Computer Networks 192 (2021) 108047

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey paper

Software defined networking architecture, traffic management, security, and placement: A survey

Madhukrishna Priyadarsini *, Padmalochan Bera

Indian Institute of Technology, Bhubaneswar, India

ARTICLE INFO

Keywords:
SDN
Network function virtualization
Energy efficiency
Load balancing
SDN security
Controller placement
OpenFlow protocols

ABSTRACT

Software-defined networking (SDN) is the potential deployment platform for the future Internet and enterprise networks. In SDN, the logical control (programs) and the forwarding logic (data) are separated. Therefore, it allows to configure the network parameters, routing policies effectively based on the requirements of the environment. The control plane (set of controllers) implements network functions such as, load balancing, energy-efficient routing, security. The data plane (set of switches) receives the forwarding logic from control plane and forwards the packets in the network. This in turn enhances the traffic management performance of the network, and strengthens the security perimeter. The researchers have worked on various directions towards the enhancement of network control functions, performance, and security. However, none of those talks about traffic management challenges in integration with security, energy efficiency and performance dependencies.

This survey presents an extensive study, analysis and report of state-of-the-art works on effective traffic management including load balancing and energy-efficient routing, SDN control implementation and deployment architecture, controller security and optimal controller placement that affect traffic management. Moreover, this survey discusses a few unexplored problems in SDN such as scalable deployment, integration with legacy network, and future research road maps to these problems. This will help researchers understanding different theoretical, experimental, and applied research in SDN, challenges in these problems; developing effective solutions; and designing future research road maps.

1. Introduction

Software-defined networking (SDN) is a modern-era network technology that allows effective management of heterogeneous network. Traditional network architecture is limited in satisfying the growing demand of implementing heterogeneous applications with real time communications requirements [1]. Traditional network supports specific policies implemented during the build time of the devices and thereby limits flexibility in dynamic configuration of the network parameters [2].

Fig. 1 shows generic views of TCP/IP and SDN network architecture. The SDN architecture separates the control plane from the data and thereby makes the network function implementation hardware independent. The interfacing between different planes are realized using a set of protocols under the OpenFlow standard. For example, data and control plane communication is controlled using northbound protocol. Thus, SDN can accommodate network devices (switches) from any vendor that supports the OpenFlow protocol standard. The SDN architecture consists of three layers namely; data plane, control plane, application plane, and two primary functional components namely; controllers and switches. The controllers generate the flow rules (instructions for packet processing and network management) and the switches forward the traffic according to these flow rules [3]. The communication between controller and switches is governed by a set of OpenFlow protocols. In contrast to traditional networks, SDN architecture only requires software (controller function and protocol) up-gradation in case of changes in network environments. This makes SDN platform cost-effective and simple for deployment in real-life network implementations. SDN provides flexibility to the programmer in developing control and application planes in any high-level languages. It allows updating network parameters on the fly based on changes in requirements [4] which in turn enhances the performance and robustness of the network. Moreover, SDN supports developing different network functions such as load balancing, energy-efficient routing, security analysis using network function virtualization for effective traffic management in the network [5,6].

One of the limitations of SDN Architecture is its heavy dependency on the controller. This may lead to various problems. Firstly, the SDN

* Corresponding author.
E-mail addresses: mp18@iitbbs.ac.in (M. Priyadarsini), plb@iitbbs.ac.in (P. Bera).

https://doi.org/10.1016/j.comnet.2021.108047
Received 26 June 2020; Received in revised form 14 December 2020; Accepted 23 March 2021
Available online 31 March 2021

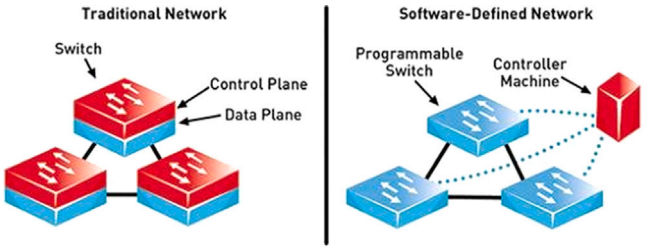

Fig. 1. Example topology showing difference between TCP/IP and SDN Network.

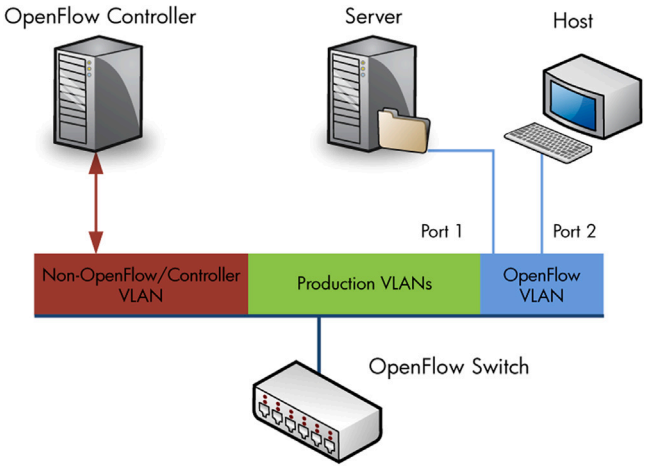

Fig. 2. OpenFlow connection architecture.

platform may potentially suffer from different network server-based attacks such as spoofing, intrusion, DoS, policy anomalies [7]. Secondly, there may be a possibility of traffic congestion in the controller. Moreover, there is a generic problem of high energy consumption by different network devices [8]. Finally, a limited number of controllers with uneven association with the switches may introduce performance problems for large-scale networks. In general, all these problems significantly affect traffic management in the network. Therefore, to utilize the benefits of SDN architecture, there is a need for developing solutions to address the above problems. An efficient load balancing technique may solve the traffic congestion problem in the controller and thereby can enhance the traffic management process. On the other hand, the introduction of sleep-active mode in the device functioning may significantly reduce energy consumption and thereby control carbon emission to the environment. Finally, it is necessary to develop dynamic security enforcement solutions to proactively detect and prevent various attacks. Also, solutions to place the controller in the network are required after the implementation of the above-mentioned network functions.

Existing survey articles lacks in covering extensive study and analysis of traffic management functions in SDN [2,3,9], advancements in SDN implementation and deployment architectures, affects of security and controller placement on traffic management. Although there exist survey articles on SDN security and controller placement, they do not discuss its dependencies with traffic management. In addition, there is no article explaining designed energy-efficient solutions for traffic management and load balancing in SDN. This motivates our current survey. The main contribution of the paper includes:

(1) We present a detailed study on SDN implementation architecture covering its evolution from OpenFlow and network function virtualization (NFV). The working procedure of each layer and the functional components (network devices) are presented along with the flow rule generation process.
(2) We discuss the traffic management challenges in SDN mainly in the form of load balancing and energy-efficient routing. Then, we detailed the state-of-the-art research in these directions.
(3) We present a detailed study on various security threats in SDN and the research directions to counter these threats.
(4) We formally state the controller placement problem in SDN. Also, we present the research insights to solve this problem considering different network and traffic management constraints.
(5) We present detailed theoretical, experimental, and analytical comparison of state-of-the-art works in all the above directions. We also discuss the dependencies of these problems with effective traffic management in SDN. Finally, we discuss various open and current research challenges in SDN for its effective realization and deployment in various applications.

The remainder of this paper is organized as follows. Section 2 discusses the SDN evolution from OpenFlow and NFV, its basic architecture, and the flow rule generation procedure. The traffic management challenges are described in Section 3 with their potential solutions. In Section 4, we present the current security threats of SDN and its countermeasures. The controller placement problem in SDN is described in Section 5 with the methodologies to solve the problem. Section 6 presents open challenges of SDN platform such as reliability and scalability with certain application challenges. Finally, we conclude the paper in Section 7.

2. Evolution of SDN

SDN has evolved as an emerging network platform since 2000 with different technologies such as OpenFlow and network function virtualization. The two major characteristics of SDN are (i) separation between data and control plane which is acquired from telephone network systems; and (ii) requirement driven control of tasks which are taken from different network models such as Tempest, capsule model, virtual network infrastructure, etc. Due to the lack of hardware support and available implementation infrastructure, these concepts were not integrated as an efficient networking platform. Since 2008 with the development of OpenFlow protocols, a significant amount of research has been performed in the field of SDN. In the following subsections, we highlight the backbone technologies of SDN.

2.1. Introduction to OpenFlow and Network Function Virtualization (NFV)

2.1.1. OPenFlow architecture

OpenFlow is a specification managed by the Open Networking Foundation (ONF) [10], which defines the functions and protocols used to manage network switches through a centralized controller. It was initiated by Martin Casado et al. at Stanford University during 2005–2009. OpenFlow is a command and control protocol that includes communication over SSL/TLS protected channels; feature discovery and configuration of devices by the controller, and managing the forwarding tables on the switches. The OpenFlow protocol is designed and managed by the network administrators and vendors for configuring and managing network devices [11]. The protocol allows the controller to guide the switches on handling incoming packets. The control instructions are generally structured as “flow rules”. Each flow contains the following fields < header match fields, flow priority, counters, packet processing instructions, flow timeouts, cookies>. An OpenFlow switch maintains a set of flow rules in a flow table. A packet is forwarded in the network based on the flow rules in the switches’ flow table. Fig. 2 presents the OpenFlow connection architecture. It consists of three types of network components such as;

- One or more switches (The switches may be physical or virtual)
- One or more controllers
- One or more applications/application servers

The controller implements the OpenFlow protocol for establishing communications with the switches. It provides a northbound API to the applications. The northbound API allows the applications to read the state of the network and to control the network to perform different tasks. These components establish communication in three possible ways: OpenFlow, Non-OpenFlow, and production VLANs. Some OpenFlow applications may require the only partial deployment of OpenFlow switches, whereas others require a network consisting of only OpenFlow switches.

2.1.2. Network Function Virtualization (NFV)

In late 2012, the initiative of NFV has been taken, involving more than 28 networking companies and over 150 technology providers from various telecommunication industries across the globe. NFV aims to reduce equipment costs and decrease time to market while attaining scalability, dynamic configuration, and a strong ecosystem between network providers, vendors, and users. By virtualization of network functions (generally implemented in dedicated hardware), network operators expect to achieve greater agility and acceleration in-service deployments while reducing both operational (OpEx) and capital costs (CapEx) [12]. In addition, NFV is intended to optimize the overhead incurred due to deployment functions (such as traffic management, firewalls, DNS, load balancers, etc.). Implementation of NFV requires dynamic network connectivity both in physical and virtual layers to interconnect network function endpoints (virtual machines).

NFV Framework:

Network function virtualization framework consists of three elements; those are described as follows:

(1) Virtualized Network Function (VNF): The virtualized network functions comprise of the software used to create the various network functions in its requested format. In general, multiple virtual machines are created that implement different network functions. The VNFs are then synthesized in the hardware, i.e., the Network Function Virtualization Infrastructure.
(2) Network Function Virtualization Infrastructure (NFVI): NFVI is a computing platform that consists of core hardware and software logics for implementation of VNFs. NFVI is platform and OS independent and can be realized in any geographic location which allows operators to access the infrastructure flexibly [13].
(3) Network Functions Virtualization Management and Orchestration (NFV-MANO): NFV-MANO consists of various functional blocks that enable storage, access, and exchange of information for operating the network correctly with improvements in efficiency and performance.

In summary, NFVI and NFV-MANO together help in managing and monitoring devices, recovering from failures, and providing effective security.

2.2. The SDN architecture

The SDN architecture has been designed for the development and execution of agile and cost-effective network functions and applications. One of the key emphasis behind SDN architecture is providing an easily configurable platform with simple troubleshooting methods. This recommends the centralization of network intelligence by separating the forwarding process of network packets (data packets) from the routing process (control process) [14]. In general, SDN is characterized by seven fundamental traits: data-control separation, simplified devices, centralized control, network automation and virtualization, vendor-independence, programmable platform, and openness. Here, we provide an overview of the basic components of a software-defined networking system, their functions, operations, and the process of communication between these components [2].

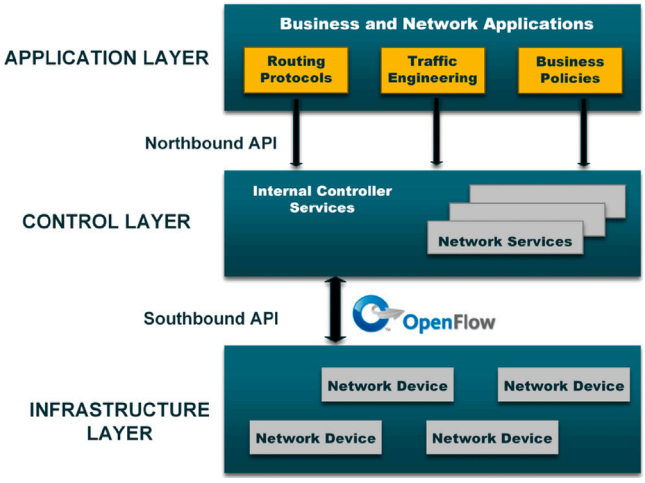

Fig. 3. SDN architecture consists of three different layers.

2.2.1. Layered architecture

The fundamental characteristic of SDN is the separation of the forwarding/data plane and the control plane. Fig. 3 shows the basic architecture of SDN, which consists of three layers. The bottom or first layer is the *infrastructure layer*, which is also called a data plane. It comprises of the traffic/data forwarding network elements. The responsibilities of the data plane are mainly data forwarding; dropping or replicating of data, as well as monitoring local network information and collecting flow statistics. Data plane communicates to the control plane if it does not possess forwarding information for a packet and subsequently control plane generates flow information for such a packet. On the other hand, if the packet forwarding rule/flow rule pertains to multicast, the packet is replicated in the data plane before forwarding the various copies through different output ports [3]. In summary, the data plane constitutes the basic distribution topology of the SDN network.

The second layer is called the *control layer* or the control plane. Configuration of the forwarding plane is realized in the control plane. The control plane determines the flow tables and forwarding logic in the data plane. Many of these protocols and algorithms require global knowledge of the network. The control plane determines how the generation of the forwarding tables and logic in the data plane should be programmed. It makes use of the information provided by the forwarding plane and defines network operation and routing-logic. It comprises one or more software controllers (which are logically centralized) that communicate with the forwarding devices (switches) through the standardized interface, the southbound interface [9].

The application layer hosts network applications for introducing new network features, such as security policy and control, network service quality, that in turn assist control plane in configuring the network with these application requirements. The application layer receives an abstract and global view of the network from the controller and uses it to provide recommendations in the form of different application policies or rules. The interface between the application layer and the control layer is referred to as the northbound interface [15].

2.2.2. Communication protocols

The most common southbound interface is OpenFlow [10], which is standardized by the Open Networking Foundation (ONF). OpenFlow is a protocol that describes the interaction of one or more control servers with OpenFlow-compliant switches. An OpenFlow controller installs a flow table in the switches so that these switches forward traffic as per the entries in the flow table. The flow rules are of different types as classified by match fields such as bit_offset, length, the pattern is similar to IP access control lists (ACLs) and may contain wildcards.

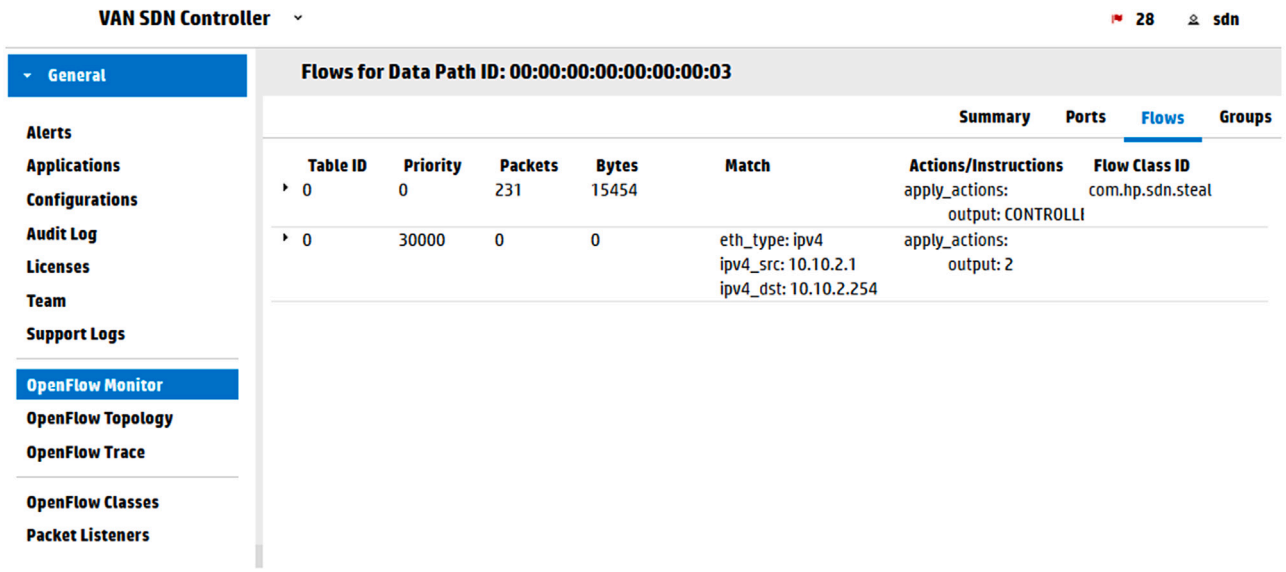

Fig. 4. An example of flow table.

The OpenFlow protocol provides an interface that allows a control software to program switches in the network. In general, the controller can change the forwarding behavior of a switch by altering the rules in the forwarding table. Controllers often provide a similar interface to the applications, which is called the *northbound interface*, to expose the programmability of the network. The northbound interface is not standardized and often allows fine-grained control of the switches. However, applications do not require the details of the southbound interface, e.g., an application does not need to know the details about the network topology and related parameters.

2.2.3. SDN forwarding devices and functionalities

An SDN device (OpenFlow switch) is comprised of an API for communication with the controller, a topology abstraction layer, and a packet-processing function. In the case of a virtual switch, this packet-processing function is packet-processing software. On the other hand, in the case of a physical OpenFlow switch, the packet-processing function/logic is embedded in the hardware. The packet-processing logic consists of the mechanisms to take actions based on the results of evaluating different fields in the header of the incoming packets and on finding the highest-priority match. When a match is found with the flow table rules, the incoming packet is processed locally unless it is explicitly forwarded to the controller. When there is no match, the packet may be forwarded to the controller for further processing. In the case of a software switch, these functions are mirrored by the software while in case of hardware switch, they are programmed in the hardware logic [3].

The flow table is the primary data structure in an SDN device that consists of a set of flow rules. The flow tables allow the switch to evaluate incoming packets and take appropriate actions based on packet headers. Incoming network packets are evaluated based on some fields in the header such as length, pattern, offset. The actions may include forwarding the packet to a specific port, dropping the packet, or flooding the packet on all ports, etc. An SDN device is not fundamentally different from a network switch except that the basic operation of an SDN switch has been rendered more generic and more programmable through the flow tables and the associated logic. The Flow tables consist of a number of prioritized flow rules, each of which typically consists of two components: a set of match fields and actions. An incoming packet is compared against the match fields in priority order, and the first complete match is selected. Subsequently, the network device takes the corresponding action specified in the flow rule. The flow table and flow entry constructs used in flow tables and flow rules provides application developer a wide range of possibilities for designing packet matching s/w or h/w logic. An example of the flow table is shown in Fig. 4.

A number of SDN device implementations are available today, both commercial and open source. At present, there are two major alternatives: Open vSwitch(OVS) from Nicira, Indigo from Big Switch. In addition, network equipment manufacturers such as Cisco, HP, NEC, IBM, Juniper, and Extreme, have added OpenFlow support to some of their legacy switches.

2.2.4. SDN operation

The three major functional components of SDN are the SDN switches, the controller, and the application layer. The SDN switches implement forwarding logic for determining actions on incoming packets. An SDN switch contains a data structure called a flow table which is a collection of a set of flow rules or flow entries. When a switch receives a packet, it checks its flow tables for a match. In case of a match, it takes appropriate configured action which usually entails forwarding the packet. If it does not find a match, the switch can either drop the packet or pass it to the controller. The SDN controller is responsible for creating an abstract view of the topology and presenting it to the running applications. The controller allows the SDN applications to define flows on switches and to manage the network functions to respond to the packets that are forwarded to the controller. Since one controller can control a large number of network devices, the forwarding rules are calculated and operated on a high-performance machine with less latency [1]. Fig. 5 shows the generic functioning of SDN.

The SDN applications are built on top of the controller. It is to be noted that these applications are different from the traditional OSI network model. The functional implementations of SDN applications cover characteristics of the network layer and data layer (not application layer) of the OSI model. The SDN applications are interfaced with the controller; use that interfaces to set proactive flows on the switches, and to receive packets that have been forwarded to the controller. When the application starts, the proactive flows are set by application and these flows persist till there are no changes in the configuration. This type of flow is called static flows. Another type of proactive flow is configured by the controller based on the current load in the device. In addition, some flow rules are defined in response to a packet forwarded to the controller. On receipt of incoming packets that have been forwarded to the controller, the SDN application sets the effective policies for the controller, and if appropriate, establishes/configures new flows in the

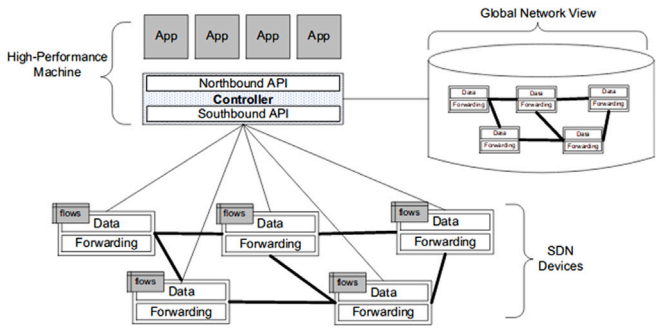

Fig. 5. SDN operation overview.

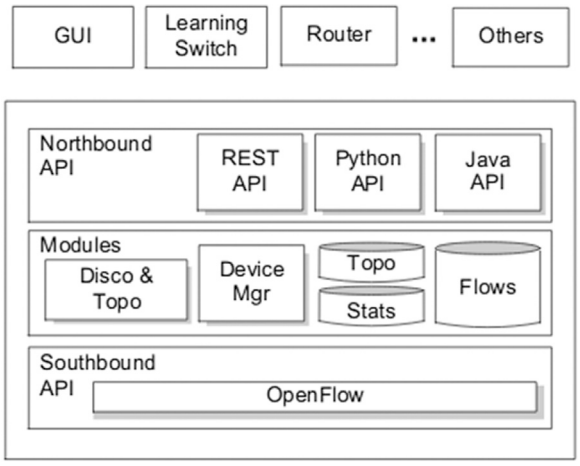

Fig. 6. SDN controller architecture.

switch. Next time, the switch responds locally when it receives a packet and is called reactive flows [3].

The controller maintains a view of the entire network, receives network functions for traffic management, implements policy decisions, controls SDN devices in the infrastructure layer, and provides a northbound API for applications. Here, we refer implementation of policy decisions as routing, forwarding, packet redirecting, load balancing, etc. The controllers often bundled with their own set of common application modules, such as a learning switch, a router, a basic firewall, and a simple load balancer [16].

Fig. 6 shows the modules that cover the core functionalities of the controller. It also shows a northbound API and a southbound API, and a few example applications that might use the controller. The core features in the controller include:

- End-user device discovery: Discovery of end-user devices such as laptops, desktops, printers, mobile devices, and so on.
- Network device discovery: Discovery of network devices that comprise the infrastructure of the network, such as switches, routers, and wireless access points.
- Network device topology management: Maintaining information about the interaction between the network devices to each other and to the end-user devices to which they are directly attached.
- Flow management: Maintaining a database of the flows being managed by the controller and perform all necessary coordination with the devices/switches to ensure synchronization of the device flow entries with that database.

The core functions of the controller are implemented by a set of software modules as shown in Fig. 6. These modules need to maintain local databases containing the current topology and flow statistics. The controller dynamically extracts the network topology by learning of the presence of switches (SDN devices) and end-user devices and checking the connectivity between them. It maintains a flow cache that mirrors the flow tables on the various switches it controls. The controller locally maintains the per-flow statistics as collected from the switches [16]. The controller functions are implemented through pluggable modules, and the features are designed according to the network requirements.

Existing SDN controllers:

There are a number of SDN controllers available today. They include both open source SDN controllers and commercial SDN, controllers. Open source SDN controllers come in many forms of implementations starting from basic C-language controllers such as NOX [17] to python-based controller POX [18], and to Java-based versions such as Beacon [19], Floodlight [20], OpenDayLight [21]. Also, there is a Ruby-based controller called Trema [22]. Interfaces to these controllers may be offered in the language in which the controller is written or other alternatives, such as REST or Python. Vendors such as NEC, IBM, and HP offer controllers that are primarily OpenFlow implementations. In summary, OpenFlow controller architecture is more prevalent for extensions and offering new features and automation. Whereas, the proprietary controllers are more network or application-specific and offer less programmability support.

In the next section, we present the traffic management challenges in SDN and the proposed methodologies to solve those issues.

3. Traffic management in SDN

SDN has evolved as an emerging network platform due to its key feature of effectively managing and monitoring heterogeneous traffic with less configuration and performance overhead. However, as the SDN platform has experimented with a large volume of traffic with various environmental constraints, researchers and network engineers found various challenges towards effective traffic management. Here, we cover two such challenges, namely, load balancing and energy-efficient routing. The first part of this section covers the load balancing problem and related works to counter this problem. The second part presents the energy consumption of the network infrastructure and the devices and its impact on traffic routing and network performance parameters. We also discuss the research works in the direction of reducing energy consumption in traffic management.

3.1. Load balancing and its implementation in SDN

With the growth in network size and usages of network applications, it is a challenge to the network industries to come up with efficient technologies and platforms for implementing large-scale heterogeneous network applications [23]. The SDN platform provides flexibility to configure networks for complying with heterogeneous applications. But, failure of managing a large volume of traffic potentially creates a load imbalance in the controllers which in turn degrades network performance and utilization, e.g., packet reordering, end-to-end delay, increase of latency, etc [24]. It also limits end-users to access the required services. Therefore, it is important to distribute the traffic load evenly among the controllers through appropriate network paths.

In this survey, we present the state-of-the-art works on control plane load balancing only. Controller is responsible for creating flow rules and the data plane switches only forward the data according to the generated flow rules. The controller also considers the resources available in the data plane, suitable routing protocols, available link bandwidth, capacity of each forwarding devices while generating the flow rules. This is why, control plane load balancing is more relevant for discussion than data plane load balancing. Here, we discuss the research work on control plane load balancing mainly in two directions. The first subsection discusses the works done on load balancing in traditional networks along with highlighting the differences of this

problem in the SDN platform. On the other hand, the second subsection presents the state-of-the-art load balancing for SDN. In addition, we summarize the survey on load balancing with our observations and analysis.

3.1.1. Load balancing in Traditional Networks

The load balancing problem in traditional networks mainly covers server and link load balancing. In this area, different technologies such as NGINX [25], HAProxy [26] have been proposed which used hardware and software implementations based on layer interactions. Islam et al. [27] proposed a water-aware workload management algorithm that controls data centers' long-term water consumption by exploiting Spatio-temporal diversities of water efficiency and dynamically dispatching workloads among distributed data centers. Liang et al. [28] investigated the problem of developing a geographical load balancing (GLB) scheme for distributed Internet data centers (IDCs). They imposed proper limits on the workloads allocated to the IDC locations with high price-sensitivity coefficients so that the total cost is decreased. Saurav et al. [29] proposed a distributed algorithm that saves resources such as the time and energy of mobile devices and improves overall system performance.

However, the traditional network does not adopt dynamic changes in network states, which makes the realization of load balancing difficult [13]. On the other hand, the SDN controller manages the network resources dynamically as per the user's requirements and network topology. Therefore, due to the flexibility in programming and configuring SDN controllers, the SDN platform can be deployed as a load balancer in traditional network applications.

3.1.2. Load balancing in SDN

The load balancing solutions in SDN are broadly classified into two categories: (i) centralized load balancing, and (ii) distributed load balancing. Table 1 presents the comparison of various load balancing techniques that have been proposed for SDN implementation.

Centralized load balancing:

In the centralized load balancing approach, a single controller is responsible for balancing the load [30]. It periodically collects load from other controllers, informs the overloaded controller to transfer some of its load to a lightly loaded controller. However, this approach does not scale well for large systems since the central controller has to make all the decision making. The system is not reliable in the sense that the failure of the central controller stops the load balancing process altogether.

Distributed load balancing:

In the distributed load balancing approach, a threshold for the load is defined for each controller based on the hardware capacity of the controller, and load balancing is not required until the traffic load crosses that value [31]. When the load in a controller crosses its threshold, it collects load from others, and then the controller may initiate a load balancing process. In this case, each controller balances its load in collaboration with other controllers through message exchanges. Yu et al. [32] proposed a load informative strategy where a load scale from zero to threshold is divided into many segments. The load is informed only if the previously informed load and the current load lies in two different segments in the load scale. But when the load approaches towards the threshold, the load is continuously informed to other controllers which require high bandwidth. In SMDM based load balancing [33], a trade-off between switch migration cost and load balancing rate is maintained. However, the migration at the source controller leads to another load migration at the target one, which causes performance degradation. In the SMCLBRT load balancing scheme [34], multiple controllers migrate their switches simultaneously depending on the response time of the target controller. In the first phase, highly loaded controllers find lightly loaded controllers for switch migration, and in the second phase, all the selected switches are migrated. This scheme consumes high network bandwidth and introduces congestion due to simultaneous switch migration. A load-balancing scheme proposed by Zhou [35] explores the migration of a group of switches while reducing the number of decisions taken by the overloaded controllers. But it increases load balancing time and migration cost, which effectively reduces the performance of the network. Li et al. [36] proposed a fuzzy synthetic evaluation mechanism for path load-balancing. It balances traffic and avoids unexpected breakdowns caused by link failure. However, it increases the utilization and reliability of network paths.

An OpenFlow-based dynamic load-balancing strategy for data center networks is proposed by Trestian [37]. Zhong et al. [38] proposed a load-balancing based on server response time. It achieves a better load-balancing effect in comparison with the traditional round-robin scheme and random allocation scheme. Kandoo [39], ONOS [40] implement distributed controller architecture for a cluster of SDN nodes to improve the reliability, scalability of the control plane. However, they experience a high bandwidth requirement, which leads to performance degradation. A hybrid approach for the hierarchical design of the control plane is proposed by Fu et al. [41], where the scalability is improved considering large scale scenarios. Recently, Priyadarsini et al. [42] proposed a self-adaptive load balancing scheme that addresses the aforementioned limitations by adjusting the threshold dynamically. Here, load balancing procedure is completed through five components, namely, *Load measurement component* that monitors the load on every controller; *Load broadcast component* that is responsible for broadcasting the load to different controllers in the network; *Load balancing component* that checks different load balancing conditions under variable load in the network; *Load migration component* that migrates the load of highly loaded source controllers to lightly loaded controllers as identified in the load balancing component; *Link reset component* resets the controller-switch links involved in the migration to their initial state. This scheme provides a low packet drop rate while ensuring a high throughput.

In summary, centralized load balancing is not scalable for large-scale heterogeneous applications. Moreover, it suffers from reliability in case of controller failures. On the other hand, distributed load balancing is proven to be more scalable satisfying specific performance objectives. The major factors to select a load balancing techniques are network size, latency requirements, variation in traffic types, link quality (failure prone), convergence requirements etc. Therefore, it is recommended to deploy adaptive load balancing strategies which are capable of effectively satisfying the above requirements.

In addition, there exists another direction of research, where link load balancing is given equal priority as controller load balancing. Authors in [43], proposed a load-balancing algorithm using the remaining bandwidth of the bottleneck link and the average link remaining bandwidth in the path for SDN-based data centers. A score is given for each path according to the link load in the path and the path with the largest score is selected to transfer the traffic. The traffic is redistributed periodically to different paths that can avoid the occurring of heavy load link. In paper [44], the authors proposed a hybrid approach combining server and link load balancing for multi-path routing in distributed storage systems. They used a process as call on-demand inverse multiplexing which calculates the overall throughput and resource usage. Authors in [45], formulated the load-balancing routing for both links and controllers (LBR-LC) problem in an SDN, and proved its NP-hardness. They proposed a rounding-based algorithm to solve this problem, and also analyzed the approximation performance. Moreover, they discussed the efficient mechanism for network status maintenance among distributed controllers.

3.2. Energy-efficiency in SDN

Today, a large number of heterogeneous applications are being executed in the backbone network of any organization or publicly

Table 1
Load balancing approaches in SDN.

Load balancing approaches	*Contributions*	*Applications*	*Proposed techniques*
Centralized	A single controller collects load from others and balances load of highly loaded controllers	1- Cloud computing 2- Defense networks	Centralized load balancing [30]
Distributed	Each controller informs others when their load exceeds defined threshold values	1- Cloud computing 2- Data centers	Threshold-based load balancing [31]
	The load is informed when the current load differs from previous load	1- Cloud computing 2- Data centers	Load informative strategy [32]
	A trade-off between switch migration cost and load balancing rate is maintained	1- Enterprise networks 2- Social networks	SMDM [33]
	Controllers migrate their load simultaneously depending on the response time of target	1- Defense networks 2- Social networks	SMCLBRT [34]
	Reduces the number of decisions taken by the overloaded controllers	1- Cloud computing 2- Data centers	Load balancing scheme [35]
	Balances traffic and avoids unexpected breakdowns caused by link failure	1- Cloud computing 2- Edge computing	Fuzzy synthetic scheme [36]
	Dynamically migrates load for data center networks	1- Edge computing 2- Fog computing	Dynamic load balancing [37]
	Migrates load considering server response time	1- Data centers 2- Social networks	Load balancing scheme [38]
	Improves scalability, reliability of control plane	1- Cloud computing 2- Enterprise networks	Distributed load balancing [39], [40]
	Improves scalability in large scale networks	1- Cloud computing 2- IoT networks	Hybrid load balancing [41]
	Dynamically adjusts threshold value for load increment and decrement	1- Cloud computing 2- Social networks 3- Enterprise networks 4- Data centers	Self-adaptive load balancing [42]

accessible network [46]. The increase of traffic with varying requirements trivially causes high energy consumption and degradation in the performance of SDN controllers. In order to maintain and enhance the network's performance, optimizing the energy consumption of controllers, switches, and links in heterogeneous and live networks with varying input is necessary. In fact, it is one of the key requirements for obtaining quality of service, reliability, robustness in various computing-intensive networks such as data centers and disaster management networks. Statistics show, even with the reduction in the growth of the data center for reliable services, the power consumption is increasing significantly [8].

In addition, with the exponential growth of Internet use, a large amount of energy is required to operate and control the cooling system in backbone network infrastructures which produces a significant amount of carbon waste. Therefore, while developing network control functions in SDN, there is a need for reducing energy consumption in order to control carbon emission and to make the environment green.

With the aforementioned summary, we now introduce the related works in the direction of energy-efficient computing in SDN. The research is broadly classified into two categories: (i) basic energy-efficient SDN, and (ii) application-aware energy-efficient SDN. Table 2 shows the highlighted research works for achieving energy-efficiency in SDN. They also highlight the benefits and drawbacks of different proposed methodologies.

3.2.1. Traffic-aware energy-efficient SDN

This subsection discusses the existing research works on designed frameworks for obtaining energy efficiency in SDN, solutions provided to reduce energy consumption of the network devices, as well as the impact of energy efficiency on traffic management.

Wei et al. [47], presented the energy-efficient traffic engineering problem in hybrid SDN/IP networks. They formulated a mathematical model considering SDN/IP hybrid routing mode and proposed one algorithm to solve energy-efficient traffic engineering problems. Their algorithm considers the IP routers which perform the shortest path routing using distribute OSPF link weight optimization based on neighboring region search and split the traffic at the SDN enabled switches by the global controller. Bolla et al. [48], proposed energy management primitives in the context of the emerging Software-defined networking. It increases networking flexibility, the OpenFlow Protocol to integrate the energy-aware capabilities offered by the Green Abstraction Layer (GAL). It also proposes an analytic model for the management of a network with these capabilities. In Nam et al. [49], a novel energy-saving scheme that can flexibly control and route traffic relying on the difference of network device's energy-profile. The energy-saving scheme using OpenFlow makes use of the energy profile of network devices and switch port under various link rates. Cruz et al. [50], focused on energy consumption optimization in SDN switches and links, their associated rates, the number of flow entries at each SDN switch. Alberto et al. [51], proposed an energy-aware and policy-based system-oriented SDN paradigm, which allows managing the mobile network dynamically at run time and on-demand through policies. In their work, they reduced energy consumption by switching off unused devices. Priyadarsini et al. [8], proposed sleep and active mode concept for energy consumption reduction by the network devices and link paths. The proposed mathematical modeling for each device's energy consumption calculation. In addition, they introduced energy consumption reduction by the routing algorithms used by the controller for a global topology view and finding the route for each packet. Priyadarsini et al. [7], presented an energy-efficient load balancing approach where the authors integrated energy consumption and load balancing in a single framework. They have shown how to load increment leads to more energy consumption and proposed a controller system model to balance traffic load as well as reduce the energy consumption of the network.

Table 2
Proposed Methodologies for energy-efficiency in SDN with benefits and drawbacks.

Category	*Methodology*	*Principle*	*Benefits (+) and drawbacks (−)*
Basic energy-efficient SDN	Energy-efficient traffic engineering [47]	OSPF link weight optimization for SDN/IP hybrid routing mode	+ Efficient link utilization − Prone to collision
	Energy-aware primitives for GAL [48]	Analytical modeling for energy-aware capabilities of GAL	+ Green communication − Not scalable
	Energy-saving scheme [49]	Routes traffic by considering energy consumption of devices	+ Flexible to control routes − Prone to failure
	Energy-aware approach [50]	Optimization model to reduce energy consumption of switches and links	+ 40% energy saving − Prone to overheads
	Energy-aware SDN paradigm [51]	Manages mobile network dynamically through policies	+ Discards unused devices + Cost effective − High processing time
	Energy optimization framework [8]	Selects energy-efficient routing routing algorithms	+ Optimizes energy of devices − Uses existing algorithms
	Energy-efficient load balancing [7]	Balances network load while optimizing energy consumption	+ Improves network performance + 25% more energy saving − Not cost effective
Application-aware energy-efficient SDN	ElasticTree [52]	Saves energy of data center networks	+ Dynamic flow control − Not scalable
	CARPO [33]	Eliminates unnecessary links in data center networks	+ 90% link utilization − Delay
	EAR [53]	Introduces infinite number of rules	+ Reduces links − Communication delay − Large memory requirement
	DevoFlow [54]	Distributes controllers work among active switches	+ Reduces energy consumption + Cost effective − Large memory requirement
	Energy-aware VM placement [55]	Combines VM placement and routing in DCN	+ Parallel processing + Uses DFS and best fit − Not fault-tolerant
	Traffic-aware VM placement [56]	Efficiently allocate resources in intra-DCN	+ Better resource utilization − Delay

The survey paper by Tuysuz et al. [57], discussed the importance of energy-efficiency in SDN and presented major research works to reduce energy consumption in the network. They also discussed the benefits and drawbacks of each proposed methodology. In addition, they discussed open issues n energy-efficiency and provided a guideline for the future direction of research.

3.2.2. *Application-aware energy-efficient SDN*

There exist research works that contribute towards achieving energy efficiency in application specific SDN fields such as data center networks, wireless sensor networks, mobile ad-hoc networks to achieve energy efficiency. We highlight some of the research contributions in this subsection.

ElasticTree [52] was proposed to save energy in data center networks (DCN) using the SDN framework. It dynamically optimizes energy by ensuring qualitative traffic flow, meeting performance constraints, and turning network devices off. It has three parts: optimizer, routing, and power control. Optimizer provides a minimum-power consumption considering a particular load of the network topology. The routing part chooses the routes for traffic flows. Finally, the power control part arranges power states and turns the device on or off according to the necessity. Correlation-Aware Power Optimization Algorithm (CARPO) [33], saves the energy of the DCN by eliminating unnecessary links. Considering maximum link capacity and the equality of incoming and outgoing data rates, CARPO tries to minimize energy consumption by turning off switches as much as possible.

There exists another direction of research which considers VM placement and rule placement for energy-aware SDN approaches. Authors in [53] proposed a method to eliminate the rule space problem of existing energy-aware routing (EAR) approaches. EAR approaches mainly assume that there is an infinite amount of rule space and hence an OpenFlow switch can hold an infinite number of rules. According to this work, if there is not any predefined rule for a packet, the default rule is applied. Authors in [54] proposed a load aware energy-efficient framework namely; DevoFlow, which disburdens the controller by assigning some of its work to switches and provides energy-efficient management of the network. The underlying reason for this modification is the interference of the controller inflows transfers a high amount of workload to the control plane. Authors in [56] focused on the traffic-aware placement of VMs to obtain better utilization of resources. They presented two algorithms: server-driven and network-driven. These algorithms aim to efficiently allocate resources of intra-DCN. The server-driven algorithm first chooses a server for a VM and then determines the switches that will provide the flow of traffic. The network-driven algorithm first selects the switches and then an appropriate server for VM. Authors in [55] combined VM placement and routing optimization in DCNs to achieve energy efficiency and proposed a joint host-network algorithm by using a depth-first search with the best-fit option. They introduced a parallel processing method that divides DCNs into clusters and found optimal paths in a parallel way.

In summary, basic energy-efficient techniques are not scalable and prone to failure. Moreover, they are highly complex in nature with large memory requirements. Therefore, there is a need to develop energy-efficient traffic Management and network control solutions for SDN which are scalable, fault-tolerant, cost-effective, and require less computation time. This direction of research is still in the pre-mature stage. Network industries need to come with effective embedded communication and control software technologies to address this challenge. On the other hand, the application-aware energy-efficient techniques are applied in data centers with rule updating methodology. In this direction, researchers require to exploit the growth in server and computing technologies along with resource sharing to develop light-weight applications that reduce carbon footprint. In the next section, we describe the SDN security challenges and the prevention methodologies to provide high-end security to the platform.

4. Security and SDN

SDN is an emerging technology with effective control and decision making solves various security challenges in traditional networks. Thus, SDN implementations introduce wide-open space to the network security research community. Here, we present the major security challenges across different functional layers in SDN along with the potential prevention and control strategies.

4.1. Security challenges

Security challenges can be realized mainly in the form of attacks. Attacks on the SDN platform consist of control plane attacks, data-plane attacks, southbound attacks, and northbound attacks.

4.1.1. Control plane attacks

The literature reported possible attacks on the SDN control plane that include: denial of service (DoS) in switch-controller and switch–switch [58], controller hijacking [59], insertion of malicious applications [60]. Researchers at Microsoft introduced a threat model for the SDN controller, called *STRIDE* [61] that consists of six different attack types such as Spoofing, Tampering, Repudiation, Information Disclosure, DoS, and Elevation of Privilege. The above attacks occur dynamically when the controller processes requests from the underlying devices. On the other hand, it is also possible to launch an attack on the controller during build time by exploiting the vulnerability of the controller's *operating system* and bugs in the network function implementation. In addition, there exist reports on two other classes of SDN-protocol attacks, namely topology tampering [62] and link-fabrication [63]. Such attack classes manifest as DoS and controller hijacking. The attacks on SDN Controller is considered as high-risk security violation as it affects the complete functioning of the network. So, there is a need for research to come up with effective solutions to address these.

4.1.2. Data plane attacks

SDN switches present in data plane are capable of only forwarding the traffic flows according to the flow rules. However, attacks on switches may cause damage to the entire network including it being the potential source of controller attacks. In general, the attacks on the switches are of two types. Firstly, there may exist fake traffic flows during the intercommunication of switches. Secondly, malicious flows may be generated/exists during the in-flow and out-flow of the traffic. The common form of attacks in the data plane are DoS, DDoS [58], spoofing [61], intrusion [2] are examples of attacks associated with data plane. Spoofing attacks lead to flow rule modification, leakage of data. On the other hand, DoS and DDoS attacks create flooding in controllers and flow entries. Network intrusions cause unauthorized data, leakage of data, injection of software bugs.

4.1.3. Southbound attacks

Southbound APIs play a major role in communication between the control plane and the data plane. If this interface is compromised then malicious traffic can be injected in both the planes. Examples of southbound attacks are man-in-the-middle [64], black-hole [59] attacks, intrusion attacks [2], DoS and DDoS attacks [58], spoofing attacks [61], etc. Black-hole attack diverts the flow rules created by the controller when forwarded to the switches. The attacker can manipulate the data plane devices according to the flow rules. The man-in-middle attack can see the flow rules and accordingly attack the switches by knowing the loopholes of the devices.

4.1.4. Northbound attacks

Northbound protocols are used to communicate between the application plane and the control plane. Intrusion attack [2] is the major attack type possible on the northbound protocol.

4.1.5. STRIDE attack model

As mentioned earlier, the STRIDE attack model is widely applicable to network threat modeling that consists of almost all types of attacks. It covers the following six classes of attacks. (i) *Spoofing* aims to hide the identity of the attacker as well as traffic origin. Its primary goal is to mislead the target using a fake IP address. The controller should able to detect IP spoofing initiated within its network. (ii) *Tampering* is an unauthorized modification or destruction of network information such as flow rules, policies, and access lists. The Transport layer security (TLS) encryption used in OpenFlow does not provide efficient isolation to prevent tampering mechanism. This is because many controllers have not adopted TLS encryption or kept it as an optional mechanism. (iii) *Repudiation* is applicable to SDN controllers during build time only. The administrator needs to provide a user name and password to all developers to prevent code injection attacks in the controller. (iv) *Information Disclosure* is the method for getting information about other users or the system in an unauthorized manner. The information can be exploited by the attacker to extract useful information from the target (controller). (v) *Denial of Service (DOS)* occurs when an attacker sends a large number of packets to the controller for processing with large payload size and SYN flag *ON*. The functioning of the controller slows down as it gets engaged in sending acknowledgment packets, and by the time the attacker gets its required information. The attacker can exploit this time to extract useful information from the controller. (vi) *Elevation of Privilege* is the ability to get privileged access to some other systems without appropriate access rights. This type of attack may results in the modification of flow rules by the attacker, and subsequently, the attacker can gain control over the network.

4.2. Proposed prevention methodologies

There exist various research efforts to address security challenges in the SDN controller. Here, we discuss the existing solutions that cover DoS attacks on the controller, Controlling Malicious Applications, policy-based Security, and game-based security models.

4.2.1. Prevention to DoS attacks in SDN

Shin et al. [65] introduced a solution, called *AVANT-GUARD* that includes a connection migration mechanism used to establish useful TCP sessions, and actuating triggers that enable data plane devices to activate flow rules under predefined conditions. Wei et al. [66] presented a *FlowRanger*, which is a request prioritizing algorithm for control plane based DoS attacks. It calculates the trust value of each node in the network based on the traffic flow request and stores in a priority queue. Entries on the priority queue control the flow rule generation inside the controller. Dhawan et al. [67] provided a flow behavioral solution, named *Sphinx* that monitors all controller communication and identifies relevant OpenFlow messages required to build the network. It analyzes and compares the current traffic flows with the previous flows and saved policy rules over a certain period.

4.2.2. Controlling malicious applications

Wang et al. [68] proposed a scheme, *PERM-GUARD*, which provides authenticity to the controllers before the generation of flow rules and avoids the *controller hijacking attack*. The authentic controllers can only generate the rules and detect the *compromised controllers*. The works in [69] and [70] proposed solutions, such as *FortNOX* and *LegoSDN*, to mainly discard *malicious applications* from the application layer and northbound API. Similarly, BroIDS [71] is a security solution to avoid man-in-the-middle and black-hole attacks, and provides security to the control and application planes. Jiang et al. [72] introduced a technique, *Combat-Sniff*, which actively scans eavesdropping nodes in the network and proactively defend sniff packets. This eliminates *eavesdropping attack mitigation* in the SDN environment and targets TCP-SYN based attacks.

4.2.3. Policy-based security mechanisms

A new direction of research provides policy-based security mechanisms to the SDN controller. Dao et al. [73] introduced a method utilizing user behavior analysis. They listed existing attack types in a repository, and every incoming packet request is checked with the repository. If the packet matches the attack category, then it is discarded; otherwise, flow rules are generated for the same. MM. et al. [74] introduced the use of a trust management system that maintains IDs of currently operational network devices. An encrypted authentication mechanism is used to check the authenticity of the devices and allow them to generate flow rules. Krishna et al. [60], presented a policy-based approach to mitigate attacks on the SDN controller. They proposed two modules inside the controller namely, *policy manager* and *topology repository*. The topology repository fetches useful topology information from the switches about the incoming packet and forwards to the policy manager, which in turn evaluates the traffic against existing security policies and generates the required flow rules using topology information.

4.2.4. Game-based security models

In the literature, there exist research works on controller security using the Stackelberg game model. Lu et al. [75] proposed a defense strategy against session hijacking that minimizes the overall cost of the SDN network. They modeled the game as a strategic interaction between the session attacker and the defender. However, this work did not address multiple attack scenarios. Chen et al. [76] proposed a dynamic scheduling method using the Stackelberg game to maximize the security reward of the defender. They mainly focused on the attacks on the control plane while the selection of master and slave controllers are occurring. Chowdhary et al. [77] proposed a dynamic game model inside the control plane, which avoids DDoS (distributed DoS) attacks only. Their game model is based on the Nash Folk Theorem, which implements a punishment mechanism for attackers and rewards for players who cooperate. None of the above game models considered the effect of time on the game as well as every possible attack scenario. Priyadarshini et al. [78] proposed a signaling game-based security enforcement framework (SEF). The framework prevents attacks on the controller prior to calculating the behavior of the attacker. It provides security with trust-based and risk-based packet analysis and detects dynamic attacks on the SDN controller.

Table 3 summarizes the scope and features of existing security solution. It covers the targeted attack defense, the application scope to SDN layers, and security objectives they aim to maintain. In summary, a number of research have been conducted on SDN security solutions that used different approaches. However, these solutions are limited in terms of defending heterogeneous attack patterns across different layers including end-to-end perimeter of SDN. Moreover, the effectiveness and usability of these solutions in real SDN platform require systematic analysis and reporting.

5. Placement of controllers in SDN

A single controller can manage the network efficiently, as it provides flow rules, routing decisions, policy management depending on the global network view. However, it has disadvantages in terms of latency between switches that are placed away from the controllers, cost of network links, processing powers, and reliability [80]. These challenges are handled by physically distributing multiple controllers across the network. All the controllers communicate among themselves to maintain a consistent global network view and to ensure proper network operation. The controllers must be placed at different locations in the network, which provide optimal performance and minimum cost. The switch set of any controller receives flow rules from that controller at any point in time. Determining the optimal location of controllers in the network, and assigning a set of switches to them is known as controller placement problem (CPP) [81,82]. The placement

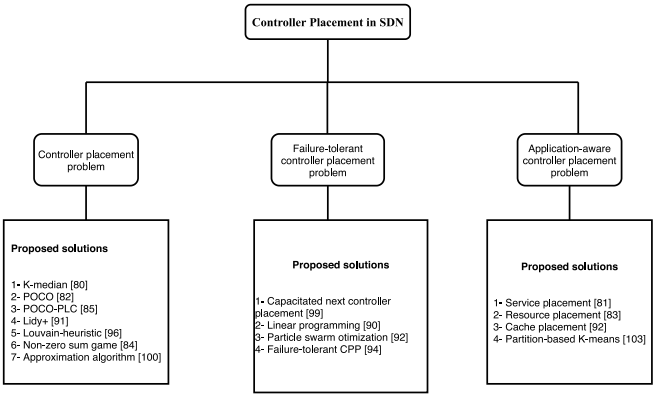

Fig. 7. Proposed solutions in controller placement problem.

of controllers introduces some challenging questions such as; how many controllers are required to manage the network? how many switches each controller can manage? what are the right locations to place the controllers? during the failure of controllers how the switches can be managed?

The controller placement problem in SDN has been solved considering various constraints of the network such as; latency, cost, load, capacity, etc. We divide the controller placement problem in SDN into three categories, namely, (i) controller placement problem, (ii) failure-tolerant controller placement problem, (iii) application-aware controller placement problem. Fig. 7 shows the highlighted contributions in SDN controller placement.

5.1. Controller placement problem (CPP)

The CPP was first introduced by Heller et al. [82] in 2012. They investigated; the number of controllers required in a given topology and their placement locations. They considered average-case latency bound, worst-case latency bound, and nodes within a latency bound metrics to find the location of the controllers. They formulated the CPP as a minimum k-median problem and minimum k-center problem to minimize the average and maximum (worst case) switch-to-controller latency, respectively. However, they did not consider multiple controllers in the network and capacity of the controller. Hock et al. [83–85] proposed POCO (Pareto Optimal Controller) and POCO-PLC for all possible controller placements in realistic networks. They considered inter-controller latency, controller failures, network disruptions constraints, and found locations for placements. They also developed GUIs for POCO and POCO-PLC. POCO-PLC is different from POCO as it combines the MATLAB analysis tool with a distributed application. But, these frameworks do not consider load factors into account for controller placement. The time complexity of these proposed algorithms is very high. Sallahi et al. [86] determined the optimal number, location as well as the interconnections between all the network elements while minimizing the cost of the network. They formulated the problem using linear programming and solved it through the optimizer. However, this model applies to only small networks and solves 10% of the total problem in 30 h. Therefore, the performance of this model is very less. Lange et al. [87] implemented the POCO framework in a large scale SDN network, and also gave a heuristic solution Pareto simulated annealing to solve the placement problem. The running time of the simulated annealing used here is high. Cheng et al. [88] proposed a QoS-Guaranteed Controller Placement problem to find the number of controllers needed, their placement locations, and the switches assigned to each of the controllers. They proposed three heuristic algorithms; incremental greedy algorithm, primal–dual-based algorithm, and network-partition-based algorithm, which guaranteed QoS in the network and provides optimal placement. However, the latency is

Table 3
Attack detection comparison of state-of-the-art prevention methodologies.

Comparison indices		Proposed solutions											
		SEF [78]	Flow Ranger [66]	Sphinx [67]	Topo-Guard [79]	Avant-Guard [65]	Game-model [77]	Perm-Guard [68]	FortNOX [69]	LegoSDN [70]	BroIDS [71]	Combat-sniff [72]	Game-model [75]
SDN Operation Layer	Data			✗				✗			✗		
	Control	✗	✗		✗	✗	✗	✗	✗	✗	✗	✗	✗
	Application							✗	✗	✗	✗		
	Northbound								✗	✗	✗		
	Southbound								✗		✗	✗	✗
Targeted Attacks	DoS	✗	✗	✗	✗	✗	✗						
	Controller hijack	✗						✗					
	Malicious Application	✗							✗	✗			
	MIM , Black hole Attack	✗									✗		
	Eavesdropping	✗										✗	✗

high in comparison to other solutions. Ul-Haque et al. [89] proposed an algorithm named LiDy+ for controller placement, which not only achieves a higher controller utilization but also incurs less energy and maintenance costs. They considered open search (the controller can be placed inside any devices) and restricted search (the controllers are placed inside switches) for controller placement. This is the only work where the open search controller placement is considered. But, the performance of the open search is less. Kim et al. [90] proposed a heuristic solution for both switch and controller placement in a distributed SDN environment. However, it does not consider the cost and load parameter into consideration.

Wang et al. [80] presented a survey on state-of-the-art controller placement solutions that considered different parameters like minimization of latency, deployment cost, energy consumption, and maximization of reliability. Chen et al. [91] proposed a Louvain-heuristic based algorithm to find optimal placement for the controllers. They partitioned the network and then found the optimal location within the partition. This solution method does not consider the capacity and cost of the controllers during placement. Rath et al. [92] solved the CPP using a non-zero-sum game and found the optimal location of the controllers using the payoff values of the game. However, the load and cost parameters are not considered, which do not provide accuracy in placement. Sood et al. [81] presented the CPP as controller selection problem (CSP). They Choose CSP instead of CPP to solve performance issues in SDN, such as the minimum number of controllers, their workload distribution, and placement locations, etc. Here, the performance of the network degraded during the execution of the proposed algorithms. Singh et al. [93] presented a survey on research works on CPP from 2012 to 2017 and critically analyzed the existing solutions, and found their limitations and future scopes. Zhang et al. [94] presented the CPP as an optimization problem, and minimized the reaction time using approximation algorithms. The throughput is less and the delay is high in this solution. Das et al. [95] formulated the CPP in hybrid SDN/legacy network over a period of time and maximized the switch-controller control channel resilience. Li et al. [96] proposed a constant approximation algorithm based on the prime dual approach to finding the minimum number of controllers in the network. They also minimized the controllers' cost and latency of the network.

5.2. Failure-tolerant controller placement problem (FCPP)

Controller failure results in the disconnection of the controller from its assigned switches. Since the controller is an external entity and it runs on hardware, both software and hardware failures are considered as controller failure. Therefore, the reliability of the controller is an important factor to be considered.

Killi et al. [97] proposed a capacitated next controller placement solution considering the controller failure constraints and minimizes the average latency of the network. However, they did not consider the effect of cost during placement and reassignment. Perrot et al. [98] solved CPP in a wide area network (WAN) using linear programming. They considered QoS and load balancing constraints and gives the optimal number of controllers and their locations. They also considered the failure of both controllers and routers. Although this work considered placement, failure, and maximum constraints, it is suitable for small-sized networks. The controller can be placed in separate hardware or inside the OpenFlow switches. Alshamrani et al. [99] presented fault-tolerant controller placement in a distributed SDN environment. But, the performance of the network is less. Fan et al. [100] considered link failure and the worst-case delay between switch-controller factors to solve CPP. They proposed a heuristic method based on particle swarm optimization. However, inter-controller communication which has a major role during link placement is not considered here.

5.3. Application-aware controller placement problem (ACPP)

There exist another direction of research where service [101], resource [102], cache [103] placement are done in heterogeneous networks. In paper [104], the authors considered delay-overhead minimization in wireless edge networks during placement. Yang et al. [105, 106] solved the controller placement problem in SDN-enabled integrated satellite-terrestrial networks. They proposed a network partition algorithm, Simulated Annealing Partition-based K-means (SAPKM), to find the placement locations.

Note that optimal and suitable placement of SDN controllers in a network can help in balancing traffic load, used to save a significant amount of energy consumed by the network, and provide efficient security defense. On the other hand, solving the load balancing issue introduces challenges and changes in the objective of optimal controller placement. Therefore, there is a need for optimal controller placement in SDN integrating load balancing, energy consumption reduction, inter-controller communication considering the changes in dynamics and multiple objectives of the network. In the next section, we discuss other existing challenges in the SDN environment which may become individual research problems.

6. Open challenges in SDN

Our study shows that there is a number of effective solutions on energy-efficient traffic management, adaptive load balancing, and security hardening for SDN. However, there exist additional challenges and open problems that require systematic study and related developments. In this section, we discuss a few important open problems in SDN.

6.1. Scalability of the controller

Scalability is one of the major challenges in SDN as it is expected to accommodate demand growth in traffic volume and heterogeneity in requirements without compromising the quality of service and performance. The controller is the key component of SDN that drives the functioning and performance of the network. In general, a single controller should scale up to a traffic volume of 100 switches [107]. However, beyond this limit, the controllers face load balancing problem, and it leads to performance degradation of the network. Therefore, there is a need for efficient software cores and libraries that enables agile development of controller supporting demands and provides a scalable execution architecture. In addition, cross-layer and inter-controller communication protocols for traffic management introduce network broadcast overhead and proliferation of flow table entries. There is a need for sustainable solutions both in terms of software core and API builds to mitigate this problem.

6.2. Reliability of the controller

Reliability plays a major role in software systems or software guided control systems. If there is any failure in a system, it should automatically generate alarms or exceptions [108]. The SDN controller configuration and functioning must intelligently assess and validate network management with changes in network dynamics so as to increase the availability of the network. Researchers studied the reliability of ONOS, a production-grade SDN controller, and the fault report shows fairly consistent behavior across the releases, in terms of the number of bugs, fault detection, and resolution time. When devices fail or stop working in legacy network, network traffic is routed through alternative nearby paths/nodes to maintain flow control and continuity. In SDN, failure of the centralized controller may lead to collapsing of the whole network. Also, this may introduce security breaches in the network and legitimate system components may be compromised. Therefore, the vendors and developers may emphasize on developing efficient, reliable and secure network control functions and development of robust, consistent software stack in integration of these functions.

6.3. Integration of SDN with legacy network

SDN architecture is flat with centralized control whereas the traditional network is hierarchical in nature. Therefore, integration of SDN devices with legacy network devices is one of the major challenges. One way, there is a need for hybridization between OpenFlow protocol stack and TCP/IP protocol stack with less transformation. On the other hand, network control functions including traffic management and monitoring, flow and congestion control (load balancing), and security solutions for both platforms require unification with suitable interfaces and software components. The integration must be done in a seamless manner so that the performance, robustness, and reliability of the combined platform are not affected.

6.4. Performance enhancement

The performance of SDN mainly depends on switches and controllers. There is a need for switches with high capacity processors embedded with advanced hardware logic. On the other hand, the energy consumption by switches is directly proportional to the in-build hardware logic. So, networking industries need research on suitable technologies for OpenFlow switches. On the other hand, controllers' performance depends on the complexities of network functions, execution architectures, and APIs. A good number of network functions including security solutions are in the scope of control functions that satisfies heterogeneous requirements as well as increase network performance. The performance of SDN depends on various network functions such as, load balancing, energy efficiency, security, and controller placement. Efficient and secure traffic management for heterogeneous and real time applications demands integration of all these network functions exploiting their inter dependencies. For example, if less number of controllers are placed in the network, it may reduce energy consumption and satisfy the load of the network, but, it may be easier for attacker to inject attack paths in the network. Similarly, more controllers may serve large number of traffic but may introduce latency and higher energy consumption. Therefore, it is important to define the network performance metric accurately and devise multi-objective problems with satisfaction of constraints on network parameters and dynamic changes in requirements and propose solutions for the same.

6.5. Flow-management

The SDN switches take decisions on data packets based on the flow rules which are set by the controller. An SDN switch maintains a flow table which is an ordered set of flow rules. These rules are stored for future use to optimize the traffic forwarding process. In general, flow tables have limited space and therefore old rules are overwritten by new rules. This may require more controller intervention for some old packets in the later stage. On the other hand, more flow rule entries may introduce overhead leading to performance degradation, an increase in energy consumption, and cost. Therefore, there is a need for efficient data structures and procedures to store more flow entries that can be dynamically updated without compromising cost, performance, and energy consumption.

7. Conclusion

In this survey, we have presented a systematic study of state-of-art technologies and research on SDN architecture, operations, functionalities, and its evolution from OpenFlow, and network function virtualization for SDN. A detailed discussion on traffic management challenges such as load balancing and energy-efficient routing is covered with the proposed solution approaches. Our observations on proposed solutions have been reported. In addition, we have discussed the security challenges in the SDN environment, and the state-of-the-art works towards mitigating these problems. Then, we presented our study on an important constraint solving the problem in SDN called as controller placement problem. The selection of network parameters and constraints for this problem depends on applications and requirements. For example, the parameter required for the smart grid is different from IoT and cyber–physical systems. Also, we have discussed the survey of the fault-tolerant controller placement problem. In order to bring more insights and future scope on this emerging paradigm, we have discussed the current challenges in SDN.

Our survey reveals that there is a need for focused research on designing efficient controller architecture and SDN framework that can overcome traffic management challenges, provide end-to-end security along with the reduction in energy consumption. The failure-tolerant controller placement problem must be explored in combination with

inter-controller communication and synchronization issues. In summary, networking research community and developers should develop technologies and solutions to address the above-mentioned problems in SDN with major objectives of enhancing network performance with high scalability and adaptability; and with effective integration with legacy networks.

CRediT authorship contribution statement

Madhukrishna Priyadarsini: Conception, Design, Formal analysis, Interpretation of data, Writing - original draft. **Padmalochan Bera:** Writing - review & editing, Final approval of the version to be published.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Goransson, C. Black, Software Defined Networks- A Comprehensive Approach, Morgan Kaufmann (imprint of Elsevier), Waltham, USA, 2014.

[2] D.B. Rawat, S.R. Reddy, Software defined networking architecture, security and energy efficiency: a survey, IEEE Commun. Surv. Tutor. 19 (1) (2017) 325–346.

[3] Y. Jarraya, T. Madi, M. Debbabi, A survey and a layered taxonomy of software-defined networking, IEEE Commun. Surv. Tutor. 16 (4) (2014) 1955–1980.

[4] Y. Zhao, L. Iannone, M. Riguidel, IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), On the performance of SDN controllers: A reality check, 2015.

[5] M. Priyadarsini, P. Bera, R. Bampal, Performance analysis of software defined network controller architecture–A simulation based survey, in: International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017.

[6] M. Priyadarsini, P. Bera, A secure virtual controller for traffic management in SDN, IEEE Lett. Comput. Soc. 2 (3) (2019) 24–27.

[7] M. Priyadarsini, S. Kumar, P. Bera, M.A. Rahman, An energy-efficient load balancing scheme for SDN controllers, Computing 27 (5) (2019) 1–26.

[8] M. Priyadarsini, P. Bera, M.A. Rahman, A new approach for energy efficiency in software defined network, in: The Fifth International Conference on Software-defined Systems, 2018.

[9] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, A survey on software-defined networking, IEEE Commun. Surv. Tutor. 17 (1) (2015) 27–51.

[10] Open Networking Foundation, OpenFlow Switch Specification, 1.4.0 ed., 2017.

[11] I.Z. Bholebawa, R.K. Jha, U.D. Dalal, Performance analysis of proposed network architecture: Openflow vs traditional network, Int. J. Comput. Sci. Inf. Secur. 14 (3) (2016) 1–10.

[12] Telecommunication Department, Ministry of communication, India, A Study Paper on Network Function Virtualization and its impact on Future Telecom Networks, http://tec.gov.in/pdf/Studypaper/Network_Function_Virtualization%20.pdf [Online], accessed 19-July-2018.

[13] T. Wood, K.K. Ramakrishnan, J. Hwang, G. Liu, W. Zhang, Toward a software-based network: integrating software defined networking and network function virtualization, IEEE Netw. 29 (3) (2015) 36–41.

[14] O. Blial, M.B. Mamoun, R. Benaini, An overview on SDN architectures with multiple controllers, J. Comput. Netw. Commun. (2016) 9396525:1–9396525.

[15] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.

[16] M. Priyadarsini, P. Bera, Traffic Management in SDN- a Road Map to Research, Scholars' Press, 2019.

[17] About NOX, http://www.noxrepo.org/nox/about-nox/ [Online], accessed 10-March-2016.

[18] About POX, http://www.noxrepo.org/pox/about-pox/ [Online], accessed 10-March-2016.

[19] D. Erickson, ACM SIGCOMM workshop on hot topics in software-defined networking (hotsdn), in: The Beacon OpenFlow Controller, 2013.

[20] FloodLight, Open SDN Controller, http://www.projectfloodlight.org/blog/2016/03/10/announcing-floodlight-v1-2/ [Online], accessed 12-January-2016.

[21] OpenDayLight Project, http://www.opendaylight.org/ [Online], accessed 16-January-2016.

[22] SDN Series Part Two: Trema, a Framework for Developing OpenFlow Controllers in Ruby and C, https://thenewstack.io/sdn-series-part-ii-trema-a-framework-for-developing-openflow-controllers-in-ruby-and-c/ [Online], accessed 29-September-2019.

[23] H. Zhong, An Efficient SDN Load Balancing Scheme Based on Variance Analysis for Massive Mobile Users, Hindawi Publishing Corporation Mobile Information Systems, 2015, 241732.

[24] J. Wu, C. Yuen, B. Cheng, Y. Shang, J. Chen, Goodput-aware load distribution for real-time traffic over multipath networks, IEEE Trans. Parallel Distrib. Syst. 26 (8) (2015) 2286–2299.

[25] W. Reese, NGINX: The high-performance web server and reverse proxy, Linux J. 2008 (173) (2008).

[26] V. Kaushal, A.G. Bala, Autonomic fault tolerance using HAProxy in cloud environment, Int. J. Adv. Eng. Sci. Technol. 7 (2) (2011) 54–59.

[27] M.A. Islam, S. Ren, G. Quan, M.Z. Shakir, A.V. Vasilakos, Water-constrained geographic load balancing in data centers, IEEE Trans. Cloud Comput. 5 (2) (2017) 208–220.

[28] L. Yu, T. Jiang, Y. Zou, Price-sensitivity aware load balancing for geographically distributed internet data centers in smart grid environment, IEEE Trans. Cloud Comput. 6 (4) (2018) 1125–1135.

[29] S. Sthapit, J. Thompson, N.M. Robertson, J.R. Hopgood, Computational load balancing on the edge in absence of cloud and fog, IEEE Trans. Mob. Comput. 18 (7) (2019) 1499–1512.

[30] Y. Hu, Balanceflow: Controller load balancing for openflow networks, in: IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), 2012.

[31] Y. Zhou, A load balancing strategy for SDN controller based on distributed decision, in: IEEE 13th International Conference on Trust, Security, and Privacy in Computing and Communications, 2014.

[32] J. Yu, Y. Wang, K. Pei, S. Zhang, J. Li, A load balancing mechanism for multiple SDN controllers based on load informing strategy, in: The 18th Asia Pacific Network Operations, and Management Symposium (APNOMS), 2016.

[33] X. Wang, Y. Yao, X. Wang, K. Lu, Q. Cao, Carpo: Correlation-aware power optimization in data center networks, in: INFOCOM, 2012 Proceedings IEEE, 2012.

[34] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, A load-balancing mechanism for distributed SDN control plane using response time, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1197–1206.

[35] Y. Zhou, Y. Wang, J. Yu, J. Ba, S. Zhang, Load balancing for multiple controllers in SDN based on switches group, in: The 19th Asia Pacific Network Operations and Management Symposium (APNOMS), 2017.

[36] J. Li, An effective path load balancing mechanism based on SDN, in: Proceedings of IEEE 13th International Conference on Trust Security Privacy Computation Communication, 2014, pp. 527–533.

[37] R. Trestian, Ofload: An openflow- based dynamic load balancing strategy for datacenter networks, IEEE Trans. Netw. Serv. Manag. 14 (4) (2017) 792–803.

[38] H. Zhong, LBBSRT: An efficient SDN load balancing scheme based on server response time, Future Gener. Comput. Syst. 68 (2017) 183–190.

[39] S.H. Yeganeh, Y. Ganjali, Kandoo: A framework for efficient and scalable offloading of control applications, in: Hot Topics Software Defined Network (HotSDN), 2012, pp. 19–24.

[40] ONOS project [Online], Available: https://onosproject.org/.

[41] Y. Fu, A hybrid hierarchical control plane for flow-based large scale software-defined networks, IEEE Trans. Netw. Serv. Manag. 12 (2) (2015) 117–131.

[42] M. Priyadarsini, J. Mukherjee, P. Bera, S. Kumar, A. Jakaria, M.A. Rahman, An adaptive load balancing scheme for software-defined network controllers, Comput. Netw. (2019) 164, http://dx.doi.org/10.1016/j.comnet.2019.106918.

[43] X. Zeng, D. Wang, X. Han, W. Yao, Z. Wang, R. Chen, An effective load balance using link bandwidth for SDN-based data centers, in: International Conference on Artificial Intelligence and Security, 2019.

[44] L. Guilen, S. Izumi, T. Abe, T. Suganuma, H. Muraoka, SDN-based hybrid server and link load balancing in multipath distributed storage systems, in: IEEE Symposium on Network Operations and Management, 2018.

[45] H. Wang, H. Xu, L. Huang, J. Wang, X. Yang, Load-balancing routing in software defined networks with multiple controllers, Comput. Netw. 141 (2018).

[46] Software defined networking (SDN) as a tool for energy efficiency approaches in information and communication technology (ICT) networks, in: Study Paper By Telecommunication Engineering Department, Government of India, 2015.

[47] Y. Wei, X. Zhang, L. Xie, S. Leng, Energy-aware traffic engineering in hybrid SDN/IP backbone networks, J. Commun. Netw. 5 (2) (2016).

[48] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, Fine-grained energy-efficient consolidation in SDN networks and devices, IEEE Trans. Netw. Serv. Manag. 12 (2) (2015).

[49] T. Nam, N. Thanah, N. Thu, H. Hieu, S. Covaci, Energy-Aware Routing Based on Power Profile of Devices in Data Center Networks using SDN, 978-1-4799-7961-5/15/, IEEE, 2015.
[50] A.F. Cruz, J.P. Muñoz Gea, P. Lopez, J. Sanahuja, Optimization of power consumption in SDN networks, in: The Ninth International Conference on Emerging Networks and Systems Intelligence, 2017.
[51] A. Celdran, M. Perez, F. Clemente, G. Perez, Policy-Based Management for Green Mobile Networks Through Software-Defined Networking, Springer Science Business Media New York, 2016.
[52] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, ElasticTree: Saving Energy in Data Center Networks, NSDI, 2010.
[53] F. Giroire, J. Moulierac, T.K. Phan, Optimizing rule placement in software-defined networks for energy-aware routing, in: Global Communications Conference (GLOBECOM), 2014.
[54] J.C. Mogul, Tourrilhes, P. Yalagandula, P. Sharma, A.R. Curtis, S. Banerjee, Devoflow: Cost effective flow management for high performance enterprise networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.
[55] M. Gharbaoui, B. Martini, D. Adami, G. Antichi, S. Giordano, P. Castoldi, On virtualization-aware traffic engineering in OpenFlow Data Centers networks, in: Network Operations and Management Symposium (NOMS), 2014.
[56] S.H. Wang, P.P.W. Huang, C.H.P. Wen, L.C. Wang, EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined data center networks, in: International Conference on Information Networking (ICION), 2014.
[57] M.F. Tuysuz, Z.K. Ankarali, D. Gozupek, A survey on energy efficiency in software defined networks, Comput. Netw. 113 (2016) 188–204.
[58] I. Gabriel, P. Victor-Valeriu, Achieving DDoS resiliency in a software-defined network by intelligent risk assessment based on neural networks and danger theory, in: Proceedings of the Fifteenth International Symposium on Computational Intelligence and Informatics, 2014, pp. 319–332.
[59] S. Shin, G. Gu, Attacking software-defined networks: A first feasibility study, in: Proceedings of HotSDN, 2013, pp. 165–166.
[60] K. Krishna, V. Vardharajan, U. Tupakula, Mitigating attacks in software defined network(sdn), in: Fourth International Conference on Software Defined Systems (SDS), 2017.
[61] F. Ruffy, W. Hommel, F.V. Eye, A STRIDE-based security architecture for software-defined networking, in: The Fifteenth International Conference on Networks, 2016.
[62] R. Skowyra, L. Xu, G. Gu, V. Dedhia, T. Hobson, H. Okhravi, J. Landry, Effective topology tampering attacks and defenses in software-defined networks, in: The 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2018.
[63] D. Kreutz, F.M.V. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software-Defined Networking, 2013, pp. 55–60.
[64] M. Hasan, M.A. Rahman, Protection by detection: A signaling game approach to mitigate co-resident attacks in cloud, in: IEEE 10th International Conference on Cloud Computing, 2017.
[65] S. Shin, V. Yegneswaran, P. Porras, G. Gu, Avantguard: Scalable and Vigilant switch flow management in software-defined networks, in: Proceedings of ACM CCS, 2013, pp. 413–424.
[66] L. Wei, C. Fung, FlowRanger: A request prioritizing algorithm for controller dos attacks in software defined networks, in: Next Generation Networking Symposium, 2015, pp. 5254-5259.
[67] M. Dhawan, R. Poddar, K. Mahajan, V. Mann, Sphinx: Detecting security attacks in software-defined networks, in: Proceedings of Network and Distributed Systems Security (NDSS), 2015.
[68] M. Wang, J. Liu, J. Chen, X. Liu, J. Mao, PERM-GUARD: Authenticating the validity of flow rules in software defined networking, in: International Conference on Cyber Security and Cloud Computing, 2015, pp. 127-133.
[69] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A security enforcement kernel for OpenFlow networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012, pp. 121–126.
[70] B. Chandrasekaran, T. Benson, Tolerating SDN application failures with LegoSDN, in: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, 2014, p. 22.
[71] S. Hayward, G. O'Callaghan, S. Sezer, SDN security: A survey, SDN for future networks and services (SDN4FNS), 2013.
[72] F. Jiang, C. Song, Z. Xu, Combat-sniff: A comprehensive countermeasure to resist data plane eavesdropping in software defined networks, Am. J. Netw. Commun. 5 (2) (2016) 27–34.
[73] N. Dao, J. Park, M. Park, S. Cho, A feasible method to combat against DDoS attack in SDN Network, in: International Conference on Information Networking, 2015, pp. 309-311.
[74] O. MM, K. Okamura, Securing distributed control of software defined networks, Int. J. Comput. Sci. Netw. Secur. 13 (9) (2013).
[75] Z. Lu, F. Chen, G. Cheng, J. Ai, A secure control plane for SDN based on bayesian stackelberg games, in: The 3rd IEEE International Conference on Computer and Communications, 2017.
[76] Z. Lu, F. Chen, G. Cheng, S. Li, The best defense strategy against session hijacking using security game in SDN, in: The 19th International Conference on High Performance Computing and Communications, 2017.
[77] A. Chowdhary, S. Pisharody, A. Alshamrani, D. Huang, Dynamic game based security framework in SDN-enabled cloud networking environments, in: The Fourth International Conference on Software-defined Network and Network Function Virtualization (SDN-NFV), 2017.
[78] M. Priyadarsini, P. Bera, M.A. Rahman, A signalling game-based security enforcement mechanism for SDN controllers, in: 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019.
[79] S. Hong, L. Xu, H. Wang, G. Gu, Poisoning network visibility in software-defined networks: New attacks and countermeasures, in: Proceedings of Network and Distributed Systems Security (NDSS), 2015.
[80] G. Wang, Y. Zhao, J. Huang, W. Wang, The controller placement problem in software defined networking: A survey, IEEE Netw. 31 (5) (2017).
[81] K. Sood, Y. Xiang, The controller placement problem or the controller selection problem? J. Commun. Inf. Netw. 2 (3) (2017).
[82] B. Heller, R. Sherwood, N. McKeown, The controller placement, in: HotSDN, 2012.
[83] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, Pareto-optimal resilient controller placement in SDN-based core networks, in: Proceedings of the 25th International Teletraffic Congress (ITC), 2013.
[84] D. Hock, S. Gebert, M. Hartmann, T. Zinner, P. Tran-Gia, POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks, in: IEEE Network Operations and Management Symposium (NOMS), 2014.
[85] D. Hock, S. Gebert, M. Hartmann, T. Zinner, P. Tran-Gia, POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks, in: IEEE Conference on Computer Communications Workshops (INFOCOM Workshop), 2014.
[86] A. Sallahi, M. St-Hilaire, Optimal model for the controller placement problem in software defined networks, IEEE Commun. Lett. 19 (1) (2015).
[87] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, Heuristic approaches to the controller placement problem in large scale SDN networks, IEEE Trans. Netw. Serv. Manag. 12 (1) (2015).
[88] T.Y. Cheng, M. Wang, X. Jia, QoS-guaranteed controller placement in SDN, in: IEEE Global Communications Conference (GLOBECOM), 2015.
[89] M.T.I. ul Huque, G. Jourjon, V. Gramoli, Large-scale dynamic controller placement, IEEE Trans. Netw. Serv. Manag. 14 (1) (2017).
[90] W. Kim, J. Ling, J. Hong, Y. Shu, HeS-CoP: Heuristic switch-controller placement scheme for distributed SDN controllers in data center networks, Int. J. Netw. Manag. 28 (3) (2018).
[91] W. Chen, C. Chen, X. Jiang, L. Liu, Multi-controller placement towards SDN based on louvain heuristic algorithm, IEEE Access 6 (2018).
[92] H.K. Rath, V. Revoori, S.F. Nadaf, V. Simha, Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2014.
[93] A.K. Singh, S. Srivastava, A survey and classification of controller placement problem in SDN, Int. J. Netw. Manag. 28 (3) (2018).
[94] T. Zhang, A. Bianco, P. Giaccone, The role of inter-controller traffic in SDN controllers placement, in: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016.
[95] T. Das, M. Gurusamy, Resilient controller placement in hybrid sdn/legacy networks, in: IEEE Global Communications Conference (GLOBECOM), 2018.
[96] T. Li, Z. Gu, X. Lin, S. Li, Q. Tan, Approximation algorithms for controller placement problems in software defined networks, in: IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018.
[97] B.P.R. Killi, S.V. Rao, Capacitated next controller placement in software defined networks, IEEE Trans. Netw. Serv. Manag. 14 (3) (2017).
[98] N. Perrot, Optimal placement of controllers in a resilient SDN architecture, in: International Conference on the Design of Reliable Communication Networks (DRCN), 2016.
[99] A. Alshamrani, S. Guha, S. Pisharody, A. Chowdhary, D. Huang, Fault tolerant controller placement in distributed SDN environments, in: IEEE International Conference on Communications (ICC), 2018.
[100] Z. Fan, J. Yao, X. Yang, Z. Wang, X. Wan, A multi-controller placement strategy based on delay and reliability optimization in SDN, in: The 28th Wireless and Optical Communications Conference (WOCC), 2019.
[101] M. Steiner, B. Gaglianello, V. Gurbani, V. Hilt, W.D. Roome, M. Scharf, T. Voith, Network-aware service placement in a distributed cloud environment, SIGCOMM, 2012.

[102] Y. Rochman, H. Levy, E. Brosh, Resource placement and assignment in distributed network topologies, in: Proceedings IEEE INFOCOM, 2013.
[103] S. He, H. Tian, X. Lyu, G. Nie, S. Fan, Distributed cache placement and user association in multicast-aided heterogeneous networks, IEEE Access 5 (2017).
[104] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella, L. Tassiulas, SDN controller placement with delay-overhead balancing in wireless edge networks, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018).
[105] K. Yang, B. Zhang, D. Guo, Partition-based joint placement of gateway and controller in SDN-enabled integrated satellite-terrestrial networks, Sensors 19 (12) (2019).
[106] K. Yang, B. Zhang, D. Guo, Controller and gateway partition placement in SDN-enabled integrated satellite-terrestrial network, in: IEEE International Conference on Communications Workshops (ICC Workshops), 2019.
[107] M. Alsaeedi, M.M. Mohamad, A. Al-Roubaiey, Toward adaptive and scalable openflow-SDN flow control: A survey, IEEE Acess 7 (2019) 107346–107379.
[108] X. Guan, B. Choi, S. Song, Reliability and scalability issues in software defined network frameworks, in: Second GENI Research and Educational Experiment Workshop, 2013.

Madhukrishna Priyadarsini is a research scholar in the School of Electrical Sciences, IIT Bhubaneswar, India. Her current work includes computer network management, software-defined network, security issues in SDN. Despite these areas, she is also interested in Image processing, Game theoretical approaches. She is a student member of IEEE as well as the secretary in IEEE student branch IIT Bhubaneswar. She has organized workshops in real-time implications of SDN.

Padmalochan Bera is working as an assistant professor in the department of Computer Science and Engineering in Indian Institute of Technology, Bhubaneswar, India. His research interest includes network security, cryptography, access control, software-defined networking (SDN), cloud computing, formal verification, and optimization. He was a postdoctoral fellow in CyberDNA Research Center, University of North Carolina Charlotte, USA from 2011–2012.