Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Review:

Novel architectures and security solutions of programmable software-defined networking: a comprehensive survey*

Shen WANG[1], Jun WU[†‡2], Wu YANG[3], Long-hua GUO[4]

[1]*Research Center for Modern Governance, Zhejiang University of Science and Technology, Hangzhou 310023, China*

[2]*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*

[3]*Information Security Research Center, Harbin Engineering University, Harbin 150001, China*

[4]*Huawei Technologies Co., Ltd., Shanghai 201206, China*

[†]E-mail: junwuhn@sjtu.edu.cn

Received Sept. 17, 2018; Revision accepted Nov. 14, 2018; Crosschecked Dec. 17, 2018

Abstract: Nowadays, cyberspace has become a vital part of social infrastructure. With the rapid development of the scale of networks, applications and services have become enriched, and the bearing function of the underlying network devices (such as switches and routers) has also been extended. To promote the dynamics architecture, high-level security, and high quality of service of the network, control network architecture forward separation is a development trend of the networking technology. Currently, software-defined networking (SDN) is one of the most popular and promising technologies. In SDN, high-level strategies are deployed by the proprietary equipment, which is used to guide the data forwarding of the network equipment. This can reduce many complicated functions of the network equipment and improve the flexibility and operability of the implementation and deployment of new network technologies and protocols. However, this novel networking technology faces novel challenges in term of architecture and security. The aim of this study is to offer a comprehensive review of the state-of-the-art research on novel advances of programmable SDN, and to highlight what has been investigated and what remains to be addressed, particularly, in terms of architecture and security.

Key words: Software-defined networking (SDN); Security; Programmable
https://doi.org/10.1631/FITEE.1800575 **CLC number:** TP393

1 Introduction

With the development of various applications and services of next generation networks, traditional networking technologies cannot satisfy the novel requirements, including dynamics architecture, high-level security, and high quality of service (QoS). To address such challenges, novel networking technologies have been widely studied (Feng et al., 2016; Liu B et al., 2016; Yang et al., 2016; Liu CF et al., 2018; Wu et al., 2018a, 2018b). The evolution history of networks is shown in Fig. 1. Software-defined networking (SDN) is one of the most popular and promising technologies (Zhang et al., 2017). It is usually used to decouple the controlling and forwarding functions, to solve the problems of the existing rigid network system; e.g., the network management and configuration are complex and prone to error, and the unified and rapid deployment of the new network architecture and technology is inconvenient. By virtue of SDN's network programmable ability and control logic centralization, network innovation can be accelerated and network management can be greatly simplified (Zhang et al., 2016a, 2016b).

There are some important advantages of novel SDN models. First, SDN can provide flexible and

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61831007)

ORCID: Jun WU, http://orcid.org/0000-0003-2483-6980

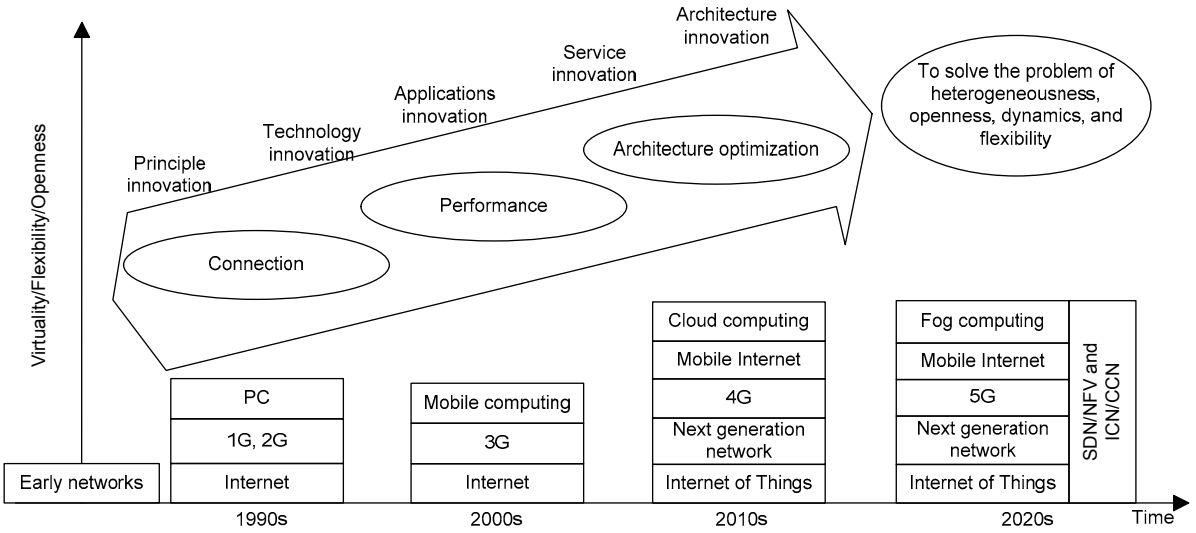

Fig. 1 Evolution of network technologies

extensible monitoring and control for future networks. There is a network operation system between the applications and networking devices, which can implement network virtualization and abstraction. Moreover, network control software can be deployed. The aforementioned factors can ensure the independent implementation of the data plane and control plane. Second, SDN can integrate heterogeneous networks and the semantic gap of different contents. This can enhance the QoS and the capacity of the network. Third, because of the usage of northbound and southbound interfaces, novel applications (e.g., big data and cloud computing) can be introduced in a seamless way. Thus, data processing and transmission can be implemented using a flattening and efficient approach. Fourth, dynamic security management and defense can be realized based on SDN, due to the programmable and redefinition features (Zhang et al., 2016a, 2016b; Hu et al., 2017).

SDN has received significant attention from all aspects. Not only has the academic community conducted in-depth research on its key technologies, but there have been large-scale applications in industry. However, note that the emergence of SDN not only brings opportunities, but also faces many challenges. The rest of this paper is organized as follows: Section 2 presents the research background of programmable SDN. Section 3 gives the mainstream reference architecture of programmable SDN. The key technologies involved in the implementation of SDN are shown in Section 4. Novel security technologies are discussed in Section 5. Application scenarios of SDN are discussed in Section 6 and promising development directions are analyzed in Section 7. Finally, Section 8 concludes this study.

2 Analysis of existing programmable SDN systems

Cyberspace has become a vital part of social infrastructure. With the rapid development of networking technologies, applications and services have become enriched, and the bearing function of the underlying network devices (such as switches and routers) has also been extended. SDN runs from the initial data forwarding unit to support many functions, including packet filtering, distinguishing the service, multicast, and QoS. However, traditional networking devices also build many complicated protocols, increasing the difficulty in deploying new protocols and leading to the rigidity of the existing network system. To solve these problems, control network architecture forward separation has been proposed. Namely, high-level strategies are deployed by the proprietary equipment which is used to guide the data forwarding of the network equipment. Doing so can reduce many complicated functions of the network equipment and improve the flexibility and operability of the implementation and deployment of new network technologies and protocols. The core ideas of SDN are logical control and data forwarding separation.

Because the current network logic decision plane is combined with distributed hardware devices, there are inflexible problems in the traditional architecture. To overcome the limitations of traditional networks, some novel networking architectures have been proposed. A typical system consists of decision, dissemination, discovery, and data (4D) architecture, secure architecture for the networked enterprise (SANE)/Ethane (Casado et al., 2007), network operating system (NOX) (Gude et al., 2008), forwarding and control element separation (ForCES), and OpenFlow (Mckeown et al., 2008). Greenberg et al. (2005) redesigned the Internet control and management structure and put forward the 4D architecture. Under the 4D architecture, the decision plane makes network control decisions through a global network view to the data plane directly, the dissemination plane establishes a reliable communication channel between the decision plane and routers, the discovery plane is responsible for finding the physical components of the network and provides the basic data of building a network view to the decision plane, and the data plane is used to realize the data forwarding functions. This architecture helps implementing a secure and robust network. SANE and Ethane are the management architecture for the enterprise network. SANE defines a protective layer used to manage all connections between the link layer and Internet protocol (IP) layer. All routing and access control decisions are controlled by a logical central server through this protective layer. The key lies in the safety control but does not realize the complex routing decisions. It does not deal with extensive text either. Therefore, the actual deployment is difficult. Ethane expands its functions based on SANE, which adds the security management strategy into the network management. Its main components are a central controller and Ethane switch, which lay the technical foundation of OpenFlow. NOX tries to change the plight of the current network system by establishing a network operating system. It does not provide the network management but provides the generic interface as much as possible for running its various applications. NOX includes switches, controllers (running NOX and multiple corresponding management applications), and network views. The network view provides a different observation and abstract analysis of network physical resources. Maestro, proposed almost simultaneously with NOX, can also be seen as an implementation of network operating systems, but it focuses more on controlling interactions between applications. ForCES divides the network element into the control element (CE) and forwarding element (FE). Both of them do the collaboration and interaction by the ForCES protocol for improving the network control and enhancing the flexibility and effectiveness of network deployment. OpenFlow is composed mainly of an OpenFlow switch and a controller. The OpenFlow switch forwards data packets according to the flow table, which is the main component of the data forwarding plane. The controller realizes the control function through the whole network view, and its control logic represents the control plane. In a prototype implementation of SDN, OpenFlow represents the technical implementation of SDN control and forwarding separation architecture. Strictly speaking, OpenFlow is one of the communication protocols between the control plane and data plane of SDN. OpenFlow with good flexibility and normativity has been seen as the de facto standard of SDN's communication protocol. In particular, Nunes et al. (2014) pointed out the difference between ForCES and OpenFlow: In ForCES, though the internal network equipment was split into the control element and forwarding element, and both of them worked in the master-slave mode, they were placed into an entity inside. In OpenFlow, the control layer was removed from the network device completely. In addition, the ForCES protocol used for the FE and CE interaction was based on per-packet to deal with the network communication, but in OpenFlow, the concept of flow was introduced to forward a series of messages matching the datagram. However, both ForCES and OpenFlow can improve the flexibility and controllability of the network by endowing the network with the programming ability and a control forwarding separation idea.

At present, many organizations have joined in the research on SDN and its relevant standards. This has greatly promoted the development of SDN.

3 Reference architecture

SDN's architecture design is the basis of its implementation. According to different requirements,

corresponding SDN reference architectures have been put forward.

SDN architecture (Fig. 2) was first proposed by the Open Networking Foundation (ONF) in a white paper (Open Networking Foundation, 2012). It is divided into a data plane, a control plane, and an application plane. The data plane is composed of network equipment such as switches, and is responsible mainly for the underlying operations such as data forwarding. The control plane contains the logic center controller, and is responsible for maintaining the global network view and running the control logic strategy. Moreover, the controller needs to abstract the network view into the network service and provide an interface for the third party to develop the application program and realize network management. The application plane contains a variety of SDN applications, so developers do not need to care about the technical details of the underlying devices but can realize the rapid deployment of new applications by simple programming. The SDN control-data plane interface is used for communication between the data and control planes. This interface is responsible for sending the forwarding rules generated in the control layer down to the data layer, and at the same time, sending the request of the data layer uploaded to the controller. Its main communication standard is the OpenFlow protocol. The northbound interface of the SDN is used for communication between the control and application planes. The controller provides the easy-to-use network resource abstraction for the upper application developers through this interface and allows the user to develop applications according to their needs. The SDN architecture proposed by ONF has been widely recognized in academia and industry.

Network function virtualization (NFV) was proposed by network operators (European Telecommunications Standards Institute, 2012). This technical standard with the aid of network virtualization technology is used for high capacity servers, memory, and switches with industry standards for various network software functions. NFV uses virtualization technology to establish a network virtual layer in hardware devices to virtualize the hardware resources, which are managed by the operators through software. NFV speeds up network deployment and adjustment, reduces the complexity of business deployment, and improves the unification and universalization of network equipment.

OpenDaylight architecture was proposed by equipment manufacturers (Linux Foundation, 2015). It is aimed to propel its specific implementation and deployment in the industry through SDN's open source development. Like ONF, this architecture is divided into application, control, and data layers. The control layer of OpenDaylight is directly implemented by the built-in JAVA virtual machine. Depending on requirements, the controller provides a series of pluggable modules and is compatible with third-party modules with strong flexibility. Unlike ONF, OpenDaylight introduces a service abstraction layer (SAL) between the control layer and data layer. This is used to convert all kinds of underlying protocols into the request service supported by the control layer. Therefore, this architecture can handle different standard protocols with strong extensibility.

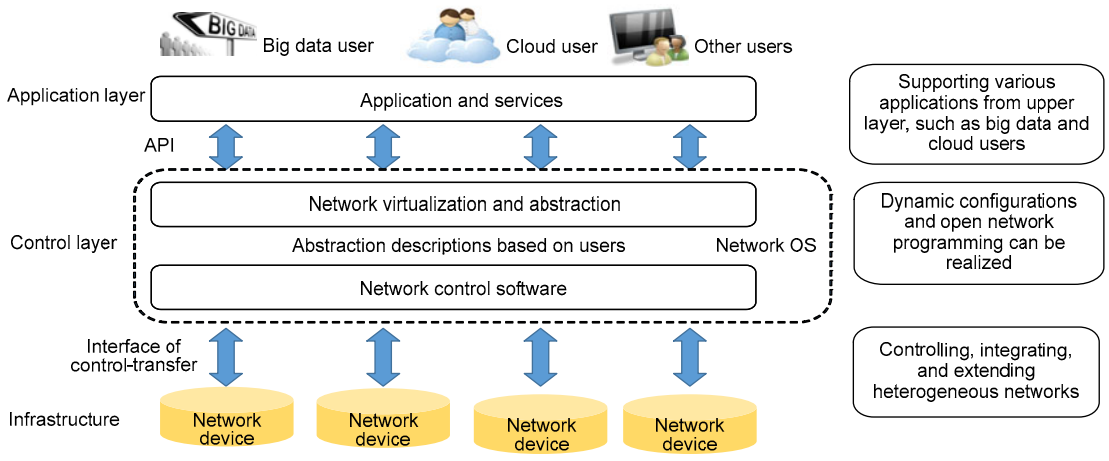

Fig. 2 Basic architecture of software-defined networking (SDN)

The comparison of the aforementioned three kinds of SDN architectures is shown in Table 1.

Because different architectures are proposed for different application scenarios, there are certain differences in emphasis and specific settings within the architectures. Gelberger et al. (2013) compared different complexity of SDN architectures with an index of throughput and delay, and the results showed that the SDN architecture with a higher complexity could support a stronger function and a higher flexibility, and this interior complexity did not mean it had depressed performance. Therefore, the design of SDN architecture should consider the actual business requirements instead of pursuing superficial simplicity blindly.

4 Key technologies

4.1 Data plane

SDN separates the data and control layers and extracts the control function of a traditional closed switch to the upper controller, thereby simplifying the switch function. Taking OpenFlow as an example, the switch forwarding data needs only to follow the control layer's decisions. For SDN, how to ensure the switch to match and forward data streams quickly and flexibly is the research core of the data layer.

4.1.1 Design of switch

A basic problem in the SDN data layer is how to achieve high performance and flexibility of the data stream in the process of data forwarding in an efficient way. High performance refers to high processing speed of a data flow, taking throughput and delay as the main parameters.

Flexibility refers to the extent to which the rule matching can be done flexibly. Usually, the switch design has two types: hardware and software. The hardware way has the advantages of fast speed, low cost, and low consumption, while the software way has the incomparable advantage of flexibility. How to design a switch that ensures both the hardware forwarding rate and the flexibility of data forwarding, is a fundamental and significant research topic.

Sezer et al. (2013) compared current mainstream technologies used for switch implementation, including general processors (CPUs/GPUs), network stream processors (NPUs/NFPs), programmable logic devices (PLDs), field programmable arrays (FPGAs), special standard products (ASSPs), and application-specific integrated circuits (ASICs), from perspectives of forwarding rate and flexibility of forwarding rules. Considering the balance between performance and flexibility, Sezer et al. (2013) proposed to incorporate the technologies properly to realize processing flexibility and to try to reduce the impact of inefficiency. Function implementation was carried out by both hardware and software reasonably, to provide SDN with an efficient solution in the data forwarding layer.

4.1.2 Implementation of the multi-level flow table in switch

In SDN, to avoid the frequent interaction between the switch and controller, a stream-based forwarding mechanism is adopted instead of a packet. However, with the expansion of the network function, the flow table size of the switch presents an increasing problem. The continuous extension of the forwarding plane abstract is difficult to store at the Ternaryu content addressable memory (TCAM) of the switch and this becomes a bottleneck in the development of the network. For a single flow table, a matching domain in multiple flows of the whole flow table generally exists repeatedly. Liu et al. (2014) put forward a high-efficiency OpenFlow multistage flow table mapping mechanism named TCAM. This mechanism performs orthogonal splitting on some matching domains in a single flow table, and compresses and stores them in multiple flow tables, effectively reducing the storage space of the flow table.

Although TCAM has acted as a virtual industry standard to find the forward information base, the

Table 1 Comparison of ONF-SDN, NFV, and OpenDaylight

SDN architecture	Interface standard	Organization	Key feature
ONF-SDN	OpenFlow	Open Networking Foundation	Control and data forwarding separation
NFV	Multiple interface protocols	European Telecommunications Standards Institute	Network function virtualization
OpenDaylight	Multiple interface protocols	Linux Foundation	Software open source and implementation

control plane needs to know the pipeline organization of each OpenFlow switch. This requires a corresponding extension of the control plane protocol. However, the versions of flow table structure of the OpenFlow switch are not the same. Thereby, it causes the control plane to maintain the heterotic flow table structure of the OpenFlow switch. This increases the maintenance cost and complexity of the control plane. So, the multistage flow table mechanism based on TCAM is not perfect. To solve this problem, the industry has made some beneficial explorations. ONF proposed the hardware abstraction layer (HAL) model to reconcile the different flow table structures supported by the switch hardware and controller. The HAL model provides a unified interface so that the controller can negotiate with the switch to determine the flow table structure supported by the switch hardware. Based on this, the controller can issue the rules of the corresponding structure. This method requires that both controllers and switches are upgraded to support the HAL model, so there are some problems of practicability. FlowAdapter (Pan et al., 2013) layered the switch to achieve a highly efficient and flexible multi-level flow table mechanism. The FlowAdapter switch is divided into three layers: (1) The upper software data plane is used to store the rules issued by the controller. (2) There is a relatively fixed hardware data plane at the bottom, which is responsible mainly for high-speed forwarding of packets. (3) In the middle there is a FlowAdapter layer used to realize the communication between the software and hardware data planes. Its operational mechanism is as follows: (1) The controller issues rules and the software data plane stores these rules and forms an *M*-phase flow table. (2) The FlowAdapter checks all rules of the software data plane. (3) The complete rules are used to convert the *M*-phase flow table into a one-phase flow table, convert the one-phase flow table into an *N*-phase flow table, and send it to the hardware data plane. This can solve the incompatibility problem of the multi-table pipeline structure between the switch hardware and the controller.

4.1.3 Trade-off in switch function settings

During the process of continuous development and optimization of an SDN architecture, there are some differences on whether more control functions should be added to the data layer of switch.

The initial SDN architecture separates the data and control. In the data layer, the dumb switch simplifies the traditional layer-2 switch, providing a simple data forwarding function to realize a fast matching and forwarding of the packet. The designer of Ethane believes that the best switch should contain few or no management function, and considers whether the core switch or edge switch should have the same "dumb" characteristics. Real improvement of network architecture can be achieved by allowing new features to be added to the centralized controller while keeping the switch focused on fast data forwarding. Similar to the idea of Ethane, the OpenFlow switch does not have any logic in the local control layer, and relies completely on the controller to fill its flow table to perform the task of data forwarding.

However, in the process of the development of SDN, there is an inconsistent trend with the above idea; namely, by giving some additional functions to switches, routers, and other data layer equipment, it makes the underlying network more intelligent, so as to improve the scalability of SDN. Sezer et al. (2013) pointed out that some query events (such as newly arrived flows) in the network could be completed by the node CPU rather than being forwarded to the upper layer and waiting for the controller to make decisions. In fact, distributed flow architecture for networked enterprises (DIFANE) and DevoFlow do exactly that. DIFANE hands flow build requests over to the authority switch, and DevoFlow considers the flow build request and obtains the statistics message to reduce directly the number of request packets. Both of them increase the control function of the data plane, reduce the data interaction between data and control planes, so as to reduce the burden on the control plane. Ali et al. (2015) deployed additional features such as customized protocol processing and path splicing on the forwarding device using programmable FPGA hardware. Cisco onePK platform attempts to provide a wider range of functions such as encryption, decoding, and deep packet inspection for packets forwarding. Narayana et al. (2014) designed an application extension framework to deploy middleware on a programmable switch to make it more versatile. Aiming at the shortcomings of the OpenFlow flexibility in programming, Hata (2013) designed a kind of SDN exchange platform by imitating the

active network architecture. This platform includes a built-in packet processor of a virtual CPU and dedicated memory. Through a secure channel, the user program can be downloaded from the controller to switch platform for storage space, and finally, its specific function can be realized by the packet processor operation. Kreutz et al. (2013) pointed out that the programmable ability of the extending data plane is very helpful in improving the reliability of SDN. It provides the switch with a higher programmability, which can give the switch a dynamic relation with more than one controller in a safe manner at the same time, effectively reducing the negative effect resulted from a single controller failure, improving the throughput of the control plane, and reducing the time delay. It is more helpful in improving the overall security of SDN, returning certain control power back to the network equipment of the data layer (Scott-Hayward et al., 2013). Resonance forces network devices to provide dynamic access control based on high-level security policies rather than attaching the security functions to the top of the network architecture (Nayak et al., 2009). If the network relies only on the centralized controller, there is a bottleneck in the network development. Nayak et al. (2009) used a protocol named ident++ to query some necessary information of the host and user and did the related decision forwarding to share the burden of the controller.

The aforementioned research shows that making the switch of the data plane to undertake some control functions in addition to forwarding data may be contrary to the original design of SDN, but it optimizes the SDN architecture, improves its extensibility, reliability, and safety to some extent. In the macro perspective of SDN architecture, it makes the issue of processing easier and more efficient. However, note that this approach will reduce the visibility of the controller to the data layer devices to some extent. To keep the switch of "simple" design principles or to realize a broader "simplicity" by paying a fee is a problem that should be weighed and balanced in future long-term practice.

4.2 Control plane

The control plane plays a vital role in the SDN architecture. Through its core component controller, the switch can be controlled logically, and the network can be managed safely and conveniently.

4.2.1 Scalability

In the controller, the network operating system (NOS) realizes the control logic function. NOX was the first to introduce this concept. NOS refers to the control software in SDN. Different logic control functions can be realized by running different applications on NOS. In the OpenFlow network based on NOX, NOX is the control core and manages the OpenFlow switch of the data plane. NOX maintains the basic information about the entire network by maintaining the global network view. Applications running on NOX manage and control the entire network and then manipulate the OpenFlow switch by invoking global data in the network view. From the perspective of function realization, NOX realizes the basic control function of the network, and provides a common application programming interface (API) for the OpenFlow network. However, it fails to provide sufficient reliability and flexibility to meet the needs of extensibility.

In addition, the original simple network can be controlled by a single controller. With the increase of the network scale and business requirements, it is necessary to study the scalability of the control plane, which means looking for a multi-controller solution. A lot of in-depth studies are required about how to control the number of units and realize their synergy and interaction of the network status, such as topology, transmission capacity, and routing restrictions, to ensure the consistency and scalability of the network status. In fact, there are two ways to extend the SDN control plane: vertical and horizontal extension. The scalability solutions (Table 2) for the SDN control plane are as follows:

1. DIFANE

As the granularity of flow processing continues to refine, more and more flow processing events increase the burden on the controller. Although the controller can deploy the control logic in advance to forward data units by a proactive decision-making mechanism, the change of the control logic is dynamic. Because the duration of most flows is actually very short, installing the flow chart in advance does not have the desired effect. Also, it will cause the waste of resources because a large flow of table space cannot be freed.

Table 2 Scalability solutions for the SDN control plane

Solution	Extension scheme	Popularity	Global status maintenance	Complexity	Main technology
DIFANE	Vertical	Lower	No need	Middle	Controller installs the partition rules and implements rule installation by authority switch
DevoFlow	Vertical	Low	No need	Middle	Rule copy, FRR, and multipath support
HyperFlow	Horizontal	Higher	Need	Middle	Distributed storage system to maintain a global network view
Onix	Horizontal	High	Need	Higher	NIB maintains global network view; Onix forms instance clustering
Kandoo	−	High	Partial need	Middle	Control layer is composed of local and root controllers

To solve this problem, DIFANE combines proactive and reactive installation of flow meters to keep the flow as much as possible in the data plane, to reduce the load of the controller. DIFANE selects the authority switch from the OpenFlow switch and lets each authority switch manage the OpenFlow switch within a certain area. The controller installs the partition rules on the OpenFlow switch and the authority rules on the authority switch according to the global network information. When the ordinary switch receives a new data flow, it communicates with the partition authority switch directly according to the partition rule. Then the authority switch installs caching rules to the ordinary switch, and at the same time forwards the request data directly to the destination rather than the source switch. This approach reduces the control flow that the controller needs to process in real time. Generally, the partition rule of DIFANE and the authority rule of the authority switch need to be processed only when the network changes, so it does not need to be updated frequently, thus reducing the load on the controller.

2. DevoFlow

To reduce the information interaction between the OpenFlow switch and controller, DevoFlow uses two approaches: regular replication and local operation.

Rule copy mode increases the clone mark on the action field of the flow table item. If this mark is zero, the matching message of the flow table will be processed normally. If this mark is set, it will establish an exact microflow to refine each microflow statistics. In this way, the ordinary flow table is implemented by the TCAM hardware, and the microflow flow table is implemented by the software. The OpenFlow switch needs only to install the ordinary flow table in advance and can substantially reduce the message interaction with the controller, reducing the resource consumption of the TCAM hardware at the same time.

Local operation includes multipath support and rerouting. The former provides multiple possible output ports for the replicable wildcard flow table items in the OpenFlow switch. The latter provides one or more alternate paths for the switch to remove high-priority flow table items when the link fails.

In addition, DevoFlow takes steps such as sampling, and approximates statistics to reduce the cost of the controller in the process of statistical information collection.

DIFANE hands the cost of the flow-establishing request over to the authority switch for sharing the controller's cost, while DevoFlow considers two processes of the flow-establishing request and obtaining the flow statistics. DevoFlow reduces the number of request messages directly so as to reduce the cost of the controller from the root. Both DIFANE and DevoFlow are vertical expansion schemes.

The vertical expansion scheme is still a centralized controller with a single structure. Initial concern about the single controller comes from the potential bottleneck in its performance. Heller et al. (2012) and Nunes et al. (2014) pointed out that in fact "A single controller can handle an alarming number of the flow of request events. Its processing capacity in theory can satisfy most of the network." However, to reduce the time delay and improve the fault tolerance, there is an inevitable trend to adopt the distributed controller of the horizontal extension scheme.

3. HyperFlow

HyperFlow manages the OpenFlow switch by deploying multiple controllers. Given that each controller can synchronize the global network view at the same time, it needs only to manage the OpenFlow switch in a particular area. Network events are transmitted through the publish-subscribe pattern between controllers. However, the global network view updates and the cycle time of network status information are related to the transmission delay. Therefore, this method may face performance bottlenecks in a data center or a large-scale network.

4. Onix

Onix uses a distributed SDN deployment scheme for large-scale networks. It uses a distributed architecture to provide the programming interface of the network state to the upper control logic. The core of this scheme is using the network information base (NIB) to maintain the global network state and distribute the network state. The entire Onix network architecture is implemented in a hierarchical topology.

5. Kandoo

Unlike DIFANE and DevoFlow which make the switch assume functions to share the cost of the controller, Kandoo (Yeganeh and Ganjali, 2012) insures the original data layer unchanged but makes the control plane hierarchical. The local controller near the switch deployment process does not need the high-frequency request events of the global network state. The root controller handles only a handful of global network state events. The communication between the root and local controllers is implemented in the manner of event subscription asynchronously.

After analyzing the root of SDN scalability, Yeganeh et al. (2013) pointed out that the factors limiting SDN scalability not only exist in the SDN architecture. The traditional network is facing the same challenge. With continuous optimization of solutions and the performance improvement of SDN related hardware, SDN scalability will be gradually developed to meet the requirements of most network environments.

4.2.2 Controller deployment

The deployment of the controller will greatly affect the control plane's ability to respond to network events. Understanding "where to place" and "how many to use" is the key to improve the overall performance of SDN and the fault tolerance of the control plane.

Heller et al. (2012) studied the controller deployment around the delay from the node to the controller. They pointed out that the delay between the control plane and the data plane in SDN would affect whether the control logic could be sent to the forwarding device effectively and in a timely manner. Therefore, Heller et al. (2012) introduced two indicators, average-case latency and worst-case latency, to analyze the deployment position of an OpenFlow controller on Internet 2. The analysis of time delay showed that the number and location of the controllers would influence both indices. Analysis also indicated that taking "whether deploying k controllers can make average delay or delay in the worst case reduce to $1/k$," namely the cost-benefit ratio, as the analysis emphasis, the ratio tended to be stable with three or four controllers. For most topological structures, increasing the number of controllers can make the delay basically conform to the rule of proportional reduction. Heller et al. (2012) also pointed out that the optimal deployment scheme of the controller could be found by calculation if a certain delay index was satisfied.

4.2.3 Control logic consistency

Research on the logical consistency of the control layer usually involves two problems. The first is the inconsistency caused in the policy update process. The second is the inconsistency caused by the policy conflict. Causes and solutions of consistency problems in the control layer are discussed from these two perspectives.

SDN dissociates the control layer from the data forwarding plane and operates the switching equipment by the logic centralized controller. However, a time delay exists inevitably between the controller and switch, and this will probably affect the receive event order of the controller and the installation order of the controller rule on the switch. On the other hand, the execution order of the control logic itself may also lead to the network being in a temporary status. It means that part of the packet is operated according to the new rule and the other part is still operated by the old configuration. Developers need to consider carefully the order, in which each update step is executed,

and find a security sequence that allows the switch to be updated according to the specific steps in that sequence. However, this way is extremely complex. The intermediate steps in the process of updating will take a lot of flow table space and they may go against some important network characteristics such as access control policy when performing these intermediate steps. In some cases, such a safe sequence will not exist.

For the consistency of network control logic update, Reitblatt et al. (2011, 2012) proposed the per-packet consistency and per-flow consistency. The consistency of per-packet means that each packet performs either the old control logic or the new control logic in the transmission process, and it is impossible to perform them together. Note that per-packet consistency is stronger than the atomic update mechanism. Even if the rules in all switches are set up and completed at the same time, this may lead to the transmission packet performing with a different control logic. The consistency of per-flow means that all packets in the same flow cannot be performed by the mixed old and new control logics during transmission. Reitblatt et al. (2012) proved that the per-packet consistency could guarantee that the network properties remained unchanged before and after the update, and a two-phase update method was proposed to achieve per-packet consistency.

Another perspective on the consistency of SDN is the inconsistency caused by its policy conflicts. This situation often occurs when the controller is used by different users and applications, or different controllers are running in the same field. It can further cause problems of loops, block holes, and access control violations. Attackers may even use this potential vulnerability of inconsistency to bypass the security policy, and redefine the new flow rules to operate the network data for their own purposes (e.g., data theft).

NICE combines two technologies, model checking and symbolic execution, to detect the correctness of applications running on the controller. FlowChecker uses FlowVisor to ensure the validity of the multi-controller OpenFlow network configuration. In this scheme, the main controller (MC) is responsible for running FlowChecker and communicating with the NOX controller. After checking, it can do the analysis and validation. FlowVisor can also intercept the requests from FlowChecker and reply to them. FLOVER translates the security policies into a series of sub-policies that can be run and validated by satisfiability modulo theories (SMT). The difference in implementation between FlowChecker and FLOVER is that FLOVER supports checking for policy conflict events caused by operations including "set" and "go to," meaning that the checking scope of FLOVER is wider. FortNOX (Porras et al., 2012) is a security service agency based on NOX which directly integrates with the NOX controller. It converts all rules into new rule alias reduced rules (ARR) that are easy to verify. Once the conflict is detected, these rules will be chosen according to the security authorization level of the requester, so the inconsistency problem can be avoided. As the middle layer of the control plane and data plane, VeriFlow can check the invariants in the network when new forwarding rules are issued and realize real-time dynamic monitoring of the consistency of logical policies in the network. Hu et al. (2015) designed a better integrated security framework of SDN, which increased features such as the role of the policy creator and the security privilege level of role, to ensure that the policies released by decision-makers at higher security levels could be implemented preferentially. Thus, it will avoid the problem that the security policy may be overwritten by a common application policy to some extent when policy conflicts occur.

4.2.4 Control sharing of the multi-controller to switch

To enable the controller to be deployed directly in the real network, network virtualization technology is used to solve the problem of control sharing of the multi-controller to switch. Through this technique, multiple logically independent networks can share the same physical infrastructure. FlowVisor achieves the virtual layer based on OpenFlow between the controller and OpenFlow switch. It enables the forwarding plane of hardware to be shared by multiple logical network slices, and each network slice has its own forwarding logic policy. In this mode, multiple controllers can manage one switch at the same time. Researchers can run multiple network experiments in the same network, and the network administrator can in parallel control the network. So, the network normal flow can run in the independent slice mode and ensure that the normal flow is not disturbed.

4.3 Design of the interface and protocol

Interface is a bridge that connects all layers in the SDN architecture. It takes the controller as the logical center with the south interface responsible for communicating with the data plane and the north interface responsible for communicating with the application layer. In the distributed multi-controller model, the east–west interface is also needed to complete the interaction between multiple controllers.

As the communication medium between the controller and data forwarding device, the south interface is the key to realizing the decoupling and integration of the control and data. Logically, it not only ensures normal communication between the data layer and control layer, but also supports the independent update between these two layers. There are relatively mature interface standards for the south interface, among which the OpenFlow protocol proposed by ONF has become the data layer–control layer interface protocol that is widely used. OpenFlow matches rules based on the concept of flow. The switch needs to maintain a flow table to support OpenFlow and forward data according to the flow table. The controller is responsible for establishing, maintaining, and issuing the flow table. With the continuous development of related work, the OpenFlow protocol gradually supports multi-table and an action set to solve the problem of combinatorial explosion, to increase the support for IPv6 to be better integrated with the next generation Internet, and support flow control mechanisms to mitigate network congestion. The support functions of OpenFlow are becoming more and more detailed and comprehensive, and the mechanism is being improved.

The north interface is responsible for communication between the control layer and various business applications. Each business of the application layer calls the required network abstract resources through programming, grasps the global network information, and facilitates users in quickly promoting the network configuration, application deployment, and other businesses. However, because of the diversity of application business, it is difficult to unify the north interface, and this causes great difficulty for development. Some researchers proposed to use network configuration language to describe policies. Voellmy et al. (2012) set up a policy layer on the controller for configuration files, graphical user interface (GUI), and interaction with external sensors. This policy layer is responsible for converting advanced logical policy to flow constraints used by the controller. The Frenetic language is important for the north interface. It implements an abstraction of policy consistency. Frenetic requires each configuration program list a series of policy clauses for each switch. When being executed, these items are matching in turn according to the priority to solve the policy conflicts of the north interface. At present, the north interface has not been unified, and research on the north interface is still the major promotion work in industry.

The east–west interface is responsible for communication between controllers. The establishment idea of the east–west interface standard is that the controller has expandable capability and can further provide technical support for load balancing and performance improvement.

In addition to the above interfaces, Scott-Hayward (2015) mentioned another interface of the controller, GUI. GUI is usually used to provide an intuitive network topology, network device information, and flow table details for simplifying network management. OpenDaylight, open network operating system (ONOS), and Ryu provide GUI to facilitate user interaction with the controller. Note that the access to and operation of GUI should be strictly restricted.

5 Cyber security

As a brand-new kind of network architecture, SDN has the characteristics of obtaining a global network view and centralized control of the network. It has shown great advantages in improving the overall security of network, and has provided a brand-new solution to assure network security (details are discussed in Section 6.1). However, with the gradual marketization of the SDN technology and products, the security problem has become a key factor restricting the wide application. Comparing SDN with other traditional network architectures, Kreutz et al. (2013) pointed out that the programmability and centralized control logic, as SDN's advantages, may nevertheless bring many new security threats. In a sense, the new characteristics of SDN could lead to a security situation worse than that of

the traditional network. In recent years, more and more researchers have begun to study the security threat of SDN technology, and made some progress.

Corresponding to the hierarchical structure of SDN, the potential security threats of the data layer, control layer, application layer, and north–south interface will be analyzed.

5.1 Security problem in the data plane

5.1.1 False flow injection

A malfunctioning network device or malicious user may trigger this threat. An attacker can use switches, servers, personal computers, and other devices to generate a large amount of useless traffic and launch denial-of-service (DoS) attack to the network switch (details on the controller DoS/DDoS are discussed in Section 5.2.1) such that the switch TCAM resources are exhausted and the normal flow in the network cannot be processed. A simple identification mechanism can prevent this kind of attack to some extent. However, this authentication mechanism will no longer be effective if an attacker gains control of an application server by an abnormal means and thus obtains more details of the user's identity. Using the intrusion detection system (IDS) to support root-cause analysis can help identify the abnormal traffic flow. This system can also be combined with the dynamic control of the switch action mechanism to test this type of attack.

5.1.2 Attack against switch

Attacks against switches can lead to malicious packet loss or packet forwarding delay, unauthorized copying or redirected traffic (typically used for data theft), and even the situation where the attacked controller overloads by being injected a large number of malicious flow requests such that the controller cannot work normally.

5.1.3 Data layer fault recovery

Similar to a traditional network, SDN also faces the problem of network node or link failure. In this situation, the SDN controller can recover the failure nodes faster through the global network information. Usually, the convergence process of the network node is: (1) When a switch fails, other switches detect changes. (2) The controller is informed of the changes by switches. (3) The controller recalculates the restored rules from the global state information. (4) Updated rules are sent to the affected network elements in the data plane. (5) Affected devices update their flow table information separately. The above discussion assumes that the secure channel between the controller and switches can be used normally. If the failure of data layer makes the switches and the controller fail to interact, the convergence process is relatively difficult. If so, the traditional interior gateway protocol (IGP) (such as open shortest path first) can be adopted for a flooding recovery.

For this problem, the internal switch should deploy a disaster recovery mechanism when designing SDN. For example, the controller installs the backup path in the flow chart in advance, which can make the switch forward the temporary traffic in the backup path if the original path is unavailable. A similar multi-path support mechanism is provided by Onix to enhance the fault tolerance of the data layer.

Reitblatt et al. (2012) used the new version of the OpenFlow statute that allowed defining multiple corresponding forward behaviors for switches in different conditions and under particular parameters to propose the high-level programming language fault tolerating regular expressions (FatTire). Using FatTire, programmers need only to specify a strategy for switches, and then the controller calculates the appropriate backup path and sends the corresponding flow rules to the switch. When an emergency occurs, the switch can automatically start using the backup rules without the intervention of the controller, to achieve a rapid convergence.

5.2 Security problems in the control plane

5.2.1 Denial-of-service attack

Flooding based distributed denial-of-service (DDoS) not only seriously affects the normal operation of the controller, but also prevents the switch from transmitting data according to the rules issued by the controller. In this attack mode, the attacker first invades some hosts in the network and changes them into puppet machines. Then it pours a mass of invalid forged flow into the network (the forged flow tends to use false source addresses to hide the true information). Under the OpenFlow protocol, when receiving the unknown flow, the switch forwards it to the controller and requests the controller to issue the appropriate flow rules. However, a large number of

requests and packet flows rushing into the controller in a short period of time would lead to the withdrawal of the normal processing ability of the controller. At the same time, the normal flow rules cannot be issued because of channel congestion. The invalid flow can fill the accepted cache of the switch, and the normal data flow is discarded directly, leading to the network paralysis. DDoS attacks can be mitigated by detecting and isolating malicious traffic from normal traffic. The Kandoo framework can restrict frequent access to the controller, so the controller does not need to process a large number of requests such as the unknown flow rules request in a short period of time, thus preventing the DoS attack to some extent. A lightweight DDoS detection method proposed by AVANT-GUARD, Defense4All, and Roduigo Brage can also be used to detect and control this kind of attack (see Section 6.1.1 for details).

5.2.2 Overwriting of the security policy

The overwriting of mandatory security policies in the controller has a significant impact on the security of SDN. Hu et al. (2015) designed a comparatively perfect integrated framework of the SDN safety strategy in the controller to increase the features of the role of the policy creator and the security privilege level of the role, which can ensure priority implementation of policies generated by high-level decision-makers. Thus, overwriting of the security policies is avoided to some extent. FortNOX converts all rules to new rules of ARRs which can be easily verified. Once the conflict is detected, the rules will be accepted or rejected according to the safety authorization level of the requesting party, to avoid the security policies that are covered by ordinary application policies. FRESCO (Shin et al., 2013) directly integrates a security enforcement kernel (SEK) in the OpenFlow controller to verify the signature of conflict rules, and to ensure that the flow rules generated by the security services are not covered by the rules of other non-security applications.

5.2.3 Control program isolation

Some control procedures are running in the SDN controller to provide all kinds of available network services to the upper layer of business and carry out some basic control to the lower layer of the network. In the security assessment of OpenFlow, Benton et al. (2013) pointed out that the isolation of these control procedures is an indispensable measure to guarantee network security. Isolation of the control program here refers to the provision of a logical space to the application running on the controller, to further support the identification of each control program and the rank authorization mechanism based on trust. This approach aims to provide the controller with a security assurance by ensuring that the error of one program does not harm the whole controller. In ROSEMARY (Scott-Hayward, 2015), each OpenFlow application is regarded as an independent ROSEMARY instance running on the controller. At the same time, these applications are running in sandboxes to protect the controller and avoid the hazard from any malicious operation of an attacked application.

5.2.4 Illegal access

Illegal access control is one of the security risks of an SDN controller. Once the controller is illegally controlled, the attacker can build botnet by controlling the underlying facilities to further expand the scope of the attack. So, the access of the SDN controller should be strictly restricted. Role-based access control (RBAC) policy based on the user role can prevent unauthorized access to the SDN controller by setting specific access rights and authenticating users. In addition, a proxy controller can be set up on the northward channel to complete some interactive operations instead of the controller, which can prevent third-party applications or users from accessing the controller directly from the outside world. The network equipment, which is the channel of the controller and the data layer, should be regularly registered. The new equipment needs to have a strict examination before building a connection with the controller to prevent illegal devices to access networks. However, Sezer et al. (2013) pointed out that in the case of multiple controllers communicating with a node or applications, the authorization and access control between them would be very complex.

5.2.5 Illegal access to controller data

The controller stores much important information such as the entire network status. Therefore, it is necessary to protect the control program of the controller and its information database to prevent the

attacker from stealing data or tampering with the control procedures. For this problem, we can take the methods of the important data backup, cyclically changing the default storage path, encrypting data (such as secret homomorphic technology), or storing sensitive information in the security mediation secondary way to ensure information security.

5.2.6 Failure recovery

If the controller fails because of a failure or an attack, an effective recovery mechanism should be adopted to ensure that the controller can resume normal operation within a short period of time and reduce the negative impact on the whole network.

In ONOS (Linux Foundation, 2015), the multiple ONOS instances are interconnected to form an ONOS cluster, and each of these ONOS instances serves as an exclusive master of a set of switches. Once an ONOS instance fails, the remaining ONOS controllers will redesign a new master controller for the affected switch to ensure the efficient operation of the whole control layer.

Data backup is important for recovering from disasters rapidly. Kreutz et al. (2013) abstracted the backup method into two models: crash fault tolerance service (CTS) and Byzantine fault tolerance (BFT) models. The former supports only benign errors such as program errors, and operating system or machine failures. The latter supports any exception caused by malicious or incorrect operations and properly handles these errors through a state machine backup. Ethane proposed three fault-tolerant backup mechanisms at the design time, namely the cold-standby approach, warm-standby approach, and fully replicated approach. SMaRtLight (Botelho et al., 2014) is a structured model of fault-tolerant backup. In SMaRtLight, only one main controller is responsible for the management of the network. The other controller as a backup is running a fault detection and leader election algorithm, to elect a new controller in time to replace the main controller when it fails. To ensure the smooth transition of old and new controllers, SMaRtLight lets all controllers in a shared data store maintain an NIB associated with the network and the application status. The main controller is responsible for updating NIB constantly to ensure that NIB of each controller is consistent with the current network status. In this way, when the new master controller first takes over the network, it needs only to read the network and application state information from the shared area.

As the core layer of the SDN architecture, the controller's security is vitally important to the overall security of SDN. Kreutz et al. (2013) proposed three main principles when designing the SDN control layer: (1) backup, namely redundancy deployment of the controller and application; (2) diversity, namely trying to obtain the same control program running on different controllers to avoid the error of the same model; (3) dynamic device, namely allowing the switch dynamic associated with more than one controller to prevent controller failure in the single correlation pattern.

5.3 Security problem in the application plane

When a new application is connected to SDN, SDN needs to carry out strict audit and certification to prevent malicious applications from running on the network. However, there are various applications based on SDN, and many application auditing models in the traditional network are only for a specific application and lack universality and flexibility. Therefore, there is no effective and reliable application auditing model for SDN. This is an urgent problem to be solved.

5.4 Security of network interfaces

5.4.1 South interface security threats

Currently, the mainstream south interface protocol is the OpenFlow protocol. To enhance the security of the channel between the controller and switch, the OpenFlow1.3.0 version provides a certificate verification service between transport layer security (TLS) and communication entities. However, this security feature is optional, and details such as the concrete implementation and certificate format have not been described. In fact, the secure sockets layer (SSL)/TLS protocol itself is not secure and is vulnerable to attack by an intermediary. The exchange of certificate for mutual authentication between the controller and switch is also not reliable. The attacker can use the security vulnerabilities of SSL/TLS to steal and forge certificates from the channel, and then establish the connection with the controller and obtain the required sensitive information.

5.4.2 North interface security threats

The north channel refers mainly to the API that the control layer opens to the third parties, through which users can develop application services and deploy their own applications on SDN. In the design of the north interface API, the security and robustness of the API should be paid attention to in addition to ensuring that the API execution process is correct. Security threats to the north interface exist primarily in the deployed API. Hu et al. (2015) indicated that the attacked application server may influence the generation of flow table entries through the north interface to carry out malicious attempts. To avoid this, authorization to the API should be supported. The authorization framework of API references IETF OAuth2.0. In particular, the API based on RESTful can reference OMA Autho4APIv1.0.0. Wen et al. (2013) designed PermOpenFlow to provide that the controller running in the kernel module can be directly called by the third-party applications. At the same time, the scheme defines a series of fine-grained permission categories for strict control over access to applications.

To ensure the safe and reliable operation of a future SDN network, it is necessary to draw lessons from the current transmission control protocol (TCP)/IP system. From the start, the corresponding security concept and mechanism should be integrated into the design of SDN, rather than waiting until a large number of security vulnerabilities have been exposed and then attempting to add various security devices. In fact, this will incur higher costs.

6 Application scenarios

The characteristics of SDN, which are acquiring a global network view through the controller and trying to centrally control the network, offer great advantages compared with the traditional network architecture when solving many problems. It has a broad application prospect.

6.1 Improving network security performance

Programmability and global network view are characteristics of SDN, which in a sense can bring their own new security threats. However, these characteristics make SDN able to programmatically monitor the network state and control network behavior in real time, thus bringing a new possibility to improve the network security performance. At present, the studies on SDN security can be roughly divided into two categories: SDN enhances network security and SDN provides security services. Some typical security threats and solutions are listed in Table 3.

Table 3 Typical security threats and solutions

Security threat	Security solution
Illegal programming	Novel access control
Suspicious flow	Flow identification at the southbound interface
Distributed denial-of-service	Security solutions based on statistics and learning
Threats from outside networks	SDN firewall

6.1.1 SDN enhancing network security

The unique architecture of SDN enables it to monitor, analyze, and respond to various events in the network at a certain height. The core of this system is the controller. The controller can collect flow statistics from network devices periodically to obtain a centralized and real-time network state view. Developers can read the network state through an open API and use these resources to write automated control programs to achieve efficient network management.

A traditional network uses middleware to provide a network security function. In fact, integrating these middleware into SDN and using the programmability of SDN to redirect the suspicious flow with some characteristics to the middleware for processing can achieve a better effect. In the Slick architecture, the controller can install the necessary functions for middleware based on the security requirements of the application and make it further process the specific flows. FlowTags (Fayazbakhsh et al., 2013) makes a small change to the traditional middleware to make it interact with SDN by FlowTags API. FlowTags containing the data flow information is set in a packet to track and route control the suspicious flow. By providing a policy enforcement layer, SIMPLE (Qazi et al., 2013) makes the network administrator need only to specify the middleware passing route in the logic, and then converts the strategy to the forwarding rule of the switch automatically at this layer. At the same time, SIMPLE generats the flow rule which is related to the physical path and also considers the load

balancing of flow in each middleware. The control program based on NOX can redirect the flow to the appropriate middleware and force packets in the flow to be detected, to make up for the data shortcoming of NOX, checking only the first few hundred bytes of flow. However, for this idea, the reasonable deployment of middleware and network performance loss, caused by the additional link overhead when the flow is transferred to the middleware for inspection, need to be further studied.

The application of SDN's special architecture in alleviating DoS attacks has also attracted attention. RadWare (Ali et al., 2015) developed the first commercial SDN application, DefenseFlow, to confront DoS attacks. DefenseFlow can make the controller collect the flow data from the forwarding device in the space within 1 s. Then it measures the benchmark parameters of flow and compares them with parameters under the DoS attack to determine the abnormal flow. Once an attack is detected, the suspicious flow will be transferred to the scrubbing center by the flow transfer mechanism for further detection. AVANT-GUARD extends the data layer and makes the switch act as an agent to conduct the TCP handshake with the data sender. Until this phase, the flow request is forwarded to the controller to respond to the saturated flow attack such as the synchronous (SYN) Flood. Braga et al. (2010) obtained the flow information of the OpenFlow exchange device and took out the characteristics of flow such as flow rate, single-flow table rate, and flow characteristic entropy, to analyze the test samples. Once the DDoS attack is detected, the policies of shunt or speed limit are selected according to requirements to alleviate the DDoS attacks effectively.

The OpenFlow equipment and Floodlight controller were used to create a firewall based on SDN (Cheng et al., 2015). This firewall is implemented as SDN application software, which can control all switches in the network by the programming interface of the SDN controller. Therefore, a firewall application system which can flexibly respond to different security requirements has been built.

In general, SDN is more flexible in dealing with threats than the traditional network. Due to the real-time programming and control capability of SDN, the emergencies response mode of SDN also presents diversity and dynamics. These responses include emergency alarms, dynamic quarantine solutions, traffic redirection, and entrapment mechanisms.

6.1.2 SDN providing security services

In addition to enhancing network security, SDN can be used to provide additional security services.

Static IP address allocation can lead to the leakage of user information to some extent, which may lead to the user being attacked actively. Existing schemes that provide anonymous services, such as ToR and Tarzan, are inefficient and have large delays. AnornyFlow provides a service that is similar to caller-ID blocking; namely, third-party applications can no longer access the IP address of the Internet service provider (ISP) assigned to users, and thereby AnornyFlow can no longer associate the user flow with a specific IP address. OpenFlow random host mutation (RHM) (Jafarian et al., 2012) is the moving target defense technology based on OpenFlow. OpenFlow RHM can complete transformation between the real IP address (rIP) and virtual IP address (vIP) of a host in a fixed time interval at a high frequency, while maintaining a high unpredictability. In this scheme, each host belongs to a subnet. The unused IP addresses are assigned to each subnet under certain constraints. Each host will associate with a new vIP by a blind mutation or weighted mutation algorithm after each conversion interval. During the whole process, the NOX controller is responsible for IP conversion, corresponding flow table installation, and domain name system (DNS) update, and the conversion process is transparent to the host. This scheme can resist scanning-based attacks such as worm and address detection.

SDN's centralized control logic architecture provides a simpler and easier implementation for security service outsourcing. For the security status of family and small commercial networks, which are vulnerable to attack because of the lack of professional protection equipment, Kim and Feamster (2013) pointed out that such users could use SDN to outsource their own networks to the third-party professional security service providers. The basic implementation of this model is to make the programmable network access point of the home network periodically distribute flow data to the remote controller, and then provide security services such as spam filtering and zombie network monitoring by the running

algorithms of the controller, and finally implement the corresponding policies to adjust the network to respond to the potential threats.

SDN can also be used to implement secure data offloading on mobile phones or other handheld devices. The enterprise-centric offloading system (ECOS) is a secure solution for intelligent data diversion in the enterprise scope. The network controller negotiates with the mobile application on security, energy consumption, delay, and other issues, and according to their corresponding requirements, uses the verification resources in the enterprise resource pool to process the split data. These split data, according to their sensitivity, are divided into three different types, user-, enterprise-, and non-private, for the controller realizing the personalized security service policy.

6.2 Cloud network and data center

The increasing demand for virtualization and cloud services has attracted much attention in industry and academia. The development requirements of rapid service provision, efficient resource management, and extensibility, which are difficult to satisfy with a traditional network, can be solved under the framework of SDN.

Data flow is large in a data center and the hierarchy management structure of the switch is complex. So, it is easy to cause network congestion and performance bottlenecks if efficient addressing and data transfer cannot be carried out in a large server cluster. New features of SDN can better realize the functions of efficient addressing, optimized path transmission, and load balancing, and further improve the efficiency of data exchange and increase the controllability of the data center. Tavakoli et al. (2009) introduced the OpenFlow technology to a data center network, and the NOX controller was adopted to realize the PortLand and VL2, which are currently used more widely in the data center for addressing and routing mechanism. They also pointed out that using the SDN technology could simplify the realization of these two solutions and provide some additional extension services.

In the data center, servers and virtual machines need rapid configuration and data migration. Virtual machine migration strategy synchronous following needs to move applications from one location to another under the premise of uninterrupted service. However, the applications are usually composed of many virtual machines (VM) and their basic functions of availability, access control, and QoS is dependent on the underlying network, which makes the real-time migration difficult. Live migration of ensembles (LIME) (Keller et al., 2012) provides a solution to this problem using SDN and its virtualization technology. LIME runs on a common network virtual layer such as FlowVisor and FlowN to block the migration process for applications on the controller. The state of the data plane is first copied to a new set of switches during migration, and then the VM is migrated incrementally. The invariance of application logic connection is maintained by merging two instances on the same switch according to the transformation mechanism of packet forwarding rules. With the support of the SDN technology, LiveCloud (Wang et al., 2012) realizes a reasonable layout of network resources and provides service-level agreement (SLA) guarantee on the bandwidth and time delay by providing three different points of view, namely physical, logical, and tenants, which are components of the network topology and service components.

Because of the characteristics of large scale, complex configuration, and being dynamic, a cloud computing network faces many difficulties in deploying existing security devices in this environment. Shin and Gu (2012) proposed a new architecture of CloudWatcher for deploying security services in the cloud environment. Under this framework, cloud administrators can use a simple policy scripting language to change the flow path according to the specific requirements and transport them forcibly to the safety equipment for checking, to realize granular flow monitoring in the dynamic cloud environment.

Since the stability and efficiency of large-scale network services are often at the expense of waste energy, energy saving has become an important issue in data center and cloud network research. However, only a small amount of energy can be saved by turning off ports that have no flow for a while. A more effective method is to master the global network information through SDN, which can close temporarily unused devices and open them when needed. Low usage also leads to high energy consumption in data centers. In the data center, each flow can increase the use of routing links by exclusive routing at each time slice. To master the whole network information using SDN, each flow can be scheduled fairly, and the

routing links can be used fully, so that the energy consumption of large-scale networks can be reduced.

6.3 Enterprise and campus networks

The enterprise and campus networks are usually large in scale and run various applications and heterogeneous network protocols, so there is a high demand for the reliability and security of the network. However, most of the existing solutions have weak control and complicated configuration, and are error prone and of high cost. SDN can achieve more effective and personalized management according to the different demands of the enterprise and campus networks.

SANE/Ethane aims mainly at the management of enterprise networks, in which SANE focuses on security control, but does not realize complex routing decisions, nor does it undergo large-scale testing, so the actual deployment is difficult. Ethane extends SANE, and requests that the controller, host, and the user must prove their identity and registration. At the same time, the packets are requested from a registered entity, which does not allow any unauthorized communication generated between any end system of the network. This would provide a strong security guarantee to the flow management and control in the enterprise network. Resonance asks the controller to maintain a state machine for each access network device. The underlying programmable network equipment with the function of security alarm analysis will process the flow according to the instructions of the controller, the host current state, and the security level. Thus, a flexible and convenient access control management can be realized between the dynamic state transitions of the host.

CERNET2 connects multiple colleges and universities using the 4over6 technology, provides services such as IPv4 and IPv6 application accesses, and exchanges visits. 4over6 describes the technology supporting the transition from the IPv4 network to the IPv6 network. It draws on the virtualization of the SDN network and separates the IPv4 and IPv6 networks from the data layer. Since IPv4 and IPv6 transmit data according to the same basic principle, the data layer can provide a transmission service for both IPv4 and IPv6, and achieve forwarding abstraction. It can also provide a more convenient management mechanism to the service providers of IPv4 and IPv6 respectively. This will contribute to the transition from the IPv4 to IPv6 networks.

7 Future work

ONF summarized the advantages of SDN in a white paper (Open Networking Foundation, 2012), including multiple environment centralized control, reducing administrative complexity, promoting service renewal, fine-grained network control, enhancing the user experience, and improving network security performance. These characteristics give SDN broad development prospects. Even so, a lot of work is required to improve SDN.

7.1 Security study on SDN

The SDN technology, as a kind of emerging technology, is still in the development stage, and is facing many security challenges, which include not only the traditional threats such as saturation flow attack and data theft tampering, but also new threats that have never been encountered, such as controller security attack and north–south channel security attack. With the gradual marketization of SDN technology and products, the security problem will become a key in restricting its extensive application and an important factor affecting the successful development of its technology. Although SDN security has received attention, there are many problems to be solved, such as SDN application audit and code inspection, or corresponding solutions to be optimized, such as further development of the dynamic adaptive security detection framework. On this issue, current mainstream development directions include making full use of the global view of SDN and its programmable control ability, integrating various strong encryptions and a strong authentication mechanism, and integrating middleware with more professional security features to review and monitor the fine-grained traffic. Moreover, anonymous and privacy-preserving technologies are promising security technologies to protect SDN (Guan et al., 2019).

7.2 Study on the coexistence of the traditional network and SDN

With the continuous development of SDN, the traditional network will coexist with SDN for a long

time. To make SDN compatible with the traditional network, it is necessary to support both SDN and traditional network equipment. The implementation principle of OpenFlow is that switches are divided into dedicated OpenFlow switches which do not support L2/L3 layer processing. Then the OpenFlow-enabled switches add the new features of the OpenFlow protocol and interface into the commercial Ethernet switch. However, embedding the SDN protocol into traditional network devices will make them more bloated. The path computation element (PCE) proposed by IETF is conducive to the gradual migration of the existing network to SDN. PCE makes the network path calculation functions be partly separated from the distributed network device into a concentrated point, while the traditional network nodes without PCE continue to perform the path calculation functions in their previous manner. However, PCE does not provide a complete SDN implementation. The adoption of protocol abstraction technology can ensure that all kinds of protocols run safely and stably in a unified module to reduce the burden of the equipment, and it becomes one of the trends of compatibility research.

7.3 Application research on SDN in the data center and cloud

The features of SDN, such as centralized control, global network information acquisition, and network function virtualization, can be used to optimize the management and service provision schemes in data centers and cloud application scenarios. For a data center network, SDN can be used to eliminate the redundancy of data transmission via global network information, and can also be used to achieve balance between reliability and flexibility of the data flow via network function virtualization. In addition, SDN will be important in improving performance and green energy conservation in a data center. Heller et al. (2012) summarized three problems that should be solved in the cloud management system: status monitoring technology components and ensuring their visibility, resource orchestration, and providing personalized service on demand. These requirements, which cannot be satisfied in the traditional network, will find a new sally port relying on the SDN network architecture of forwarding and control separation, flexible business deployment mode, centralized global network management model, and developed API framework.

7.4 Study on SDN scalability

Scalability determines the further development of SDN. The OpenFlow protocol has become a commonly used standard for the SDN south interface. However, this protocol is not mature, and its version is still being updated. The diversity of SDN business applications increases the design difficulty of the north interface, and the balance of flexibility and performance should also be considered. Therefore, the abstract interface language supported by mathematical theory has become a research trend. Distributed controller architecture avoids the possible single point failure of a single controller, but other related problems such as state synchronization and data backup between multiple controllers still need to be explored in the future.

7.5 Integration of SDN with other network architectures

Currently, SDN technologies bring a lot of benefits into 5G, such as seamless mobility across, dynamic security protection, and effective radio resource allocation. Beyond 5G, 6G is regarded as the next generation network, which has the features of terahertz-frequency networks and spatial multiplexing with multiple simultaneous beams of data transfer (David and Berndt, 2018). Based on the opinions of the Federal Communications Commission (FCC), dynamic spectrum sharing can be realized based on the blockchain. SDN can provide flexible resource scheduling and dynamic spectrum management for 6G.

Information-centric networking (ICN) is another new architecture for the future Internet (Wu et al., 2017). ICN aims to improve the efficiency of content distribution, storage, and fetch. The separation of information processing and forwarding in ICN is consistent with the decoupling of the control and data in SDN. In fact, SDN can be regarded as an improvement of the current Internet (Li et al., 2017; Wu et al., 2018a), which keeps some important features of IP networks. ICN is the complete revolution of the current Internet (Wang et al., 2018), which is based on the novel schemes of naming, caching, and content distribution. The novel features of ICN provide a flat

networking architecture to joint edge computing, big data, and cloud. However, SDN and ICN can be integrated efficiently. On the one hand, if the concentrated control capability of SDN is used in ICN, the intelligence of ICN will be enhanced. On the other hand, the name and caching concepts can be borrowed by SDN to enhance its flexibility. Using the SDN technology to separate the control information and making these two architectures complementary will promote further development of the network.

7.6 SDN promoting heterogeneous network communication

The heterogeneous nature of the future Internet will greatly increase. Different types of network, such as wired network, wireless network based on the fixed infrastructure, wireless network without fixed infrastructure, and optical network, will blend together and constitute a heterogeneous network to meet the various demands of terminal business. The SDN technology has the characteristics of resource optimization and collaborative control among these networks. The coordination ability of the SDN controller is very important for realizing intelligent communication and resource joint scheduling of heterogeneous networks.

7.7 SDN development and testing tools

At present, the tools to support the safe and efficient development and debugging of SDN are still very limited. A network debugging tool was proposed for the SDN architecture named ndb (Handigol et al., 2012). It makes a packet producing a "postcard" including matching flow table and output port on the passing switch to carry on the path backtrack when the error occurs, and then ndb determines the logic errors that exist in the process of switch protocol conformance and program development. FRESCO implements a framework for rapid development of OpenFlow security applications in a modular manner.

Developers can use simple FRESCO scripting language to write different function modules, and make multiple modules dynamically linked by specifying parameters including input, output, and events to deal with possible security incidents in the network. FRESCO also provides interfaces to receive information from the traditional security service applications and work with them. These platforms and tools are significant to the development of SDN.

8 Conclusions

We have made a comprehensive survey of the novel architecture and security solutions for SDN. Compared with the traditional network, two substantial changes are brought about by SDN. The first is that the deployment of new applications is accelerated by replacing decentralized closed devices running private protocols with general-purpose switch hardware that supports standard interfaces and functions. The second is that it integrates network control from distributed network components to controllers with global network views to use its programmability to manage and monitor the network uniformly. The creative structural adjustment to the existing network has endowed SDN with great development potential. Various studies have been carried out in industry to promote the application of SDN in various fields, and this promotion has been quite successful. However, SDN is still at the initial stage of development, and there are various problems that need to be solved urgently. At present, SDN has become the mainstream technology of the next generation Internet.

References

Ali ST, Sivaraman V, Radford A, et al., 2015. A survey of securing networks using software defined networking. *IEEE Trans Reliab*, 64(3):1086-1097. https://doi.org/10.1109/TR.2015.2421391

Benton K, Camp LJ, Small C, 2013. OpenFlow vulnerability assessment. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.151-152. https://doi.org/10.1145/2491185.2491222

Botelho F, Bessani A, Ramos FMV, et al., 2014. On the design of practical fault-tolerant SDN controllers. 3rd European Workshop on Software Defined Networks, p.73-78. https://doi.org/10.1109/EWSDN.2014.25

Braga R, Mota E, Passito A, 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. IEEE Local Computer Network Conf, p.408-415. https://doi.org/10.1109/LCN.2010.5735752

Casado M, Freedman MJ, Pettit J, et al., 2007. Ethane: taking control of the enterprise. Conf on Applications, Technologies, Architectures, and Protocols for Computer Communications, p.1-12. https://doi.org/10.1145/1282380.1282382

Cheng YN, Dong C, Chu LW, et al., 2015. Design and implementation of software-defined networking based firewall system. *Comput Appl Softw*, 32(1):286-288, 312 (in Chinese). https://doi.org/10.3969/j.issn.1000-386x.2015.01.072

David K, Berndt H, 2018. 6G vision and requirements: is there any need for beyond 5G? *IEEE Veh Technol Mag*, 13(3): 72-80. https://doi.org/10.1109/MVT.2018.2848498

European Telecommunications Standards Institute, 2012. Network Functions Virtualisation. https://portal.etsi.org/nfv/nfv_white_paper.pdf

Fayazbakhsh SK, Sekar V, Yu ML, et al., 2013. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.19-24. https://doi.org/10.1145/2491185.2491203

Feng MJ, Mao SW, Jiang T, 2016. Enhancing the performance of future wireless networks with software-defined networking. *Front Inform Technol Electron Eng*, 17(7):606-619. https://doi.org/10.1631/FITEE.1500336

Gelberger A, Yemini N, Giladi R, 2013. Performance analysis of software-defined networking (SDN). IEEE 21st Int Symp on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, p.389-393. https://doi.org/10.1109/MASCOTS.2013.58

Greenberg A, Hjalmtysson G, Maltz DA, et al., 2005. A clean slate 4D approach to network control and management. *ACM SIGCOMM Comput Commun Rev*, 35(5):41-54. https://doi.org/10.1145/1096536.1096541

Guan ZT, Zhang Y, Wu LF, et al., 2019. APPA: an anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT. *J Netw Comput Appl*, 125:82-92. https://doi.org/10.1016/j.jnca.2018.09.019

Gude N, Koponen T, Pettit J, et al., 2008. NOX: towards an operating system for networks. *ACM SIGCOMM Comput Commun Rev*, 38(3):105-110. https://doi.org/10.1145/1384609.1384625

Handigol N, Heller B, Jeyakumar V, et al., 2012. Where is the debugger for my software-defined network? 1st Workshop on Hot Topics in Software Defined Networks, p.55-60. https://doi.org/10.1145/2342441.2342453

Hata H, 2013. A study of requirements for SDN switch platform. Int Symp on Intelligent Signal Processing and Communication Systems, p.79-84. https://doi.org/10.1109/ISPACS.2013.6704525

Heller B, Sherwood R, McKeown N, 2012. The controller placement problem. 1st Workshop on Hot Topics in Software Defined Networks, p.7-12. https://doi.org/10.1145/2342441.2342444

Hu ZY, Wang MW, Yan XQ, et al., 2015. A comprehensive security architecture for SDN. 18th Int Conf on Intelligence in Next Generation Networks, p.30-37. https://doi.org/10.1109/ICIN.2015.7073803

Jafarian JH, Al-Shaer E, Duan Q, 2012. OpenFlow random host mutation: transparent moving target defense using software defined networking. 1st Workshop on Hot Topics in Software Defined Networks, p.127-132. https://doi.org/10.1145/2342441.2342467

Keller E, Ghorbani S, Caesar M, et al., 2012. Live migration of an entire network (and its hosts). 11th ACM Workshop on Hot Topics in Networks, p.109-114. https://doi.org/10.1145/2390231.2390250

Kim H, Feamster N, 2013. Improving network management with software defined networking. *IEEE Commun Mag*, 51(2):114-119. https://doi.org/10.1109/MCOM.2013.6461195

Kreutz D, Ramos FMV, Verissimo P, 2013. Towards secure and dependable software-defined networks. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.55-60. https://doi.org/10.1145/2491185.2491199

Li GL, Wu J, Li JH, et al., 2017. Battery status sensing software-defined multicast for V2G regulation in smart grid. *IEEE Sens J*, 17(23):7838-7848. https://doi.org/10.1109/JSEN.2017.2731971

Linux Foundation, 2015. OpenDaylight. http://www.opendaylight.org

Liu B, Chen M, Xu B, et al., 2016. An OpenFlow-based performance-oriented multipath forwarding scheme in datacenters. *Front Inform Technol Electron Eng*, 17(7): 647-660. https://doi.org/10.1631/FITEE.1601059

Liu CF, Samarakoon S, Bennis M, et al., 2018. Fronthaul-aware software-defined wireless networks: resource allocation and user scheduling. *IEEE Trans Wirel Commun*, 17(1):533-547. https://doi.org/10.1109/TWC.2017.2768358

Liu ZJ, Li Y, Su L, et al., 2014. TCAM-efficient flow table mapping scheme for OpenFlow multiple-table pipelines. *J Tsinghua Univ* (*Sci Technol*), 54(4):437-442 (in Chinese). https://doi.org/10.16511/j.cnki.qhdxxb.2014.04.009

McKeown N, Anderson T, Balakrishnan H, et al., 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput Commun Rev*, 38(2):69-74. https://doi.org/10.1145/1355734.1355746

Narayana S, Rexford J, Walker D, 2014. Compiling path queries in software-defined networks. 3rd Workshop on Hot Topics in Software Defined Networking, p.181-186. https://doi.org/10.1145/2620728.2620736

Nayak AK, Reimers A, Feamster N, et al., 2009. Resonance: dynamic access control for enterprise networks. 1st ACM Workshop on Research on Enterprise Networking, p.11-18. https://doi.org/10.1145/1592681.1592684

Nunes BBA, Mendonca M, Nguyen XN, et al., 2014. A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor*, 16(3):1617-1634. https://doi.org/10.1109/SURV.2014.012214.00180

Open Networking Foundation, 2012. Software-Defined Networking: the New Norm for Networks. http://www.valleytalk.org/wp-content/uploads/2012/05/wp-sdn-newnorm.pdf

Pan H, Guan HT, Liu JJ, et al., 2013. The FlowAdapter: enable flexible multi-table processing on legacy hardware. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.85-90. https://doi.org/10.1145/2491185.2491209

Porras P, Shin S, Yegneswaran V, et al., 2012. A security enforcement kernel for OpenFlow networks. 1st Workshop on Hot Topics in Software Defined Networks, p.121-126. https://doi.org/10.1145/2342441.2342466

Qazi ZA, Tu CC, Chiang L, et al., 2013. SIMPLE-fying middlebox policy enforcement using SDN. *ACM SIGCOMM Comput Commun Rev*, 43(4):27-38. https://doi.org/10.1145/2486001.2486022

Reitblatt M, Foster N, Rexford J, et al., 2011. Consistent updates for software-defined networks: change you can believe in! 10th ACM Workshop on Hot Topics in Networks, Article 7. https://doi.org/10.1145/2070562.2070569

Reitblatt M, Foster N, Rexford J, et al., 2012. Abstractions for network update. *ACM SIGCOMM Comput Commun Rev*, 42(4):323-334. https://doi.org/10.1145/2377677.2377748

Scott-Hayward S, 2015. Design and deployment of secure, robust, and resilient SDN controllers. 1st IEEE Conf on Network Softwarization, p.1-5. https://doi.org/10.1109/NETSOFT.2015.7258233

Scott-Hayward S, O'Callaghan G, Sezer S, 2013. SDN security: a survey. IEEE SDN for Future Networks and Services, p.1-7. https://doi.org/10.1109/SDN4FNS.2013.6702553

Sezer S, Scott-Hayward S, Chouhan PK, et al., 2013. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun Mag*, 51(7):36-43. https://doi.org/10.1109/MCOM.2013.6553676

Shin S, Gu GF, 2012. CloudWatcher: network security monitoring using OpenFlow in dynamic cloud networks. 20th IEEE Int Conf on Network Protocols, p.1-6. https://doi.org/10.1109/ICNP.2012.6459946

Shin S, Porras P, Yegneswaran V, et al., 2013. FRESCO: modular composable security services for software-defined networks. ISOC Network and Distributed Security Symp, p.1-16.

Tavakoli A, Casado M, Koponen T, et al., 2009. Applying NOX to the datacenter. 8th ACM Workshop on Hot Topics in Networks, p.1-6.

Voellmy A, Kim H, Feamster N, 2012. Procera: a language for high-level reactive network control. 1st Workshop on Hot Topics in Software Defined Networks, p. 43-48. https://doi.org/10.1109/10.1145/2342441.2342451

Wang K, Li JH, Wu J, et al., 2018. QoS-predicted energy efficient routing for information-centric smart grid: a network calculus approach. *IEEE Access*, 6:52867-52876. https://doi.org/10.1109/ACCESS.2018.2870929

Wang X, Liu Z, Qi YX, et al., 2012. LiveCloud: a lucid orchestrator for cloud datacenters. 4th IEEE Int Conf on Cloud Computing Technology and Science, p.341-348. https://doi.org/10.1109/CloudCom.2012.6427544

Wen XT, Chen Y, Hu CC, et al., 2013. Towards a secure controller platform for openflow applications. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.171-172. https://doi.org/10.1145/2491185.2491212

Wu J, Dong MX, Ota K, et al., 2017. FCSS: fog computing based content-aware filtering for security services in information centric social networks. *IEEE Trans Emerg Top Comput*, in press. https://doi.org/10.1109/TETC.2017.2747158

Wu J, Dong MX, Ota K, et al., 2018a. Big data analysis-based secure cluster management for optimized control plane in software-defined networks. *IEEE Trans Netw Serv Manag*, 15(1):27-38. https://doi.org/10.1109/TNSM.2018.2799000

Wu J, Luo SB, Wang S, et al., 2018b. NLES: a novel lifetime extension scheme for safety-critical cyber-physical systems using SDN and NFV. *IEEE Internet Things J*, in press. https://doi.org/10.1109/JIOT.2018.2870294

Yang EZ, Zhang LK, Yao Z, et al., 2016. A video conferencing system based on SDN-enabled SVC multicast. *Front Inform Technol Electron Eng*, 17(7):672-681. https://doi.org/10.1631/FITEE.1601087

Yeganeh SH, Ganjali Y, 2012. Kandoo: a framework for efficient and scalable offloading of control applications. 1st Workshop on Hot Topics in Software Defined Networks, p.19-24. https://doi.org/10.1145/2342441.2342446

Yeganeh SH, Tootoonchian A, Ganjali Y, 2013. On scalability of software-defined networking. *IEEE Commun Mag*, 51(2):136-141. https://doi.org/10.1109/MCOM.2013.6461198

Zhang D, Chang Z, Yu FR, et al., 2016a. A double auction mechanism for virtual resource allocation in SDN-based cellular network. IEEE 27th Annual Int Symp on Personal, Indoor, and Mobile Radio Communications, p.1-6. https://doi.org/10.1109/PIMRC.2016.7794896

Zhang D, Chang Z, Hämäläinen T, 2016b. Reverse combinatorial auction based resource allocation in heterogeneous software defined network with infrastructure sharing. IEEE 83rd Vehicular Technology Conf, p.1-6. https://doi.org/10.1109/VTCSpring.2016.7504455

Zhang D, Chang Z, Hämäläinen T, et al., 2017. Double auction based multi-flow transmission in software-defined and virtualized wireless networks. *IEEE Trans Wirel Commun*, 16(12):8390-8404. https://doi.org/10.1109/TWC.2017.2762300