Network augmentation by dynamically splitting the switching function in SDN

Rawal, Bharat S.; Manogaran, Gunasekaran; Singh, Raman; M., Poongodi; Hamdi, Mounir

Published in:
2021 IEEE International Conference on Communications Workshops (ICC Workshops)

DOI:
10.1109/ICCWorkshops50388.2021.9473709

Published: 09/07/2021

Document Version
Peer reviewed version

Rawal, B. S., Manogaran, G., Singh, R., M., P., & Hamdi, M. (2021). Network augmentation by dynamically splitting the switching function in SDN. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)* (IEEE Proceedings).
IEEE. https://doi.org/10.1109/ICCWorkshops50388.20

Network Augmentation by Dynamically Splitting the Switching Function in SDN

Bharat S Rawal, [2]Gunasekaran Manogaran, [3]Raman Singh [4]Poongodi M, [5]Mounir Hamdi
[1&2] Cyber Department, Gannon University, Erie, PA, USA.
[3]School of Comp Sci & Stats, Trinity College Dublin, Dublin, Ireland
[1]rawalksh001@gannon.edu, [2]manogara001@gannon.edu, [3]rasingh@tcd.ie, [4]dr.m.poongodi@gmail.com and [5]mhamdi@hbku.edu.qa

Abstract— **Software-defined networking is a synonym for the term programmable network and is the Key for 5G and beyond networking paradigms. Software-defined network (SDN) provides network management and controlling features irrespective of the hardware configurations and network infrastructure. Network slicing is the process of separation of multiple virtual networks based on different functions or tasks such that one application does not interfere with another network. Network slicing allows separating the control plane from the user's plane. Various investigators have investigated how the slices that have been used for splitting paths can be measured to improve durability. In SDN, the dynamic migration of switches offers a method of offloading the load from one controller to another controller. We have reintroduced the concept of the network (dataflow) splitting for load balancing in SDN. In this paper, we compare the performance techniques and found that the splitting paradigm with dynamic migration offers the most balanced network flow and least overhead on the SDN controller.**

Keywords: Slices, Splices, Software-defined networks, splitting, 5G, Open-flow

I. INTRODUCTION

The programmable network devices' behavior and control flow is independently handled by software that is apart from the hardware. A valid program network can allow the engineer to reprogram any infrastructure of the network instead of doing it manually. SDN offers the following benefits to 5G network designers [1].

1. Long term costs can be decreased.
2. The device information protection capabilities for any application are granted.
3. Resource constraints and application status can be better answered through SDN.
4. Network bandwidth and resources can have a better allocation.
5. Traffic determining can be done via packet prioritizing.
6. Transparency augmentation and flexibility of operation can be increased.
7. Better privacy and security machinery can be reinforced.

Software defined network (SDN) has network programmability as the center. Open-flow is currently the most active software-defined network creation tool, in which the routing tables can be controlled remotely. Open-flow has a centralized packet-moving decision capability, which allows the system to have data center gear and individual switches that are independently programmed. Software-defined networking is the synonym for the term programmable network it is widely used in recently introduced 5G network.

In this paper, we study mainly three processes, namely network slicing, network splicing, and network splitting. As shown in Fig 1., network slicing is the process of separation of multiple virtual networks based on different functions or tasks such that one application does not interfere with another network [17]. Path Splicing: Path Splicing: Fig 2 shows, if a connection fails on each side of the n edge-disjoint, the pairs of nodes on each side of the graph will be separated [18]. Network splitting is the network process functions spread over different network elements. The role must be delegated to one node and another node [2-5].

The paper is organized as follows. Section 2 talks about our motivation and contribution. Section 3 describes related work. Network Slicing, Splicing, and Splitting algorithms are explained formally in Section 4. Section 5 describes the analytical analysis. Experimental results and performance analysis are presented in Section 6. We conclude the paper in Section 7.

II. MOTIVATIONS AND CONTRIBUTIONS

Benefits of SDN are already discussed in previous section. In this section, we discuss our motivations and the contributions of our work. Hardware-based encryption performs encryption and decryption with the help of an onboard device. It is self-reliant and does not need the use of any extra software. Consequently, it is free from the likelihood of infection or vulnerability. A hardware-based security system is not suitable for all available systems and does not offer flexibility.

On the other hand, the software-based system provides a great deal of flexibility. However, at the same time, it leaves much room for cyber-attack. SDN is not an exception from the risk of cyber-attacks. We have studied several latest articles on SDN and listed possible cybersecurity risks and proposed mitigation techniques in SDN. Security problem identified [13]: The transfer of data, capacity to make right decisions in the future of the

internet of vehicles. In 5G era, even in the absence of roadside units, a vehicle-to-vehicle communication is going to be connected with many IoT devices. The primary problem of the Internet of Vehicle (IoV) would be to meet the quality of service. The IoV should be capable of catering to all the required changes in the quality of service.

A hierarchical load-based routing scheme has been proposed, which makes use of the SDN central controller system and the designing of this hierarchical routing can be used for both locally based and globally exploited perspectives [13]. The single logical point of the SDN-based network framework makes it simpler for the design and the operations, which makes it easier for enabling the carriers to acquire vendor-independent control on the whole network [15]. We have demonstrated slicing, splicing, and splitting networks for optimal load distribution and better quality of services for IoT and IoV.

III. RELATED WORKS

Ren et al. [6] presented a two-level hierarchy that is defined between master and slave nodes in the IoT platform. The objective of this work is to address the master and slave issue for this slave controller placement strategy (SCPS) method is introduced. By processing this, it decreases control delay, whereas to address the significant nodes in IoT binary particle swarm optimization algorithm is proposed. Nguyen et al. [7] proposed a decentralized and revised content-centric networking (CCN) to reduce the discovery time of services. It is processed based on MEC services, which are associated with three-tier architecture. Here, it decreases service delay and processing time by performing a round trip calculation. Information-Centric Networking (ICN) is developed to resolve Content-Centric Delivery Networks (C-CDN).

In [8], the author discovers the processing time, which is based on partial Quorum configurations. In this work, a standard duplication technique is transformed to be a scalable and intelligent technique; it is performed by evaluating Quorum-replicated consistency. IoT data transmission is proposed based on the MEC technique to address latency, jitter, and packet drop during transmission. In [9], an efficient dynamic resource sharing scheme (E-DRSS) is developed in IoT devices by combining radio remote head (RRH) for reliable QoS communication. The communication is enhanced based on the narrowband Internet of Things (NB-IoT).

Ateya et al. [10] implemented a Salp Swarm Optimization Algorithm (SSOA), which is processed as a dynamic optimization method. Here, a reliable connection is established between switches and controllers in SDN networks. A meta-heuristic way is introduced to improve the run-time and reliability, which is performed based on game theory.

Sharma, P. K et al. [16] proposed Open cloud software-defined wireless network security for the Internet of Things (OpCloudSec) security attack mitigation architecture with a highly programmable monitoring network to allow attack identification. The results have shown that our OpCloudSec proposal is successful when it comes to tackling new challenges. The detection algorithm for deduction line packages is quick enough and carries out a high identification rate. Qiao Yan et al. [11] presented a multi-level DDoS mitigation framework (MLDMF) to protect against IIoT DDoS attacks, including the edge level of computing, the fog level, and the cloud level. The networking software specified is used for managing several IIoT devices and for mitigating DDoS attacks in IIoT. Experimental findings demonstrate the efficiency of the proposed system.

Li-Jun et al. [12] analyzed Software-Defined Networking Enhanced Edge Computing (SDNEEC) to determine how SDN and related technologies are incorporated to promote edge server and IoT system management and service. They concentrate on how SDN can be used to provide coherent and programmable system management interfaces. The funding of the SDN network is proposed for edge computing is explored further through our discussion. Tryfon Theodorou et al. [13] proposed a Multi-Protocol Software-Defined Networking (MPSDN) for IoT, which uses the required SDN interface to enforce service knowledge of complex, resource-constrained IoT environments. Several on-demand networking protocols and an Interface that provides a tailor-made dashboard and real-time visualization.

Even though SDN has some favorable aspects that it brings to the administrative framework of the system, it can be relatively defenseless among security attacks since these attacks can be made on the information plane using components of the system from inside and using SBI-API through the control layer being attacked [14]. The single logical point of the SDN based network framework makes it simpler for the design and the operations, which makes it easier for enabling the carriers to acquire independent vendor control on the whole network. Devices over the network are also simplified due to the SDN framework as they are not required to be familiar with all the standard protocols but instead of the basic instructions from the controllers of the SDM framework [15]. The CISCO systems have employed slicing techniques for better utilization of wireless network and improve quality of service [19, 20,21]

IV. NETWORK SLICING, SPLICING, AND SPLITTING

As shown in Fig 1. network slicing is the method of the parting of multiple virtual networks based on different roles or tasks such that one application does not interfere with another network [17]. The virtual network architecture operates on the principle of software-defined network and network function virtualization on a fixed network for better flexibility and efficiency. Network slicing allows

separating the control plane from the user's plane to controls the flow of data.

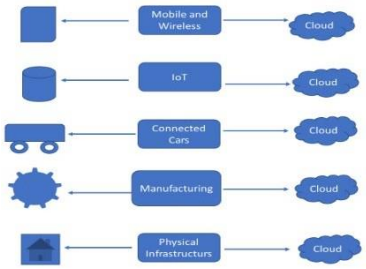

Figure 1. Network slicing

Path Splicing: As seen in Fig 2., if a connection is lost on either side of an edge disjoint, the pair of nodes will be disconnected from each side of the graph. We have already explored split-protocol (task delegation) for load balancing, reliability, security, and migration purpose.

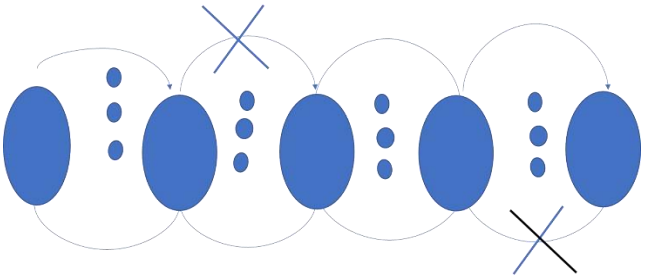

Figure 2. Network splicing analysis

1) The split protocol allows task delegation [2], in case of a dedicated slice of mobile and wireless communication fails at that moment, the slice handling communication will also work for mobile communication [3].
2) If any link fails to notify to other corresponding link and it will change from unidirectional to by directional link [4].
3) If any node fails immediately migrates all communication and backup node and role changeover take places [5]. The split-protocol components are treated as parallel components. **Network Configurations:** To illustrate the effect proposed techniques, we demonstrate with three different configurations.

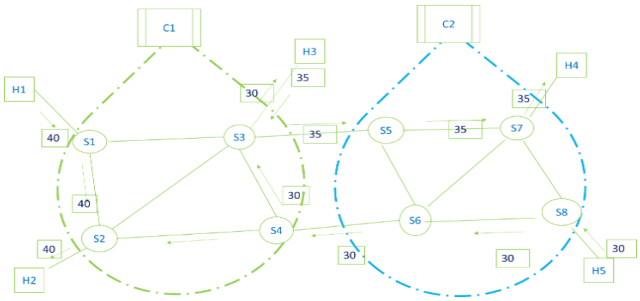

Figure 3. Configuration1 [18]

Configuration 1.

As shown in Fig 3., Host H5 generates 30 new flows/seconds to H3, this flow is routed through H5→S8→S6→S4→S3→H3.

Suppose path computation for single flow requires α unit of load and β is the load necessary for installation of single rules in the switch. Let delegated (splitting) task α* and β*; α =100 α* and β= 100β*, here α=1 and β=.1 The delegation task is a process of merely forwarding tasks to somebody you trust and willing to do your job. The nodes/ switches which are delegated simply hands over the task. They will not spend time understanding how to process and perform the task.

Load at Controller C1 is

The flow from H1→H2 generates 40 new flows/seconds and 40 rules on S1 and S2.

Load = 40α+(40+40) β = 48 --------(1)

And the 30flows/seconds from S3->H3 will generate load = 30α+(30+30) β =36 --------- (2)

The 35 flows/seconds from H3-> H4

35α+35β= 38.5 ----------(3)

Total load C1 from equation 1,2 and3.

=40α+(40+40) β +30α+(30+30) β +35α+35β = 105α+185 β

If α =1 and β=.1

= 105+ 18.5 = 122.5 Units ----------(4)

And At controller C2 =

65α+130 β= 78 Units ----------(5)

Total of Both controller **C1+C2 = 200.5 Units ---- (A)**

From Configuration 2. As shown in Fig 4., the load at controller C1 is

= 40α+40β = 44 units

Load at Controller C2 is

(30+35) α + (120+ 105) β = 65+22.5 = 87.5 units.

C1+C2 = 131.5 -------(B)

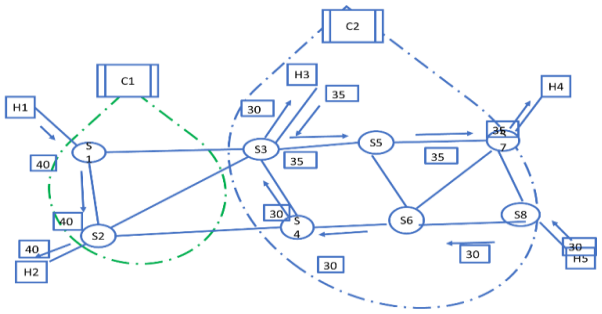

Figure 4. Configuration 2 With Switch Migration [18]

Split configuration -3:

As shown in Fig 5., the traffic for generated the host remains the same only switch split the task. Host H5 generates 30 flows/seconds for Host H3, and the traffic load is split into equal two routes.

H5→S8→S6→S4→S3→H3.&

H5→S8→S7→S5→S3→H3

H5→S8→S6→S4→S3→H3.

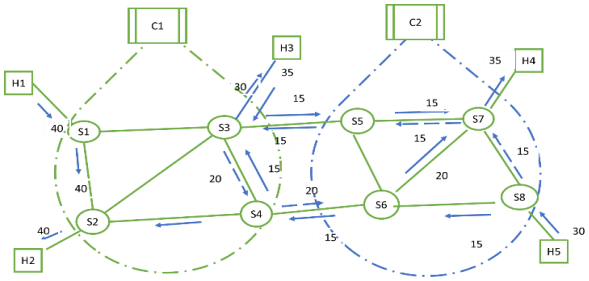

Figure 5. Configuration 3. Split configuration.

$(15\alpha + 15\alpha^*) + (15+15)\beta + (15+15)\beta^*$

$15.15\alpha + 30.03\beta = 15.15 + 3.003 = 18.153$ Units -------(4)

Since, α and α* is taken in to account in Eq (4)

H5->S8->S7->S5->S3->H3

Since S8 delegates 50% flow to S7, so there is no cost β* for S8.

$(15+15)\beta = 30\ \beta = 3$ Units ----------(5)

For S6→S7→H4 Route

$20\alpha + (40)\beta = 24$ Units ----------(6)

For S5→S7→H4 Route

$15\alpha + (30)\beta = 18.0$ Units ----------(7)

Load at Controller C2 from Equations 4-7 = 63.153 Units.

Load at Controller C1

H1-> S1->S2->H2:

$40\ \alpha + 80\ \beta = 48$ Units ……. (8)

Route H3->S3->S5

$35\alpha + 15\ \beta = 36.5$ Unit ----------(9)

Route H3->S3->S4

$40\beta^* = 0.04$ Units ----------(10)

Load at Controller C1 **83.8 Units**

Total Load at C1 & C2 = 146.953 ---- C

As shown in Fig 6., the load at Controller C1 is the flow from H1->H2 generates 40 new flows/seconds and 40 rules on S1 and S2.

Load = $40\alpha + (40+40)\ \beta = 40.80$-------- (11)

S3->S1->H1

$15\alpha + 15\ \beta = 16.5$----------(12)

Load at C1 from Eq 11& 12 = 57.3

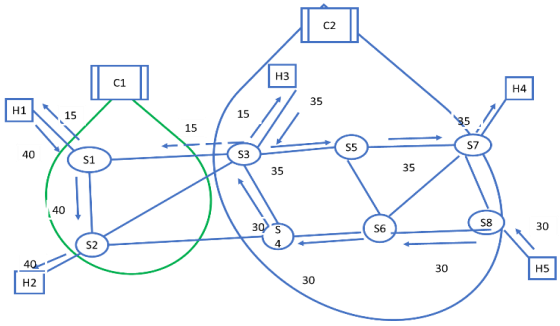

Figure 6. Configuration 4. Dynamic migration with split configuration.

And the 35 flows/seconds from H3→H4 will generate load

$= 35\alpha + (35+35+35)\ \beta = 35 + 10.5 = 45.5$----------(13)

The 15 flows/seconds from H5→S8→S6→S4→S3→ H3

$15\alpha + 15\alpha^* + (30+30+30+15)\ \beta + 15\beta^* = 15 + .15 + 10.5 + .015 = 25.665$ ---------- (14)

Total load C2 from Eq 13 and 14**. 71.165**

Total of Both Controllers C1+C2 = 128. 465 Units ---- (D)

Based on the mathematical formulations, there are many methods based on ear decompositions that allow us to measure two slices that can tolerate arbitrary single link failures, that does not split up the underlying network. In SDN, the dynamic migration of switches offers a method of offloading the load from one controller to another controller. We have introduced the third concept of network (dataflow) splitting. In this paper, we compared the performance of all three technics and noticed that the splitting paradigm offers better network flow control and less overhead on the SDN controller.

V. ANALYTICAL ANALYSIS

The three configuration settings, i.e., 1, 2, and 3, reduce the network data flow problem into a classic max-flow min-cut problem of network flow theorem. There are two unique vertices in this network, and the first is the source from where all the flow will originate or the switch or host which wants to forward the network traffic. The second unique vertex is the sink of the flow, which is the destination for that particular network flow. As shown in Fig 3, 4,5, and 6, the network graph is divided into two disjoint sets of vertices known as set S and set T. The source and destination node of network traffic should fall in either of these two disjoint sets in a mutually exclusive

fashion. The capacity of the flow is always the sum of the weights of the edges in the cut-set.

The source vertexes belong to set S and destination 't' belongs to set T. The capacity can be defined as the sum of weights of edges that are leaving any particular vertex to say 's', as shown here in equation 11.

$$Capacity\ (s) = sum\ of\ weight\ of\ edges\ leaving\ s$$

The capacitated network can be defined as given in equation 12 and represented as G. V is the set of vertices of the network whereas C is the capacity function defined on each edge and function can be described as C: $V^2 \rightarrow \mathbb{R}^+$.

$$G = (V, C, s, t) \qquad 12$$

A flow 'f' is an assignment of weights to the edges in such a way that $0 \leq f(e) \geq u(e)$, where f(e) is the maximum flow of the cut and u(e) is the capacity (S, T). The net flow which can be transferred across the cut will be equal to the amount leaving as shown in equation 13.

$$v(f) = \sum_{e\ out\ from\ S} f(e) - \sum_{e\ into\ S} f(e)$$

Consider a situation where host H5 wants to send data to host H3. In this case, the H5 will represents' whereas the H3 will be considered as 't'. The number of flows can be regarded as the capacity or weight of the edge as we want to maximize the flow through the edges.

In the S-T cut, its capacity is always the sum of the capacities of all the edges, which starts from S and ends at T, as shown in equation 14.

$$Capacity\ (S, T) = \sum_{v \in S} \sum_{w \in T} capacity(v \rightarrow w)$$

If equation 14 is applied to configurations 1, 2, and 3 given in Figure 6, Figure 7, and Figure 8, the capacity of these configurations is computed as 65, 80, and 35 respectively. [in the absence of weights/number_of_flows I assumed weight of S1-S3=30, S2-S3=20, and S2-S4=30].

The computation of load along the path depends upon the number of flows on each edge $u \rightarrow v$. If the overall bandwidth is considered as bidirectional, the residual flow can be computed using equation 15.

$$f_p(u, v) = \begin{cases} C_f(p) & if\ flow\ is\ u \rightarrow v \\ -C_f(p) & if\ flow\ is\ v \rightarrow u \\ 0 & in\ all\ other\ cases \end{cases} \qquad 15$$

The bidirectional nature of the network edge can be used in both ways. If flows are moving in both directions, the capacity of the edge can be calculated as given in equation 15. For the simplicity of computations, the total number of flows is given as unidirectional in Figures 3, 4 5, and 6 as we are focusing on transmission of data from one host/switch to another host/switch in these scenarios. The total number of flows can be computed by summing all the flows of edges falling in the path, as shown in equation 16.

$$Flow\ (s, t) = \sum_{s_i \rightarrow t_i} flow(u \rightarrow v)\ \ where\ (u, v) \in path\ (s, t) \qquad 16$$

Equation 16 computes the total number of flows along a path from source vertex's' to destination vertex 't'. Each intermediary forwarding host/ switch represents as vertex 'u' whereas as host/ switch which receives flow is represented as vertex 'v'. The flow along the path has associated overhead which is represented as α and β. The overhead α depicts the time consumed by the controller to compute the path for a single flow, whereas the overhead β represents the total load required to install the single rule in the switch configuration. The total load along a path can be computed as given in equation 17.

$$load\ (s, t) = \alpha \sum_{s_i \rightarrow t_i} flow(s) + x\beta \sum_{s_i \rightarrow t_i} flow\ (s) \qquad 17$$

In equation 17, the *load (set)* represents the total load computed along the path from vertex's' to vertex 't'. The *flow(s)* computes the total number of flows leaving source vertex's' whereas variable 'x' represents the total number of switches pass off along the full path from source vertex's' to destination vertex 't'.

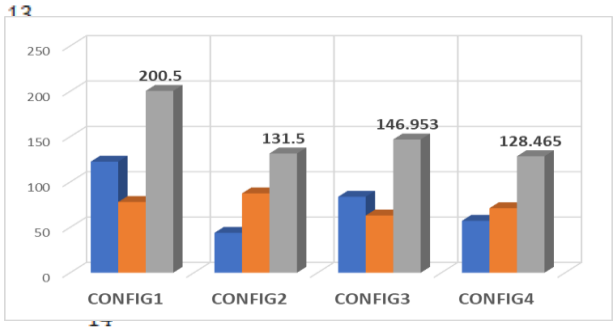

Figure 7. Load distribution in all three configurations

IoT comprises various instruments, such as embedded and communications systems. Data compression will reduce the content which the IoT has transmitted, processed, and stored to everybody. Finally, the protocol agreed can be used to boost the assigned bandwidth and to reduce contract latency, which improves the integration between the IoT and the SDN, as shown in Fig 7. based on load distribution.

VI. PERFORMANCE ANALYSIS

Fig 3., illustrates that configuration1. offers to higher average load for the same average number of flows. In Fig 4., configuration 2 shows the average load of 120 as highest for 100 numbers of flows. Fig 5., configuration 1, has the highest load of 200.5 units on the controller, and configuration 2, shows the second-lowest overall 131.5 units load on the controller for the same average flows. And configuration 3, the load is between configuration 1&2. And configuration 4, offers the lowest load in the system 128.465 units. Fig 8., shows the variance of load distribution between controllers 1&2. We can notice that configuration 4 provides the most downward variance of 24.63 % between controller C1&C2.

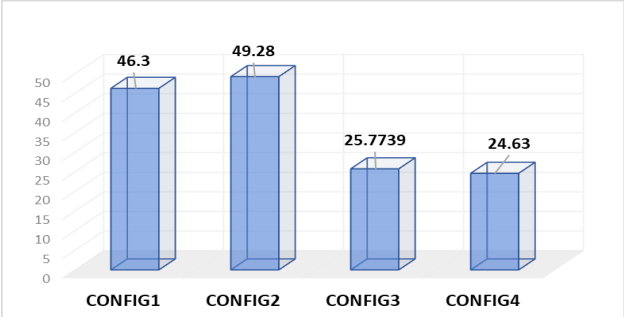

Figure 8. Variance in load distribution in all three configurations

VII. CONCLUSION & FUTURE WORK

The promising frameworks of SDN technologies based on split-architecture applications have been explored in this paper. The virtual network architecture operates on the principle of software-defined network and network function virtualization on a fixed network for better flexibility and efficiency. In 5G network slicing allows separating the control plane from the user's plane to monitor the flow of data. Many researchers have explored how the slices used for path splicing can be computed in such a way that the resilience to edge failures improves. In this paper, we compared the performance of various techniques and configurations and found that the splitting paradigm with dynamic migration offers the most balanced network flow and least overhead on the SDN controller. In large network splitting of the task is adopted based on optimal splicing of network slice. The split protocol also offers additional reliability and security because the service delegation slice is transparent to network users.

ACKNOWLEDGMENT: we really thank our graduate student Tarun Sai, Gollapudi for helping us in conducting a literature survey.

REFERENCES

1. Margaret rouse," programmable network", may 2013, https://searchnetworking.techtarget.com/definition/programmable-network-PN. Access date 10/10/2020.
2. Rawal, Bharat S., Ramesh K. Karen, and Alexander L. Wijesinha. "Mini Web server clusters for HTTP request splitting." In *2011 IEEE International Conference on High Performance Computing and Communications*, pp. 94-100. IEEE, 2011.
3. Rawal, Bharat S., Ramesh K. Karne, Alexander L. Wijesinha, Harold Ramcharan, and Songjie Liang. "A split protocol technique for web server migration." In 2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012), pp. 1-6. IEEE, 2012.
4. Rawal, Bharat S., Ramesh K. Karne, and Alexander L. Wijesinha. "Splitting HTTP requests on two servers." In *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pp. 1-8. IEEE, 2011.
5. Rawal, Bharat S. "Attack countermeasure tree (ACT) meets with the split-protocol." *International Journal of Computer Networks & Communications* 7 (2015): 99-113.
6. W. Ren, Y. Sun, H. Luo, and M. Guizani, "A Novel Control Plane Optimization Strategy for Important Nodes in SDN-IoT Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3558–3571, 2019.
7. T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and Revised Content-Centric Networking-Based Service Deployment and Discovery Platform in Mobile Edge Computing for IoT Devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4162–4175, 2019.
8. F. Bannour, S. Souihi, and A. Mellouk, "Adaptive distributed SDN controllers: Application to Content-Centric Delivery Networks," *Future Generation Computer Systems*, vol. 113, pp. 78–93, 2020.
9. S. Math, P. Tam, A. Lee, and S. Kim, "A NB-IoT data transmission scheme based on dynamic resource sharing of MEC for effective convergence computing," *Personal and Ubiquitous Computing*, 2020.
10. A. A. Ateya, A. Muthanna, A. Vybornova, A. D. Algarni, A. Abuarqoub, Y. Koucheryavy, and A. Koucheryavy, "Chaotic salp swarm algorithm for SDN multi-controller networks," *Engineering Science and Technology, an International Journal*, vol. 22, no. 4, pp. 1001–1012, 2019.
11. Noman, Haeeder Munther, and Mahdi Nsaif Jasim. "POX Controller and Open Flow Performance Evaluation in Software Defined Networks (SDN) Using Mininet Emulator." In *IOP Conference Series: Materials Science and Engineering*, vol. 881, no. 1, p. 012102. IOP Publishing, 2020.
12. Li-Jun, M. A. O., and Yan ZHANG. "Research on Dynamic Migration Technology of OpenFlow Switch." *DEStech Transactions on Social Science, Education and Human Science* icesd (2020).
13. Al-Heety, Othman S., Zahriladha Zakaria, Mahamod Ismail, Mohammed Mudhafar Shakir, Sameer Alani, and Hussein Alsariera. "A Comprehensive Survey: Benefits, Services, Recent Works, Challenges, Security, and Use Cases for SDN-VANET." IEEE Access 8 (2020): 91028-91047.
14. Arif, Muhammad, Guojun Wang, Oana Geman, Valentina Emilia Balas, Peng Tao, Adrian Brezulianu, and Jianer Chen. "SDN-based VANETs, Security Attacks, Applications, and Challenges." Applied Sciences 10, no. 9 (2020): 3217.
15. Iqbal, Waseem, Haider Abbas, Mahmoud Daneshmand, Bilal Rauf, and Yawar Abbas. "An In-Depth Analysis of IoT Security Requirements, Challenges, and their Countermeasures via Software Defined Security." IEEE Internet of Things Journal (2020).
16. Sharma, P. K., Singh, S., & Park, J. H. (2018). OpCloudSec: Open cloud software defined wireless network security for the Internet of Things. Computer Communications, 122, 1-8.
17. H. Xue, S. Chen, and Q. Yang, "Structural Regularized Support Vector Machine: A Framework for Structural Large Margin Classifier," IEEE Transactions on Neural Networks, vol. 22, pp. 573-587, April, 2011. Article (CrossRef Link).
18. Motiwala, Murtaza, Megan Elmore, Nick Feamster, and Santosh Vempala. "Path splicing." In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pp. 27-38. 2008.
19. Hiremagalur, D., and G. Grammel. "Internet Engineering Task Force G. Galimberti, Ed. Internet-Draft Cisco Intended status: Experimental R. Kunze Expires: December 27, 2018 Deutsche Telekom."
20. Miguel, Constantino, Francisco Neto, and Paulo Sampaio. "The CAARF approach towards Monitoring and analysis of Contextual data within SDN networks." In 8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020), pp. 1-8. 2020.
21. Santos, Omar. CCNP and CCIE Security Core SCOR 350-701 Official Cert Guide: Implementing and Operating Cisco Security Core Technologies. Cisco Press, 2020.