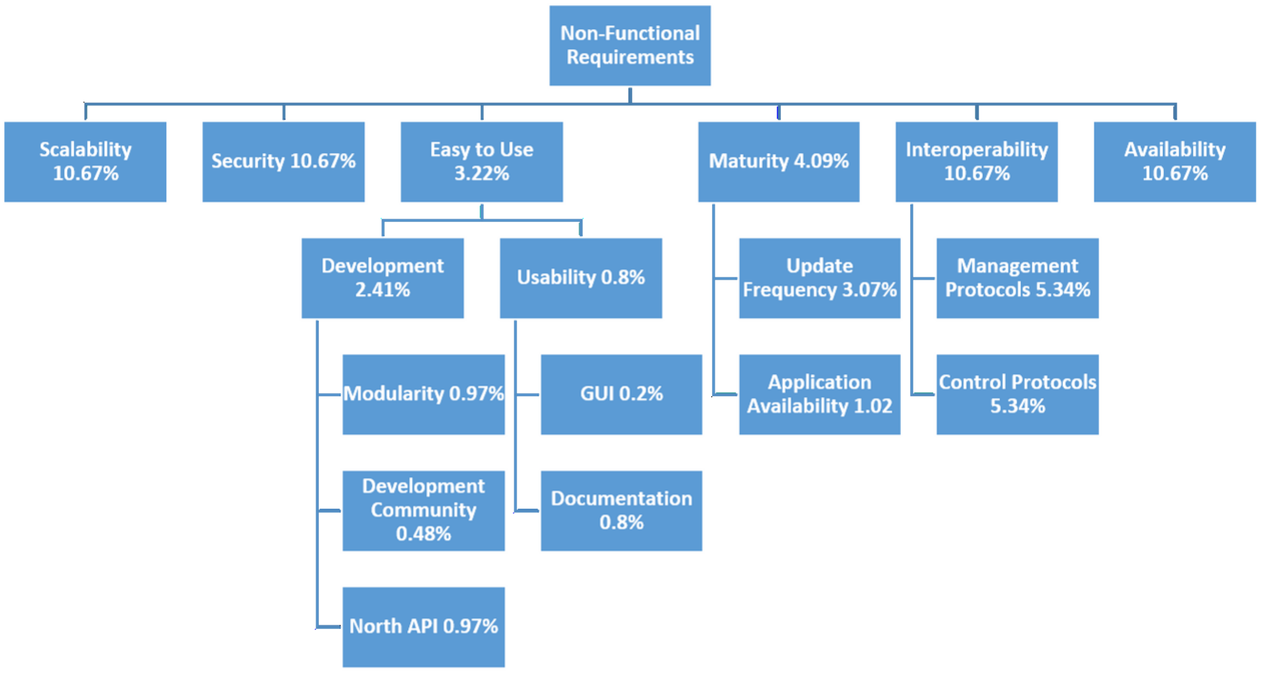

**Figure 3.** Non-functional Requirements

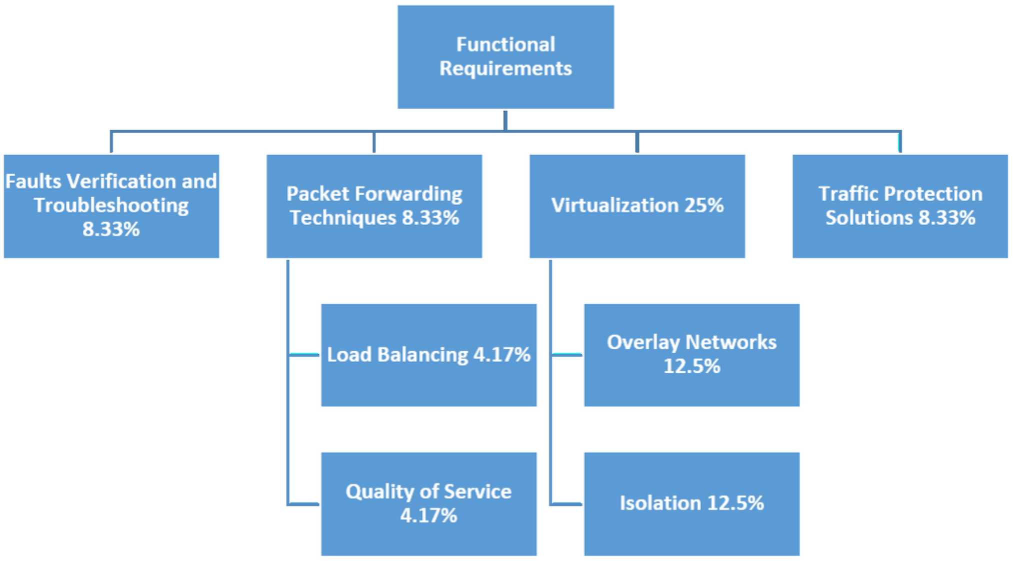

**Figure 4.** Functional Requirements

**Table 5.** Values of Random Index

| **Scale** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| RI | 0 | 0 | 0.58 | 0.9 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 |

Tables 6-15 present the matrices resulted from Eq. 1. The pair-wise comparisons of criteria show that functional and non-functional criteria have equal importance. In relation to the non-functional requirements, we set weights of Ease of Use and Maturity less than that of the remaining criteria since all considered NOSs are open-source, and they can be developed according to the requirements. On the other hand, Development has a higher weight than Usability as NOSs might be modified to propose new functions and services. As a result, Modularity has a higher weight in comparison to Development Community and North API as it simplifies NOSs modification. As shown in Table 10, Documentation has a higher weight than GUI, where it is essential to adopt, use and develop NOSs. Finally, all remaining criteria have similar weights except Virtualization since we aim to find the optimal NOS for CDC. We present pair-wise comparison of the alternatives resulted from Eq. 8 in Table 16-33. Table 35 presents $\acute{W}$ $\boldsymbol{n} \times \boldsymbol{m}$ of all alternatives according to all leaf criteria. Final values of all alternatives, $\boldsymbol{X}_{\boldsymbol{n}\times\mathbf{1}}$, are presented in Figure 5.

## 5. Results and Discussion

### 5.1. Non-functional features

As shown in Figure 5, ODL is the best suited NOS for CDC according to the specified criteria. The final values of ONOS and ODL are close since they are being developed continuously by a large and active community, and they are supported by many industrial and research groups in comparison to the other NOSs. This presents the importance of adopting open-source software projects by relevant companies and institutions. Table 34 summarizes the detailed features of NOSs.

#### 5.1.1. Scalability

ONOS, ODL and Tungsten are more scalable than the other NOSs, since they support distributed model in which multiple NOS instances can inter-operate simultaneously. However, ONOS can assign segments of Wide Area Network (WAN) or DCN to a specific instance to mitigate the instance load. ONOS employs Atomix database to store network information, which is also used for coordinating between the instances. In particular, ONOS represents a logically centralized control plane since the infrastructure might consist of multiple domains with one ONOS instance for each domain. On the other hand, ODL employs Advanced Message Queuing Protocol (AMQP)[7] to exchange information between ODL instances where each instance maintains information of its own domain. ODL leverages Federation and NetVirt projects to create virtual networks across many OpenStack instances, but these inter-site

[7] "Amqp is the internet protocol for business messaging", OASIS, last modified 2021, https://www.amqp.org/about/what

communications are managed at ODL levels, so overlapping is likely at Neutron level, e.g., duplicated IPs in different OpenStack instances is probable. Tungsten has control nodes run in a cluster to maintain scalability.

### 5.1.2. Security

All NOSs provide secure connections with the devices in the data plane by applying Transport Layer Security/Secure Sockets Layer (TLS)/(SSL). Moreover, ODL has Unified Secure Channel (USC), which consists of an agent, plugin and manager to initiate and maintain the connection with ODL, authenticates ODL and presents the state of USC in ODL DLUX user interface. Besides, ODL provides authentication, authorization, and accounting (AAA) framework controlling access to ODL features, applying policies to use those features and auditing the usage.

**Table 6**. Top level criteria

| Criteria | non-Functional Requirements | Functional Requirements |
|---|---|---|
| non-Functional Requirements | 1 | 1 |
| Functional Requirements | | 1 |

inconsistency ratio = 0.00%

**Table 7.** non-Functional Requirements

| Criteria | Scalability | Security | Easy to Use | Maturity | Interoperability | Availability |
|---|---|---|---|---|---|---|
| Scalability | 1 | 1 | 3 | 3 | 1 | 1 |
| Security | | 1 | 3 | 3 | 1 | 1 |
| Easy to Use | | | 1 | 1/2 | 1/3 | 1/3 |
| Maturity | | | | 1 | 1/3 | 1/3 |
| South API | | | | | 1 | 1 |
| Availability | | | | | | 1 |

inconsistency ratio = 0.22%

**Table 8.** Easy to Use

| Criteria | Development | Usability |
|---|---|---|
| Development | 1 | 1 |
| Usability | | 1 |

inconsistency ratio = 0.00%

**Table 9.** Development

| Criteria | Modularity | Development Community | North API |
|---|---|---|---|
| Modularity | 1 | 2 | 1 |
| Development Community | | 1 | 1/2 |
| North API | | | 1 |

inconsistency ratio = 0.00%

**Table 10.** Usability

| Criteria | GUI | Documentation |
|---|---|---|
| GUI | 1 | 1/3 |
| Documentation | | 1 |

inconsistency ratio = 0.00%

**Table 11.** Maturity

| Criteria | Update Frequency | Application Availability |
|---|---|---|
| Update Frequency | 1 | 3 |
| Application Availability | | 1 |

inconsistency ratio = 0.00%

**Table 12.** Interoperability

| Criteria | Management Protocols | Control Protocols |
|---|---|---|
| Management Protocols | 1 | 1/3 |
| Control Protocols | | 1 |

inconsistency ratio = 0.00%

**Table 13.** Functional Requirements

| Criteria | Fault Verification & Troubleshooting | Packet Forwarding Techniques | Virtualization | Traffic Protection Solutions |
|---|---|---|---|---|
| Fault Verification & Troubleshooting | 1 | 1 | 1/3 | 1 |
| Packet Forwarding Techniques | | 1 | 1/3 | 1 |
| Virtualization | | | 1 | 3 |
| Traffic Protection Solutions | | | | 1 |

inconsistency ratio = 0.00%

**Table 14.** Packet Forwarding Techniques

| Criteria | Load Balancing | Quality of Service |
|---|---|---|
| **Load Balancing** | 1 | 1 |
| **Quality of Service** | | 1 |

inconsistency ratio = 0.00%

**Table 15.** Virtualization

| Criteria | Overlay Networks | Isolation |
|---|---|---|
| **Overlay Networks** | 1 | 1 |
| **Isolation** | | 1 |

inconsistency ratio = 0.00%

### 5.1.3. Easy to use & Maturity

In terms of Modularity, new modules can be added to all NOSs feasibly either as Python component or Apache Karaf. However, ONOS provides Yang management system that abstracts data modelling details of applications and device drivers by generating YANG modelled JAVA objects corresponding to device and application schema so that the business logic will be the sole concern. Hence, it can seamlessly support any interface like Representational state transfer (REST), Network Configuration Protocol (NETCONF), Extensible Markup Language (XML), etc. ONOS and ODL have the richest documentations. In addition, ODL and ONOS have the fastest update frequency and the most varied and numerous applications, but Tungsten provides richer GUI than other NOSs as configurations in terms of QoS, IP addressing, Ports, policies, etc can be depicted and defined using the web-based GUI. Finally, ONOS and ODL have the largest development communities as they are supported by many technological companies researching centers, and organizations.

### 5.1.4. Interoperability

Tungsten, ONOS and ODL provide Simple Mail Transfer Protocol (SNMP) as a management protocol which provides so many facilities in an environment such as CDC. In relation to south API control protocols, ODL provides Model-Driven Service Adaptation Layer (MD-SAL), which simplifies the communications between ODL modules and data plane through the available southbound APIs by supporting user-defined payload formats, including payload serialization and adaptation. ODL provides OpFlex[8], which is an extensible policy protocol designed to exchange abstract policy between ODL and the devices support policy rendering. On the other hand, Tungsten provides Extensible Messaging and Presence Protocol (XMPP)[9] as a southbound API to communicate with vRouter[10], which can be deployed either as a kernel module, Data Plane Development Kit (DPDK)[11] in user space, SmartNIC programmable network interface[12], or by using Single Root I/O Virtualization (SR-IOV) for accessing Network Interface Card (NIC) from VMs directly.

### 5.1.5. Availability

Availability has been boosted in all NOSs except POX by applying Master/Slave model. The continuous mirroring operates between master and slave instances so that the slave takes responsibility upon master failure. However, since ODL maintains the virtual network information on ODL level in case of multiple CDC sites, non-disconnected ODL instances can provide inter-site services during network disconnections where ODL instances employ AMQP to communicate with each other so that no need to share information between instances. On the contrary, ONOS instances present a logically distributed control plane by sharing information using Atomix database[13].

[8] "Opflex control protocol", IETF, last modified 2016, https://datatracker.ietf.org/doc/html/draft-smith-opflex-03

[9] "An overview of xmpp", last modified 2021, https://xmpp.org/about/technology-overview/

[10] "vrouter", ONOS project, last modified 2016, https://wiki.onosproject.org/display/ONOS15/vRouter

[11] "About dpdk", Linux Foundation, last modified 2021, https://www.dpdk.org/about/

[12] "What is a smartnic?", Nvidia, last modified 2021, https://blogs.nvidia.com/blog/2021/10/29/what-is-a-smartnic/

[13] "Atomix ", Open Network Foundation, last modified 2021, https://atomix.io/

## 5.2. Functional features

### 5.2.1. Fault verification and troubleshooting

Most outstanding troubleshooting capabilities have been found in ODL and ONOS as well. ODL provides a diagnostics framework to report the status of ODL itself by performing continuous monitoring of registered modules and built-in services to maintain the overall health of the system, and it additionally triggers alarms. On the other hand, ONOS provides a framework to break the routing loop, routing black holes and applications conflicts. As well, ONOS integrates with services like CPMan[14] and Ganglia[15] to maintain the information of the ONOS state. In addition, ONOS provides fault management by polling SNMP capable network devices to track events like device adding, updating, link down, etc. Users can access and manipulate alarms by CLI, Rest API or GUI. FlowScale[16] provided by Floodlight diverts all traffic directed to a down port to other up ports. Tungsten's analytic nodes collect metrics from compute, storage and network nodes as well as their workloads to facilitate troubleshooting and monitoring. In particular, Zookeeper[17] is used to distribute the responsibility of collecting data among analytic nodes to avoid nodes overwhelming.

### 5.2.2. Packet forwarding technique

For the sake of improving the availability and the performance, ONOS balances the load between the instances in a cluster. Tungsten balances the load among the virtual resources by providing load balancing as a service employing different drivers such as HAproxy[18]. On the other hand, ONOS improves QoS by providing SDNi, which manages flow setup by considering information such as path requirement, QoS and Service Level Agreement (SLA). Besides, ONOS provides intent framework by which applications can specify their network control desires as a policy (e.g., tunnel provisioning, flow rule installation). Hence, different packet forwarding techniques can be applied at run-time. However, the other NOSs support QoS by creating queues, Openflow meters, Differentiated Services (DiffServ) or creating Label-Switched Path (LSP) by Path Computation Element Configuration Protocol (PCEP).

### 5.2.3. Virtualization

In relation to virtualization, all NOSs except POX integrate with Openstack orchestrator, support Virtual Local Area Network (VLAN), moreover, Tungsten, ODL and ONOS support Virtual Extensible LAN (VxLAN) and L3 tunnelling. In this context, ODL manages overlay tunnels established within a transport zone using VxLAN, Generic Routing Encapsulation (GRE), Generic Protocol Extension for VxLAN (VxLAN-GPE) and MPLS over GRE (MPLSoGRE). Besides, ODL provides L2, L3 tunneling, network address translation (NAT) and access control list (ACL) services for the virtualized resources using Neutron northbound API, which communicates with ODL driver in Openstack. On the other hand, ONOS provides a tenant network virtualization service for CDC provisioning using L2 VxLAN or L3 GRE. Furthermore, ONOS provides L2, L3 tunneling and NAT services using Open Virtual Switch (OVS)[19] in the compute nodes and provides horizontal scalability of the gateway node, which connects the virtual networks with the outside by BGP. ONOS isolates the virtual network traffic by Ethernet Virtual Connections (EVC), which uses VLAN ID to tag traffic of different EVCs uniquely and define the associated components like User Network Interface (UNI) and bandwidth profile. Similarly, ODL provides UNI manager, which provisions EVC in physical and virtual network elements of multi-vendor using YANG based APIs and drivers. In addition, ONOS Simplified Overlay Network Architecture (SONA) and ODL NetVirt integrate with Kubernetes as well. On the other hand, Tungsten employs BGP Ethernet Virtual Private Network (EVPN) and VxLAN to connect VMs in different orchestrator domains like Kubernetes, OpenStack and VMWare vCenter. Furthermore, Tungsten enables virtual networks to connect with external network by BGP peering with gateway routers. Tungsten employs vRouter instead

[14] "Control plane management application", ONOS Project, last modified 2016, https://wiki.onosproject.org/display/ONOS/Control+Plane+Management+Application

[15] Ganglia monitoring system, http://ganglia.sourceforge.net/

[16] Flowscale, https://floodlight.atlassian.net/wiki/spaces/FlowScale/overview

[17] "What is zookeeper?", Apache, last modified 2021, https://zookeeper.apache.org/

[18] "the reliable high performance tcp/http load balancer", Haproxy, http://www.haproxy.org/

[19] "Production quality, multilayer open virtual switch", Linux Foundation, last modified 2021, https://www.openvswitch.org/

of Linux bridge or OVS in hosts where Tungsten configures vRouter to implement the network and security policies. Tungsten maintains the isolation using MPLS over User Datagram Protocol MPLSoUDP or VxLAN encapsulation tunneling between fabric Virtual Routing and Forwarding (VRF) and VM VRFs.

**Table 16.** Scalability

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/3 | 1/4 | 1 | 1 |
| RYU | | 1 | 1/3 | 1/4 | 1 | 1 |
| ODL | | | 1 | 1/3 | 3 | 3 |
| ONOS | | | | 1 | 5 | 3 |
| POX | | | | | 1 | 1 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 1.55%

**Table 17.** Security

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 2 | 1/3 | 1/2 | 2 | 1 |
| RYU | | 1 | 1/4 | 1/3 | 1 | 1/2 |
| ODL | | | 1 | 2 | 4 | 3 |
| ONOS | | | | 1 | 2 | 3 |
| POX | | | | | 1 | 1/2 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.66%

**Table 18.** Modularity

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/3 | 1/3 | 1 | 3 |
| RYU | | 1 | 1/3 | 1/3 | 1 | 3 |
| ODL | | | 1 | 1 | 3 | 6 |
| ONOS | | | | 1 | 3 | 6 |
| POX | | | | | 1 | 3 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.44%

**Table 19.** Development community

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/6 | 1/6 | 3 | 1/4 |
| RYU | | 1 | 1/6 | 1/6 | 3 | 1/4 |
| ODL | | | 1 | 1 | 6 | 2 |
| ONOS | | | | 1 | 6 | 2 |
| POX | | | | | 1 | 1/5 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 3.29%

**Table 20.** North API

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 3 | 1/3 | 2 | 5 | 2 |
| RYU | | 1 | 1/5 | 1 | 3 | 1 |
| ODL | | | 1 | 3 | 7 | 3 |
| ONOS | | | | 1 | 3 | 1 |
| POX | | | | | 1 | 1/2 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 1.87%

**Table 21.** GUI

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 2 | 1/3 | 1/3 | 3 | 1/6 |
| RYU | | 1 | 1/3 | 1/3 | 1 | 1/6 |
| ODL | | | 1 | 1 | 5 | 1/3 |
| ONOS | | | | 1 | 5 | 1/3 |
| POX | | | | | 1 | 1/9 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 2.08%

**Table 22:** Documentation

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/3 | 1/3 | 3 | 1 |
| RYU | | 1 | 1/3 | 1/3 | 3 | 1 |
| ODL | | | 1 | 1 | 5 | 3 |
| ONOS | | | | 1 | 5 | 3 |
| POX | | | | | 1 | 1/3 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.93%

**Table 23.** Update frequency

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1/3 | 1/6 | 1/6 | 3 | 1/5 |
| RYU | | 1 | 1/3 | 1/3 | 7 | 1/2 |
| ODL | | | 1 | 1 | 9 | 2 |
| ONOS | | | | 1 | 9 | 2 |
| POX | | | | | 1 | 1/7 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 2.22%

**Table 24.** Application availability

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 2 | 1/3 | 1/3 | 4 | 1 |
| RYU | | 1 | 1/6 | 1/6 | 3 | 1/2 |
| ODL | | | 1 | 1 | 9 | 3 |
| ONOS | | | | 1 | 9 | 3 |
| POX | | | | | 1 | 1/4 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.73%

**Table 25.** Management protocols

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1/2 | 1/6 | 1/5 | 3 | 1 |
| RYU | | 1 | 1/6 | 1/3 | 4 | 2 |
| ODL | | | 1 | 2 | 6 | 6 |
| ONOS | | | | 1 | 6 | 5 |
| POX | | | | | 1 | 1/3 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 4.06%

**Table 26.** Control protocols

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/2 | 1 | 4 | 2 |
| RYU | | 1 | 1/2 | 1 | 4 | 2 |
| ODL | | | 1 | 2 | 6 | 3 |
| ONOS | | | | 1 | 4 | 2 |
| POX | | | | | 1 | 1/2 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.22%

**Table 27.** Availability

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 2 | 1/5 | 1/5 | 5 | 1 |
| RYU | | 1 | 1/7 | 1/7 | 4 | 1/2 |
| ODL | | | 1 | 1 | 7 | 5 |
| ONOS | | | | 1 | 7 | 5 |
| POX | | | | | 1 | 1/4 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 4.87%

**Table 28.** Faults verification and troubleshooting

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/3 | 1/3 | 3 | 1/2 |
| RYU | | 1 | 1/3 | 1/3 | 3 | 1/2 |
| ODL | | | 1 | 1 | 5 | 2 |
| ONOS | | | | 1 | 5 | 2 |
| POX | | | | | 1 | 1/4 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 1.11%

**Table 29.** Load balancing

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 5 | 1 | 1/7 | 1 | 1 |
| RYU | | 1 | 1/5 | 1/9 | 1/5 | 1/5 |
| ODL | | | 1 | 1/7 | 1 | 1 |
| ONOS | | | | 1 | 7 | 7 |
| POX | | | | | 1 | 1 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 3.54%

**Table 30.** Quality of service

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/3 | 1/5 | 3 | 1 |
| RYU | | 1 | 1/3 | 1/5 | 3 | 1 |
| ODL | | | 1 | 1/3 | 5 | 3 |
| ONOS | | | | 1 | 7 | 5 |
| POX | | | | | 1 | 1/3 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 1.99%

**Table 31.** Overlay networks

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| Floodlight | 1 | 1 | 1/5 | 1/5 | 1 | 1/5 |
| RYU | | 1 | 1/5 | 1/5 | 1 | 1/5 |
| ODL | | | 1 | 1 | 5 | 1 |
| ONOS | | | | 1 | 5 | 1 |
| POX | | | | | 1 | 1/5 |
| Tungsten | | | | | | 1 |

inconsistency ratio = 0.00%

**Table 32.** Isolation

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| **Floodlight** | 1 | 1 | 1/5 | 1/5 | 1 | 1/5 |
| **RYU** | | 1 | 1/5 | 1/5 | 1 | 1/5 |
| **ODL** | | | 1 | 1 | 5 | 1 |
| **ONOS** | | | | 1 | 5 | 1 |
| **POX** | | | | | 1 | 1/5 |
| **Tungsten** | | | | | | 1 |

inconsistency ratio = 0.00%

**Table 33.** Traffic protection solutions

| NOS | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
|---|---|---|---|---|---|---|
| **Floodlight** | 1 | 3 | 1/7 | 1/5 | 3 | 1/3 |
| **RYU** | | 1 | 1/9 | 1/7 | 1 | 1/5 |
| **ODL** | | | 1 | 3 | 7 | 5 |
| **ONOS** | | | | 1 | 5 | 3 |
| **POX** | | | | | 1 | 1/5 |
| **Tungsten** | | | | | | 1 |

iconsistency ratio = 5.51%

### 5.2.4. Traffic protection solutions

Finally, Floodlight provides proactive ACL and basic firewall by installing Openflow flow entries reactively in terms of secure traffic. Besides, POX can block connections based on Media Access Control (MAC) address. RYU can apply firewall principles based on VLAN and Switch id. However, Tungsten, ONOS and ODL provide Service Function Chaining (SFC) by which an ordered list of network services (e.g. firewalls, intrusion detection system, etc.) can be defined, consequently deploying softwarized security functions in CDC becomes more flexible. In this context, OpenStack provides integrating SFC drivers to implement SFC using ODL and ONOS. Moreover, ODL provides a logical service function forwarder that facilitates service function mobility, load balancing and failover. In addition, ONOS provides a policy framework as an abstraction layer that hides the details of the control and data planes. This framework simplifies the enforcement of actions to the network by installing OpenFlow rules to provide functionalities like firewall and NAT. ODL provides NetVirt to apply ACL in ingress and egress modes to VMs created by Neutron northbound API. Tungsten provides security policies based on tags applied to projects, networks, vRouters, VMs and interfaces. Consequently, more granular policies can be created as well as security policy administration and troubleshooting are more feasible.

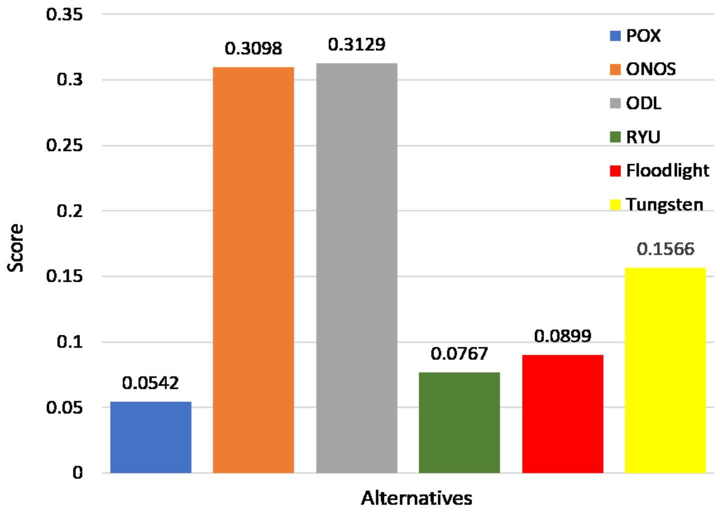

**Figure 5:** Total final scores

## 6. Conclusion

In this study, we presented the best suited NOS to be used in CDC by assessing the specifications of six NOSs, according to the criteria imposed by cloud data center requirements. Besides, we presented to which extend each NOS meets these criteria. To the best of our knowledge, this is the first study in the literature tackling such a problem. For this sake, first, we determined the requirements of CDC by investigating the characteristics of such an environment. Then, we classified the CDC requirements into functional and non-functional criteria based on which we inspected the specifications of the investigated NOSs. AHP was used, which requires much input conducted by pair-wise comparisons of the criteria to determine their importance by calculating their weights. As well as, we pair-wise compared the alternatives to determine the best one accurately against each criterion. Therefore, AHP's flaw is that it imposes many comparisons. We found that ODL and ONOS have the higher scores in comparison to the other NOSs. In particular, ODL has 31.29%, ONOS has 30.98% whereas Tungsten, Floodlight, Ryu and POX have 15.66%, 8.99%, 7.67% and 5.42%, respectively. A continuous follow-up of the NOSs should be conducted since Software Defined Cloud Computing (SDCC) is still in the stages of maturity and adoption. Moreover, the related technologies are developing rapidly because of the tendency to convert CDC into Software Defined Environment (SDE). In this work, we also presented the essential requirements for SDN based CDC. However, several requirements need more investigation under other use cases. In this context, it is essential to extend the current NOSs to provide network services in case of inter-CDCs, e.g., deploying SFC whose VMs belong to multiple CDCs managed by different cloud orchestrating instances. Furthermore, maintaining the synchronization between multiple NOSs belong to different CDCs managed by different cloud orchestrating instances is still problematic, and it needs more investigation to resolve potential synchronization conflicts. In addition, considering container based cloud environment sheds the light on new challenges in relation to the integration between the container orchestrator and SDN NOS. The previous challenges will be in our future work plans, where we need to investigate the NOSs capabilities and deficiencies to propose new solutions addressing the mentioned challenges.

**Table 34.** Functional and non-functional feature comparison between NOS

| **Criterion** | **Alternatives** | | | | | |
|---|---|---|---|---|---|---|
| | **Floodlight** | **RYU** | **ODL** | **ONOS** | **POX** | **Tungsten** |
| **Scalability** | Centralized | Centralized | Distributed | Distributed (Network Segmentation) | Centralized | Distributed |
| **Security** | TLS/SSL | TLS/SSL | USC | TLS/SSL, HTTPs | TLS/SSL | TLS/SSL |
| **Modularity** | Maven | Python components | Apache Karaf | Apache Karaf | Python Components | Unknown |
| **Development Community** | Good | Good | Excellent | Excellent | Poor | Very Good |
| **North API** | Rest API | Rest API | Rest/Restconf API | Rest API | Rest API | Rest API |
| **GUI** | Yes | Yes | Yes | Yes | NO | Yes |
| **Documentation** | Good | Good | Excellent | Excellent | Poor | Excellent |
| **Update Frequency** | Poor | Good | Excellent | Excellent | Poor | Very Good |
| **Application Availability** | Good | Good | Excellent | Excellent | Poor | Very Good |
| **South API – Management Protocols** | N/A | OF-config, NETCONF, OVSDB, Nicira ext | NETCONF, OVSDB, SNMP | NETCONF, OVSDB, SNMP | N/A | NETCONF, SNMP |
| **South API – Control Protocols** | OF 1.0, 1.3, OpenFlowJ-Loxigen | OF 1.0-1.5, BGP | OF 1.1, 1.3, OpFlex, PCEP, BGP, LISP, MD-SAL | OF 1.1, 1.3, PCEP, BGP | OF 1.0 | XMPP, BGP |
| **Availability** | Master/Slave | Master/Slave | Active/Backup | Master/Standby | N/A | Active/Backup |
| **Faults Verification & Troubleshooting** | Oftest, Flowscale | ofctl, of13, ofctl_test | Status And Diagnostics Framework, networking-odl ML2 | CPMan, Ganglia Troubleshooting module | N/A | Analytic Nodes |
| **Load Balancing** | Basic | N/A | Basic | Mastership rebalancing | Basic | Third party |
| **Quality of Service** | OF meter & queue | DiffServ | PCEP | SDNi | N/A | DiffServ |
| **Overlay Networks** | Virtual Network | VLAN | VLAN, VxLAN, L3 GRE | VLAN, VxLAN, L3VPN | VLAN | BGP EVP, VxLAN |
| **Isolation** | L2 Based | L2 Based | L2 & L3 Tunneling, EVC | L2 & L3 Tunneling, EVC | N/A | MPLSoUDPGRE, VxLAN |
| **Traffic Protection Solutions** | Forwarding rules, ACL | VLAN-id & SW-id based | SFC | SFC | MAC blocker | SFC, Applications tag |

**Table 35.** Final numbers of the pair wise comparison of all alternatives regarding all criteria

| Criterion | Alternatives | | | | | |
|---|---|---|---|---|---|---|
| | Floodlight | RYU | ODL | ONOS | POX | Tungsten |
| Scalability | 0.0876 | 0.0876 | 0.2345 | 0.4129 | 0.0842 | 0.0932 |
| Security | 0.131 | 0.0736 | 0.3631 | 0.2276 | 0.0736 | 0.131 |
| Modularity | 0.1111 | 0.1111 | 0.312 | 0.312 | 0.1111 | 0.0428 |
| Development Community | 0.0628 | 0.0628 | 0.3226 | 0.3226 | 0.0361 | 0.1932 |
| North API | 0.2164 | 0.1 | 0.4162 | 0.115 | 0.0446 | 0.1077 |
| GUI | 0.0837 | 0.0548 | 0.1857 | 0.1857 | 0.0403 | 0.4498 |
| Documentation | 0.1128 | 0.1128 | 0.3075 | 0.3075 | 0.0466 | 0.1128 |
| Update Frequency | 0.0499 | 0.1216 | 0.3068 | 0.3068 | 0.0252 | 0.1898 |
| Application Availability | 0.117 | 0.0636 | 0.3359 | 0.3359 | 0.0306 | 0.117 |
| South API – Management Protocols | 0.0692 | 0.1121 | 0.434 | 0.2789 | 0.0367 | 0.0692 |
| South API – Control Protocols | 0.1782 | 0.1782 | 0.3252 | 0.1782 | 0.0468 | 0.0936 |
| Availability | 0.0953 | 0.0598 | 0.3621 | 0. 3621 | 0.0302 | 0.0905 |
| Faults Verification & Troubleshooting | 0.1016 | 0.1016 | 0.2897 | 0.2897 | 0.0447 | 0.1726 |
| Load Balancing | 0.0988 | 0.0274 | 0.0988 | 0.5772 | 0.0988 | 0.0988 |
| Quality of Service | 0.0916 | 0.0916 | 0.2292 | 0.4564 | 0.0395 | 0.0916 |
| Overlay Networks | 0.0556 | 0.0556 | 0.2778 | 0.2778 | 0.0556 | 0.2778 |
| Isolation | 0.0556 | 0.0556 | 0.2778 | 0.2778 | 0.0556 | 0.2778 |
| Traffic Protection Solutions | 0.0703 | 0.0331 | 0.4684 | 0.2516 | 0.0377 | 0.1389 |

## Acknowledgement

Project no. 135074 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary under the FK_20 funding scheme.

## References

[1] Shuo Fang, Yang Yu, Chuan Heng Foh and Khin Mi Mi Aung, "A Loss-Free Multipathing Solution for Data Center Network Using Software-Defined Networking Approach", *IEEE Transactions on Magnetics*, pp. 2723-2730, Vol. 49, No. 6, 2013, DOI: 10.1109/tmag.2013.2254703, Available: https://ieeexplore.ieee.org/document/6522268.

[2] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson *et al*., "OpenFlow: Enabling Innovation in Campus Networks", *SIGCOMM Computer Communication Review*, pp. 69-74, Vol. 38, No. 2, DOI: 10.1145/1355734.1355746, Available: https://dl.acm.org/doi/10.1145/1355734.1355746.

[3] Christina Delimitrou, Sriram Sankar, Aman Kansal and Christos Kozyrakis, "ECHO: Recreating Network Traffic Maps for Datacenters with Tens of Thousands of Servers", in *Proceeding of the IEEE International Symposium on Workload Characterization (IISWC)*, 4 November 2012, pp. 14-24, DOI: 10.1109/IISWC.2012.6402896, published by IEEE, Available: https://ieeexplore.ieee.org/document/6402896.

[4] Yaomin Wang, Xia Wang, Haiyan Li, Yi Dong, Qing Liu *et al.*, "A Multi-Service Differentiation Traffic Management Strategy in SDN Cloud Data Center", *Computer Networks*, Vol. 171, No. C, April 2020, DOI: 10.1016/j.comnet.2020.107143, Available: https://dl.acm.org/doi/abs/10.1016/j.comnet.2020.107143.

[5] Rahamatullah Khondoker, Adel Zaalouk, Ronald Marx and Kpatcha Bayarou, "Feature-Based Comparison and Selection of Software Defined Networking (SDN) Controllers", in *Proceeding of the IEEE World Congress on Computer Applications and Information Systems (WCCAIS)*, 17 January 2014, pp. 1-7, published by IEEE, DOI: 10.1109/WCCAIS.2014.6916572, Available: https://ieeexplore.ieee.org/document/6916572.

[6] Firas Fawzy Zobary, "Applying TOPSIS Method for Software Defined Networking (SDN) Controllers Comparison and Selection", in *Proceedings of the International Conference on Communications and Networking*, 24-26 September 2016, pp. 132–141, Print ISBN: 978-3-319-66627-3, DOI: 10.1007/978-3-319-66628-0_13, Available: https://link.springer.com/chapter/10.1007%2F978-3-319-66628-0_13.

[7] Omayma Belkadi and Yassin Laaziz, "A systematic and generic method for choosing a SDN controller", *International Journal of Computer Networks and Communications Security,* Vol. 5, No. 11, 2017, PP. 239-247, available: https://ijcncs.org/published/volume5/issue11/.

[8] Jehad Ali, Byeong-hee Roh and Seungwoon Lee, "QoS improvement with an optimum controller selection for software-defined networks", *Plos One*, Vol. 14, No. 5, 2019, pp. 1-37, Published by Public Library of Science, DOI: 10.1371/journal.pone.0217631, Available: https://doi.org/10.1371/journal.pone.0217631.

[9] Mohammed Abdul Rahman AlShehri and Shailendra Mishra, "Feature based comparison and selection of SDN controller", *International Journal of Innovation and Technology Management*, Vol. 16, No. 5, 2019, DOI: 10.1142/S0219877019500305, Available: https://www.worldscientific.com/doi/abs/10.1142/S0219877019500305.

[10] Esmaeil Amiri, Emad Alizadeh and Mohammad Hossein Rezvani, "Controller selection in software defined networks using best-worst multi-criteria decision-making", Bulletin of Electrical Engineering and Informatics, Vol. 9, No. 4, 2020, pp. 1506-1517, DOI: https://doi.org/10.11591/eei.v9i4.2393, Available: https://beei.org/index.php/EEI/article/view/2393.

[11] Jehad Ali, Byungkyu Lee, Jimyung Oh, Jungtae Lee and Byeong-hee Roh, "A Novel Features Prioritization Mechanism for Controllers in Software-Defined Networking", *Computers, Materials & Continua,* Vol. 69, No. 1, 2021, pp. 267–282, DOI: 10.32604/cmc.2021.017813, Available: http://www.techscience.com/cmc/v69n1/42766.

[12] Jehad Ali and Byeong-hee Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering", *Computers, Materials & Continua,* Vol. 67, No. 1, 2021, pp. 849-875, DOI: 10.32604/cmc.2021.014576, Available: http://www.techscience.com/cmc/v67n1/41200.

[13] Abdelrahman Abuarqoub, "A review of the control plane scalability approaches in software defined networking", *Future Internet,* Vol. 12, No. 3, 2020, Published by MDPI, DOI: 10.3390/fi12030049, Available: https://www.mdpi.com/1999-5903/12/3/49.

[14] David Espinel Sarmiento, Adrien Lebre, Lucas Nussbaum and Abdelhadi Chari, "Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey", *IEEE Communications Surveys & Tutorials*, Vol. 23, No. 1, pp. 256-281, DOI: 10.1109/COMST.2021.3050297, Available: https://ieeexplore.ieee.org/document/9319748.

[15] Mohammad Riyaz Belgaum, Shahrulniza Musa, Muhammad Mansoor Alam and Mazliham Mohd Su'ud, "A systematic review of load balancing techniques in software-defined networking", *IEEE Access,* Vol. 8, 2020, pp. 98612-98636, DOI: 10.1109/ACCESS.2020.2995849, Available: https://ieeexplore.ieee.org/document/9097181.

[16] Mosab Hamdan, Entisar Hassan, Ahmed Abdelaziz, Abdallah Elhigazi, Bushra Mohammed *et al.*, "A comprehensive survey of load balancing techniques in software-defined network", *Journal of Network and Computer Applications*, Vol. 174, pp. 102856, 2021, DOI: 10.1016/j.jnca.2020.102856, Available: https://www.sciencedirect.com/science/article/pii/S1084804520303222.

[17] Bassey Isong, Reorapetse Ramoliti Samuel Molose, Adnan M. Abu-Mahfouz, and Nosipho Dladlu, "Comprehensive review of SDN controller placement strategies", *IEEE Access,* Vol. 8, 2020, pp. 170070-170092, DOI: 10.1109/ACCESS.2020.3023974, Available: https://ieeexplore.ieee.org/document/9195810.

[18] Liehuang Zhu, Md M. Karim, Kashif Sharif, Chang Xu, Fan Li *et al.*, "SDN controllers: A comprehensive analysis and performance evaluation study", *ACM Computing Surveys,* Vol. 53, No. 6, 2020, pp. 1-40, DOI: https://doi.org/10.1145/3421764, Available: https://dl.acm.org/doi/fullHtml/10.1145/3421764.

[19] Thomas L. Saaty, *The Hierarchy Analytical Process*, New York, USA: McGraw-Hill, 1980.

[20] Diego Kreutz, Fernando Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky *et al.,* "Software-defined networking: A comprehensive survey", in *Proceedings of the IEEE,* Vol. 103, No. 1, 2014, pp. 14-76, DOI: 10.1109/JPROC.2014.2371999, Available: https://ieeexplore.ieee.org/document/6994333.

[21] Surbhi Saraswat, Vishal Agarwal, Hari Prabhat Gupta, Rahul Mishra, Ashish Gupta *et al.,* "Challenges and solutions in Software Defined Networking: A survey", *Journal of Network and Computer Applications,* Vol. 141, 1 September 2019, pp. 23-58, Published by Elsevier Ltd., DOI: 10.1016/j.jnca.2019.04.020, Available: https://www.sciencedirect.com/science/article/pii/S1084804519301444.

[22] Jiuxin Cao, Zhuo Ma, Jue Xie, Xiangying Zhu, Fang Dong *et al.,* "Towards tenant demand-aware bandwidth allocation strategy in cloud datacenter", *Future Generation Computer Systems,* Vol. 105, 2020, pp. 904-915, DOI: 10.1016/j.future.2017.06.005, Available: https://www.sciencedirect.com/science/article/pii/S0167739X17311834.

[23] Davide Adami, Barbara Martini, Andrea Sgambelluri, Lisa Donatini, Molka Gharbaoui *et al.,* "An SDN orchestrator for cloud data center: System design and experimental evaluation", *Transactions on Emerging Telecommunications Technologies,* Vol. 28, No. 11, 2017, DOI: 10.1002/ett.3172, Available: https://onlinelibrary.wiley.com/doi/10.1002/ett.3172.

[24] Richard Cziva, Simon Jouët, David Stapleton, Fung Po Tso and Dimitrios P. Pezaros, "SDN-based virtual machine management for cloud data centers", *IEEE Transactions on Network and Service Management,* Vol 13, No. 2, 2016, pp. 212-225, DOI: 10.1109/TNSM.2016.2528220, Available: https://ieeexplore.ieee.org/document/7416246.

[25] Jungmin Son and Rajkumar Buyya, "A taxonomy of software-defined networking (SDN)-enabled cloud computing", *ACM Computing Surveys,* Vol. 51, No. 3, 2018, pp. 1-36, DOI: 10.1145/3190617, Available: https://dl.acm.org/doi/10.1145/3190617.

[26] Yaser Jararweh, Mahmoud Al-Ayyoub, Elhadj Benkhelifa, Mladen Vouk and Andy Rindos, "Software defined cloud: Survey, system and evaluation", *Future Generation Computer Systems,* Vol. 58, 2016, pp. 56-74, DOI: 10.1016/j.future.2015.10.015, Available: https://www.sciencedirect.com/science/article/pii/S0167739X15003283.

[27] Seung Won Shin, Phillip Porras, Vinod Yegneswara, Martin Fong, Guofei Gu *et al.,* "Fresco: Modular composable security services for software-defined networks", In *proceeding of 20th Annual Network & Distributed System Security Symposium*, 26 February 2013, Published by NDSS, Available: http://hdl.handle.net/10203/205914.

[28] Jungmin Son and Rajkumar Buyya, "SDCon: Integrated control platform for software-defined clouds", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 1, 2019, pp. 230-244, DOI: 10.1109/TPDS.2018.2855119, Available: https://ieeexplore.ieee.org/document/8409965.

[29] Qiao Yan, F. Richard Yu, Qingxiang Gong and Jianqiang Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges", *IEEE communications surveys & tutorials*, Vol. 18, No. 1, 2016, pp. 602-622, DOI: 10.1109/COMST.2015.2487361, Available: https://ieeexplore.ieee.org/document/7289347.

[30] Peng Qin, Bin Dai, Benxiong Huang and Guan Xu, "Bandwidth-aware scheduling with sdn in hadoop: A new trend for big data", *IEEE Systems Journal*, Vol. 11, No. 4, 2015, pp. 2337-2344, DOI: 10.1109/JSYST.2015.2496368, Available: https://ieeexplore.ieee.org/document/7332913.

[31] Murat Karakus and Arjan Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)", *Computer Networks*, Vol. 112, 2017, pp. 279-293, DOI: 10.1016/j.comnet.2016.11.017, Available: https://www.sciencedirect.com/science/article/pii/S138912861630411X.

[32] Fouad Benamrane, Mouad Ben Mamoun and Redouane Benaini, "An East-West interface for distributed SDN control plane: Implementation and evaluation", *Computers & Electrical Engineering*, Vol. 57, January 2017, pp. 162-175, Published by Elsevier Ltd., DOI: 10.1016/j.compeleceng.2016.09.012, Available: https://www.sciencedirect.com/science/article/pii/S0045790616302798.

[33] Basem Almadani, Abdurrahman Beg and Ashraf Mahmoud, "DSF: A distributed sdn control plane framework for the east/west interface", *IEEE Access*, Vol. 9, 2021, pp. 26735-26754, DOI: 10.1109/ACCESS.2021.3057690, Available: https://ieeexplore.ieee.org/document/9349440.

[34] Ziad A Al-Sharif, Yaser Jararweh, Ahmad Al-Dahoud and Luay M. Alawneh, "ACCRS: autonomic based cloud computing resource scaling", *Cluster Computing*, Vol. 20, No. 3, 2017, pp. 2479-2488, DOI: 10.1007/s10586-016-0682-6, Available: https://link.springer.com/article/10.1007/s10586-016-0682-6.

[35] Aaqif Afzaal Abbasi, Almas Abbasi, Shahaboddin Shamshirband, Anthony Theodore Chronopoulos, Valerio Persico *et al.*, "Software-defined cloud computing: A systematic review on latest trends and developments", *IEEE Access*, Vol. 7, 2019, pp. 93294-93314, Published by IEEE, DOI: 10.1109/ACCESS.2019.2927822, Available: https://ieeexplore.ieee.org/document/8758941.

[36] Gwo-Hshiung Tzeng and Jih-Jeng Huang, *Multiple attribute decision making: methods and applications*, Florida, USA: CRC press, 2011, ISBN 9781439861578.

[37] Thomas L. Saaty, *Decision making with dependence and feedback: The analytic network process*, Pittsburgh, USA: RWS Publications, Vol. 4922, No. 2, 1996, ISBN: 9780962031793.

[38] Jafar Rezaei, "Best-worst multi-criteria decision-making method", *Omega*, Vol. 53, 2015, pp. 49-57, DOI: 10.1016/j.omega.2014.11.009, Available: https://www.sciencedirect.com/science/article/pii/S0305048314001480.

[39] Mehmet Şahin, "A comprehensive analysis of weighting and multicriteria methods in the context of sustainable energy", *International Journal of Environmental Science and Technology*, Vol. 18, No. 6, 2021, pp. 1591-1616, DOI: 10.1007/s13762-020-02922-7, Available: https://link.springer.com/article/10.1007/s13762-020-02922-7.

[40] Meenu Singh and Millie Pant, "A review of selected weighing methods in MCDM with a case study", *International Journal of System Assurance Engineering and Management*, Vol. 12, No. 1, 2021, pp. 126-144, DOI: 10.1007/s13198-020-01033-3, Available: https://link.springer.com/article/10.1007/s13198-020-01033-3.