Journal of Network and Systems Management (2021) 29:9
https://doi.org/10.1007/s10922-020-09575-4

Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers

Suhail Ahmad[1] · **Ajaz Hussain Mir**[1]

Received: 17 November 2019 / Revised: 17 September 2020 / Accepted: 19 October 2020 /
Published online: 5 November 2020

Abstract

Software Defined Networking simplifies design, monitoring and management of next generation networks by segregating a legacy network into a centralized control plane and a remotely programmable data plane. The intelligent centralized SDN control plane controls behavior of forwarding devices in processing the incoming packets and provides a bird-eye view of entire network at a single central point. The centralized control provides network programmability and facilitates introduction of adaptive and automatic network control. The SDN control plane can be implemented by using following three deployment models: (i) physically centralized, in which a single SDN controller is configured for a network; (ii) physically distributed but logically centralized, wherein multiple SDN controllers are used to manage a network; and (iii) hybrid, in which both legacy distributed control and centralized SDN control coexist. This manuscript presents all these control plane architectures and discusses various SDN controllers supporting these architectures. We have analyzed more than forty SDN controllers in terms of following performance parameters: scalability, reliability, consistency and security. We have examined the mechanisms used by various SDN controllers to address the said performance parameters and have highlighted the pros and cons associated with each mechanism. In addition to it, this manuscript also highlights number of research challenges and open issues in different SDN control plane architectures.

Keywords SDN · SDN control plane · Centralized SDN control plane · Multiple SDN controllers · Hybrid SDN control plane · OpenFlow

✉ Suhail Ahmad
suhail.sam008@gmail.com

Ajaz Hussain Mir
ahmir@rediffmail.com

[1] Electronics & Communication Engineering Department, National Institute of Technology Srinagar, Srinagar, J & K 190006, India

1 Introduction

The ever-increasing demand of online services like cloud computing [1], big data applications [2] and automated networking platform for IOT [3] have stretched traditional networks to breaking points. With the unprecedented growth of such online services, the network industry is compelled to change its conventional architecture. Open Networking Foundation (ONF) [4] proposed Software Defined Networks (SDN) which supports network programmability and automation of network operations. This networking paradigm fosters innovations by separating the data and control plane, removing hurdles for advances in each plane. SDN [5] is a radical approach for next generation networks which provides bird-eye view of entire network at a centralized controller and promotes use of open and programmable forwarding devices. The multilayered architecture of SDN comprising of forwarding, control and management plane is shown in Fig. 1.

The bottom layer is termed as forwarding plane or data plane. It consists of distributed forwarding devices usually switches that forward packets as per the flow rules communicated by the remote controller. The open vendor-agnostic interface between control and data plane is termed as southbound interface. This interface is used by the controller to communicate flow rules and to retrieve flow statistics information from the data plane devices. The most widely used southbound interface is OpenFlow [6, 7].

The middle layer or control plane consists of software-based SDN controller(s) which controls and manages the underlying data plane devices and defines traffic flows as per the network policy. The control plane can be implemented as a single physically centralized controller or distributed but logically centralized controllers or an amalgam of centralized SDN control and legacy distributed control. The controllers use east and westbound interfaces to exchange inter-domain network information, as shown in Fig. 1. As per the authors in [8], the eastbound interface is used

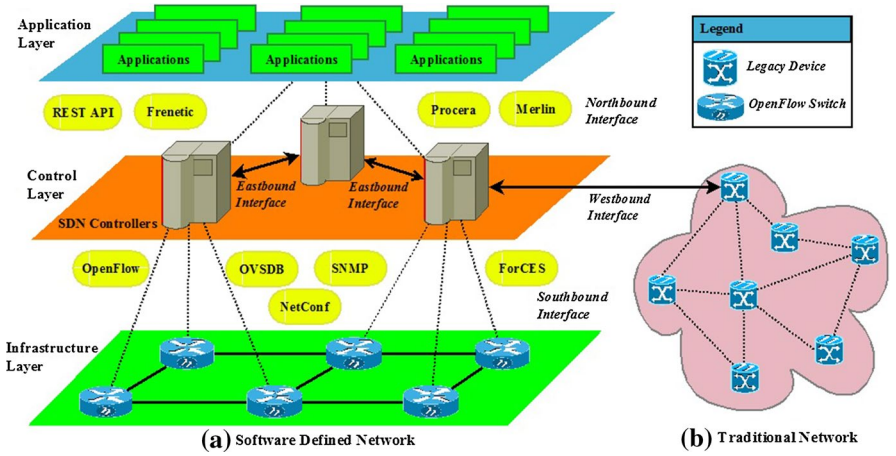

Fig. 1 SDN architecture and SDN interfaces

between two SDN controllers whereas the westbound interface is used to exchange information between SDN control plane and legacy distributed control. Since the main focus of this manuscript is on SDN control plane, a detailed account on it is provided in the next section.

The top layer or application layer is also termed as management plane. It comprises of various SDN applications which are designed to implement specific control and management strategies. Traffic engineering (TE), load balancing, firewalls, etc., are some common SDN applications. These control applications use open northbound interface to interact with the SDN control plane. The northbound interface can be compared to win32 or POSIX standard of operating systems, providing abstractions that guarantee programming language and controller platform independence. Keeping in view the importance of this interface, the ONF formulated a working group namely Open Networking Foundation North Bound Interface Working Group (NBI-WG) [9] for standardization of this API.

SDN is an agile, programmable and centrally controlled architecture which supports vendor-agnostic open devices. To meet changing traffic demands in next generation networks, the logically centralized controller in SDN enables network administrators to dynamically regulate network-wide traffic flows. With the help of centralized SDN control, dynamic network topologies can be defined in data centers and policy based routing can be implemented in service providers or enterprise networks. Numerous mechanisms have been used in various controllers to provide a scalable, fault-tolerant, consistent and secure control platform. The main aim of this paper is to perform an analysis of such mechanisms with emphasis on prospective trends which may drive further research in SDN control plane.

1.1 Contributions and Related Work

The main aim of this manuscript is to survey the industrial/academic projects and publications related to SDN controllers implemented and published over the last one decade. We have presented a systematic discussion of various prominent SDN controllers and have analyzed such controllers in terms of scalability, reliability, consistency and security. To the best of our knowledge, this is the first manuscript which considers four performance parameters of SDN controllers and analyzes a wide range of SDN controllers. The main contributions of this survey are as follows:

- Comprehensive background of SDN control plane: We present a detailed account on SDN control plane, SDN control plane architectures and classification of various SDN controllers based on their architecture.
- Research challenges in SDN control plane architectures: The research challenges associated with different SDN control plane architectures are identified and emphasized.
- Evaluation of SDN Controllers: Most prominent SDN controllers are evaluated in terms of following four performance parameters: scalability, consistency, reliability and security.

- Classification of Hybrid SDN models: We present the classification of hybrid SDN models and map various hybrid SDN controllers to these models.
- Future research directions: We specify potential future research areas in SDN control plane along with recommendations on probable solutions.

Covering every aspect of SDN in a single survey is a difficult task as it is an intricate field. There have been numerous surveys [10–18] addressing different aspects of SDN paradigm including SDN's historical perspective [11, 12], SDN architecture, design challenges and applications [13–18], programming languages for SDN [19], fault management in SDN [8, 20], traffic engineering with SDN [21, 22], security issues in SDN [23–26] and SDN applicability in diverse domains [27–32]. However, SDN control plane being a vital component in SDN architecture has been discussed as a section in some papers [10] and some surveys [33–36] have limited discussion to specific controllers and have considered only few performance parameters.

In [33], the authors have addressed only scalability issue of SDN controllers. They have discussed the contributors for scalability issues in SDN architecture including control and data plane separation, request to a single centralized controller and switch-controller communication delay. Further, the authors in [33] have classified the control plane scalability approaches into two broad categories: topology based and mechanism based approaches. On the other hand, authors in [34–36] have limited the discussion to distributed SDN control plane. In [34], the authors have highlighted the scalability and consistency challenges in distributed SDN controllers, whereas authors in [35] have highlighted differences between multi-controller architectures and have discussed the communication mechanisms and state distribution methods used in some prominent multi-controllers. Hu et al. in [36] have elaborated the mechanisms proposed by researchers to handle controller placement, domain partition, state consistency, strategy consistency, path reliability, node reliability and load balancing in multi-controller architectures. The authors have elaborated the strategies and mechanisms proposed by researchers to address scalability, consistency and load balancing in multi-controller architecture.

Even though Hybrid SDN controllers provide transitional approach to introduce programmability or OpenFlow devices in traditional networks but have been least analyzed by the research community. Authors in [37] have explained five hybrid SDN models and have described their transitional and long-term design use cases. They have provided a tradeoff analysis of these hybrid models interms of robustness, scalability, deployment cost, flexibility and complexity. On the other hand, authors in [38] have discussed the various hybrid SDN models suitable for transition of conventional networks to SDN. They have compared these models on the basis of traffic management, automation, investment and scalability.

Despite these surveys, none of the paper has discussed in detail the possible SDN control plane implementations, classification of controllers based on the architecture, and mechanisms used by various controllers to address the following important performance parameters: scalability, reliability, consistency and security. In this paper, we have analyzed more than forty SDN controllers in terms of said performance parameters. The controllers discussed in this manuscript along with the year

of inception are shown in Fig. 2. Figure 2 also depicts the research challenges associated with different control plane architectures.

1.2 Paper Outline

The manuscript is organized as follows: Sect. 2 provides background information about SDN control plane and defines the four performance parameters. Section 3 provides a detailed account on centralized SDN control plane, the research challenges in centralized SDN control plane and examines various centralized SDN controllers. Section 4 explains distributed SDN control plane, challenges in distributed SDN control plane and presents analysis of various distributed SDN controllers. Section 5 provides a classification of various hybrid SDN network models, research challenges in Hybrid SDN control plane and detailed analysis of various Hybrid SDN controllers. The future research perspectives in SDN control plane are discussed in Sect. 6 and finally the paper is concluded in Sect. 7.

2 Background

In this section, we have presented a detailed account on SDN control plane, SDN control plane architectures, and relevant terminology. The performance parameters including scalability, reliability, consistency and security used in this manuscript to evaluate diverse SDN controllers are also expounded.

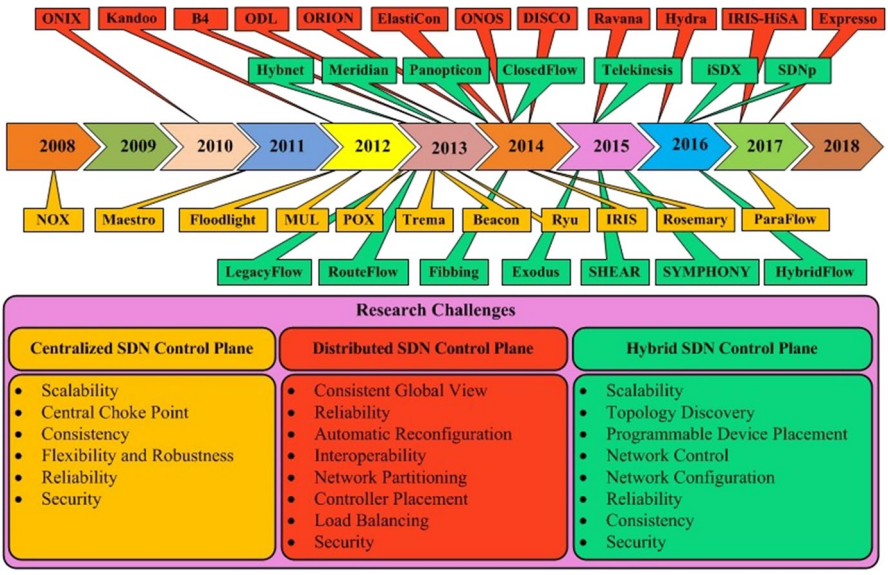

Fig. 2 Diverse SDN controllers and research challenges in various SDN control plane architectures

2.1 SDN Control Plane

The SDN control plane acts as a bridging layer between management and data plane. It plays a vital role in network control and monitoring. The network control involves programming forwarding devices as per the policy directives defined by the applications in management plane. On the other hand, in network monitoring, the control plane retrieves traffic flow statistics information from the data plane devices which can be analyzed by various applications in management plane, for instance, traffic engineering, security, etc. If congestion or a security attack is detected by such applications, the control plane can dynamically re-program the flow tables of data plane devices in such a way so that the traffic will be diverted to under-utilized paths or to intrusion detection system (IDS), respectively. The traffic flow statistics information is also useful for network provisioning to meet future traffic demands.

Most often control plane consists of a general purpose hardware executing a Network Operating System (NOS). The NOS consists of basic control software necessary to operate and manage a network. It provides global view of entire network to the applications, simplifies network programming by hiding the complex control logic implementation details, analogous to an operating system of a PC. In essence, NOS involves basic control programs necessary for topology detection and traffic management. Some of the core modules commonly found in various NOSs are shown in Fig. 3.

The topology manager module along with the link detection module maintains up-to-date topology information which involves discovery of hosts, switches and

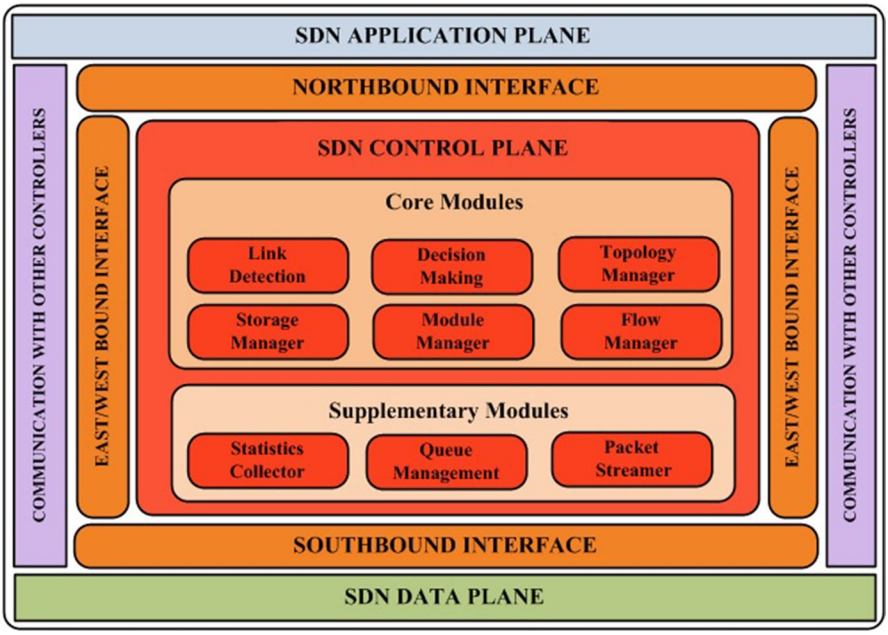

Fig. 3 SDN controller modules

links. The SDN controller discovers hosts, switches and links in a network with the help of packet_in messages, initial handshake process and LLDP protocol, respectively [39]. With the help of topology information, the decision module determines optimal paths across the network. The two other core modules are storage manager and flow manager. The storage manager stores all the necessary network state information whereas the flow manager module utilizes southbound interface to define and modify flow rules in flow tables. Other than these core modules, the controller may also have various supplementary modules like dedicated queue manager module, statistics collector module and module manager for management of queues, flow statistics collection and orchestration of information exchange between various controller modules, respectively.

In brief, SDN controller provides programmability, virtualization, centralized monitoring and dynamic network control. Providing all functions at a single central point simplifies job of a network operator and enables efficient network management. However, in multi-domain networks or in large scale networks a single controller may face scalability and control latency issues. In such domains, instead of a single centralized SDN control plane, a physically distributed but logically centralized SDN control plane is used. Moreover, a clean slate or green field deployment of SDN paradigm seems impossible with tremendous number of legacy devices deployed globally and proprietaries adamant to change due to cost and technical constraints. These factors force to use an incremental approach to introduce programmability or OpenFlow enabled devices into a traditional network (TN). Such a networking scenario, where there is an amalgam of legacy and OpenFlow devices is termed as Hybrid SDN or transitional SDN and the controller used to manage such a network is termed as Hybrid SDN controller [40, 41]. Therefore, in SDNs, we can have a single centralized SDN controller, physically distributed but logically centralized SDN controllers or Hybrid SDN controller as shown in Fig. 4.

2.2 Performance Parameters

The four performance parameters we have considered to evaluate various SDN controllers in this manuscript are scalability, consistency, reliability and security. The

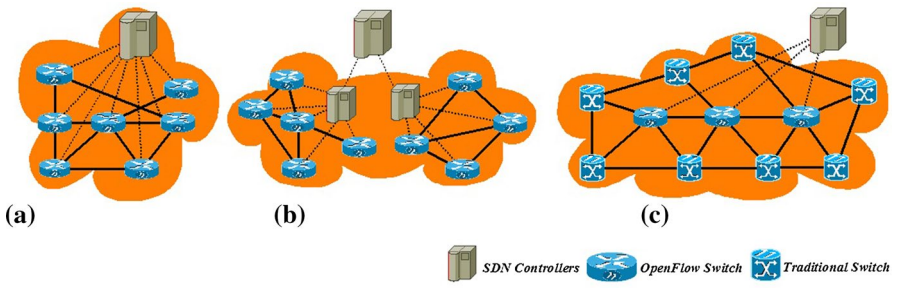

Fig. 4 SDN control plane implementations: **a** Centralized SDN control plane, **b** Distributed SDN control plane and **c** Hybrid SDN control

different control architectures face different challenges with respect to the said performance parameters and here we have emphasized all those challenges.

2.2.1 Scalability

It is a multi-dimensional topic which does not educe the same meaning in every system. In some systems, it means parallel execution of multiple applications on different CPUs whereas in others, it can be optimization of resources of the system with dynamic workload. There is no precise agreement on its definition or content [33]. However, in this manuscript, by scalability we broadly refer to controller's performance in handling flow requests, installation of flow rules in forwarding tables, delay incorporated to respond to flow requests. The physically centralized controllers try to achieve scalability by using parallelism, whereas distributed controllers achieve scalability by breaking the control plane into horizontally distributed or hierarchically organized controllers. In hybrid SDN control, the scalability largely depends on performance of central SDN controller and efficient mechanisms used for interoperability between legacy distributed control and centralized SDN controller.

2.2.2 Consistency

In general, consistency means having a stable and updated network wide view, unswerving policy update across the network and coherent fault tolerance in the network. Achieving consistent network-wide view under a single centralized controller is facile as compared to distributed control architecture. The centralized controller only has to ensure consistent forwarding during network policy update whereas in distributed control architecture consistency involves three aspects: state consistency, rules update consistency and version update consistency. State consistency ensures that the distributed controllers within a cluster have an identical global view whereas the rule update consistency ensures that switches under a controller are having the same forwarding policies for stable forwarding. Lastly, the version update consistency ensures consistent version update in distributed controllers. Ensuring all the three aspects of consistency is a complex task and involves trade-offs between performance and availability. On the other hand, in hybrid control there should be proper coordination, cooperation and translation mechanisms in place between the centralized SDN control and legacy distributed control to have a consistent network state.

2.2.3 Reliability

The reliability of a system refers to perform its functions or operations without failure or abjection. Reliability in SDN pertains to resilience in both SDN control plane and data plane. In case of SDN control plane, reliability refers to perform seamless network operations even if the primary controller fails whereas in data plane it means resilient connectivity between the forwarding nodes. The controller failures can be due to technical snag in the hardware, bugs in a software

module or sometimes deadlock due to race-condition between controller modules. In centralized SDN controllers, a simple monitoring software module and backup controller can provide fault-tolerance in control plane whereas in distributed SDN controllers designing fault-tolerant control architecture faces consistency and performance issues. In hybrid SDN control, the legacy distributed control is resilient but backup controllers are required to achieve fault tolerance in centralized controllers. To achieve reliability in data plane, the SDN controller has to compute and program flow tables with backup routes for all traffic flows.

2.2.4 Security

Security vulnerabilities in a SDN controller will compromise the security of an entire network. Within a controller, there should be measures like process containment, application permission structure and resource utilization monitoring so that the attacks like spoofing, tampering, denial of service (DoS) and privileges elevation can be mitigated. In a single centralized controller, a malformed OpenFlow header can crash a controller and frequent flow-requests from an attacker can degrade overall controller performance and in the worst case can deny services to legitimate requests. In distributed control architecture, other than the said security issues there is an urgent requirement of authentication protocols for validation and verification of controller instances. Such protocols can enable mutual authentication of controller instances before proceeding for state information exchange. Impersonation and DoS attacks are very likely in hybrid SDN control plane due to mix of two control planes and diverse devices in the network. Further, irrespective of SDN control plane architecture, the exchange at SDN interfaces need to be secured with security protocols to mitigate various integrity threats.

In this paper, we have identified the mechanisms used by various SDN controllers to address the aforesaid performance parameters. We have classified such mechanisms as good (G), limited (L) and very limited (V), depending on the number of issues addressed by those mechanisms. If majority of the issues are addressed, for instance, in case of a single centralized SDN controller, if a security mechanism ensures process containment, application permission structure and resource utilization monitoring then we have classified it as good, and if only few of these issues are addressed then limited and in the worst case very limited.

3 Centralized SDN Control Plane

The centralized controllers emerged with the inception of NOX [42] and thereafter number of centralized SDN controllers have been proposed. In this section, we have highlighted pros and cons, research challenges of centralized SDN control plane and have analyzed various centralized SDN controllers interms of said performance parameters.

3.1 Pros and Cons of Centralized SDN Control Plane

Unlike legacy distributed control, where each device is having limited knowledge of the network, a single centralized controller in SDN is having information of entire network toplogy, traffic flows and switch load. The physically centralized SDN controller monitors and manages the entire network. It remains connected either in-band or out-of-band to all forwarding devices and can define optimal paths for traffic flows across the network. In conventional networks, a number of specialized devices termed as middle boxes are used to perform functions like load balancing, firewalls, intrusion detection and prevention, etc. In SDN, such functions can be easily implemented as specialized applications over the SDN controller, hence reducing cost and complexity of the network. Additionally, developing applications for a single central controller is easy for an application developer as it has to consider requirements only of a single system, rather than considering complex issues like multiple concurrent accesses and events in case of distributed control.

The service provider networks include limited number of nodes distributed over a wide geographical area. Using a single centralized SDN controller in such a network may face high control latency and bandwidth issues. On the other hand, in multi-tenant data centers, where virtual machines (VMs) are brought up and down very rapidly, tens of thousands of network elements are required to connect such VMs. In such a dynamic environment, a huge number of networking events are generated in a short span of time and such events are sufficient enough to overload a single SDN controller. Authors in [43] have studied traffic characteristic of diverse data centers, varying in scale and have observed that for even 100 edge switches the controller may encounter 10 million flow setup requests per second. To handle such requests, they have suggested to use either parallelism in a single central controller or set up multiple controllers in the network. Likewise, Software Defined WANs (SD-WANs) impose stringent resiliency requirements. As per the authors in [44], it is difficult to achieve scale-out behaviours and desired failure resiliency with a single centralized controller in SD-WANs.

3.2 Research Challenges in Centralized SDN Control Plane

3.2.1 Scalability

A single central entity to provide all networking functions require high computation power and efficient data management techniques to respond to flow requests of forwarding switches. If flow setup requests arrive at a rapid rate and there are brisk changes in the network, such requests may overwhelm the centralized controller and can degrade the overall response time. The authors in [45] have concluded that if the network scales-up by increasing number of switches and end hosts, the SDN controller can become a bottleneck.

To address this issue, one solution is to extend limited control logic back to forwarding devices [57]. However, such a solution requires modifications in the design

of OpenFlow switches and is against the basic principles of SDN paradigm. Another possibility which is considered as an effective solution by the research community is to mold the control plane in such a way so that scalability and reliability issues are mitigated i.e., using physically distributed SDN controllers [46, 47].

3.2.2 Central Choke Point

Whenever a new traffic flow request arrives at a switch, the first packet is forwarded to the controller. The controller inspects the received packet's header, determines the path for traffic flow using topology information and then programs the forwarding tables of all data plane devices from source to destination. However, if such flow setup requests arrive at a rapid pace, the controller can become a bottleneck in handling such requests. As observed by researchers in [48], the failure of a centralized controller disrupts the overall network traffic and halts the flow setup process. Further, an upgrade in hardware/software at the controller will obstruct all services provided by the centralized SDN controller.

3.2.3 Consistency

In SDN, the data plane devices forward packets as per the policy defined by applications in management plane. Now, if a policy update takes place, there might be packets of various flows in transit which may be forwarded by a mix of old and new policies, leading to inconsistency in packet forwarding. To address this issue, the researchers have proposed various solutions including reverse update [49], consistent update [50], and a detailed survey of such solutions is given in [51]. However, it has been observed that achieving time stringent policy update across the entire network is a complex task.

3.2.4 Flexibility and Robustness

Incase of fine-grained flow matching, every new flow request very often results in modification of multiple flow table entries in various switches. Frequent such requests in a network may lead to explosion of flow table modification messages. On the other hand, handling such requests frequently with a single controller may not be robust and any failure while handling such requests may result in instable network state. Studies on NOX controller have shown that it can handle 30 K flow request per second [52]. This may suffice a campus network but it is not enough for large scale networks.

3.2.5 Security

SDN controller can provide inline network functions using global view and centralized control of data plane devices. However, this single point of network control is itself vulnerable to security attacks. From the security point of view, it is easy for an attacker to subvert a single point rather than multiple distributed devices. If attackers gain access to the controller, they can tamper/damage every corner of the network

and it will be "game over". Further, in a single central control model, if an attacker floods the controller with new flow setup requests, it will render controller inaccessible for other legitimate traffic flows. To address these issues, the management plane applications and data plane devices must be authenticated before gaining access to control plane and message integrity measures need to be taken into account for both south and northbound interface.

The majority of aforesaid issues have been confirmed by various studies including [34, 53, 54] by evaluating the performance of centralized controllers such as NOX [42], POX [55], Floodlight [56], etc.

3.3 Centralized SDN Controllers

NOX [42] is an event-based, first generation network operating system which can handle 30 K flow requests per second [54]. This controller is applicable in small enterprizes, home networks or campus networks and is not suitable for environments that generate high flow setup requests like data-center [43]. Nicira networks has developed successors of NOX, a muti-threaded NOX termed as NOX-MT to provide better performance. On the other hand, POX [55] inherited from NOX, provides better application development environment for programmers. Both NOX and POX are vulnerable to DoS, repudiation and information disclosure attacks [58]. Repudiation is possible in these controllers as they fail to maintain log of communication with switches and applications. All these controllers have gained good eminence in research and education but fail to address requirements of large-scale networks interns of throughput and reliability.

Maestro [59] is a java based multi-threaded controller which can handle 600 K flow requests per second (rps), still far-off from the requirements imposed by a large-scale data center (more than 10 million rps). It is optimised for a small domain, comprising of four main applications namely discovery, intradomain routing, authentication and routeflow. Like NOX, Maestro also crashes when it receives a malformed OpenFlow header and is very vulnerable to security attacks [54]. Another popular java based multi-threated controller from Big Switch Networks is Floodlight [56]. Although very popular in research community, but suffers from serious resiliency and security issues as reported by authors in [60]. They have also reported that floodlight controller is innately vulnerable to DoS attacks. To overcome such security issues, another version of Floodlight called SE-Floodlight has been released by Big Switch Networks. This security enhanced version still has a limitation of single point of failure. However, due to apparent functionality and performance advantages [61], open-source Floodlight controller has been extensively used to construct distributed SDN architectures such as ONOS [62], DISCO [63], etc.

Ryu [64] is a muti-threaded, component based SDN controller developed in python using gevent wrapper of libewent. It supports various Southbound APIs including OpenFlow (all versions), Netconf, OF-config, etc. It provides a convenient application development environment for developers and has a module called Ryu BGPSpeaker that can be extended for writing BGP application to define inter-domain flows. On the other hand, MUL [65] is also a mutithreaded SDN controller,

written in C using services of libevent and glib. It is a flexible, modular and easy to use controller but faces reliability and security issues as highlighted by authors in [54]. Another Ruby and C based SDN Controller framework is Trema [66]. It helps programmers to create simple, modular, customized controller in a network by defining messaging scripts in Ruby and C. It provides various libraries and supports a network emulator which can be used to create simple OpenFlow-based networks. Such a custom controller framework provides developers an efficient environment to develop and test OpenFlow networks. However, all the three controllers (Ryu, MUL and Trema) fail to address spoofing, tampering, DoS and repudiation attacks.

Beacon [67] is a modular, cross-platform, java-based controller which supports both threaded and event-based operations. Like Floodlight, Beacon also uses OpenFlowJ library for working with OpenFlow messages. Unlike, Floodlight which supports start-time modularity, Beacon has run-time modularity i.e., the capability to start and stop applications while it is running without shutting down the main Beacon process. OSGi specification [68] enables this run-time modularity in Beacon. In comparison with other centralized SDN controllers, Beacon provides high scalability but fails to address security and reliability issues [54]. Although, it can withstand privilege elevation attack due to slicing architecture (in which each application has a limited domain) but fails to resist spoofing, repudiation and DoS attacks [58].

Another java based re-factored Floodlight controller is Iris [69]. It tries to resolve scalability and reliability issues of most popular controllers like Floodlight and Beacon controller, by proposing horizontal scalability for carrier grade networks and high availability with transparent failover. It also provides multi-domain support with the help of recursive network abstraction based on Openflow. However, it also faces the challenges of single point failure and other security vulnerabilities. Since Iris controller is derived from Floodlight controller, hence involve same security issues as that of Floodlight controller.

Rosemary [70] distinguishes itself from other control platforms by proposing concepts of process containment, application permission structure and resource utilization monitoring in order to prevent common network application failures from halting the operation of SDN controller. It implements a resilient strategy and concept of network application containment based on the notion of spawning applications separated within a micro-NOS. It employs sandbox approach for access control and authentication of applications in order to prevent malicious applications to access internal data structures and modify them without restriction. It resists repudiation, tampering, spoofing and disclosure of information attacks due to auditing service and micro-NOS permission structure [71].

The latest multi-threaded centralized controller with fine-grained parallelism is ParaFlow [72]. Unlike, conventional parallelism used by various multithreaded SDN controllers, ParaFlow exploits parallelism not only in event handling but also in event processing by event handlers. ParaFlow introduces flow-based programming interface that enables application developers to create application programs using network flows instead of low-level assorted events. ParaFlow is basically a lightweight controller written in C++, using Boost library to achieve parallel asynchronous I/O. In this multithreaded SDN controller, multiple threads operate concurrently on network state, which is stored in the shared memory. The consistency

is ensured by the mutex-based synchronization mechanism; however, such mechanisms may result in errors if fine-grained parallel applications perform concurrent accesses.

3.4 Insights

Most of the centralized SDN controllers face scalability, reliability and security issues due to single point of network control, lack of resiliency measures and absence of security mechanisms, respectively. Apart from Rosemary [70], all other centralized SDN controllers lack security measures as shown in Table 1. Majority of the centralized SDN controllers support multithreading and use OpenFlow as southbound interface. On the other hand, the centralized SDN controllers either use an inbuilt API or REST APIs as northbound interface. The centralized SDN controllers are applicable in small enterprises, campus networks, domain specific networking in small-scale data centers and edge networks.

4 Distributed SDN Control Plane

In the last few years, physically distributed SDN control plane has received much attention from the reseach community [34, 35] and a number of distributed SDN control frameworks have been proposed. In this section, we have presented pros and cons, research challenges in distributed SDN control plane and have analyzed various distributed SDN controllers.

4.1 Pros and Cons of Distributed SDN Control Plane

The distributed SDN control plane overcomes scalability, reliability, performance and single point failure problem of centralized SDN control plane by introducing multiple controllers in a SDN. The distributed SDN control plane is more robust, scalable and responsive which can effectively react to diverse networking events like link failures, new flow setup requests, intrusion, etc. In dynamic environments like multi-tenant data centers where millions of networking events are generated frequently the distributed control plane architecture provides flexible and scalable solutions to manage such events [43].

Likewise in WANs, placing network controllers at strategic points can facilitate quick response and consistent view of network changes [46, 112]. To have better response time and performance, the distributed SDN controllers use load sharing mechanisms to distribute data plane switches among the controller instances. However, flexible load balancing involves continous state sharing communication overhead among the distributed controllers. Further, the distributed SDN control plane architecture faces numerous challenges in terms of interoperability, consistency, controller placement, etc., and we have presented all these challenges in detail in the next sub-section.

Table 1 Centralized SDN controllers

Controller	Programming Language	Mutithreading Support	Interface	Modularity	Northbound interface	Southbound interface	Partner	Performance				Platform	Applications
								Scalability	Reliability	Consistency	Security		
NOX [42]	C++	NOX-MT	Python + QT4	Low	ad-hoc	OF 1.0	Nicira Networks, USA, 2008	V	V	G	V	Mostly Linux	Campus
POX [55]	Python	No	Python + QT4	Low	ad-hoc	OF 1.0	Nicira Networks, USA, 2013	V	V	G	V	Linux, MAC and Windows	Campus
Maestro [59]	Java	Yes	Web Based	Fair	REST API	OF 1.0	RICE University, USA, 2011	V	V	G	V	Linux, MAC and Windows	Research
Floodlight [56]	Java	Yes	Web Based	Fair	REST API	OF 1.0, 1.2, 1.3, 1.4 and 1.5	Big Switch Networks, USA, 2012	V	V	G	L*	Linux, MAC and Windows	Campus, Research (* for SE-Floodlight)
Ryu [64]	Python	Yes	CLI	Good	REST API	OF 1.0, 1.2, 1.3, 1.4, 1.5 Netconf., OF-config	Nippo Telegraph And Telephone Corporation, Japan 2013	L	V	G	V	Mostly Linux	Campus, Research

Table 1 (continued)

Controller	Programming Language	Mutithreading Support	Interface	Modularity	Northbound interface	Southbound interface	Partner	Performance				Platform	Applications
								Scalability	Reliability	Consistency	Security		
MUL [65]	C	Yes	Web Based	Fair	REST API	OF 1.4, 1.3, 1.0, OVSDB, OF-config	Kulcloud, 2012	V	V	G	V	Linux	Data center
Trema [66]	Ruby and C	Yes	CLI	Fair	ad-hoc	OF 1.3, 1.0	NEC, 2013	V	V	G	V	Linux	Research
Beacon [67]	Java	Yes	Web Based	High	REST API	OF 1.0	Stanford University, USA, 2013	L	L	G	V	Linux, MAC and Windows	Research
Iris [69]	Java	Yes	Web Based	Fair	REST API	OF 1.0, 1.3, OVSDB	ETRI, 2014	L	L	G	V	Linux, MAC and Windows	Carrier-Grade
Rosemary [70]	C	Yes	N/A	High	ad-hoc	OF 1.0, 1.3	-, 2014	L	L	G	G	Linux	Campus, Research
ParaFlow [72]	C++	Yes	N/A	High	Abstract APIs	OF 1.0, 1.3	-, 2017	L	L	G	N/A	Linux	Carrier-Grade

V Very Limited, *L* Limited, *G* Good

4.2 Research Challenges in Distributed SDN Control Plane

4.2.1 Consistent Global View

In distributed SDN control plane, domain-specific controllers address data plane failures or traffic flow congestion in their respective domains. In order to have a consistent global view such changes should be communicated to all other controller instances within a cluster in a timely manner. However, achieving such time stringent level of consistency while maintaining good performance is a complex task [73].

The strong consistency model guarantees that all distributed controllers have latest network information, but at the cost of communication overhead and increased synchronization. Such strong consistency models introduce new scalability challenges and retaining sturdy consistency during recurrent state updates might obstruct the state progress and can render network unavailable, resulting in higher switch to controller latencies. On the other hand, eventual or weak consistency models allow concurrent reads and such read operations may return different values from the actual updated values for a short transient period. Consequent to such dissimilar values retrieved by the SDN controllers, there can be inconsistent global view of the network which may result in incorrect application behaviour. As per authors in [74], inconsistency in control plane can have considerable effect on the network performance. So maintaining a consistent global view across all controllers is a design challenge that involves trade-offs between policy enforcement and performance [74].

4.2.2 Reliablility

Unlike traditional networks, the emphasis in SDN is not only on resilient data plane but also on resilient distributed control plane [75]. The fault tolerance in control plane is commonly achieved through active or passive replication mechanisms [76, 77]. In active replication strategy, OpenFlow switches keep simultaneous connection with multiple controllers and if one controller fails, the others can still control the switches. On the other hand, in passive replication or primary-backup replication, each switch is connected with only one controller (termed as primary controller) and if the primary fails then the backup controllers can take control of the network.

In centralized SDN control, a simple Master/Slave approach can be used for fault-tolerance, whereas in distributed SDN control, such a simple approach will not work as the network state information is partitioned among many controllers which exchange information to maintain a consistent logically centralized global view [78]. So in distributed SDN control plane, there should be coordination strategies to solve and reach agreements on concurrent state updates and to maintain consistent network state. In large-scale networks, a simple self-healing approach can be implemented wherein if one controller goes down, the load will be redistributed among the remaining active controllers. However, such an approach involves overhead of state maintenance and frequent distribution of domain state among the participating controllers.

Additionally, implementation of replication strategy is a challenging aspect of distributed control plane [79] as there should be mechanisms also for state storage replication. Some approaches store network state of a replication controller locally and use specific group coordination framework for communication [80], others delegate state storage, replication and management of state information to external data stores like distributed file system and distributed data structures [81, 82]. In designing a reliable fault-tolerant distributed SDN control architecture issues like consistency, scalability and performance should also be taken into account.

4.2.3 Automatic Reconfiguration

The mapping between distributed controllers and forwarding devices must be automated rather than using static configurations. Static configurations may result in uneven load distribution among the controllers within a cluster. For distribution of switches to different SDN controllers, there should be an application operational on all active controllers which monitors and shares the network load information with the neighbouring controller instances. However, this approach may overload the controllers with load sharing information, leading to scalability issue. Further, in the absence of standard northbound and eastbound interface, communication among applications and application portability is hard to achieve.

4.2.4 Interoperability

To foster development and adoption of SDN in next generation networks, there is an urgent need of ensuring interoperability among heterogenous distributed SDN controllers operating under different administrative domains and using different technologies. The main reason for the lack of interoperability is absence of standard east/westbound interfaces and heterogeneity in data models used in various SDN controllers.

YANG [83] has emerged as a data modelling language to represent state and configuration data in a standard form. This NETCONF based IETF contribution is expected to be extended in future to pave way for standard data models enabling interoperability in SDNs. Another initiative in this direction is from OpenConfig's efforts on building a standard data model for management and configuration operations, which is vendor-neutral written in YANG [84]. ONF's OF-Config protocol has implemented YANG based data models called Core Data Model to enable remote configuration of OpenFlow capable devices [85]. However, such data models and protocols should be integrated into various heterogenous distributed controllers to achieve interoperability.

4.2.5 Network Partitioning

The topology based partitioning of network in distributed SDN control may result in performance degradation of latency sensitive (e.g., monitoring) or compute intensive (e.g., route computation) applications. As reported by authors in [86], if latency sensitive and compute intensive applications are co-located within a controller

which controls a particular network partition, achieving low response time and convergence time concurrently might be challenging. They have proposed functional slicing in which different applications are placed in different partitions in order to reduce inter-controller communication between applications. They have observed that functional slicing and communication aware placement of control applications can minimize network convergence time and response time. We believe that network partitioning is an optimization problem and requires further attention from the research community.

4.2.6 Controller Placement and Load Balancing Problem

Decoupling control plane from the forwarding devices into a logically centralized SDN controller raises questions like where to place these controllers and how many controllers are required in a network? Such questions need to be answered, particularly in WANs, where propagation latency becomes a decisive factor. In other areas, like data center or enterprize, researchers are focused more towards load balancing and fault tolerance.

Undoubtedly, the distributed SDN control architecture is a scalable option as compared to the centralized one, but achieving scalability and at the same time good performance requires a strategy that takes into account both physical placement of controllers and numbers of SDN controllers required. Several placement schemes have been proposed by researchers including [46, 87, 88] and all these schemes use different approaches and heuristic algorithms to achieve optimal values of performance parameters like link delay or flow set-up time, etc. In [89], authors have proposed a framework that combines hierarchical clustering and betweenness centrality models to reduce switch-to-controller latency and concurrently balance the load using centrality scores of the nodes. These performance parameters are interdependent and finding the optimal solution is an NP hard problem. We believe that controller placement problem should be investigated as an optimization problem with focus on multiple performance parameters.

4.2.7 Security

Since the entire network intelligence resides at distributed SDN controllers, compromising security of a distributed controller instance can jeopardize the entire SDN network. In large scale networks, if security issue is not addressed, SDN might lose the control plane availability [90, 91]. Although, distributed control architecture overcomes risk of subverting a single central controller, but faces authentication and message integrity challenges. Without proper authentication mechanisms in place, any attacker can easily introduce its node within the network which will behave like other SDN controller instances and can corrupt the entire network. Securing information exchange between controllers is also very vital to ensure consistent network wide view in distributed SDN Controllers. Addressing such issues require new strategies and application of security protocols, with the aim of securing distributed SDN control environment.

4.3 Distributed SDN Controllers

The physically distributed controllers can be broadly classified into two categories based on the physical organization of controller instances. The first category includes hierarchical control plane wherein the network control logic is partitioned into multiple layers as shown in Fig. 5a. The second category is a flat structure in which network control is partitioned horizontally to handle multiple areas as shown in Fig. 5b.

The hierachical control plane also referred as vertical model provides better scalability and performance. Each layer of controllers provide specific services, for instance, in two layer architecture as shown in Fig. 5a, the lower layer of controllers handle local events and provide services to switches, whereas top layer provides global view and interoperability between the lower layer area-specific controllers. The lower layer controllers (or local controllers) does not have a direct connection with each other, instead each local controller uses services of upper layer controller (or root controller) for interdomain connectivity.

In flat SDN control plane, the network is partitioned into multiple areas and each area is under the control of a local controller. Organizing controllers in such a fashion provides good resiliency and less control latency. This architecture is also referred as horizontal architecture. To have a global view, all controllers in a flat cluster remain connected with each other using eastbound interface and share/receive network state information to/from other controllers as shown in Fig. 5b. Every local network event like link or node failure is shared with other SDN controllers within the cluster in order to have a consistent global view. Consequent to it, this model is also called as peer-to-peer model or replicated state machine model.

4.3.1 Hierachical SDN Controllers

Google's B4 [92] is the first and largest private intradomain SD-WAN connecting multiple data centers across the globe. It implements various control applications

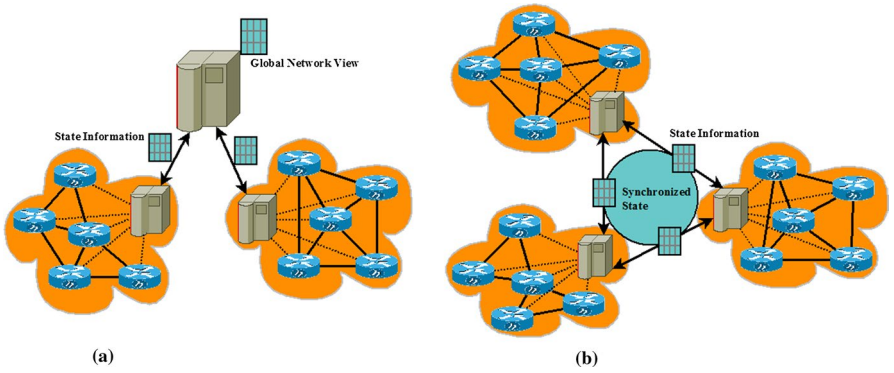

Fig. 5 Types of distributed SDN control plane: **a** Hierarchical SDN Control Plane, **b** Flat SDN Control Plane

to achieve cost efficient networking between scalable WAN sites. As per designers of B4, it consists of two-level hierarchical control plane to meet elastic bandwidth demands of interconnected data centers. The bottom layer comprises of Onix based [82] SDN controllers managing each data center site and using site-level control applications. At the upper layer or global level, a global SDN Gateway manages the site-level controllers and provides the site-level TE services. It collects necessary network information from the lower level distributed controllers and forwards this information to a logically centralized TE server. The TE server operates at the upper layer and enforces high-level TE policies that optimize the elastic bandwidth demands among the competing control applications working across the different data center sites. The TE server uses gateway APIs to program the TE entries into high-priority forwarding switch tables alongside with shortest path routing (SPR). The routing and TE are deployed as separate independent services with former acting as the base and later deployed as an overlay.

To reduce complexity in B4 network, the topology abstraction is used wherein each data center site is represented as a super-node and aggregated links connecting these super-nodes as super-trunk. This network abstraction reduces the complexity and allows the centralized TE server to run protocols at coarse granularity level. For reliability and fault tolerance, the redundant controllers are deployed at both site-level and global-level. These availability mechanisms have been largely improved after experiencing a large-scale outage in B4 in the intial deployment. At site-level, Paxos [93] is employed to detect and handle the failure of a primary controller. Incase a primary controller fails, it selects a new leader among the set of standby controller instances placed on different physical servers in the data center. On the other hand, at global-level, the logically centralized TE controller is guarded against failures by geographically replicating TE servers across multiple WAN sites. More precisely, one master and four secondary hot standby TE servers are used in B4. Further, if TE service goes down, the standard shortest-path routing is employed as a sovereign service.

Kandoo [94] proposes a two layer hierarchical control architecture in which bottom layer controllers manage OpenFlow switches of a domain without having network-wide state information. On the other hand, the top layer consists of a logically centralized controller or root controller that maintains the network-wide state information. The root controller can install flow entries into the OpenFlow switches by delegating requests to the particular local controllers. These local controllers in kandoo framework can scale linearly and provide adaptive control wherein default configuration can be pushed proactively and can be refined afterwards. However, this framework lacks measures to safeguard controllers from security attacks and failures.

Logical xBar [95] introduces a recursive building block to design SDN control plane for a large-scale and worldwide distributed network. Logical xBar is actually a programmable entity which switches packets between ports, using aggregation of smaller units (like OpenFlow switches) for forwarding into larger ones. The control plane computations and management of forwarding table are carried out in management CPUs termed as Logical Servers. These logical servers are replicated for fault tolerance and scalability. The hierarchical architecture is achieved by recursive

aggregation of logical xBars and their respective logical servers. Each xBar stores state information and configuration information to formulate table of topology at each recursive level. The control plane acquires summarized state by upwards aggregation and configuration information is disseminated downwards to forwarding devices. This approach uses virtualization to provide extensible, scalable, and locally-scoped control planes for large scale networks. However, no information about state storage and consistency is provided in [95].

With the success of B4 [92] in optimizing the overall network performance and reducing the complexity in management of inter-data center WAN motivated designers to use resonant concepts of SDN in Google's edge network. Google designed Expresso [96] an architecture which provides cost-effective, reliable and exponentially scalable peering edge network integrated with global traffic systems. It provides application-aware routing and routes over 22% of Google's overall traffic to Internet.

Like B4, Espresso also uses two layer hierarchical control plane architecture, with top layer comprising of global controllers and lower layer of local controllers. The local controllers react to local networking events, for instance performing local repairs like link or peering port failure whereas the global controllers optimize global traffic to improve efficiency. Espresso is designed with intent-driven manageability to support large scale operations which are safe, automated and incremental. Further, it uses software development design principles that allows network evolution with changing application requirements and enables new innovative features to be deployed with high velocity.

Espresso provides full interoperability with traditional heterogenous peers or with rest of the Internet. It provides a fail static system in which if a local controller fails, the data plane works as per the last known good state without impacting BGP peering and other data plane operations. All these features implemented in Espresso provide high reliability, interoperability and supports incremental deployment.

4.3.2 Flat SDN Controllers

Onix [82] is a distributed control architecture comprising of one or more physical servers with each server executing multiple Onix instances. To provide scalability and resiliency in large scale networks (connecting millions of ports), each Onix controller instance disseminates network state information to other instances within the cluster. It uses a data structure termed as Network Information Base (NIB) to store network state. This data structure is partitioned among multiple controller instances that hold responsibility for the subset of NIB. Onix achieves scalability and resiliency by partitioning, aggregation and replication of NIB among controller instances. Onix ensures consistency by using distributed locking and consensus algorithms. It uses application-specific logic for both detection and conflict resolution of network state. ONIX also supports distributed hash table (DHT) to provide general APIs which guarantee weak consistency of network view. Fault tolerance of link or node is handled by the control applications whereas controller failure is handled by using distribution coordination function among controller replicas. Such a controller is applicable in environments which require high availability and frequent

updates. The Onix control framework lacks confidentiality and integrity measures to ensure secure state sharing among controllers within a cluster.

On the other hand, ONOS [62] uses multiple instances of floodlight controller to build a distributed control platform for scale-out performance and fault tolerance. This framework executes controller instances on numerous servers, each of which handles a subset of OpenFlow switches. Fault tolerance is achieved by connecting each switch with multiple controllers, one acting as a master controller and others operate as backup controllers. For consistency and integrity of network information, ONOS uses Titan's transactional semantics (that maintains graph's structural integrity) on top of Cassandra's consistent data store. For better performance, it employs load balancing mechanisms to balance the number of switches under a master controller. Further for controller break down, it uses Anti-Entropy protocol [97] recovery mechanism for healing lost updates. ONOS employs TLS and HTTPS at southbound and northbound interfaces, respectively to prevent tempering and information disclosure threats. Authorized access and fine grained control to internal data-structures and libraries are provided in Secure-Mode ONOS (SM-ONOS). It uses strict access control measures and security audit service to prevent repudiation and elevation of privilege threats.

The designers of Ravana [98] argue that using simple primary/backup methods [62, 82] or replicated state machines (RSM) for controllers can provide fault-tolerant consistent state only in control plane and fail to provide resilient and consistent switch state. Such mechanisms do not capture the switch state accurately. Consequent to it, if a master controller crashes while configuring a switch, the new master may not know from where to resume the switch configuration process. So, in Ravana, instead of keeping only the controller state consistent, the entire event-processing cycle (including events from switches, event processing at controllers, and commands forwarded to switches) is treated as a transaction; either all or none of the events of this transaction are executed. This architecture ensures that transactions are totally ordered across replicas and executed only once across the entire system.

In Ravana architecture three components including switch runtime, Ryu-based controller runtime and control channel interface works collectively and cooperatively to ensure desired correctness and robustness of fault-tolerance in logically centralized controller. Further, the authors propose Ravana protocol that detects the failure of master controller and performs leader election among the slave controllers like the Zookeeper [99] mechanism. The new elected leader before proceeding with the normal operations first, finishes the job of a failed master controller. Since Ravana is based on Ryu controller, it faces the same security challenges as that of Ryu controller.

The two popular controllers using flat distributed control architecture are OpenDaylight (ODL) [100] and DISCO [63]. The former is a logically centralized control architecture whereas the later is a logically distributed control architecture (logical classification adopted by [101]). The ODL is a general-purpose framework administered by Linux Foundation and supported by the industry. It is an open source community driven framework providing full functionalities of a Network Operating System. It was conceived to address requirements of multiple domains including data center, enterprise and service provider networks. The latest release of this

framework is Magnesium [102] which supports wide range of Southbound APIs including OpenFlow, NETConf, OVSDB, PCEP, etc. The main architectural feature of Magnesium is Model-Driven Service Abstraction Layer (MD-SAL) based on YANG models, that allows simple and flexible integration of network services requested by the application layer via northbound APIs (supporting RESTful interface, OSGi Framework and intents). The Magnesium framework integrates ODL with the Container Orchestration Engine for Kubernetes environments.

Another aim of ODL project was to accelerate integration of SDN with traditional networking environments, to automate management and configuration of legacy network devices and enabling them to communicate with OpenFlow enabled devices. The distributed controllers in ODL maintain a logically centralized view by using Akka framework [103] and RAFT concensus algorithm [104]. It's latest release provides base for running business logic and other control algorithms as applications in management plane. Such control applications e.g., BGPCEP, BGP/MPLS, OpenROADM, etc. enable this framework to control devices in various domains like WAN, cloud or edge networks. ODL is the most secure distributed SDN control platform with specialized security modules like Secure Network Bootstrapping Infrastructure (SNBI), AAA service, Defense4All, etc. [71]. It uses Apache Karaf security framework to control accesses of OSGi services, console commands, java management extension layer and WebConsole. Such security features in ODL enables it to mitigate various security threats like DoS, spoofing, repudiation, etc.

On the other hand, DISCO's [63] logically distributed control architecture is applicable in multi-domain heterogeneous environments particularly overlay networks and WANs. DISCO control architecture is based on Floodlight controller in which each distributed controller instance controls a single SDN domain and interacts with other instances for end-to-end transport service. A unique lightweight channel is used between two controller instances to share summarized domain information. The DISCO architecture separates intra and inter-domain control logic wherein the intra-domain control part monitors the local network and responds to the local issues and the inter-domain part handles communication with other domains with the help of AMQP-based messenger [101] service. This AMQ-based messenger uses publish/subscribe mechanism and provides communication channel for agents operating at inter-domain level. Like Floodlight, DISCO controller instances also lack security measures which make them vulnerable to various security threats.

The distributed architecture of DISCO is envisioned to work in large scale networks under different administrative domains like Internet [101]. However, this logically distributed architecture has numerous drawbacks. First, the local controllers having limited network vision can optimize the local network performance rather than providing global optimal performance. Second, in DISCO a single controller is responsible for each domain, but in large scale networks controlling entire domain with a single controller instance raises scalability and resiliency issues. In DISCO if a local controller fails the nearby neighbouring controllers take care of the affected switches, however, it may result in increased control plane latency. Lastly, this decomposition of network into logically independent entities is in contrary to emerging theory [105] which proposes manageable network infrastructure by a separate plane termed as Knowledge Plane on top of the logically centralized controller.

IRIS-HiSA [106] involves a pool of distributed controller instances and each instance uses global topology information for network management. This global network information is obtained by using publish-subscribe mechanism among the controller instances. IRIS-HiSA proposes dynamic mechanism for load sharing among the controllers and take-over load mechanism from the failed controller. The load balancing among the controller instances is ensured by a session management module which uses load balancing algorithms to distribute switches among the controller instances. The main advantage of IRIS-HiSA is that it provides transparent control to switches, i.e., the switches in data plane simply connect to one or other controller instances without having any knowledge of internal architecture of control cluster. This feature in HiSA controller cluster is achieved by assigning dynamically a new connection request from a switch to a specific controller instance within the controller cluster based on the load. Fault tolerance in HiSA controller cluster is achieved by receiving a periodical status message (running/down) from each controller instance. Once failure is detected, a new controller instance is brought up and takes role of a failed controller. Moreover, IRIS-HiSA achieves consistency and state sharing by using Hazelcast [107]. Hazelcast involves a distributed in-memory database to synchronize state-information with other controller instances in the cluster and IMap data structure for state sharing.

Hydra [86] is another Floodlight based distributed SDN control framework which focuses mainly on network partitioning problem. The authors argue that conventional partitioning based on topology in controllers like Onix [82] results in performance issues. The performance of latency-sensitive applications (e.g., monitoring) associated with a given network partition may be obstructed by another co-located compute-intensive application (e.g., route computation). Achieving simultaneous low response time and convergence time might be challenging and communication between various applications across different partitions may encounter high latency. To mitigate such issues, authors have proposed functional slicing in which applications performing a particular control function are placed in physically distinct servers irrespective of the network partition. So partitioning of control applications is based on the functionality rather than using the conventional topological partitioning. In Hydra, the primary metric used by authors is network convergence time and response time. To achieve fault tolerance in control plane, authors have used master–slave controller replication and paxos [93] algorithm, but have not addressed the security issues of distributed floodlight controller instances.

Elasticon [108] is an elastic, load adaptive distributed control architecture in which controller instances are increased or decreased depending upon the network load. The Elasticon control framework comprises of following three important modules load measurement module, load adaptation module and decision module. Load measured by the measurement module is forwarded to the load adaptation and decision module, which initiate actions to shift switches between controller instances or to add/remove controller instances. Under heavy load, new controller instances are added into control cluster and switches are migrated to these controllers in order to balance the load. The load rebalancing algorithm balances the number of switches under the control of different controller instances. A specialized switch migration

protocol has been proposed by authors in [108] that provides consistent, disruption free and serializability of events during switch migration.

Additionally, in Elasticon, SIGAR API [109] has been used to retrieve CPU usage information from various controllers. The REST API architecture of controller has been enhanced to respond to CPU usage queries. The adaptation decision algorithm queries all controller instances about the switch load using these REST APIs. After retrieving the CPU usage information, reflecting the current load on a particular controller, the decision alogrithm may initiate the switch migration process. However, the elasticon control framework lacks fault tolerance and security measures.

Orion [110] proposes a hybrid control plane architecture, combining features of both flat and hierarchical SDN control plane. This hybrid hierarchical architecture comprises of three layers: (i) Bottom Layer: consists of large number of connected OpenFlow switches, (ii) Middle Layer: comprises of area controllers which are responsible for collecting physical device and link information for topology management and handling network events within an area. The area controller abstracts network view of an area and forwards it to the upper layer controllers; and (iii) Upper layer: consists of domain controllers which abstract the area controllers as simple devices and synchronizes the abstract network-wide view with the help of a distribution protocol. Orion minimizes computational complexity of the control plane from super-linear growth to linear by dividing network into smaller areas and builds abstract hierarchical area views. Orion developers have developed number of controller modules including routing module which uses Dijiktras algorithm [111] for area routing or domain routing, storage module which uses NoSQL database for dynamic clustering and storage of abstract topology information, etc. The Orion architecture is scalable, fault-tolerant, with limited consistency measures but lacks security mechanisms.

4.4 Insights

The physically distributed but logically centralized SDN controllers like Onix [82], ODL [100], etc. are more suitable for enterprises and data centers, as these use cases are under a single administrative control. Such controllers provide strong consistency measures and require limited inter-controller communication. On the other hand, physically as well as logically distributed controllers like DISCO [63] are applicable in multi-domain heterogeneous environments, particularly overlay networks and WANs. Most of the distributed SDN controllers support REST APIs as the northbound interface and lack security measures as shown in Table 2. The different mechanisms used for consistency and reliability (for both Control Plane (CP) and Data Plane (DP)) in each controller framework are summarized in Table 2.

5 Hybrid SDN Control Plane

In a hybrid network, SDN controller controls both programmable and legacy devices. The level of control depends on the mechanisms used by hybrid controllers to control legacy distributed devices. In this section, we have discussed pros and cons of hybrid SDN control plane, research challenges in hybrid SDN control architecture and classification of hybrid SDN models. Further, we have presented the analysis of various hybrid controllers in terms of said performance parameters and a mapping of hybrid controllers to different hybrid SDN models.

5.1 Pros and Cons of Hybrid SDN Control Plane

Transitional or hybrid networks provide an interim option for enterprizes or cloud operators to change the networking paradigm incrementally from TN to SDN. Transitional networks reduce need of skilled man-power, initial investment for organizations by enabling them to implement programmable devices incrementally into a legacy network. In hybrid control, the centralized SDN control plane and legacy distributed control plane communicate at diverse levels to configure, manage and control forwarding devices. The two control planes with effective communication mechanisms can complement each other and can provide better end-user experience and optimization of network resources. For instance, the Routing Control Platform (RCP) [113] uses a hybrid approach wherein the centralized control plane simplifies and augments the BGP decision logic to provide optimized inter-domain IP routing. Several other studies including [40, 41, 114, 115] have implemented hybrid control architecture and have reported that changing legacy devices to programmable devices incrementally in a network can also provide better control and performance.

With the help of centralized global view, fine-grained control of traffic flows can be achieved in hybrid networks. If such fine-grained flow control is required for a limited set of traffic flows, then such traffic flows can be managed by the centralized SDN control plane whereas the rest of traffic flows can be controlled by traditional networking [116]. Additionally, TN is very effective in handling certain tasks due to resiliency and reliability features and having centralized control only on limited traffic flows reduces burden on SDN controller and guarantees controller scalability for large-scale networks. On the other hand, hybrid SDN control faces numerous limitations due to amalgam of various devices in a hybrid network. The network configuration, topology discovery and overall network control in hybrid network is very complex and involves numerous issues which we have presented in detail in the next sub-section.

5.2 Research Challenges in Hybrid SDN Control Plane

5.2.1 Scalability

With the increase in number of OpenFlow devices in a hybrid network, the load on the centralized SDN controller increases and may lead to scalability issue. Further,

Table 2 Distributed SDN controllers

Controller	Programming Language	Architecture	Southbound interface	Northbound interface	Reliability Mechanism	Consistency mechanism	Performance				Partner
							Consistency	Reliability	Scalability	Security	
ONOS [62]	Java	Flat	OF1.0, 1.3, NETCONF	REST API, Neutron	CP: Redundant instances	Raft Cassandra database	G L	G	G	G	ON.LAB, AT&T, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, Nec, Nsf. Ntt Communication, Sk Telecom, 2014
Onix [82]	Python, C	Flat	OF 1.0–1.5	NVP –NB API	DP: Backup Paths CP: Active replication	Replicated NIB DHT	G L	G	G	V	Nicira Networks, Google, NEC ICSI & UC Berkeley, 2010

Table 2 (continued)

Controller	Programming Language	Architecture	Southbound interface	Northbound interface	Reliability Mechanism	Consistency mechanism	Performance				Partner
							Consistency	Reliability	Scalability	Security	
B4 [92]	Python, C	Hierarchical	OF (Version not mentioned)	N/A	CP: Controller Redundancy using Paxos for leader selection SPR is used in absence of Global TE mechanism	N/A	–	G	G	N/A	Google, USA, 2013
Kandoo [94]	C, C++, Python	Hierarchical	OF 1.1 and 1.2	Java RPC	N/A	N/A	–	V	G	V	University of Toronto, 2012
Expresso [96]	N/A	Hierarchical	OF (Version not mentioned)	N/A	CP: Hot standby controllers	N/A	–	G	G	N/A	Google, USA 2017
Ravana [98]	Python	Flat	OF with extensions	N/A	CP: Custom protocol for master failure and leader election among slave controllers	Transactional DB	G	G	L	V	Princeton University, USA 2015

Table 2 (continued)

Controller	Programming Language	Architecture	Southbound interface	Northbound interface	Reliability Mechanism	Consistency mechanism	Performance				Partner
							Consistency	Reliability	Scalability	Security	
ODL [100]	Java	Flat	OF 1.1–1.5, NETConf, OVSDB, PCEP, etc	REST API and OSGi Framework	CP: Nearby controllers as hot standby	Raft	G	G	G	G	Linux Foundation, USA, 2013
DISCO [63]	Java	Flat	OF 1.1–1.3	REST API	N/A	Messenger, agent	G	L	G	V	Thales Communications & Security, France 2014
Hydra [86]	Java	Flat	OF (Version not mentioned)	REST API	CP: Replication of controller configuration	N/A	–	G	G	V	Purdue University, 2016
IRIS-HiSA [106]	Not Mentioned	Flat	OF 1.0 and 1.3	REST API	CP: Runtime Controller switching	Hazelcast	G	G	G	V	Electronics and Telecommunications Research Institute, Korea 2017
ElastiCon [108]	Java	Flat	OF (Version not mentioned)	REST API	–	Hazelcast	G	V	G	V	Purdue University, 2014
Orion [110]	Java	Flat and Hierarchical	OF 1.2	N/A	CP: Master/Slave approach	NoSQL	L	G	G	V	Tsinghua University, China, 2014

V Very Limited, *L* Limited, *G* Good

in a hybrid network, other than programming of flow-tables of OpenFlow devices, the SDN controller has an additional responsibility of legacy device protocol translation for interoperability and configuration of legacy devices. To address this issue, additional controllers may be put into the action in the network, however, it will result in more complex network management due to presence of physically distributed but logically centralized SDN control plane and legacy distributed control plane in the network. Ensuring consistency and coordination of diverse control planes in such a network becomes an uphill task.

5.2.2 Topology Discovery

In hybrid networks, topology discovery is more challenging than pure SDNs due to presence of diverse vendor-specific devices, supporting different protocols such as IS-IS, OSPF, BGP, SNMP, etc. To discover such heterogenous devices, the SDN controller has to support different protocols and conversion mechanisms in order to obtain a global network view. Performing such protocol translations and supporting all protocols can easily overwhelm a single centralized controller. Several proposals have been given by researchers including [117–120] for topology discovery, but most of them have extended support for a particular protocol which works at a specific layer like layer 2 or layer 3. To have seamless connectivity in a hybrid network, hybrid SDN controllers need to be enhanced in terms of topology discovery protocols.

5.2.3 Programmable Device Placement

Hybrid SDN controller is able to manoeuver traffic in a hybrid network with the help of programmable or SDN devices. However, in such a network there are limited number of programmable devices and a network operator has to decide the number and location of these devices. The decision is based on multiple factors such as CAPEX, performance benefits and network topology. This decision making is an optimization problem with focus on performance parameters like link utilization, minimum disruption and overall cost–benefit. For such a complex problem, some researchers have used many heuristics including node degree by Hong et al. in [121] and node degree and traffic volume by authors in [41]. However, such parameters may affect the scalability of SDN controllers in a physically centralized control architecture or consistency in a distributed control architecture. Further, the number and place of programmable devices depends on multiple parameters like bandwidth, latency, traffic-flows, etc. and we believe that this issue requires further attention from the research community.

5.2.4 Complex Network Control

In a hybrid network, the traffic forwarding in data plane is controlled by both legacy distributed control plane and centralized SDN control plane. The legacy distributed control forwards traffic as per the limited view of the network whereas the centralized SDN controller does so as per the global view of the network. If there is no

coordination or cooperation between the two control planes, any network update may lead to forwarding inconsistencies which can result in traffic flow disruption and in the worst case formation of loops in the data plane. To address this issue, there should be network state verification and conflict resolution mechanisms in place in the hybrid networks. However, till date no such verification and resolution mechanism has been designed for hybrid networks.

5.2.5 Network Configuration

Network Configuration is a critical aspect in network management. Whenever there is a change in network like link or node updation, policy change or expansion of the network, the forwarding devices need to reconfigured accordingly. TNs lack sophisticated network configuration mechanisms and makes network management very challenging and error prone process. On the other hand, SDN provide abstractions and support for reconfiguration with the help of single central point for network control. However, in a hybrid network, network configuration is a complex task as the devices involved are both programmable and closed vendor-specific. Several researchers have proposed various solutions for configuration of hybrid networks including some device specific or protocol specific approaches [114, 122, 123] and some general frameworks [124, 125]. However, we believe a more robust approach is required to have interoperability and cooperation in network configuration in a hybrid scenario.

5.2.6 Security

In hybrid SDN network, there are more security challenges due to presence of diverse type of devices in the network. Authentication of devices becomes necessary to mitigate impersonation attacks, otherwise attackers can easily masquerade and disrupt the network operations. In various Hybrid SDN models, only OpenFlow devices are completely controlled by the SDN controller, so a security application running on the controller cannot mitigate all the threats confronted by the legacy devices. Moreover, the different legacy devices understand varied protocols which makes it challenging for a security application running on the SDN controller to detect all possible security threats. In such a network, if the controller has to secure entire network than it has to understand all legacy device protocols but it may lead to complex controller implementation.

Other than these, there are consistency and resiliency issues in a hybrid network. In such a network, where there are diverse devices, ensuring fault-tolerant connectivity and consistent network-wide view is a tough task. However, despite such challenges the hybrid network is envisioned to pave way for deployment of programmable devices in traditional networks or gradually shifting from legacy distributed control to centralized SDN control.

5.3 Hybrid SDN Models

We have extended the classification given by authors in [37, 38] and have provided here a brief overview of various types of hybrid network models.

5.3.1 Service Based Hybrid SDN (S-hSDN)

In S-hSDN, as shown in Fig. 6i, the two networking paradigms coexists and each provide different services. For instance, SDN provides end-to-end tunelling or network function virtualization whereas TN provides other services like MPLS-VPN. Other than these, few services like packet forwarding can be handled collectively by the two networking paradigms.

5.3.2 Traffic Class Based Hybrid SDN (TC-hSDN)

In TC-hSDN, the entire network traffic is segregated into various classes, some are under the control of centralized SDN control plane while as rest are managed by the traditional distributed control. In this approach, both SDN and legacy control spans all forwarding devices and each paradigm controls disjoint set of forwarding table entries in the devices as shown in Fig. 6ii.

5.3.3 Topology Based Hybrid SDN (T-hSDN)

In this approach, the entire network is divided into isolated islands controlled by either of the two domains as shown in Fig. 6iii. As discussed in Sect. 4.3, B4 [92] uses SDN to interconnect backbone data centers whereas non-SDN approach is used

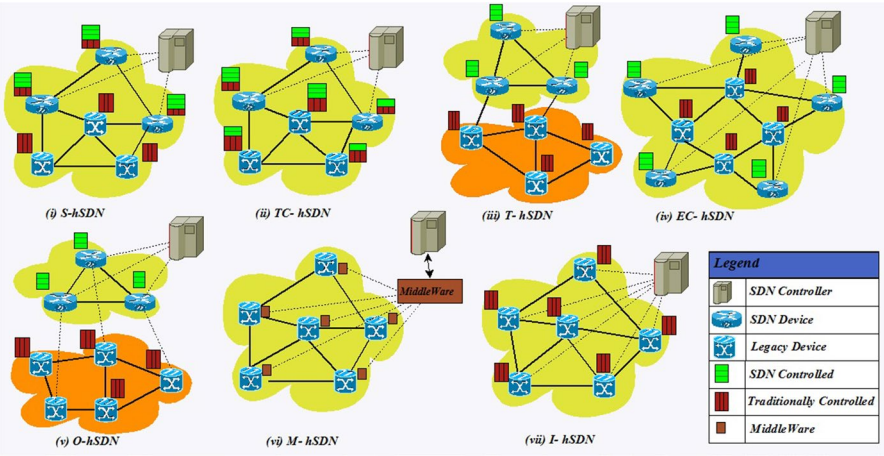

Fig. 6 Hybrid SDN models

to connect to remote data centers and storage servers. In T-hSDN, an organisation in new geographical regions may invest to achieve benefits from SDN and use TN in already existing regions. This approach enables organisations to begin with limited investment in a smaller region, acquire expertise and confidence with the new paradigm and then move to the next region.

5.3.4 Edge Controlled Hybrid SDN (EC-hSDN)

This approach is based on the idea of segregating edge from the core network as proposed by authors in [126]. In EC-hSDN, the programmable nodes are placed at the edge of the network and the SDN controller controls traffic that moves in and out of the organisation. This edge controlled strategy can enable customized routing in the legacy core networks wherein incoming packet's destination IP addresses can be mapped to unused IP addresses [127].

5.3.5 Overlay Hybrid SDN (O-hSDN)

In this approach, an overlay SDN network is built atop of TN. Some chosen legacy devices are replaced by programmable devices to ease traffic management and application of dynamic forwarding policies. In essence, an abstract centralized view of topology is generated by the controller which is used to built overlays over the TN. Big Virtual Switch [128] of Big Switch Networks is an example of O-hSDN.

5.3.6 Middleware Based Hybrid SDN (M-hSDN)

In middleware based approach, a mediator is used either at the legacy switch or on the controller side to have interoperability between the two paradigms. At legacy devices, the mediator may be a hardware agent attached to the device or a software upgrade which can understand the OpenFlow messages sent by the controller. This enables the controller to communicate configuration information to the legacy device and in return can receive flow statistics information from the forwarding device. On the other hand, if middleware is used at the controller end, it will translate the legacy protocol information into controller understandable format and the OpenFlow enabled devices will be handled in the standard manner. Author in [129, 130] have proposed intermediate hardware which extends the centralized control over the legacy devices.

5.3.7 Integrated Hybrid SDN (I-hSDN)

In this model, the SDN controller is responsible for all the network services and controls conventional devices using legacy protocols. The SDN controller can adjust protocol settings like IGP weights and can change selected routes in the legacy routing system. RCP [113] uses I-hSDN model to optimize inter-autonomous system routing.

5.4 Hybrid SDN Controllers

Panopticon [40, 41] proposes a prototype of Hybrid SDN controller which operates in a virtual SDN network comprising of programmable (or SDN) and legacy devices to derive benefits of a pure SDN network. The Panopticon architecture breaks the network into cells and connects such cells by programmable switches which are monitored and controlled by the central controller (like the Panopticon prison architecture where central watch tower monitors the prison cells). The central controller controls the ports of these programmable switches and SDN controlled ports which are on legacy devices lying adjacent to the programmable switches. The controller ensures that traffic flows traverse atleast one programmable switch by using Solitary Confinement Trees (SCTs). The isolation of SCTs is ensured by assigning a distinct VLAN ID to each SCT. This constrained traffic flow is termed as Waypoint Enforcement in Panopticon architecture and it may result in increased path-length for certain traffic flows. Further, the authors have proposed a planner for upgrading legacy switches to SDN switches to maximize benefits with limited cost. The Panopticon architecture is scalable since the VLAN IDs are assigned to SCTs rather than to end-to-end paths. The data plane fault-tolerance in panopticon is achieved by using redundant connectivity in the SCTs. Although this hybrid approach is robust, but it may lead to inconsistent traffic forwarding due to hybrid network control.

Hybnet [124] is a network management framework build on top of Openstack [131] (an open source cloud computing platform) using neutron to control and manage hybrid network devices through virtualization. It maintains a persistent network state view with a topology database and implements a path finder module in the controller that determines end-to-end paths between the host VMs. The controller parses the network operators requests and uses services of path finder module to create virtual links between the end hosts. These virtual link may span multiple OpenFlow switches and legacy devices. The Hybnet controller creates SDN-slices and these slices are mapped to different VLAN IDs. In Hybnet, OpenFlow and NETCONF protocols have been used to control SDN and legacy devices, respectively. HybNET is basically a simple custom module over OpenStack which lacks automatic topology discovery and fault-tolerance measures.

Unlike Panopticon and Hybnet, ClosedFlow [122] extends the centralized control over legacy devices without using SDN devices. The ClosedFlow controller mimic the following four main functions of a pure SDN: (i) *Control Channel between legacy device and controller:* an in-band overlay control channel between controller and switch is implemented using minimum OSPF instance; (ii) *Topology Discovery:* network-wide view at the controller is achieved either by remote login at each legacy device or by running OSPF instance at the controller end as well. In [122], authors have used the first approach to retrieve topology information and to detect link failures; (iii) *Programming Flow Tables:* legacy devices are modified by changing access control lists, route maps and interface configurations to achieve a limited control over the forwarding process in such devices; and (iv) *Packet-In Events:* The packets are forwarded to the controller using either of the following approaches: (a) Using remote login with explicit deny, i.e., if a packet fails to match access control list specified in route map, then the controller is notified by the packet header and

the payload is dropped, (b) specifying a forwarding entry in the legacy device to forward entire packet to controller if there is a no match event. The prototype of ClosedFlow has been implemented with Cisco 3550 switches, but it lacks scalability due to support for limited protocols. Consequent to this limited protocol support and partial control of legacy devices, network management is very difficult in ClosedFlow as compared to pure SDN.

Telekinesis [132] is a hybrid SDN controller which programs forwarding table entries of both SDN and legacy devices. It introduces a new flow control primitive termed as LegacyFlowMod which enables controller to modify flow table entries of a legacy switch. With the help of LegacyFlowMod, Telekinesis influences mac-learning mechanism of layer-2 and whenever new flow data packets are received by a legacy switch, it forwards them to the nearest OpenFlow switch. The controller manages flow table entries with the help of two modules namely path verification and path update. The former checks path feasibility and if feasible, the later instructs the OpenFlow switches to implement the path by making necessary changes in the network. The proposed approach is applicable only in layer 2 networks and may result in topological loops in the network. Further, it does not support other networking protocols and networking features like ACL violations, fine-grained flow control etc. It also faces high latency issue in computing routes and provides low link utilization due to absence of efficient link optimization technique. On the other hand, Magneto [133] proposes a similar but refined approach to overcome networking loops and strives to provide fine-grained flow control over legacy and OpenFlow devices.

The common hurdle that most enterprizes face to shift from TN to SDN paradigm is complex and large configurations of existing networks. Exodus [115] proposed a two stage translation system in which router configurations are first converted into an intermediate form and afterwards transformed into SDN rules which can be inferred by an SDN Controller. With this conversion mechanism, it paves way for translation of TN configurations like packet filtering configurations, ACLs, NAT, VLAN and routing policies into SDN configurations, consequently, making the migration process easy from TN to SDN. The Exodus system consists of IOS parser and compiler, the former generates the intermediate network specification and flowlog libraries which the later converts into SDN rules. The problems with this approach are: (i) it cannot translate complex network functions and; (ii) for each vendor specific device a separate parser/compiler is required. Although this translation based mechanism simplifies the migration from TN to SDN, but it needs further investigation in order to minimize translation delay and handling translation errors.

SDN Hybrid Embedded Architecture (SHEAR) [134] mainly focuses on fault-tolerance and has been evaluated for LANs particularly ethernet networks. In SHEAR, few legacy switches are upgraded to SDN switches which act as observatory points and decompose the network into loop free network fragments. These network fragments are connected using pathlets which are defined interms of VLAN IDs and switch ports. With the help of these pathlets between SDN nodes, the SDN controller responds quickly to out of order routes and configures alternate routes. With only 2 to 10% SDN nodes in the network, authors in [134] have observed that the SHEAR architecture can re-route traffic in less than 3 s in case of a route failure.

The SHEAR framework uses spanning tree protocol to prevent loops in data plane but it may face scalability issue in dense networks.

Vissicchio et al., propose multiple ideas to achieve centralized control over distributed routing [135–138]. In [135], they introduced fake nodes in a legacy network to achieve centralized control over distributed routing. In [136], they have proposed Fibbing controller, a centralized controller which provides flexible network control by performing tasks like traffic steering, load balancing and backup path planning in legacy network by controlling input to legacy devices. This flexible control is achieved by introducing fake nodes into the network through fake LSAs announcing destination reachability. These fake nodes are introduced as per the path requirements given by network operator and at the same time considering the network topology and directed acyclic graph for each destination node.

The authors have evaluated performance of Fibbing controller interns of load, topology expansion and performance gain in [137]. The Fibbing controller introduces very meagre memory and CPU overhead on legacy nodes. The time taken to install thousands of entries in switches and convergence of distributed routing protocols is almost constant. The topology augmentation is achieved by two algorithms viz, simple and merger, the former introduces fake nodes for each destination and the later works in phases. In the primary phase it introduces numerous fake nodes and determines lower and upper bound costs. In the second phase, fake nodes are merged based on the upper and lower bound cost. The controller introduces these fake nodes along the unused links, to increase the overall throughput. In [138], Tilmans et. al. have used Fibbing controller to load balance the on demand delivery of video. They have observed that incase of unexpected congestion in the network, the fibbing controller provides better and quick load balancing. However, there are two major disadvantages associated with the Fibbing controller, first this approach is limited to destination based routing and second it makes network vulnerable to security attacks as any compromised router can inject fake LSAs in the network.

SYMPHONY [139] is a framework which orchestrates SDN's centralized control and legacy distributed control with the help of two main modules namely Legacy Route Server (LRS) and Packet Forwarder. Specifically, these two modules work together to compute end-to-end paths in a hybrid network. The LRS module comprises of a Linux container that executes Quagga [140] routing engine and maintains information of complete network topology. The packet forwarder module is implemented on POX [55] along with path finder module and next-hop module. The POX controller communicates with LRS as if communicating with any host over OVS. The packet forwarder module listens to OpenFlow events like PacketIn, PortStatus etc., and enables the controller to respond to such events. The path finder module uses Shortest Path First algorithm and LLDP module (LLDP generates Switches.txt file that stores network graph information) to compute optimal paths across the network. The next-hop module uses POX and LRS interfacing over OVS to query LRS for optimal edge router to reach a destination node. This edge router is nominated as target by packet forwarder module and installs necessary flow entries on the intermediate OpenFlow nodes to reach to a destination node. Thus, the packet forwarder module replicates the working of a legacy layer-3 router in handling OpenFlow

events. The problem with this framework is that it is limited to layer-3 and does not provide important networking features like ACLs, load balancing, etc.

Authors in [141] have proposed a solution to interoperate two or more SDN islands via legacy network devices. They have proposed LegacyFlow controller that understands the configuration of legacy devices with the help of Switch Control Server (SCS). This SCS accesses the device configuration using Telnet or SNMP and translates it into controller understandable format and in return translates OpenFlow messages into switch configurations. LegacyFlow controller provides resilient connectivity, scalable dynamic flow installation, and incorporates least overhead. However, LegacyFlow controller faces flexibility and complexity issues due to diverse vendor-specific devices with each device using different mechanism for configuration. Another centralized approach to determine and supplement working of BGP protocol is Routing Control Platform [113]. RCP is a logically centralized control architecture segregated from legacy routers to augment IP routing between Autonomous Systems (AS). RCP employs SDN and in particular builds upon RouteFlow [142] to evaluate BGP routes with the help of centralized SDN control. The RCP acts as an indirection layer and optimizes the evaluation of routing information from BGP routers. A major limitation of this approach is that it focuses only on BGP protocol and does not consider other protocols.

Huang et al. propose HybridFlow [143], a lightweight control architecture to control legacy devices with the help of SDN control plane. HybridFlow in essence is an abstraction which abstracts a hybrid SDN network as a logical SDN network and facilitates use of SDN applications to control and operate the logical SDN network like a real SDN network. The HybridFlow controller maintains the logical SDN network by mapping the logical ports of SDN network to the physical ports of actual network. Authors have implemented HybridFlow on POX controller [55] and have determined that it works efficiently and incorporates marginal overhead. Since the authors have implemented HybridFlow as an application on POX controller, so the performance largely depend on the capabilities of POX controller. Further, in a dynamic environment where there are rapid topological changes this simple approach may face scalability and reliability issues.

Another Cloud based hybrid architecture is Meridian [144] which proposes following three logical layers to control and orchestrate cloud networking: (i) *API layer:* provides a declarative and query based interface to the cloud controller which allows it to specify traffic prioritization, traffic destined to middle boxes and in general access control policies; (ii) *Network orchestration layer:* maps the logical entities along with the commands to physical entities; (iii) *Interface layer:* comprises of drivers necessary to interact with legacy devices and SDN switches. This network service model enables network operators to construct, control and manage logical topologies for their complex workloads in the cloud environment. With the help of centralized SDN control, the authors have addressed numerous issues of multi-tenant cloud networking including conflicts in network configuration, optimization of device resources (e.g., flow tables), etc. Since the Meridian framework has been implemented and evaluated using Floodlight [56] controller, so the performance largely depends on the said controller.

Authors in [145] have extended their work [146] of partitioning OSPF networks and connecting them with programmable SDN devices. They have proposed SDNp in [145], a hybrid OSPF/SDN control plane to have centralized SDN control over distributed routing by using SDN or OpenFlow devices to connect OSPF domains. The OpenFlow devices communicate routing information about the OSPF domains to the centralized controller. With this domain routing information, the centralized controller can efficiently steer the inter-domain traffic. The size of the domains defines the degree of centralized controller control and the authors have presented an algorithm for balanced topology partitioning. They have evaluated the performance of pure SDN deployment, OSPF and hybrid SDN/OSPF and have concluded that with few SDN nodes between the OSPF domains, better network control can be achieved. The main objective of this study is to provide a migration path to the organizations in which they can partition their network into various domains and connect such domains with the help of OpenFlow devices. The domains can be iteratively partitioned into smaller sub-domains, resulting in more centralized control with each iteration and efficient traffic forwarding between sub-domains. The proposed approach attempted to achieve best of the two networking paradigms i.e., fault tolerant and reliable connectivity of legacy distributed routing and dynamic programmability of SDN.

5.4.1 Software Defined eXchanges (SDXes)

Some prominent projects have used centralized SDN control at Internet eXchange Points (IXPs), hence introducing the concept of Software Defined eXchanges (SDXes). As depicted in Fig. 7, the SDXes comprises of participant ASs which are connected through an IXP switch controlled by a software based controller. The software-based centralized controller creates opportunities to have innovations in inter-domain peering, rapid change and implementation of peering policies, thus providing effective and promising control over the inter-domain traffic. The main design challenges in SDXes are handling complexity of policy combination in control plane (software platform) and efficient coding methods to reduce the number of forwarding table entries in an Internet Exchange Point (IXP) switch. The most popular SDXes projects are SDX [147, 148] at Princeton, Atlantic-Wave-SDX [149, 150], Google's Cardigan [151] in New Zealand and French TouIX [152]. Here we

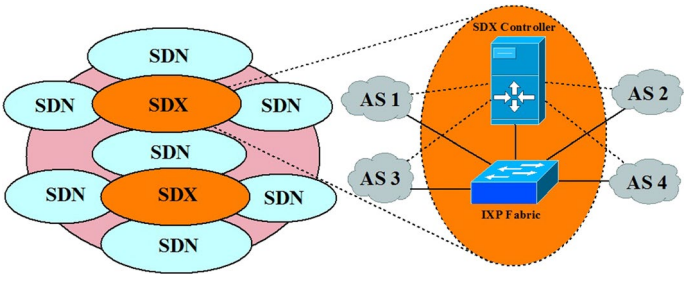

Fig. 7 SDX controller controlling the IXP fabric

have provided an overview of iSDX [148], a latest, most prominent and state-of-the-art SDX controller.

iSDX [148] is a large-scale SDX project based on Ryu [64] controller which has addressed both the aforementioned issues. iSDX proposes independent and parallel compilation of policies of participating peers which results in fast compilation. For consistent inter-domain routing, the participant domains advertise BGP routes and in an industrial-scale IXP there can be around 700 participants which exchange traffic for hundreds of thousands of prefixes. Addditionally, these participants may have different policies for different type of traffic flows and as a result it may lead to intractable forwarding table entries in an IXP switch. To address this challenge, the authors in [148] have proposed an opaque tag which occupies packet's destination MAC address. This opaque tag encodes both the AS information (which advertises the BGP routes for the packet's destination) and next-hop of the packet and therefore, this information can be removed from the IXP switch table. It also prevents BGP route updates to trigger recalculation and recompilation of the forwarding table records. Further, the authors have used matching with arbitrary bitmask feature of OpenFlow 1.3 [153] which can reduce the size of the forwarding table by grouping tags having common bitmasks. They have also suggested partitioning and compression approaches for composition of forwarding policies in an industrial scale operation.

On the other hand, the SDXes can only influence participating ASs which are connected through a software based IXP. At the inter-domain level, the SDX controllers are still logically decentralized as no exchange of information takes place between them apart from reachability. Consequent to it, the SDXes can attain local optima rather than the global one. To address this issue, authors in [123] have proposed Multi-AS Routing Controller which can have a centralized view over multiple ASes and can improve BGP convergence. However, ASes under different administrative control follow a local policy and share only reachability information, so achieving global optima is still a distant dream.

Additionally, SDX controllers face reliability and security issues. SDX controllers are more prone to security attacks as any adversary can attack a single central entity and after gaining access to it can disrupt the inter-domain traffic. Security protocols particularly authentication protocols must be employed for gaining access to the SDX controllers. Authors in [150] have classified SDX architectures into various categories and highlighted the security concerns in each. Layer 2 and layer 3 SDXes inherits vulnerabilities of shared ethernet and BGP, respectively. Further, SDX controller being the central entity is prone to security threats like DDoS attack or malicious access through compromised controller instances or applications running within a controller.

5.5 Insights

In the last few years, many centralized SDN controllers [55, 56, 64] have been extended to operate in a hybrid network. The hybrid SDN controllers either use virtualization, protocol translation mechanisms or extended OpenFlow messages to

interoperate two networking paradigms as shown in Table 3. The Panopticon [40, 41], Hybnet [124], Telekinesis [132] and SHEAR [134] hybrid controllers use virtualization mechanism (VLANs) for hybrid control. The VLAN based control mechanisms conceil the underlying configurations from the end users and provide limited control over diverse devices. Although, VLAN based approach provides an interim solution but face numerous limitations particularly in dense networks. The hybrid controllers which use translation based mechanisms are ClosedFlow [122], Exodus [115], and LegacyFlow [141]. However, these translation based mechanisms may result in issues like misconfigurations, violation of network policies and in the worst case formation of loops in the network. On the other hand, the centralized control at IXPs face complexity and security challenges. All these hybrid SDN controllers along with their main features, objectives, and mechanisms used for hybrid control are summarized in Table 3. It is clear from Table 3 that majority of these hybrid SDN controllers fail to address all the four performance parameters.

6 Future Research Perspectives in SDN Control Plane

The network-wide view at the central SDN controller enables it to optimize various network operations like flow management, load balancing, policy enforcement, etc. The efficient and effective forwarding in SDN data plane mainly depends on performance of software based SDN controller. The selection of algorithms in control software plays a vital role in achieving efficient utilization of network resources. However, using a single centralized SDN controller to perform all these functions faces scalability and reliability issues. The distributed SDN control plane addresses these problems at the cost of interoperability and consistency issues. On the other hand, the hybrid SDN control plane provides an interim solution for upgradation of legacy networks but it also faces robustness and complexity issues due to diverse devices in a hybrid network. In this section, we have highlighted the various issues in SDN control plane and have presented future research perspectives in SDN control plane.

6.1 Monitoring and Measurement Support in SDN Controllers

Network monitoring is a fundamental aspect of network management [154]. The applications in management plane require real-time appropriate network state statistics at diverse aggregation levels. The monitoring frameworks proposed for SDNs include OpenSketch [155], Payless [156], OpenNetMon [157], FlowCover [158] and Probe-SDN [159]. These frameworks either use active polling or sampling to collect monitoring data. Majority of these frameworks are implemented as a separate controller module and have attempted to address a particular monitoring challenge. For instance, OpenNetMon [157] and Probe-SDN [159] focus on tradeoff between monitoring overhead and accuracy. OpenNetMon mainly focuses on traffic engineering whereas Probe-SDN on estimation of bandwidth utilization. On the other hand, for hybrid networks, authors in [160] have provided a limited resource monitoring functionality. However, such solutions are not universal and are specific to a particular

Table 3 Hybrid SDN controllers

Controller	Hybrid model	Management mechanism	Protocols Used	Aim	Performance				Evaluation environment
					Scalability	Consistency	Reliability	Security	
Panopticon [40, 41]	O-hSDN	Virtualization using SCTs and VLAN IDs	OpenFlow	Limited replacement of legacy devices to realize SDN benefits	L	V	L	–	Custom Simulations
Hybnet [124]	O-hSDN	Virtualization using VLAN IDs	OpenFlow and NETCONF	Virtualization and central SDN control	L	G	V	–	Open Stack with Neutron
ClosedFlow [122]	I-hSDN	Remote Logging in legacy devices and VLAN	OSPF	Centralized Control over legacy devices	V	G	L	–	Test Bed using Cisco 3550 ML Switches and custom control application
Telekinesis [132]	O-hSDN	Flow control primitive LegacyFlowMod	OpenFlow	Fine grained flow control in legacy devices using OpenFlow	L	L	L	V	Mininet
Exodus [115]	M-hSDN	Parses and compiles legacy network configurations to SDN rules	OSPF, OpenFlow	Conversion of legacy configurations into controller understandable rules	V	L	V	V	Mininet
SHEAR [134]	O-hSDN	Virtualization using VLAN IDs with hybrid control	OpenFlow, STP, OSPF	Framework for interconnecting legacy devices via programmable nodes for resilient connectivity	V	G	L	V	Ryu controller

Table 3 (continued)

Controller	Hybrid model	Management mechanism	Protocols Used	Aim	Performance				Evaluation environment
					Scalability	Consistency	Reliability	Security	
Fibbing [136]	I-hSDN	Introduced fake nodes in legacy network through fake LSAs to have centralized control	OSPF	Centralized flexible control for traffic steering, load balancing and resilient networking	G	G	G	V	Extended Quagga [170] to implement Fibbing algorithms in Python and OSPF interaction with C
SYMPHONY [139]	O-hSDN	Legacy Route Server executing Quagga routing engine connected with POX Controller	OSPF, LLDP, OpenFlow	Destination based routing with centralized control	L	G	G	V	MiniNExT, POX
LegacyFlow [141]	M-hSDN	Conversion of legacy device configuration into SDN controller understandable format and vice versa	JSON, OSPF	Legacy path for connecting two or more SDN domains via legacy devices	L	L	G	–	GENI
RouteFlow [142]	I-hSDN	BGP based centralized routing with RFCP as indirection layer	BGP, MPLS, OpenFlow	Centralized inter-domain routing	L	G	L	V	NOX, JSON, Quagga
HybridFlow [143]	O-hSDN	Lightweight control plane to control hybrid network using abstractions	OpenFlow and Legacy Protocols	Integration of distributed control with centralized SDN control	V	G	V	V	POX and Mininet

Table 3 (continued)

Controller	Hybrid model	Management mechanism	Protocols Used	Aim	Performance				Evaluation environment
					Scalability	Consistency	Reliability	Security	
Meridian [144]	O-hSDN	Introduces abstractions through multiple layers to handle dynamic network updates, orchestration of network tasks on numerous network devices	OpenFlow	Control and orchestrate cloud networking	V	G	V	V	Floodlight Controller
SDNp [145]	EC-hSDN	Centralized control over inter-domain OSPF traffic by connecting these subdomains with OpenFlow switches	OSPF, OpenFlow	Enabling centralized control by partitioning the OSPF domains and connecting them via programmable switches	G	G	G	–	Simulations using SNDlib [177] library
iSDX [148]	I-hSDN	Independent and parallel processing of policies of various peers and using opaque tags for destination MAC addresses	OpenFlow	Innovations in inter-domain peering, rapid and immediate implementation of peering policies, better control over inter-domain traffic	G	G	–	V	Ryu controller

V Very Limited, *L* Limited, *G* Good

application scenario. Most of these approaches fail to innovate the ways of flow statistics collection from switches and does not address varied application requirements. There is not a single comprehensive monitoring and measurement framework for hybrid networks.

Nowadays, the application domain to be monitored is incessantly changing and growing. With emerging fine-grained monitoring requirements, the monitoring support provided by OpenFlow and SDN controllers is very limited. Currently, the SDN controllers provide elementary features and services for monitoring like cardinal [161] in ODL, OFMon [162] for ONOS, etc. However, to have diverse traffic statistics and real-time view of network statistics available to the applications we believe both controllers and data plane devices need to be extended and modified. In data plane, there should be provisions to define network events and event triggers in order to reduce the bi-directional communication overhead of monitoring data. Further, we believe OpenFlow should be extended in line with IETF's NETCONF [163] protocol which provides efficient mechanisms for statistics collection and network device configuration. Likewise, the controllers need to implement complex data structures and measurement algorithms to handle data plane events and present the collected statistics to applications as per the requirement.

6.2 Automated Network Mangement

Network verification and debugging are important facets of network management. Some approaches have been proposed to verify and debug network operations and configuration for pure SDN including ndb [164] and Veriflow [165]. ndb [164] follows the postcard model wherein switches creates a postcard (which includes switch id, version number and output port) and forwards it to controller's collector module for analysis. The controller captures and recreates the series of events which lead to an irregular behavior. On the other hand, Veriflow [165] performs real time verification of networking events by analysing rules sent by the controller towards the switches. Veriflow generates forwarding graphs and runs queries on the graphs to determine bad or good rule. After observation, either an alarm is raised for a bad rule or good rule is forwarded to the data plane. Such mechanisms involve communication overhead and control latency, therefore network operators may have to trade-off between accuracy and overhead. On the other hand, in hybrid networks, verification and debugging becomes more challenging due to disparity in protocols of two networking paradigms. Till date, to the best of our knowledge, there is not a single tool which can perform verification and debugging in a hybrid network.

Rule or policy update are other critical issues in network management. Upgrades in hardware or software or in general maintenance of forwarding equipment may change the traffic flow pattern in the network. Under such conditions, congestion should be evaded and desired level of services should be retained. Several solutions have been proposed to address such issues including [112, 166, 167] and a detailed survey of these approaches is presented in [51]. However, there are still some open challenges which need to be addressed by the research community:

- Atomic reconfiguration in distributed SDN control plane and hybrid SDN control plane.
- Providing seamless traffic level services in WANs while performing updates in geographically distributed nodes.
- Mechanisms for load balancing and congestion avoidance while performing network updates in pure SDN and hybrid SDN.
- Mechanisms for policy verification while performing the network updates.

6.3 Standard East/Westbound and Northbound Interfaces

The absence of industry recognised standard communication protocols for east/westbound SDN interfaces hampers interoperability between distinct SDN controllers and deployment of logically centralized SDN control plane in large-scale networks [168]. Alongside the concern of interoperability between hetergenous SDN controllers, there is also an urgent need of ensuring interoperability between centralized SDN control and legacy distributed control which can enable organziations to gradually embrace this new radical paradigm. Even though ONF's standardization efforts are underwary but rapid and effective standardization of these interfaces is hampered by heterogeneity in data models of SDN controllers and high performance communication mechanisms required to ensure consistent state exchange among diverse SDN controllers. The state distribution among controllers has to be clandestine, secure and consistent. In addition to it, other issues associated with state information exchange are what and when to exchange between controllers. On the other hand, to have interoperability with legacy distributed control various approaches like virtualization, message translation, etc. are used in different Hybrid SDN controllers. However, to have seamless connectivity and consistent forwarding in hybrid networks a more flexible and scalable approach is required for inter-control plane communication.

Likewise, a standardized northbound interface is very important to hide the heterogeneity in diverse SDN controllers and to facilitate application portability. In this manuscript, we have discussed numerous approaches used in various SDN controllers for northbound interface which can be broadly classified into three categories. First category includes low-level, proprietary APIs tightly coupled with controller platform, written in their native language. Second category uses client–server approach based on REST architecture [169] in which external applications (clients) use services provided by the controller (server). The third category comprises of high level APIs which use indirect way to interact with the controller by using domain-specific programming languages [170–174]. This category raises level of abstraction by allowing programmers to specify high-level network policies in a flexible application development environment. However, such programming languages need to be further extended to provide support for latest OpenFlow versions, code reusability, modularization, and libraries for developers. Further, to express network policies effectively in hybrid networks, a policy language for hybrid networks need to be developed in future research.

Another approach which has gained momentum over the last few years is Intent Based Networking (IBN). Intent-based northbound interfaces allow applications in management plane to specify policy based directives termed as intents. These intents are transformed into forwarding rules and communicated to data plane devices by control plane. With the help of intents, applications in management plane can simply specify the necessary requirements without bothering about implementation and execution of such directives in control plane. Two distributed SDN controllers ONOS [62] and ODL [100] discussed in this manuscript provide limited support for intents. ODL framework is working on Network Intent Composition (NIC) [175, 176], an intent support project which enables external applications to give directives to core ODL modules. Likewise, limited intent support [177] in ONOS allows applications to specify their requirements. IBN simplifies application development and needs to be further enhanced and extended in distributed SDN controllers. Moreover, intent support need to be developed in centralized and hybrid controllers in future research.

6.4 Adaptable SDN Controller

The majority of SDN controllers discussed in this manuscript have attempted to address a particular control plane design issue. For instance, Elasticon [108] proposed a mechanism for controller load balancing by bringing up and down the controller instances as per the load. Likewise, in other controllers [86, 94, 98], the authors have considered a particular issue without addressing other problems like control latency, functional slicing, convergence time, etc. On the other hand, most of the Hybrid SDN controllers can control and manage few legacy devices and fall short to control and manage diverse set of legacy devices. In brief, majority of SDN controllers fail to provide necessary functions and features which can be tuned to address requirements of diverse application areas like multi-tenant data centers, WANs, enterprize networks, hybrid networks, etc.

The ODL [100] project is one such SDN control framework which provides numerous advanced network control functions and features. This project was originally initiated to address requirements of multiple application areas like data center, enterprise and service provider networks. ODL's latest release Magnesium [102] supports wide range of APIs and uses Model-Driven Service Abstraction Layer (MD-SAL) based on YANG models which allows simple and flexible integration of network services requested by the application layer via Northbound APIs. The main focus of various ODL releases is around S3P which stands for stability, security, scalability and performance. The data import/export project termed as Daexin in Magnesium has improved scalability of control cluster and provides support for processing of huge data sets. The two other major projects which have been incorporated into Magnesium are DetNet and Plastic. The former focuses on deterministic networking at layer 3 and time-sensitive networking at layer 2, whereas the later is an intent based facility which performs model-to-model transformations. The DetNet provides features like optimal path calculation, QoS and end-to-end communication flow and service configuration. ODL is in active development process and

provides various control applications with every new release but still needs further research contribution to address requirements of diverse application domains.

6.5 Open Issues in Distributed SDN Control Plane

In addition to the research challenges discussed in Sect. 4.2, there are still some open issues in distributed SDN control plane which need to be addressed by the research community:

- How to determine the number of controllers required in a network?
- How concurrent consistent state sharing and dynamic load balancing can be achieved in distributed SDN controllers?
- Where to place distributed controllers in different network scenarios?
- How distributed routing can be integrated with distributed controllers in a hybrid network?

6.6 Virtualization

Network Function Virtualization (NFV) and SDN highly complement each other, since both advocate creativity, virtualization and automation to realize their respective goals. NFV aims to decouple network functions from specialized hardware whereas SDN segregates network control logic from packet forwarding. The common goal of NFV and SDN is to promote standard network hardware and open software [178]. Currently, numerous approaches are being worked out in the research community to combine these two paradigms to augment either of them [179–184].

The SDN controller can play a vital role in implementing virtual network infrastructure between various virtual network functions (VNFs) and can automate settings according to the changing network requirements. Hence, the centralized SDN control plane can accelerate NFV deployment by providing automation of control operations, flexible re-configuration and provisioning of network connectivity. Numerous controllers have been used to implement SDN/NFV architectures (ODL [180, 181], ONOS [182], Floodlight [183, 184], etc.). However, to control and manage globally distributed varied network resources (IP, MPLS, Optical, etc.) federated and hierarchical controllers must be deployed. To meet the requirements of NFV, the existing SDN controllers need to be improved in terms of scalability, reliability and interoperability [178]. Further, in NaaS, the network hypervisor creates logically-isolated network slices and each virtual slice is under the control of a virtual SDN controller. Such network slices need to be protected from each other and security measures need to be taken into account while designing such virtual SDN controllers.

7 Conclusion

SDN segregates network control logic and forwarding equipment to provide agile, responsive, adaptable and more importantly automated control in next generation networks. The performance of data plane devices mainly depends on the performance of network control logic. In SDN, this network control logic can be implemented either as a physically centralized SDN control plane or physically distributed but logically centralized SDN control plane or even hybrid SDN control plane. In this manuscript, we have discussed all these SDN control plane architectures and research challenges associated with them. Further, we have classified numerous SDN controllers on the basis of their architecture and have analyzed them in terms of following performance parameters: scalability, reliability, consistency and security. The mechanisms used to address the said performance parameters have been examined and the shortcomings associated with each mechanism are highlighted.

The physically centralized SDN controllers try to achieve parallelism not only in event handling but also in event processing to improve efficiency. In spite of it, the centralized SDN controllers face scalability and performance issues. It was observed that majority of these controllers fail to mitigate issues which may arise when a control application contains flaws, malicious logic or vulnerabilities and in the worst case such issues can obstruct the overall control plane operations. However, these controllers (like Floodlight [56], POX [55], Ryu [64], etc.) have been extensively used to design complex distributed SDN controllers (like ONOS [62], Hydra [86], DISCO [63], Ravana [98], etc.). In the last one decade, both academic and corporate sectors have realized that distributed SDN control plane can address the demands of next generation networks and consequently, numerous distributed SDN controllers have been proposed by researchers which either follow a hierarchical or flat architecture. Scalability and reliability are commonly considered as major challenges in physically centralized SDN controllers; however, we have observed that such challenges are also major concern in distributed SDN control plane.

In the distributed SDN control plane, the consistent global network view can be achieved at the cost of high control latency. Strong consistency models require huge communication overhead and increased synchronization, resulting in new scalability challenges. Preserving sturdy consistency during recurrent state updates may obstruct the state progress and it may render network unavailable and can also result in higher switch to control latency. On the other hand, weak consistency models allow simultaneous read/write operations by multiple controller instances and such read/write operations may result in different values from the actual updated values for a short transient period. Consequent to such dissimilar values retrieved by the controller instances, the distributed control plane can have an inconsistent network view which may cause improper application behavior. We have analyzed the consistency mechanisms used by various popular distributed controllers like ONOS [62], ODL [100], Onix [82], DISCO [63], etc., and have observed that consistent global network view across all controller instances is a design challenge which involves trade-off between performance and availability.

Currently, the SDN domains using different control platforms form isolated islands with no interoperability and application portability due to absence of standard interfaces. The interoperability between heterogeneous SDN controllers and control application portability is possible only with standard east/westbound and northbound interfaces, respectively. For wide-spread adoption of this radical networking paradigm, the interoperability among diverse SDN control platforms must be ensured. Further, we have highlighted various other design issues of distributed SDN control plane like number and placement of distributed SDN controllers, load balancing among distributed controllers and criteria for network partitioning. Addressing such challenges is very important to have a robust and efficient distributed SDN control plane as these issues affect the overall network convergence time, consistent global view and control plane response time.

The hybrid SDN networks provide an interim path for SDN adoption as it reduces the initial cost and other technical constraints. Moreover, hybrid networks provide SDN-like control over legacy network infrastructure. We have categorized the hybrid SDN networks into seven models and mapped various hybrid SDN controllers to these models. These hybrid SDN controllers either use virtualization or translation based mechanisms to interoperate the two networking domains. Such simple mechanisms may face scalability and reliability issues in dense or large-scale networks. We believe that an efficient and scalable hybrid SDN controller should be designed in future research which can be tuned to various parameters to manage diverse set of legacy devices in a hybrid SDN network. Further, for performance evaluation and testing of hybrid SDN networks, the simulation/emulation tools need to be developed for such networks.

The scalability, consistency, reliability and security are key factors in designing an efficient and robust SDN controller. In this manuscript, we have highlighted various open issues in SDN control plane which can be future research perspectives for the research community. We have observed that existing SDN controllers lack standard data models, anomaly detection and security mechanisms. We also believe that developing a brand-new SDN controller may not be a best solution; however, the existing SDN control frameworks need to be enhanced, refined and improved to address the said issues.

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Intern. Serv. Appl. **1**(1), 7–18 (2010)
2. Rossi, F.D., Rodrigues, G.D.C., Calheiros, R.N., Conterato, M.D.S.: Dynamic network bandwidth resizing for Big Data applications, In: Proceedings of thirteenth IEEE International Conference on eScience, (2017).
3. Samaan, N., Karmouch, A.: Towards autonomic network management: an analysis of current and future research directions. IEEE Commun. Surv. Tutorials **11**(3), 22–36 (2009)
4. ONF, "Open Networking Foundation", (online) https://www.opennetworking.org/
5. "Software Defined Networking: The New Norm for Networks", ONF White Paper, April 13, (2012).

6. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., et al.: OpenFlow: Enabling Innovation in Campus Networks, In: SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, Mar. (2008).
7. Limoncelli, T. A.: Openflow: a radical new idea in networking, In: ACM Queue, vol. 10, no. 6, (2012).
8. Yu, Y., Li, X., Leng, X., Song, L., Bu, K., Chen, Y., Yang, J., Zhang, L., Cheng, K., Xiao, X.: Fault Management in software defined networking: a survey. IEEE Commun. Surv. Tutorials **21**(1), 349–392 (2018)
9. Open Networking Foundation North Bound Interface Working Group (NBI-WG) Charter; VERSION: V 1.1 (online) https://www.opennetworking.org/images/stories/downloads/working-groups/charter-nbi.pdf
10. Kreutz, D., Ramos, F.M.V., Verissimo, P., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**, 1 (2015). https://doi.org/10.1109/JPROC.2014.2371999
11. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T.: A survey of software defined networking: past, present and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
12. Feamster, N., Rexford, J., Zegura, E.: The Road to SDN: An Intellectual History of Programmable Networks, In: ACM SIGCOMM Computer Communiation Review, vol. 44, no. 2, pp. 87–98, (2014).
13. Jarraya, Y., Madi, T., Debbabi, M.: A survey and a layered taxonomy of software defined networking. IEEE Commun. Surv. Tutorials **16**(4), 1955–1980 (2014)
14. Hu, F., Hao, Q., Bao, K.: A survey on software defined network and openflow: from concept to implementation. IEEE Commun. Surv. Tutorials **16**(4), 2181–2206 (2014)
15. Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T.: Software defined networking: challenges and research opportunities for future internet. Comput. Netw. **75**, 453–471 (2014)
16. Farhady, H., Lee, H., Nakao, A.: Software defined networking: a survey. Comput. Netw. **81**, 79–95 (2015)
17. Gong, Y., Huang, W., Wang, W., Lei, Y.: A survey on software defined networking and its applications. Front. Comput. Sci. **9**(6), 827–845 (2015)
18. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software defined networking. IEEE Commun. Surv. Tutor. **17**(1), 27–51 (2015)
19. Trois, C., Fabro, M.D.D., de Bona, L.C.E., Martinello, M.: A survey on SDN programming languages: toward a taxonomy. IEEE Commun. Surv. Tutor. **18**(4), 2687–2712 (2016)
20. Fonseca, P., Mota, E.: A survey on fault management in software defined networks. IEEE Commun. Surv. Tutor. **19**(4), 2284–2321 (2017)
21. Akyilidiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W.: A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Netw. **71**, 1–30 (2014)
22. Mendiola, A., Astorga, J., Jacob, E., Higuero, M.: A survey on the contributions of software defined networking to traffic engineering. IEEE Commun. Surv. Tutor. **19**(2), 918–953 (2017)
23. Ahmad, I., Namal, S., Ylianttila, M., Gurtov, A.: Security in software defined networks: a survey. IEEE Commun. Surv. Tutor. **17**(4), 2317–2346 (2015)
24. Hayward, S.S., Natarajan, S., Sezer, S.: A Survey of Security in Software Defined Networks. IEEE Commun. Surv. Tutor. **18**(1), 623–654 (2016)
25. Yan, Q., Yu, F.R., Gong, Q., Li, J.: Software-defined networking SDN and Distributed Denial Of Service (DDOS) attacks in cloud computing environments: a survey, some research issues and challenges. IEEE Commun. Surv. Tutor. **18**(1), 602–622 (2016)
26. Dargahi, T., Caponi, A., Ambrosin, M., Bianchi, G., Conti, M.: A survey on the security of stateful SDN data planes. IEEE Commun. Surv. Tutor. **19**(3), 1701–1725 (2017)
27. Kerpez, K.J., Cioffi, J.M., Ginis, G., Goldburg, M., Galli, S., Silverman, P.: Software Defined Access Networks. IEEE Commun. Magaz. **52**(9), 152–159 (2014)
28. Alvizu, R., Maier, G., Kukreja, N., Pattavina, A., Morro, R., Capello, A., Cavazzoni, C., Keller, E.: Comprehensive survey on T-SDN: software-defined networking for transport networks. IEEE Commun. Surv. Tutor. **19**(4), 1701–1725 (2017)
29. Baktir, A.C., Ozgovde, A., Ersoy, C.: How can edge computing benefit from software-defined networking: a survey, use cases & future directions. IEEE Commun. Surv. Tutor. **19**(4), 2359–2391 (2017)

30. Modieginyane, K.M., Letswamotse, B.B., Malekian, R., Abu-Mahfouz, A.M.: Software defined wireless sensor networks application opportunities for efficient network management: a survey. Comput. Electr. Eng. **1**, 274–287 (2018)
31. Bera, S., Misra, S., Vasilakos, A.V.: Software defined networking for internet of things: a survey. IEEE Internet Things J. **4**(6), 1994–2008 (2017)
32. Thyagaturu, A.S., Mercian, A., McGarry, M.P., Reisslein, M., Kellerer, W.: Software defined optical networks (sdons): a comprehensive survey. IEEE Commun. Surv. Tutor. **18**(4), 2738–2786 (2016)
33. Karakus, M., Durresi, A.: A Survey: Control Plane Scalability Issues and Approaches in Software-Defined Networking. Computer Networks **112**, 279–293 (2017)
34. Bannour, F., Souihi, S., Mellouk, A.: Distributed SDN control: survey, taxonomy, and challenges. IEEE Commun. Surv. Tutor. **20**(1), 333–354 (2018)
35. Blial, O., Ben Mamoun, M. and Benaini, R.: An Overview on SDN Architectures with Multiple Controllers. J. Comput. Netw. Commun. vol. 2016, Hindawi, Article ID 9396525, (2016).
36. Hu, T., Guo, Z., Yi, P., Baker, T., Lan, J.: Multi-controller based Software-Defined Networking: A Survey. IEEE Access **6**, 15980–15996 (2018)
37. Vissicchio, S., Vanbever, L., Bonaventure, O.: Opportunities and research challenges of hybrid software defined networks. ACM SIGCOMM Comput. Commun. Rev **44**(2), 70–75 (2014)
38. Sinha, Y., Haribabu, K.: A survey: Hybrid SDN. J. Netw. Comput. Appl. **100**, 35–55 (2017)
39. Khan, S., Gani, A., Wahab, A.W.A., Guizani, M., Khan, M.K.: Topology discovery in software defined networks: threats, taxonomy, and state-of-the-art. IEEE Commun. Surv. Tutor. **19**(1), 303–324 (2016)
40. Canini, M., Feldmann, A., Levin, D., Schaffert, F., Schmid, S.: Software-defined networks: incremental deployment with panopticon. IEEE Computer **47**(11), 56–60 (2014)
41. Levin, D., Canini, M., Schmid, S. and Feldmann, A.: Panopticon: Reaping the Benefits of Partial SDN Deployment in Enterprize Networks. In: the Proceedings of USENIX Annual Technical Conference, (2014).
42. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: NOX: Towards an Operating System for Networks", Computer Communication Review, (2008).
43. Benson, T., Akella, A., Maltz, D.A.: Network Traffic Characteristics of Data Centers in the Wild. In: Proceedings of the tenth ACM SIGCOMM Conference on Internet Measurement, New York, USA: ACM, pp. 267–280. (2010).
44. Michel, O., Keller, E.: SDN in Wide Area Networks: A Survey. In: the Proceedings of Fourth International Conference on Software Defined Systems (SDS) , pp. 37–42, May, (2017).
45. Voellmy, A. Wang, J. C.: Scalable Software Defined Network Controllers. In: Proceedings of ACM SIGCOMM conference on Applications, Technologies, Architectures and Protocols for Computer Communication, pp. 289–290, (2012).
46. Ul-Huque, M.T.I., Si, W., Jourjon, G., Gramoli, V.: Large Scale Dynamic Controller Placement. IEEE Trans. Netw. Serv. Manag. **14**(1), 63–76 (2017)
47. Heller, B., Sherwood, R., McKeown, N.: The Controller Placement Problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, New York, USA: ACM, pp. 7–12, (2012).
48. Macapuna, C.A., Rothenberg, C.E., Maurício, M.F.: In: Packet Bloom Filter-Based Data Center Networking with Distributed OpenFlow Controllers", in the Proceedings of IEEE GLOBECOM Workshops, pp. 584–588, (2010).
49. Mattos, D.M.F., Duarte, O.C.M.B., Pujolle, G.: Reverse update: a consistent policy update scheme for software defined networking. IEEE Commun. Lett. **20**(5), 886–889 (2016). https://doi.org/10.1109/LCOMM.2016.2546240,May
50. Mahajan, R., Wattenhofer, R.: On Consistent Updates in Software Defined Networks. In: The Procedings of ACM Symposium on SDN Research, Santa Clara, CA, April 3–4, (2017).
51. Foerster, K.T., Schmid, S., Vissicchio, S.: Survey of Consistent Software-Defined Network Updates. https://arxiv.org/abs/1609.02305, (2018).
52. Tavakoli, A., Casado, M., Koponen, T., Shenker, S.: Applying NOX to the Data Center. In: Proceedings of Ninth ACM SIGCOMM workshop on Hot Topics in Networks, Oct-(2009).
53. Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R.: On Controller Performance in Software-Defined Networks. In: Proceedings of the Second USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, (2012).

54. Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., Smeliansky, R.: Advanced Study of SDN/OpenFlow Controllers. In: Proceedings of the Ninth Central & Eastern European Software Engineering Conference in Russia, New York, USA: ACM, pp. 1–6, (2013).
55. POX. (online) https://www.noxrepo.org/pox/about-pox/
56. Floodlight Project. (online) https://www.projectfloodlight.org/
57. Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S.: Devoflow: Scaling Flow Management for High-Performance Networks", in Proceedings of the ACM SIGCOMM Conference, New York, NY, USA: ACM, pp. 254–265, (2011).
58. Ilyas, Q., Khondoker, R.: "Security analysis of FloodLight, ZeroSDN, beacon and POX SDN controllers" L. Notes Netw. Syst. **30**, 85–98 (2018)
59. Cai, Z., Cox, A.L., Ng, T.S: Maestro: A System for Scalable Open Flow Control", (2011).
60. Dhawan, M., Poddar, R., Mahajan, K., Mann, V.: Sphinx: Detecting Security Attacks in Software Defined Networks. In: The proceedings of Network and Distributed System Security (NDSS) Symposium. The Internet Society, ISBN 1–891562–38-X, (2015).
61. Zhu, L., Karim, M.M., Sharif, K., Li, F., Du, X., Guizani, M.: SDN Controllers: Benchmarking and Performance Evaluation. arXiv:1902.04491v1. [cs. NI] (2019).
62. Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., Parulkar, G.: ONOS: towards an open, distributed SDN OS. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, New York, USA: ACM, pp. 1–6, (2014).
63. Phemius, K., Bouet, M., Leguay, J.: DISCO: Distributed Multi-Domain SDN Controllers", CoRR, vol. abs/1308.6138, (2013).
64. Ryu SDN Framework. (online) https://ryu-sdn.org/
65. MUL SDN Contoller. (online) https://www.openmul.org/
66. TREMA SDN Controller Framework. (online) https://trema.github.io/trema/
67. Erickson, D.: The Beacon Openflow Controller. In: The Proceedings of Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, New York, USA: ACM, pp. 13–18, (2013).
68. OSGi Specification. (online) https://www.osgi.org/developer/what-is-osgi/
69. Lee, B., Park, S.H., Shin, J., Yang, S.: IRIS: The Openflow-based Recursive SDN controller. In: Proceeding of Sixteenth IEEE International Conference on Advanced Communication Technology,DOI: https://doi.org/10.1109/ICACT.2014.6779154, 16–19 Feb. (2014).
70. Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., Kang, B.B.: Rosemary: A Robust, Secure and High-Performance Network Operating System. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, pp. 78–89, (2014).
71. Arbettu, R.K., Khondoker, R., Bayarou, K., Weber, F.: Security Analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN Controllers. In: the Proceedings of Seventeenth IEEE International Telecommunications Network Strategy and Planning Symposium (Networks), Sept. (2016).
72. Song, P., Liu, Y., Liu, C., Qian, D.: ParaFlow: fine-grained parallel SDN controller for large-scale networks. J Netw. Comput. Appl. (2017). https://doi.org/10.1016/j.jnca.2017.03.009
73. Panda, A., Scott, C., Ghodsi, A., Koponen, T., Shenker, S.: CAP for networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, New York, USA-ACM, pp. 91–96, (2013).
74. Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A.: Logically centralized? State Distribution Trade-offs in Software Defined Networks. In: Proceedings of the first workshop on Hot topics in Software Defined Networks, pp. 1–6, (2012).
75. Sridharan, V., Gurusamy, M., Truong-Huu, T.: On multiple controller mapping in software defined networks with resilience constraints. IEEE Commun. Lett. **21**, 1763–1766 (2017)
76. Fonseca, P., Bennesby, R., Mota, E., Passito, A.: Resilience Of SDNs based on Active and Passive Replication Mechanisms. In: Proceedings of Global Communications Conference, pp. 2188–2193, (2013).
77. Fonseca, P., Bennesby, R., Mota, E., Passito, A.: A Replication Component for Resilient Openflow-Based Networking", in the Proceeding of Symposium on Network Operations and Management, pp. 933–939 , (2012).
78. Botelho, F., Bessani, A., Ramos, F.M., Ferreira, P.: On the Design of Practical Fault-Tolerant SDN Controllers. In: Third European Workshop on Software Defined Networks, pp. 73–78, Sept (2014).
79. Spalla, E.S., Mafioletti, D.R., Liberato, A.B., Ewald, G., Rothenberg, C.E., Camargos, L., Villaca, R.S., Martinello, M.: Ar2c2: Actively Replicated Controllers for SDN Resilient Control Plane. In:

The Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) , pp. 189–196, April (2016).
80. Yazici, V., Sunay, M.O., Ercan, A.O.: Controlling a Software-Defined Network via Distributed Controllers. CoRR, vol. abs/1401.7651, (2014).
81. Oktian, Y.E., Lee, S., Lee, H., Lam, J.: Distributed SDN controller system: a survey on design choice. Comput. Netw. **121**, 100–111 (2017)
82. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Shenker, S.: Onix: A Distributed Control Platform for Large-Scale Production Networks. In: Proceedings of the Ninth USENIX Conference on Operating Systems Design and Implementation, CA, USA: USENIX Association, pp. 1–6, (2010).
83. Wallin, S., Wikstrom, C.: Automating Network and Service Configuration Using Netconf and Yang. In: Proceedings of the Twenty Fifth International Conference on Large Installation System Administration, Berkeley, CA, USA, pp. 22–22, (2011).
84. OpenConfig. (online) https://www.openconfig.net/
85. OF-CONFIG 1.2: Openflow Management and Configuration Protocol, ONF, Tech. Report, (2014).
86. Chang, Y., Rezaei, A., Vamanan, B., Hasan, J., Rao, S., Vijaykumar, T.N.: Hydra: Leveraging Functional Slicing for Efficient Distributed SDN Controllers, arXiv preprint arXiv:1609.07192.
87. He, M., Basta, A., Blenk, A., Kellerer, W: Modeling Flow Setup Time for Controller Placement in SDN: Evaluation for Dynamic Flows. In: Proceedings of IEEE International Conference on Communications (ICC), pp. 1–7, (2017).
88. Zhang, T., Giaccone, P., Bianco, A., De Domenico, S.: The Role of the Inter-Controller Consensus in the Placement of Distributed SDN Controllers. Comput. Commun. **113**, 1–13 (2017)
89. Alhazmi, K., Moubayed, A., Shami, A.: Distributed SDN Controller Placement Using Betweenness Centrality & Hierarchical Clustering. In: The Proceedings of Eighth ACM Symposium on Design and Analysis of Intelligent Vehiclular Networks and Applications, Canada Oct 28-Nov 2, (2018).
90. Dacier, M.C., Konig, H., Cwalinski, R., Kargl, F., Dietrich, S.: Security challenges and opportunities of software-defined networking. IEEE Secur. Priv. **15**(2), 96–100 (2017)
91. Qi, C., Wu, J., Hu, H., Cheng, G., Liu, W., Ai, J., Yang, C. An Intensive Security Architecture with Multi-Controller for SDN" in Proceedings of IEEE International Conference on Computer Communications Workshops, pp. 401–402, (2016).
92. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J.: B4: Experience with a Globally Deployed Software Defined WAN. In: Proceedings of the ACM SIGCOMM Conference on SIGCOMM, New York, USA, ACM, pp. 3–14. (2013).
93. Chandra, T.D., Griesemer, R., Redstone, J.: Paxos Made Live: An Engineering Perspective. In: Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing, New York, USA: ACM, pp. 398–407, (2007).
94. Hassas Yeganeh, S., Ganjali, Y.: Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, USA, ACM, pp. 19–24, (2012).
95. McCauley, J., Panda, A., Casado, M., Koponen, T., Shenker, S.:. Extending SDN to Large-Scale Networks. In: Proceeding of ONS, (2013).
96. Yap, K.K., Motiwala, M., Rahe, J., Padgett, S., Holliman, M., Baldus, G., Hines, M., Kim, T., Narayanan, A., Jain, A., Lin, V.: Taking the Edge Off With Espresso: Scale, Reliability and Programmability for Global Internet Peering", in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, New York, USA, ACM, pp. 432–445, (2017).
97. A. Bianco, P. Giaccone, S. D. Domenico, and T. Zhang, "The Role of Inter-Controller Traffic for Placement of Distributed SDN Controllers", CoRR, vol. abs/1605.09268, (2016).
98. Katta, N., Zhang, H., Freedman, M., Rexford, J.: Ravana: Controller Fault-Tolerance in Software Defined Networking", in Proceedings of the First ACM SIGCOMM Symposium on Software Defined Networking Research, pp. 4:1–4:12, (2015).
99. Hunt, P., Konar, M., Junqueira, F.P., Reed, B.: Zookeeper: Wait-Free Coordination for Internet-Scale Systems. In: Proceedings of the Annual USENIX Technical Conference, (2010).
100. OpenDayLight Project. (online) https://www.opendaylight.org/
101. Benamrane, F., Benaini, R: Performances Of Openflow-Based Software Defined Networks: An Overview. J. Netw. 10(6): 329–337, (2015).

102. Magnesium. (online) https://www.opendaylight.org/what-we-do/current-release/magnesium, March (2020).
103. Akka Framework. (online) https://akka.io/
104. Ongaro, D., Ousterhout, J.: In Search of an Understandable Consensus Algorithm. In: Proceedings of USENIX Annual Technical Conference, CA, USA, pp. 305–320, (2014).
105. Clark, D.D., Partridge, C., Ramming, J.C., Wroclawski, J.T.: A Knowledge Plane for the Internet. In: The Proceedings of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, USA-ACM, pp. 3–10, (2003).
106. Shin, J., Kim, T., Lee, B., Yang, S.: IRIS-HiSA: highly scalable and available carrier-grade SDN controller cluster. Mobile Netw. Appl. **22**(5), 894–905 (2017). https://doi.org/10.1007/s11036-017-0853-6,Oct
107. Johns, M.: Getting Started with Hazelcast. Packt Publishing Ltd (2013).
108. Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V., Kompella, R.R.: ElastiCon: An Elastic Distributed SDN Controller. In: Proceedings of Tenth ACM/IEEE symposium on Architectures for networking and communications systems, Los Angeles, California, USA, pp. 17–28, October 20 - 21, (2014).
109. Hyperic SIGAR API. (online). https://www.hyperic.com/products/sigar.
110. Fu, Y., Bi, J., Gao, K., Chen, Z., Wu, J., Hao, B.: Orion: A Hybrid Hierarchical Control Plane of Software-Defined Networking for Large-Scale Networks. In: The Proceedings of Twenty Second IEEE International Conference on Network Protocols, Raleigh, NC, USA, 21–24 Oct. (2014).
111. Dijkstra, E.W.: Dijkstra: A Note on Two Problems in Connexion With Graphs. Numer. Math. **1**(1), 269–271 (1959)
112. Hong, C.Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., Wattenhofer, R.: Achieving high utilization with software-driven WAN. In: Proceedings of the ACM SIGCOMM Conference on SIGCOMM, New York, USA-ACM, pp. 15–26, (2013).
113. Rothenberg, C.E., Nascimento, M.R., Salvador, M.R., Corrêa, C.N.A., Cunha de Lucena, S., Raszuk, R.: Revisiting routing control platforms with the eyes and muscles of software-defined networking. In: Proceedings of ACM Workshop on Hot Topics in Software Defined Networks, pp. 13–18, (2012).
114. Gamperli, A., Kotronis, V., Dimitropoulos, X.: Evaluating the effect of Centralization on Routing Convergence on a hybrid BGP-SDN Emulation Framework. In: ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pp. 369–370, Oct. (2014).
115. Nelson, T., Ferguson, A.D., Yu, D., Fonseca, R., Krishnamurthi, S.: Exodus: Toward Automatic Migration of Enterprise Network Configurations to SDNs. In: The Proceedings of ACM SIGCOMM Symposium on Software Defined Networking Research, pp. 1–7, (2015).
116. Vissicchio, S., Vanbever, L., Cittadini, L., Xie, G.G., Bonaventure, O: Safe update of hybrid SDN Networks", IEEE/ACM Transactions on Networking, vol. 25, no. 3, pp. 1649–1662, June (2017).
117. Ochoa Aday, L., Cervelló Pastor, C., Fernández Fernández, A.: Current Trends of Topology Discovery in Openflow-based Software Defined Networks. (online) https://upcommons.upc.edu/handle/2117/77672
118. Pakzad, F., Portmann, M., Tan, W.L., Indulska, J.: Efficient Topology Discovery in Software Defined Networks. In: Proceedings of Eighth IEEE International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1–8. (2014)
119. Pakzad, F., Portmann, M., Tan, W.L., Indulska, J.: Efficient Topology Discovery in Openflow-Based Software Defined Networks. In: ACM SIGCOMM Computer Communication Review, vol. 77, no. C, 52–61, March (2016).
120. Saha, A.K., Sambyo, K., Bhunia, C.: Topology Discovery, Loop Finding and Alternative Path Solution in POX Controller. In: the Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 2, pp. 553–557, (2016).
121. Hong, D. K., Ma, Y., Banerjee, S., Mao, Z. M.: Incremental Deployment of SDN in Hybrid Enterprise and ISP Networks. In: The Proceedings of Symposium on SDN Research , USA, March (2016).
122. Hand, R., Keller, E.: ClosedFlow: OpenFlow like Control over Proprietary Devices. In: The Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ACM, pp. 7–12, (2014).
123. Kotronis, V., Gamperli, A., Dimitropoulos, X.: Routing Centralization across Domains Via SDN: A Model and Emulation Framework for BGP Evolution. In: Computer Networks, vol. 92, pp. 227–239, (2015)

124. Lu, H., Arora, N., Zhang, H., Lumezanu, C., Rhee, J., Jiang, G.: Hybnet: Network Manager for a Hybrid Network Infrastructure. In: Proceedings of the Industrial Track of the Thirteenth ACM/IFIP/USENIX International Middleware Conference, (2013).
125. Katiyar, R., Pawar, P., Gupta, A., Kataoka,: Auto-configuration of SDN Switches in Sdn/Non-SDN Hybrid Network. In: The Proceedings of the Asian Internet Engineering Conference, ACM, pp. 48–53, (2015).
126. Casado, M., Koponen, T., Shenker, S., Tootoonchian, A.: Fabric: A Retrospective on Evolving SDN, In: The Proceedings of the First Workshop on Hot topics in Software Defined Networks, ACM, pp. 85–90, (2012).
127. Mishra, A., Bansod, D., Haribabu, K.: A Framework for Openflow-Like Policy-Based Routing in Hybrid Software Defined Networks. In: The Proceedings of Eleventh International Network Conference, pp. 97–102, (2016).
128. Big Virtual Switch Network Virtualization with the Open SDN Architecture. (online) https://www.bigswitch.com/sites/default/files/sdnresources/bvsdatasheet.pdf
129. D. J. Casey and B. E. Mullins, "SDN Shim: Controlling Legacy Devices," in the Proceedings of IEEE Conference on Local Computer Networks, pp. 169–172, (2015).
130. Parniewicz, D., Doriguzzi Corin, R., Ogrodowczyk, L., Rashidi Fard, M., Matias, J., Gerola, M., Fuentes, V., Toseef, U., Zaalouk, A., Belter, B., Jacob, E., et al.: Design and Implementation of an OpenFlow Hardware Abstraction Layer. In: The Proceedings of the ACM SIGCOMM workshop on Distributed cloud computing, pp. 71–76, (2014).
131. OpenStack, (online) https://www.openstack.org/
132. Jin, C., Lumezanu, C., Xu, Q., Zhang, Z.L., Jiang, G.: Telekinesis: Controlling Legacy Switch Routing with Openflow in Hybrid Networks. In: Proceedings of ACM SIGCOMM Symposium on Software Defined Networking Research, pp. 1–7, (2015).
133. Jin, C., Lumezanu, C., Xu, Q., Mekky, H., Zhang, Z. L., Jiang, G.: Magneto: Unified Fine-Grained Path Control in Legacy and Openflow Hybrid Networks. In: Proceedings of ACM Symposium on SDN Research, pp. 75–87, (2017).
134. Markovitch, M., Schmid, S.: SHEAR: A Highly Available And Flexible Network Architecture Marrying Distributed and Logically Centralized Control Planes. In: Proceedings of IEEE International Conference on Network Protocols (ICNP) , pp. 78–89, (2015).
135. Vanbever, L., Vissicchio, S.: Enabling SDN in Old School Networks with Software-Controlled Routing Protocols. In: The Proceedings of the Open Networking Summit, (2014).
136. Vissicchio, L., Vanbever, J.: Rexford, "Sweet little lies: Fake Topologies for Flexible Routing" , in Proceedings of the Thirteenth ACM Workshop on Hot Topics in Networks, ACM, (2014).
137. Vissicchio, S., Tilmans, O., Vanbever, L., Rexford, J.: Central control over distributed routing. ACM SIGCOMM Comput. Commun. Rev. **45**(4), 43–56 (2015)
138. Tilmans, O., Vissicchio, S., Vanbever, L., Rexford, J.: Fibbing in action: on demand load-balancing for better video delivery. In: Proceedings of ACM SIGCOMM Conference on SIGCOMM, pp. 619–620, (2016).
139. Chemalamarri, V.D., Nanda, P., Navarro, K.F.: SYMPHONY: A Controller Architecture for Hybrid Software Defined Networks. In: The Proceedings of European Workshop on Software Defined Networks (EWSDN) , pp. 55–60, (2015).
140. GNU Quagga Project, (online) https://www.quaggaproject.org
141. Farias, F., Salvatti, J., Victor, P., Abelem, A.: Integrating Legacy Forwarding Environment to Openflow/SDN Control Plane. In: The Proceedings of Asia-Pacific Network Operations and Management Symposium (APNOMS) , pp. 1–3, (2013).
142. Vidal12, A., Verdi, F., Fernandes, E.L., Rothenberg, C.E., Salvador, M.R.: Building upon RouteFlow: A SDN Development Experience. In: The Proceedings of Thirty First Simpsio Brasileiro de Redes de Computadores (SBRC) , pp. 879–892, May (2013).
143. Huang, S., Zhao, J., Wang, X.: HybridFlow: A Lightweight Control Plane for Hybrid SDN in Enterprise Networks. In: The Proceedings of IEEE/ACM Twenty Fourth International Symposium on Quality of Service (IWQoS) , pp. 1–2, (2016).
144. Banikazemi, M., Olshefski, D., Shaikh, A., Tracey, J., Wang, G.: Meridian: An SDN Platform for Cloud Network Services. In: IEEE Communications Magazine, vol. 51, no. 2, pp. 120–127, (2013).
145. Caria, M., Jukan, A., Hoffmann, M.: SDN Partitioning, A Centralized Control Plane for Distributed Routing Protocols, In: IEEE Transactions on Network and Service Management, (2016).

146. Caria, M., Das, T., Jukan, A., Hoffmann, M.: Divide and Conquer, Partitioning OSPF networks with SDN, In: IFIP/IEEE International Symposium on Integrated Network Management (IM), 11–15 May, (2015).
147. Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S.P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., Katz-Bassett, E.: SDX: A Software Defined Internet Exchange, In: ACM SIGCOMM, (2014).
148. Gupta, A., MacDavid, R., Birkner, R., Canini, M., Feamster, N., Rexford, J., Vanbever, L.: An Industrial-Scale Software Defined Internet Exchange Point. In: Thirteenth USENIX Symposium on Networked Systems Design and Implementation (NSDI), (2016).
149. Morgan, H.: Atlanticwave-sdx: A Distributed Intercontinental Experimental Software Defined Exchange for Research and Education Networking, Press Release, (2015).
150. Chung, J., Cox, J., Ibarra, J., Bezerra, J., Morgan, H., Clark, R., Owen, H.: Atlanticwave-sdx: An International SDX To Support Science Data Applications," in Software Defined Networking (SDN) for Scientific Networking Workshop, Austin, Texas, (2015)
151. Stringer, J., Pemberton, D., Fu, Q., Lorier, C., Nelson, R., Bailey, J., Corrêa, C.N., Rothenberg, C.E.: Cardigan: SDN Distributed Routing Fabric Going Live at an Internet Exchange. In: IEEE Symposium on Computers and Communications, Madeira, Portugal, pp. 1–7, (2014).
152. Lapeyrade, R., Bruyère, M., Owezarski, P.: Openflow-based Migration and Management of the TouIX IXP. In: IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, pp. 1131–1136, April 25–29, (2016).
153. OpenFlow Switch Specification Version 1.3. (online):https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.
154. Kim, J., Sim, A., Suh, S.C., Kim, I.: An Approach to Online Network Monitoring using Clustered Patterns. In: Proceedings of the International Conference on Computing, Networking and Communications,Silicon Valley, USA, 26–29, pp. 656–661, (2017).
155. Yu, M., Jose, L., Miao, R.: Software Defined Traffic Measurement with Opensketch. In: The Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI), vol. 13, pp. 29–42, (2013).
156. Chowdhury, S.R., Bari, M.F., Ahmed, R., Boutaba, R.: Payless: A Low Cost Network Monitoring Framework for Software Defined Networks. In: The Proceedings of IEEE Network Operations and Management Symposium (NOMS) , pp. 1–9, (2014).
157. Van Adrichem, N.L., Doerr, C., Kuipers, F.A.: OpenNetMon: Network Monitoring in Openflow Software Defined Networks. In: The Proceedings of IEEE Network Operations and Management Symposium (NOMS) , pp. 1–8, (2014).
158. Su, Z., Wang, T., Xia, Y., Hamdi, M.: FlowCover: Low-Cost Flow Monitoring Scheme in Software Defined Networks. In: Proceedings of the Global Communications Conference, Austin, TX, USA, pp. 1956–1961, (2015).
159. Henni, D.E., Hadjaj-Aoul, Y., Ghomari, A.: Probe-SDN: A Smart Monitoring Framework for SDN-Based Networks. In: The proceedings of the Global Information Infrastructure and Networking Symposium, France, pp. 1–6, (2017).
160. Choi, T., Kang, S., Yoon, S., Yang, S., Song, S., Park, H.: SuVMF: Software-Defined Unified Virtual Monitoring Function for SDN-Based Large-Scale Networks. In: Proceedings of ACM International Conference on Future Internet Technology, pp. 1–6, (2014).
161. Cardinal: OpenDaylight Monitoring as a Service, (Online), https://docs.opendaylight.org/en/stable-fluorine/user-guide/cardinal_opendaylight-monitoring-as-a-service.html#cardinal-architecture
162. Kim, W., Li, J., Hong, J.W.K., Suh, Y.J.: OFMon: OpenFlow Monitoring System in ONOS Controllers. In: Proceedings of IEEE NetSoft Conference and Workshops (NetSoft), (2016).
163. Enns, R., Bjorklund, M. Schoenwaelder, J.: NETCONF Configuration Protocol. RFC 6241, (2011)
164. Handigol, N., Heller, B., Jeyakumar, V., Maziéres, D., McKeown, N.: Where is the Debugger for my Software-Defined Network? In: The Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks, pp. 55–60, (2012)
165. Khurshid, A., Zhou, W., Caesar, M., Godfrey, P.: Veriflow: verifying network wide invariants in real time. ACM SIGCOMM Comput. Commun. Rev. **42**(4), 467–472 (2012)
166. Jin, X., Liu, H.H., Gandhi, R., Kandula, S., Mahajan, R., Zhang, M., Rexford, J., Wattenhofer, R.: Dynamic scheduling of network updates. In: ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pp. 539–550, (2014).

167. Hu, S., Chen, K., Wu, H., Bai, W., Lan, C., Wang, H., Zhao, H., Guo, C.: Explicit Path Control in Commodity Data Centers: Design and Applications. In: IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2768–2781, (2016).
168. Yu, H., Li, K., Qi, H., Li, W., Tao, X.: Zebra: An East-West Control Framework For SDN Controllers. In: The Proceedings of Forty Fourth IEEE International Conference on Parallel Processing, (2015).
169. Richardsonan L., Ruby S.: RESTful web services, O'Reilly Media Inc. , (2008).
170. Kim, H., Reich, J., Gupta, A., Shahbaz, M., Feamster, N., Clark, R.: Kinetic: Verifiable Dynamic Network Control. In: Proceedings of USENIX NSDI, Oakland, CA, USA, 59–72. 2015.
171. Voellmy, A., Kim, H., Feamster, N.: Procera: A Language for High Level Reactive Network Control. In: Proceedings of the First Workshop on Hot topics in Software Defined Networks, ACM, (2012).
172. Reitblatt, M., Canini, M., Guha, A., Foster, N.: FatTire: Declarative Fault Tolerance for Software Defined Networks. In: Proceedings of the Second Workshop on Hot Topics in Software Defined Networks, ACM, (2013).
173. Foster, N., Harrison, R., Freedman, M.J., Monsanto, C., Rexford, J., Story, A. Walker, D.: Frenetic: A Network Programming Language. In: The Proceedings of SIGPLAN, (2011).
174. Layeghy, S., Pakzad, F., Portmann, M.:SCOR: Constraint Programming-Based Northbound Interface for SDN, In: Twenty Sixth International Telecommunication Networks and Applications Conference (ITNAC) , pp. 83–88, Dec 2016.
175. Network Intent Composition (NIC) Developer Guide (Online). https://docs.opendaylight.org/ en/stable-oxygen/developer-guide/network-intent-composition-(nic)-developer-guide.html
176. Rodrigues,Y.: OpenDaylight ODL: Network Intent Composition (NIC) - A real Intent-based solution, challenges and next steps Intent Framework, (online) https://www.serro.com/opendaylight-network-intent-composition-a-real-intent-based-solution-challenges-and-next-steps/
177. "Intent Framework," (Online). https://wiki.onosproject.org/display/ONOS/Intent+Framework
178. Bonfim, M.S., Dias, K.L., Fernandes, S.F.:Integrated NFV/SDN Architectures: A Systematic Literature Review. In: ACM Computing Surveys, Vol. 51, No. 06, (2019).
179. Moyano, R.F., Fernández, D., Merayo, N., Lentisco, C.M., Cárdenas, A.: NFV and SDN-Based Differentiated Traffic Treatment for Residential Networks, IEEE Access, (2020).
180. Martínez, R., Mayoral, A., Vilalta, R., Casellas, R., Muñoz, R., Pachnicke, S., Szyrkowiec, T., Autenrieth, A. Integrated SDN/NFV orchestration for the dynamic deployment of mobile virtual backhaul networks over a multilayer (packet/optical) aggregation infrastructure. In: IEEE/OSA Journal of Optical Communications and Networking Vol. 9, No. 2, (2017).
181. Saridis, G.M., Peng, S., Yan, Y., Aguado, A., Guo, B., Arslan, M., Jackson, C., Miao, W., Calabretta, N., Agraz, F., Spadaro, S., et al.: Lightness: a function-virtualizable software defined data center network with all-optical circuit/packet Switching. J. Lightwave Technology **34**, 7 (2016)
182. Schulz-Zander, J., Mayer, C., Ciobotaru, B., Schmid, S, et al.:. Unified Programmability of Virtualized Network Functions and Software-Defined Wireless Networks, In: IEEE Transactions on Network and Service Management (2017).
183. Schulz-Zander, J., Mayer, C., Ciobotaru, B., Schmid, S, et al.:. OpenSDWN: Programmatic Control over Home and Enterprise WiFi. In: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). ACM, New York, NY, USA, (2015).
184. Cheng, G., Chen, H., Hu, H., Wang, Z., Lan, J.: Enabling network function combination via service chain instantiation. Comput. Netw. **92**(2), 396–407 (2015)

Suhail Ahmad received B.E degree in Computer Science & Engineering from the University of Kashmir, India, in 2008 and the M. Tech degree in Communication and Information Technology from National Institute of Technology, Srinagar, Jammu & Kashmir in 2011. He has been working as Assistant Professor in Department of Computer Science and Engineering, University of Kashmir since 2012. Currently pursuing Ph.D. at NIT Srinagar and his research interests include Network Security, MPLS, Network Programming and Software Defined Networking.

Ajaz Hussain Mir is Professor in Electronics and Communication Department, National Institute of Technology, Srinagar, Jammu & Kashmir. He received the Ph. D and M. Tech degree from IIT-Delhi. His current research interests include Network Security, Mobile Networks, IOT, Next Generation Networks, Software Defined Networking and Image Processing.