Solutions for Adopting Software Defined Network in Practice

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Solutions for Adopting Software Defined Network in Practice

Nadir Shah*[1] | Paolo Giaccone[2] | Danda B. Rawat[3] | Ammar Rayes[4] | Nan Zhao[5]

[1]Department of Computer Science, COMSATS University of Islamabad, Wah Campus, Punjab, Pakistan
[2]Department of Electronics and Telecommunications, Politecnico di Torino, Italy
[3]Department of Electrical Engineering & Computer Science, Howard University, Washington, DC, USA
[4] Cisco Systems , USA
[5]School of Information and Communication Engineering, Dalian University of Technology, China

Correspondence
*Corresponding Nadir Shah, This is corresponding address. Email: nadirshah82@gmail.com

Summary

Software Defined Networking (SDN) proposed by academia has many advantages over traditional networks. However, SDN has not been widely adopted in practice by the organizations due to several reasons, addressed by this special issue. The articles collected in this special issue provide a broad view, since covering the data, control, and management planes. Moreover, we explain the future research directions that could need the immediate attention of the research community to address in order to enable the practical adoption of SDN.

KEYWORDS:
Software Defined Networking (SDN), SDN in practice, Hybrid SDN, Certification Programs

1 | INTRODUCTION

1.1 | The basics of Software Defined Networking (SDN)

Software Defined Networking (SDN) has emerged as a new network paradigm, which decouples both control and management planes from data plane by implementing control and management at a (logically) centralized node, i.e. the controller. The data plane communicates with controller by using an open programming interface (like Openflow protocol[1]). The reference SDN architecture is given in Figure 1 and explained as follows.

Data plane: The data plane consists of standard switching devices, acting as routers, switches and access points depending on the how they are programmed. A forwarding table is available in each switching device. When a data packet arrives at the switching device, the latter looks up its forwarding table whether an entry exists for the corresponding flow. If yes, then the data packet is forwarded/dropped as per action specified in the entry. Otherwise, the switching device buffers the data packet and sends a message to the controller to ask for instructions on how to process the packet. The controller sends back a forwarding rule to install in the forwarding table, which programs the action that the switching device must apply for the buffered data packet, and eventually for the future packets belonging to the same flow.

Control plane: The controller acts as the "network brain" and is responsible for computing the forwarding behavior for the data packets based on the network topology, the network requirements (e.g., specified throughout access control list (ACL) policies), and the configuration of other network management functions (e.g., load-balancing). The controller computes the forwarding decision and installs the corresponding rules in the forwarding tables present in the switching devices.

Management plane: The management plane specifies the network applications (like load-balancing, firewall, monitoring, etc.) developed by the network operator.

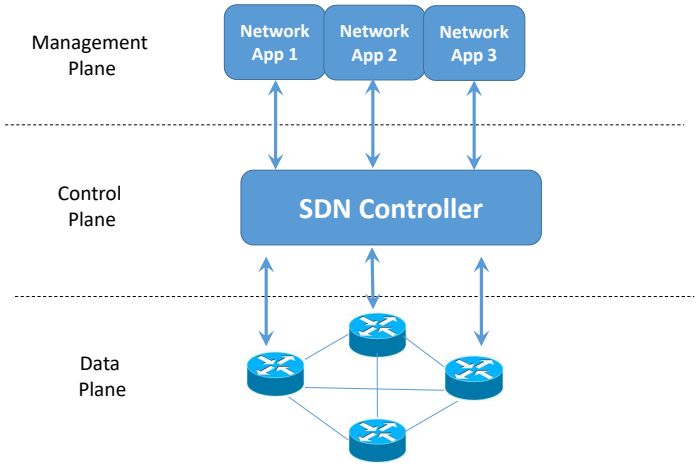

FIGURE 1 Reference SDN Architecture

Plane interfaces: These planes in SDN interact with each other through standardized interfaces. The basic interfaces are (i) the southbound interface, used for communication between data plane and control plane, and (ii) the northbound interface, used for communication between control plane and management plane.

The SDN architecture has several advantages over the traditional networks. First, it centralizes the control in the SDN controller allowing a single and coherent view of the network state, simplifying the development of complex network applications, and enabling the network administrator to push specific policies across the entire network. Second, the SDN architecture allows to evolve the control plane without changing the switching devices, since their actual behavior is programmed at the controller and can be reprogrammed at wish. Third, in traditional networks, the network administrator has to utilize management systems and/or manually define and implement security policies (e.g., ACL) at several switching devices. Instead, in SDN security and/or traffic engineering policies can be defined just at the controller and these will be forced in the whole network.

1.2 | Adopting SDN in Practice

SDN has many advantages over traditional networking and several approaches have been proposed in the research community to leverage them [2]. However, SDN is not widely adopted by organizations due to the several reasons:

* **Budget constraints:** Organizations are often reluctant to invest too much budget at a time in a new technology which requires to renew the equipment.
* **Performance and reliability:** The SDN centralized approach poses a high computation load on the server hosting the controller, which becomes a possible single point of failure. To cope with that, a cluster of controllers are adopted to provide backup controllers and to distribute the processing workload. Nevertheless, performance and reliability are perceived lower than a traditional distributed control plane.
* **Scalability:** The centralized nature of the SDN approach (even if at logical level) poses many concerns about its applicability in large networks, where large geographical distances between the SDN nodes affect negatively the reactivity to network events observed by the end-users' applications.
* **Level of maturity:** The SDN paradigm has emerged recently and is perceived as not yet mature, differently from legacy networking technologies.
* **Lack of experts and certification programs:** Due to the novel approach, it is still difficult to identify and certify the correct level of skills required for the network designers and administrators to operate on a SDN network. Only recently Open Networking Foundation has been offering a certified SDN Professional Program, mimicking well-established certification programs in networking [3].

The possible options to cope with the above issues depend on the level of SDN adoption. In the case of total SDN deployment, the research community should focus the research on the core functionalities of SDN, like network connectivity, security, and management. The research community should also demonstrate the maturity of their proposals by providing evidence through experimental results and proof-of-concepts. In the case of incremental SDN deployment, network operators can incrementally deploy SDN in their organization avoiding to replace traditional (legacy) equipment at once. One feasible approach, denoted as "hybrid SDN", is to deploy a limited number of SDN devices along with traditional devices. This solution eases the transition from legacy to SDN devices since a network operator can incrementally deploy more and more SDN devices and evaluate the impact in practice. For instance, Google has adopted SDN in multiple stages over several years for their network management and control system[4]. Similarly, other large, as well as small and medium size organizations, may want to adopt SDN technology gradually. Several challenges must be addressed for adopting SDN in practice, to guarantee synergic operations among legacy switches, SDN switches and controllers. These issues can be related to the different planes:

Data plane: Several research challenges related to the data plane arise when the SDN network is to be adopted in real life. For example, to incrementally deploy SDN and to exploit traffic engineering efficiently[5], the SDN switches should be deployed among the legacy switches in such a way that the maximum number of flows traverse SDN switches[6]. Similarly, it is possible to install a hardware SDN shim[7] in legacy switches such that the legacy switches can communicate with both legacy switches and the SDN switches/controller by running both traditional network protocols and SDN-specific protocols. Similarly, to replace the whole legacy network by a SDN-based one, the limited Ternary Addressable Memory (TCAM) memory in SDN switches can degrade the performance as follows. The paper[8] states that indeed in a data center network the flow arrival rate can be so large that the flow table overflows and, subsequently, some flow entries are replaced by other flow entries. If the data packet arrives at the switch corresponding to the replaced flow entries, then the switch will interact again with the controller to receive the corresponding flow entries and this will increase the end-to-end latency. Therefore, it is important to devise scheme to exploit the TCAM memory efficiently.

Control plane: In hybrid SDN, the centralized control plane at the SDN switches synergically cooperate with the distributed control plane running at each legacy switch. To provide a consistent control mechanism, several design alternatives are possible, as follows. First, as in majority of existing approaches of hybrid SDN, the SDN controller controls the forwarding behaviour of the legacy switches through SDN switches. For example, Telekinesis[9] configures the legacy switches to forward the traffic via SDN switches so that a data packet arriving at the first legacy switch is forwarded to the nearest SDN switch, subsequently the packet is forwarded using SDN-based protocols. Second, a specialized controller can be deployed to control both SDN switches and legacy switches. For example, ClosedFlow controller[10] can control legacy switches by exploring the configuration commands of legacy switches in similar manner as the SDN controller controls SDN switches. Similarly, if the SDN is adopted as whole in the network, then SDN controller is more vulnerable to security attacks. Therefore, new mechanisms are required to make the controller more scalable, resilient and secure.

Management plane: In SDN, the management plane encompasses several types of applications running at the controller, which configure and operate the network. These requirements are specified at higher abstract level, like Frenetic language[11]. However, legacy switches/routers are configured by using low level commands, like ACL rules. In hybrid SDN, managing the legacy switches/routers require new mechanisms and protocols. For example, to enforce ACL rules at the legacy switches, Amin et al.[12] suggest a decision tree mechanism to find the accurate placement for ACL rules. The model proposed in[8] implements SDN-like policies using the OpenFlow protocol at legacy switches in the hybrid SDN as follows. Each subnet of legacy switches is connected to at least one SDN switch so that the SDN switch monitors and controls its subnet. This is achieved technically by exploiting unused IP addresses to implement SDN-like policies at legacy switches, since they process the packets only based on IP addresses.

Plane interfaces: Besides northbound and southbound interfaces, hybrid SDN define other interfaces used for communication between SDN switches and legacy switches, and between legacy switches and the SDN controller. One of the challenges is to standardize such interfaces in a flexible way to support a diverse set of functionalities.

2 | SUMMARY OF RESEARCH PAPERS ACCEPTED IN THIS SPECIAL ISSUE

This Special Issue (SI) contains the papers presenting state-of-the-art research studies pertaining to the solutions for adopting SDN in practice. We received 30 total number of papers from the authors belonging to different countries including USA, Brazil,

China, Spain, Italy, and India. The number of papers eventually accepted has been 6, with an acceptance ratio of 20%. We are summarizing the contribution of each accepted paper in the following section.

2.1 | Data plane

Yeh et al. in "SEAL2: A SDN-Enabled All-Layer2 Packet Forwarding Network Architecture for Multi-Tenant Data Center Networks" state that existing approaches use IP address for both host's identification and its location, causing problems for mobile hosts. In a multi-tenant data center network, it is inevitable that virtual machines move/migrate. To address this issue, the authors suggest a new approach that separates virtual machine (VM) identification (ID) and its location information by using two values independent from the IP addresses. Moreover, they redefine the semantics of Ethernet address by using MAC address to as VM ID information. Further, they redesign the MAC/IP two-stage routing schemes by MAC routing. This combines MAC addresses (for VM identification and forwarding in local server groups under an edge switch gateway) and Multiprotocol Label Switching (MPLS) labels (for packet transportation between edge switch gateways across the core label switching network connecting all the edge gateways). The proposed approach is evaluated by comparing with existing approach in term of packet processing overhead incurred in packet forwarding/routing components.

2.2 | Control plane

Wang et al. in "SDN-Based Dynamic Multipath Forwarding for Inter-Datacenter Networking" explain that Equal-Cost Multipath (ECMP) is used in intra-data center (DC) networks to increase the efficiency of network transmission for east-west traffic (i.e., exchanged between servers within the data center). They further point out that, using ECMP for the data transmission in inter-DC wide area network (WAN) may decrease the efficiency of data transmission, due to the typical irregular topologies in inter-DC WANs. Furthermore, ECMP maintains multiple paths, causing scalability issues due to the limited size of TCAM memories in the SDN switches. The authors suggest a new mechanism, called Dynamic Flowentry-Saving Multipath (DFSM), for inter-DC traffic forwarding. DFSM uses source-and-destination IP addresses based multi-path forwarding and latency-aware traffic splitting at a switch in order to decrease the number of flow entries and to achieve load balancing. The evaluation results show that DFSM reduces the system flow entries to 15-30% in realistic topologies and reduces the standard deviation of path latencies from 10% to 7% compared to label-switched tunneling, and also decreases average latency by 10-48% by consuming 6-20% more flow entries than ECMP in less interconnected topologies.

2.3 | Management plane

Nehra et al. in "TILAK: A token-based prevention approach for topology discovery threats in SDN" state that SDN controller maintains the network view (more specifically the network topology) and the accuracy of such information is vital for the correct and efficient operation of the SDN. They elaborate the working of mechanisms used in different available controllers (like POX, Ryu, OpenDaylight, FloodLight, Beacon, ONOS, and HPEVAN) in order to get and compute the network view. Then, they identify the possible security vulnerabilities in these mechanisms that can subsequently create incorrect network topology at the controller. More specifically, they analyze three types of security vulnerabilities, namely link layer discovery protocol (LLDP) poisoning, LLDP flooding, and LLDP replay attack. After this, they suggest a new mechanism, named as TILAK, that generates random MAC destination addresses for LLDP packets and inserts the flow entries at the flow tables based on these MAC addresses for the LLDP packets. Finally, they show that TILAK outperforms the existing mechanisms.

Dayal et al. in "FloodKnight: An Intelligent DDoS Defense Scheme to Combat Attacks Near Attack Entry Points" suggest a new mechanism, called FloodKnight, to detect distributed denial of service (DDoS) attacks by analyzing at the controller the data packets patterns generated by the hosts. For this purpose, FloodKnight uses Radial Basis Function network (RBF) having Particle Swarm Optimization (PSO) algorithm to differentiate between DDoS attacks and the legitimate data traffic. Moreover, FloodKnight computes the DDoS attack entry points in the network by using a port-based source trace-back scheme, to prevent the attack as early as possible. Through experimental results, the authors show that their proposed mechanism can efficiently detect DDoS attacks earlier and contrast these attacks at entry points in the network, as compared to existing approaches.

Gharsallah et al. in "SDN/NFV-based handover management approach for ultradense 5G mobile networks" propose a novel approach, called Software-Defined Handover Management Engine (SDHME), to improve the handover process in term of handover failure rate and handover latency in future ultra-dense 5G networks. SDHME assumes that each mobile node periodically

sends to SDN controller through point-of-attachment (PoA) the information of Radio Signal Strength Indicator (RSSI), Signal-to-Interference-plus-Noise Ratio (SINR) and Packet Error Rate (PER). The controller transfers such information together with the network conditions (like bandwidth, latency, packet loss rate, mobile node moving speed, direction and location) to SDHME running at the management layer. SDHME takes the handover decision for a mobile node based on this information and on the QoS requirements of the application. Through simulation results, the authors show that SDHME decreases the handover failure rate and handover latency as compared to the handover approach used in conventional LTE networks.

2.4 | Northbound interface

Castellano et al. in "A Model-Based Abstraction Layer for Heterogeneous SDN Applications" attempt to provide a standardized interface for northbound interface that enables a service application running at management plane to collect the status information from SDN applications running at control plane and subsequently to control the SDN applications by configuring the SDN application accordingly. The importance of these two functionalities is shown by describing the use cases of intrusion prevention system, migration services, and service orchestrators. The proposed mechanism is based on YANG model language. The proposed mechanism can handle all existing SDN applications without modifying their original code and without incurring additional overhead. Moreover, the proposed mechanism has the advantage to perform cross-platform inspection. Finally, the authors present an implementation to show the effectiveness of their proposed approach.

3 | FUTURE RESEARCH DIRECTIONS

Even if the papers in this SI have significantly contributed to make the smooth adoption of SDN in real life, however, there are still several open research challenges that need to be addressed. In this section, we describe these research challenges related to each layer of SDN. We expect that the research community will pursue these challenges by proposing new mechanisms for making it easy to adopt SDN in practice.

3.1 | Data plane

i. Amin et al.[12] explain the mechanism that enables an SDN controller to change the ACL configuration at the legacy switches as the network topology changes. However, the ACL configuration requires to solve an optimization problem, as explained in[13]. Even in pure SDN, the configurations (like ACL policies) are defined at controller. The flow rules for these ACL policies should be installed proactively because their behavior is already specified. This will reduce the end-to-end delays for these flows and will also offload the traffic at SDN controller. Installing proactively the flow rules for implementing these ACL policies in pure SDN requires also to solve an optimization problem. To the best of our knowledge, this aspect has not been explored in the current literature. Therefore, the placement of flow rules for ACL policies should be devised in such a way that minimizes the total number of redundant transmissions, the total number of flow rules installed in the whole network, and the maximum number of flow entries at a SDN switch.

ii. The SDN architecture advocates the concept that data plane (SDN switches) should be simple and all the intelligence of the network is implemented at SDN controller. This makes controllers more complex and eventually overloaded. In order to offload the controller from managing some events, one has to add some intelligence at data plane. For example, multiple paths are installed at the switch for the flow[14] so that if one path at the switch gets failed then the switch reroutes the data over another path. By installing multiple paths at a switch, the controller can overflow the small TCAM-based flow tables at the switch. To avoid this problem, one approach could be to use DHT-based routing[15], according to which the routing and forwarding is performed by using logical identifier (LID) instead of IP address[16]. The benefits of DHT-based routing is that the LID of a node provides multiple paths without computing and installing multiple paths[16].

iii. Information centric networking (ICN) performs routing and forwarding based on the packet content instead of the destination IP address, as in standard Internet. ICN also enables a router to cache the content passing through so that subsequent requests for the same content can be handled by the router without going to origin content provider. Thus, ICN provides in-network caching capabilities. ICN also makes the routers more complex to enable content based routing, forwarding, content caching, content lookup, content cache replacement decision, and other ICN functionalities[17]. However, the

SDN switches are designed to be simple. Therefore, to support the ICN functionalities, the SDN switches should be designed such that these ICN functionalities are efficiently implemented by considering the SDN switch limited capabilities (like small TCAM memory). Moreover, southbound protocols must be designed so that SDN switches can support ICN functionalities in efficient and correct manner.

iv. In a stateful data plane, the switches are storing some internal states and can take local decisions based on programmable finite state machines. This approach is different from the canonical SDN architecture in which switches are stateless and all the states are kept in the controller. The big advantage of adopting stateful switches is that the switch can directly react to network events (e.g., arrival of a new flow, link failures) and apply a pre-programmed policy, without the intervention of the controller. Thus, the time required to react to network events is very short and is independent from the delays to reach the controller, with a beneficial effect on the network performance, especially when the network is large. In general, the stateful approach mimics a traditional distributed data plane, but with the crucial difference that the internal behavior of the switch can be completely programmed (and also re-programmed on-the-fly). Notably, new technologies are enabling the approach, as P4 [18,19] or Open Packet processor [20], which are currently available on dedicated chipsets [21] or network FPGA [20].

3.2 | Control plane

i. The existing controllers are running on general purpose processors. Nowadays, network processors are available that are designed specifically for traditional network protocols. The researchers should also design and implement specialized processors (i.e., more generally a computer system architecture) for SDN controllers. This will reduce the processing delay and subsequently will reduce the end-to-end delay experienced by the data packets, thus by the applications.

ii. Many existing controllers are designed to control the SDN switches using SDN-tailored protocols (as Openflow). However, as described above, a hybrid SDN is a viable solution at the moment to adopt SDN in practice in most of the organizations. Therefore, the SDN controllers should be designed such that the controllers control not only the SDN switches but also the legacy switches.

iii. To provide scalability and resilience to failures in SDN, a multi-controller architecture is needed that deploys multiple controllers in the network. The existing approaches (a survey can be found in [22]) place the multiple controllers in the network according to different criteria, e.g., minimizing the maximum distance between switches and controllers, or minimizing the inter-controller delays, or the latency to synchronize the shared data structures among the controllers [23]. Typically, the controller placement problem is NP-hard and many approximation solutions can be computed tailored for the specific considered optimization criteria. For example, traffic matrices in ISP networks are typically approximately low-rank [5], thus, most of the flows traverse only a few switches (denoted as "important" switches). Now the controller placement can be computed by considering only important switches instead of all switches. This will reduce the computation time of the algorithm used for controller placement problem.

iv. Segment Routing (SR) is a novel IETF protocol to allow the integration of SDN with legacy networks [24,25]. The main idea is to program the network routing behavior through source routing, i.e. the first node in the network is programmed to label the traffic with the precise sequence of nodes to route the packet. Each SR-enabled node processes the label present in the packet and routes it to the specified next-hop router. Thus, the traffic is routed according to a programmed sequence of segments connecting SR-enabled nodes, and the routing along one segment occurs through the standard IPv4/IPv6 routing. This hybrid approach guarantees a natural coexistence with legacy network devices and the level of network programmability depends on the pervasiveness of the SR-enabled devices. Notably, segment routing allows to label packets in a very flexible way, thus traffic can carry in piggybacking also "programming code" to run at the SR-enabled nodes, allowing a very wide spectrum of network applications. In summary, SR can be considered an alternative approach to the canonical SDN architecture to support network programmability.

3.3 | Management plane

i. Programming and abstraction languages: Several programming languages have been proposed for pure SDN in order to express the network configuration and policies at abstract level, a survey of these languages is given in [26]. However, these

programing languages still lack certain features. For example, there are several alternatives, like[27], to cope with the link and node failures. Each of these approaches can perform differently in different scenarios. The existing programming languages should provide the high level abstraction so that a network operator can specify to use a particular approach for link and node failures depending upon the current scenario. Similarly, these programming languages are proposed for pure SDN. However, the researchers should suggest new programming languages that can configure both SDN switches/routers and legacy switches/routers, in order to simplify the adoption of the hybrid SDN architecture.

ii. Intent-based approach: Through SDN, network policies can be expressed in terms of high-level, declarative "intents", which specify which services the network must provide without describing how they are provided. Thus, intents are defined completely obliviously of the network details and state, allowing the network operator concentrating on what the network should do (seen as black box) and not on how it is implemented.
An example, an intent could be "to provide a communication with given QoS requirements between a given set of nodes". The role of the controller is to compile such abstract intents and generate the specific forwarding rules to program all the switches needed to implement the intent. Notably, this approach is very amenable for network operators, who could just concentrate their efforts in defining new profitable services in terms of network intents, independently from the below network. The key challenge in this context is to define the correct level of abstraction to allow a simple network management from the network operator point of view and a feasible implementation from the control plane point of view.

3.4 | Plane interfaces

The existing basic interfaces (like northbound and southbound) need to be open source and flexible by supporting different functionalities. Moreover, these interfaces should enable a network operator and application developer to specify and configure the network in high abstract level. This will enable an easier network debug. Further, beside these basic interfaces, the interfaces that connect a legacy switch with SDN switch, and a SDN controller with legacy switches, should be standardized.

ACKNOWLEDGMENTS

We are extremely thankful to the authors that submitted their papers to our SI. We indebted to all the anonymous reviewers who provided the meaningful, thorough and meticulous review for the papers in a timely manner. We are thankful to Mohammad S. Obaidat Editor-in-Chief of the International Journal of Communication System, Wiley, for his guidance and support. We are grateful to Abhijeet Das (the present Editorial Office Assistant), and Anne Margaritte Pertible and Sheen Lee Locre (the former Editorial Office Assistant) for their assistance and cooperation during the review process.

References

1. Openflow Version1.5.1. https://www.opennetworking.org/software-defined-standards/specifications/OpenflowVersion1.5.1; . Accessed on Nov. 2018.

2. Kreutz D, Ramos FM, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE* 2015; 103(1): 14–76.

3. OpenNetworking Training. https://www.opennetworking.org/training-certification/skills/; . Accessed on Nov. 2018.

4. Vahdat A, Clark D, Rexford J. A Purpose-built Global Network: Google's Move to SDN. *ACM Queue* 2015; 13(8).

5. Cheng TY, Jia X. Compressive traffic monitoring in hybrid SDN. *IEEE Journal on Selected Areas in Communications* 2018.

6. Levin D, Canini M, Schmid S, Schaffert F, Feldmann A. Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks. In: USENIX Association. ; 2014: 333–345.

7. Casey DJ, Mullins BE. SDN shim: Controlling legacy devices. In: IEEE. ; 2015: 169–172.

8. Mishra A, Bansod D, Haribabu K. A Framework for OpenFlow-like Policy-based Routing in Hybrid Software Defined Networks. In: ; 2016: 97–102.

9. Jin C, Lumezanu C, Xu Q, Zhang ZL, Jiang G. Telekinesis: Controlling legacy switch routing with Openflow in hybrid networks. In: ACM. ; 2015: 20.

10. Hand R, Keller E. Closedflow: Openflow-like control over proprietary devices. In: ACM. ; 2014: 7–12.

11. Foster N, Harrison R, Freedman MJ, et al. Frenetic: A network programming language. *ACM Sigplan Notices* 2011; 46(9): 279–291.

12. Amin R, Shah N, Shah B, Alfandi O. Auto-configuration of ACL policy in case of topology change in hybrid SDN. *IEEE Access* 2016; 4: 9437–9450.

13. Sung YWE, Sun X, Rao SG, Xie GG, Maltz DA. Towards systematic design of enterprise networks. *IEEE/ACM Transactions on Networking (TON)* 2011; 19(3): 695–708.

14. Thorat P, Raza SM, Kim DS, Choo H. Rapid recovery from link failures in software-defined networks. *Journal of Communications and Networks* 2017; 19(6): 648–665.

15. Dumba B, Mekky H, Jain S, Sun G, Zhang ZL. A virtual ID routing protocol for future dynamics networks and its implementation using the SDN paradigm. *Journal of Network and Systems Management* 2016; 24(3): 578–606.

16. Abid S, Othman M, Shah N, Ali M, Khan A. 3D-RP: A DHT-Based Routing Protocol for MANETs. *The Computer Journal* 2015; 58(2): 258–279.

17. Zhang QY, Wang XW, Huang M, Li KQ, Das SK. Software defined networking meets Information Centric Networking: a survey. *IEEE Access* 2018; 6: 39547–39563.

18. P4. https://p4.org/; . Accessed on Nov. 2018.

19. Barefoot P4 Studio. https://www.barefootnetworks.com/; . Accessed on Nov. 2018.

20. Bosshart P, Gibb G, Kim HS, et al. Forwarding Metamorphosis: Fast programmable match-action processing in hardware for SDN. In: ACM. ; 2013.

21. Bonola M, Bifulco R, Petrucci L, Pontarelli S, Tulumello A, Bianchi G. Implementing advanced network functions for datacenters with stateful programmable data planes. In: ; 2017.

22. Zhang Y, Cui L, Wang W, Zhang Y. A survey on software defined networking with multiple controllers. *Journal of Network and Computer Applications* 2018; 103(C): 101–118.

23. Zhang T, Giaccone P, Bianco A, Domenico SD. The role of the inter-controller consensus in the placement of distributed SDN controllers. *Computer Communications* 2017; 113: 1 - 13.

24. Filsfils C, Previdi S, Ginsberg L, Decraene B, Litkowski S, Shakir R. Segment Routing Architecture. https://tools.ietf.org/html/rfc8402/; 2018. Accessed on Nov. 2018.

25. Segment Routing. http://www.segment-routing.net; . Accessed on Nov. 2018.

26. Trois C, Del Fabro MD, Bona dLC, Martinello M. A survey on SDN programming languages: Toward a taxonomy. *IEEE Communications Surveys & Tutorials* 2016; 18(4): 2687–2712.

27. Thorat P, Jeon S, Raza SM, Choo H. Pre-provisioning of local protection for handling dual-failures in OpenFlow-based networks. In: IEEE. ; 2017: 1–6.