Computer Science Review 40 (2021) 100386

Contents lists available at ScienceDirect

Computer Science Review

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

Review article

A survey of low-latency transmission strategies in software defined networking☆

Binghao Yan [a], Qinrang Liu [a,*], JianLiang Shen [a], Dong Liang [a], Bo Zhao [a], Ling Ouyang [b]

[a] Information Engineering University, Zhengzhou 450001, China
[b] Zhongyuan University of Technology, Zhengzhou 450002, China

ARTICLE INFO

Article history:
Received 22 June 2020
Received in revised form 3 February 2021
Accepted 10 February 2021
Available online 5 March 2021

Keywords:
Software defined network
Low-latency transmission
Traffic management
Congestion control
Load balancing
Flow table management

ABSTRACT

Software-defined networking (SDN), as a revolutionary networking paradigm, provides a new solution for future network development and equipment manufacturing by separating the control plane from the data plane to make the management model more simple and efficient. However, in the era of Industry 4.0 and Big Data, data transmission latency is beginning to replace traditional performance requirements, such as stability and flexibility, as the primary design consideration for network applications and users with more stringent end-to-end latency. Especially the high latency in the delivery of key information not only makes it difficult to provide high-quality services, but also restricts the development of high-tech industries such as the Internet of Vehicles (IoV) and the Internet of Things (IoT) severely. In this paper, with the goal of low-latency communication under the SDN architecture, we review and analyze relevant technologies and research findings in terms of traffic management, congestion control, load balancing, and flow table management in SDN. In addition, the contributions of some of the currently popular emerging network technologies to SDN low latency communications are listed for discussion. Finally, attention is given to the challenges that remain to be addressed and the next research trends for a promising way forward. This paper aims to provide interested researchers with insights to promote their ongoing research in low-latency communications, so as to obtain more achievements.

Contents

☆ This document is the results of the research project funded by the High Security Level Network Infrastructure Key Equipment Core Chip and Software Development, China under Grant 2017ZX01030301 and is supported in part by Industrial Internet Innovation Development Project of China — Universal Forwarding Chip of Time-Sensitive Network under Grant TC190A446-2.

* Corresponding author.
E-mail address: ieulqr@hotmail.com (Q. Liu).

https://doi.org/10.1016/j.cosrev.2021.100386
1574-0137/

1. Introduction

The Internet has evolved from serving only regional communications to a globalized information services infrastructure that underpins modern social development and technological progress. However, the rapid expansion of network scale and the increasing enrichment of network application models have forced the complexity of network structures to increase, resulting in a weakened ability of network managers to manage and control the network. Especially in the current network environment with a huge amount of data, the rigid network control method is not only high in operation and maintenance costs, but also in the process of configuration is prone to a variety of human errors and vulnerabilities, resulting in a significant drop in service quality.

In addition, the non-open nature of existing network equipment also greatly limits the deployment speed of new network technologies. On the one hand, the highly centralized and fixed logic control modules and data forwarding modules in traditional network devices make it impossible for developers to develop and verify the validity and stability of new network functions and protocols on existing devices flexibly. High renewal cost prolongs the transition time from technical theory to industrial application. On the other hand, different network devices mostly use private protocols or private configuration interfaces, making it impossible for management and developers to manage and adjust network structures quickly and uniformly, greatly reducing management efficiency and fault tolerance. Therefore, there is a need to improve the existing rigid network architecture or propose new ideas to solve the problems and dilemmas faced by the existing networks.

The emergence of SDN has revolutionized the existing network architecture and provided a new paradigm for the future development of networks. The core idea of SDN is to separate the tightly coupled control and forwarding functions in traditional network equipment to form the logically centralized device management model and the programmable underlying device that focuses only on data forwarding. The former, known as the control plane, has centralized control over all underlying forwarding devices and collects status information for statistical analysis periodically. The latter is called data plane, which only retains simple data matching and forwarding functions to enable fast processing of messages. How and where it is forwarded is determined by the rules that the controller sends. This revolutionary architecture of SDN not only frees operators from complicated network management, allowing them to control the network more flexibly, but also takes a big step toward low-cost and high-performance networks.

However, the adoption of SDN in most areas has hit a new bottleneck. In order to meet more demands of users and provide better quality of service (QoS), the end-to-end data transmission latency is continuously compressed. For example, the latency between vehicle communications systems used to ensure safe driving must be less than a few milliseconds [1]. The medical auxiliary equipment carried with the patient or person with a disability must also keep the response time within 10ms [2]. For industrial development, the advent of the Industry 4.0 era has also placed high demands on latency. Delayed delivery of data will result in mechanized equipment not operating in the set timing. The need for automated control of sensor networks ranges from a few tens of microseconds to a few tens of milliseconds [3]. Even in the financial field, reducing the time by 1 ms will bring substantial profits to large financial companies. Therefore, how to use SDN to achieve architecture simplification while meeting strict latency requirements has become an inevitable development trend and a key research direction [4].

To sum up, this paper comprehensively reviewed the latest technical achievements in the field of SDN in recent years to achieve low-latency communications, with a view to providing researchers with valuable references. The full roadmap is shown in Fig. 1, and the remaining organizational structure of this article is as follows. First, in Section 2, we provide an overview of the latest relevant surveys involving SDN, and point out how our survey differs from the existing literature. Then in Section 3, the

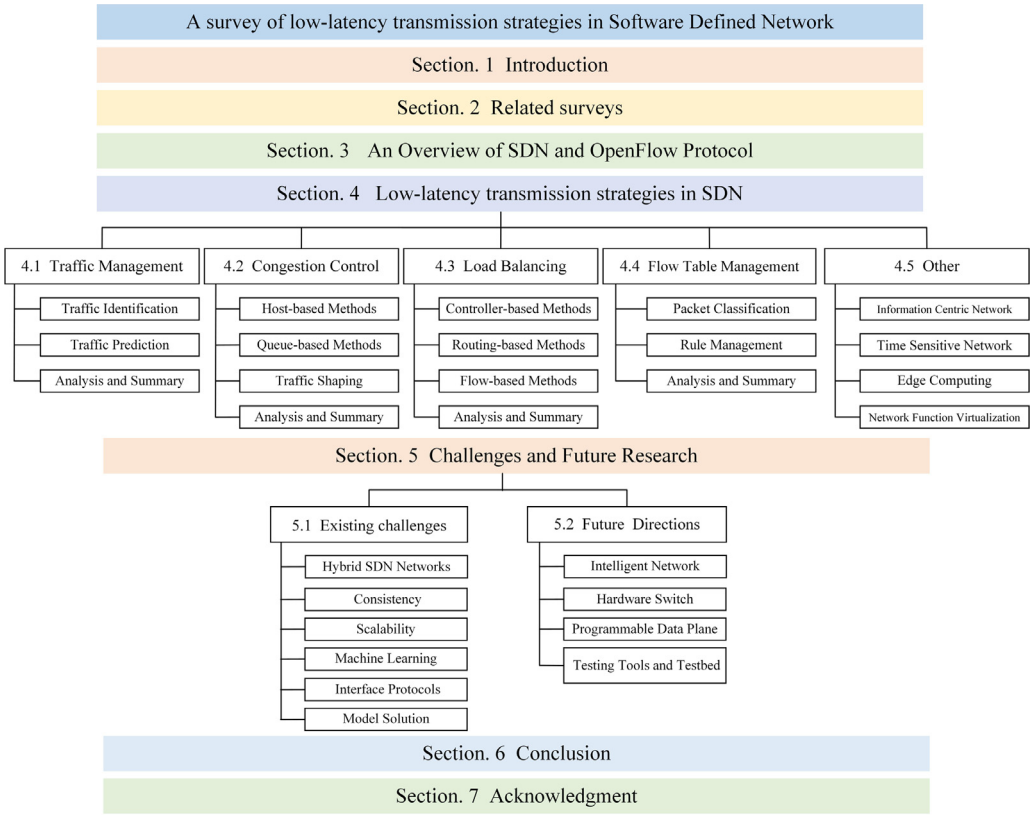

Fig. 1. The organizational framework of this paper to survey the low-latency communication technology in SDN.

development history of SDN and related theoretical knowledge is introduced. Section 4 provides a detailed review and summary of existing low-latency communication technologies and results in four aspects: traffic management, congestion control, load balancing, and flow table management. The benefits to be gained from multidisciplinary integration are also analyzed in this section. Section 5 summarizes and discusses the challenges still faced and the future research focus and development trend. Finally, the work of this paper is concluded in Section 6, and the last Section is the acknowledgment.

2. Related surveys

The high efficiency of SDN in the computer application field has drawn wide attention from all walks of life. Many reviews and surveys have been made to investigate various aspects of SDN. In [5–7], the authors have presented the detailed overviews of the development history, basic theory, typical structure and current application results of SDN, which is helpful for readers to get started with SDN.

On this basis, several works have further reviewed and summarized the security problems faced in the development of SDN, and introduced various security defense methods applied to SDN, while focusing on the future development direction of secure SDN network [8–10]. For the specific problems encountered in the development and application of SDN, the latest research works on the deployment and scheduling of distributed multi-controller are surveyed in [11] and [12]. Foerster et al. [13] have conducted a comprehensive study on the issue of consistent network updates in SDN and discussed the balance between the level of consistency that can be achieved and the speed of network updating. Ian et al. [14] have presented the traffic engineering mechanism in SDN based on traditional traffic engineering technology. Abdullah et al. [15] have further described in detail the key technologies and methods of segment routing (SR), which are widely concerned in SDN, and determine the new direction which is helpful to the development of SR. In [16], Yu et al. have conducted a comprehensive and systematic study of SDN failure management solutions and analyzed the gap between theoretical research and practical deployment. Tsai et al. [17] have outlined the work of SDN network monitoring including traffic engineering, QoS and anomaly detection, and focus on the issues to be solved in development. Xie et al. [18] have reviewed the application of ML algorithms in the SDN field and several new development areas of SDN are explored. In addition, [19–21] have devoted to solving the compatibility problem in the transition from traditional network to SDN network, which is called hybrid SDN, and many instructive problems are put forward.

However, we found no literature has paid attention to the low-latency communication technology in SDN. Therefore, based on the contributions of the existing work, this paper reviews and surveys the relevant research findings of low-latency communication technology in SDN to fill the existing gaps. At the same time, we also hope that our research can help researchers and readers in related fields and promote the birth of new research.

3. An overview of SDN and openflow protocol

In this section, we first introduce the basic background and concept of SDN, including the development process, the main architecture and the difference from the traditional network. Then the related content of the most concerned OpenFlow protocol in SDN is introduced.

3.1. Architecture and basic concepts of SDN

The study of programmable network breaks the bondage of traditional network and provides theoretical support for the development of SDN. In the traditional network architecture as

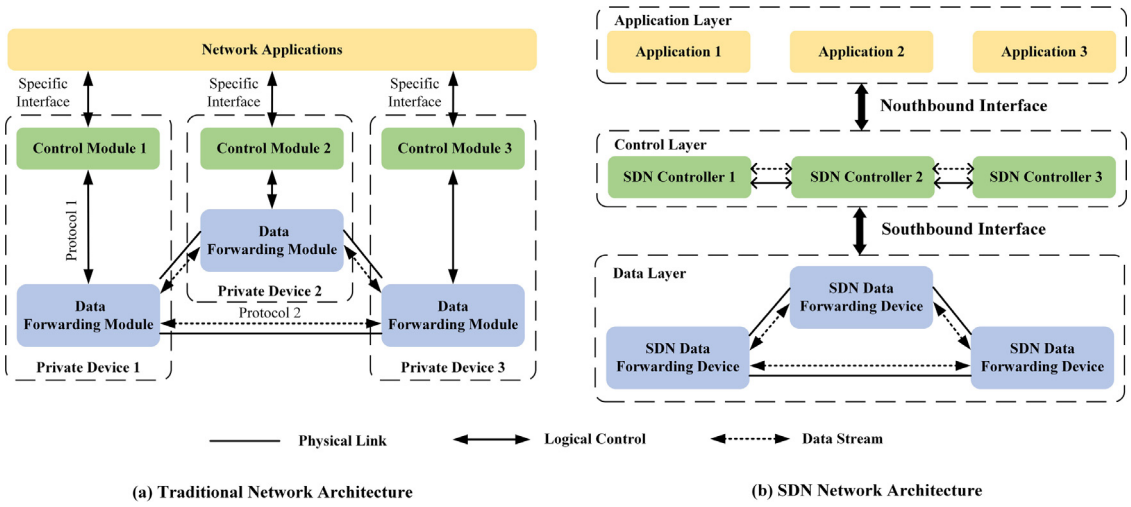

Fig. 2. Comparison of traditional network structure and SDN network structure.

shown in Fig. 2(a), the control module responsible for behavior decision-making and the data module responsible for high-speed forwarding are coupled together, which is highly independent and closed, resulting in difficulties in finding a unified way in the process of network equipment maintenance. In addition, due to the distributed deployment characteristics of different functional network devices, routing, congestion control and all other network functions between traditional devices need different network protocols to control, which also greatly reduces the flexibility of traffic path adjustment [12]. Furthermore, as network services diversify and user demand grows, network protocols become more complex and network performance increases have reached saturation point.

As a new network architecture that separates control functions from forwarding functions and can be directly programmed, SDN achieves flexible and dynamic network configuration goals. Before this, many new network structures have been proposed, among which representative ones include FIND [22], ForCES [23], AKARI [24], SANE [25], and 4WARD [26]. However, due to the high cost of modifying the original modules or the excessively advanced thinking, it is difficult to be accepted by equipment manufacturers and operators. Most of them only stay in the experimental or theoretical stage, and cannot really be implemented on a large scale. SDN originated from the Clean Slate research project of Stanford University in 2006 [27]. In 2009, Mckeown et al. [28] formally proposed the concept of SDN, which became a widely recognized thinking in academia and industry. In order to meet the needs of different industries, various organizations have proposed different SDN architectures, among which the vertical three-layer SDN architecture led and proposed by the Open Network Foundation (ONF) is widely accepted and represented. The current most popular SDN architecture is divided into data plane, control plane and application plane from bottom to top (or from south to north), and the communication between the layers depends on standardized interfaces. The overall schematic diagram is shown in Fig. 2(b).

Data Plane: The data plane consists of forwarding devices, such as routers and switches, and physical links located at the bottom of the network, which mainly are responsible for the processing, forwarding and state collection of traffic data. At the same time, the traffic forwarding rules are updated by receiving the forwarding requests from the control layer through the interfaces between the layers, and the forwarding requests that cannot be processed are delivered to the control plane for decision-making. Packets entering the data plane acquire corresponding operations by querying the flow tables stored in the forwarding device.

Control Plane: The control plane is the brain of SDN. On the one hand, it maintains all devices under its jurisdiction by coding the different user requirements issued by the application layer into a unified instruction set, so as to achieve global network topology management and flexible resource allocation. On the other hand, the control plane obtains the underlying information through the interface between the control plane and the data plane, sends forwarding strategy to the data plane, and arranges the resources of the data plane according to the status of the forwarding device, so as to ensure the effective operation of the underlying network facilities. In addition, in the distributed SDN controller system, information exchange between adjacent controllers is carried out to coordinate the overall network.

Application Plane: The application plane programs the underlying devices through the extensible programming interface provided by the control plane upwards, to realize application programs with different functions. Meanwhile, the application plane opens up control of the network according to different application needs to meet the personalized needs of users, which ensures the rapid deployment of new applications and achieves a rich variety of business innovation.

Communication Interface: The inter-layer communication interfaces in SDN include the northbound interface between the control plane and the application plane, and the southbound interface between the control plane and the data plane. These two interfaces jointly realize the seamless connection from the applications to the controller and then to the forwarding device. In addition, in a distributed SDN network structure, there are east–west communication interfaces between multiple controllers or control domains, allowing members of the controller cluster to exchange information at any time, meeting the requirements of scale and scalability. The multi-controller architecture is of key value in avoiding single points of failure and maintaining the robustness of the control plane.

3.2. Openflow protocol

As the first standard communication interface between the control plane and the data plane in SDN, the OpenFlow specification defines the communication method between the controller and the switch and the operation flow of the switch to the packets, implementing the idea of a programmable network. With its openness and flexibility, OpenFlow gives administrators the

Table 1
Improvements to various versions of the OpenFlow protocol.

Version	Main Importment
1.0	Slicing, Flow cookies, Match on IP ToS/DSCP bits
1.1	Tables, Groups, Virtual ports
1.2	Extensible match support, IPv6 support, Multiple Controllers
1.3	Per flow meters, More flexible table miss support, IPv6 Extension Header, Per connection event filtering
1.4	More extensible wire protocol, Optical port properties, Flow monitoring, Eviction, Synchronized tables
1.5	Egress Tables, Extensible Flow Entry Statistics, TCP flags matching, Scheduled Bundles

ability to access and manipulate data plane devices directly, physically or virtually, avoiding vendor disclosure and excessive modification to their device design solutions [29]. Currently, OpenFlow has been promoted as the most widely recognized standard in SDN communication, which mainly involves OpenFlow switches, OpenFlow controllers and OpenFlow channels.

OpenFlow Switch: OpenFlow switches utilize flow tables (consisting of many flow entries that represent forwarding rules) in place of the routing tables in traditional switches to enable flow-based forwarding capabilities. Among them, the flow entry mainly consists of information such as matching fields, priority, counters and actions. For each packet that enters the switch, the OpenFlow switch finds the forwarding rule with the highest priority by exact matching. Further, the operation of the packet, such as forwarding, discarding, or deleting, are completed in accordance with the operations specified in the rule, while updating the counter. If the packet does not find any rules in the flow table that match it, the switch can send the packet to the controller for decision-making or discard. When the flow rule expires, the switch needs to contact the controller to update the flow table. Currently major equipment providers around the world including Cisco, Broadcom, IBM, Juniper, Huawei and others have launched OpenFlow switches to meet different market needs. These OpenFlow switches are mainly divided into OpenFlow-dedicated switches and OpenFlow-compatible switches. The former is specifically designed to support OpenFlow and all packets that pass through the switch are subject to OpenFlow specifications. The purpose of the latter is to add OpenFlow-enabled functional modules to existing business switches and more to help operators complete the transition from legacy networks to SDN [8].

OpenFlow Controller: As the core of SDN, the OpenFlow controller abstracts control functions of traditional network switching devices to form a centralized control center to simplify network architecture. On the one hand, the OpenFlow controller is responsible for the control and management of the underlying forwarding device via the southbound interface, including: (a) maintaining the flow table information in the switch and determining how to handle data packets that do not match the rules. (b) Regularly collect and analyze the operation information of the underlying equipment, and adjust the network layout to the best operation state according to the analysis results continuously. On the other hand, the OpenFlow controller provides network resource calls to the upper layer application via a Northbound interface. Due to the subordinate relationship between the controller and the switch, one OpenFlow controller can control multiple OpenFlow switches simultaneously, and an OpenFlow switch can also be controlled by multiple OpenFlow controllers simultaneously. The controller's control of the switch can be transferred between multiple controllers, especially if one controller fails. For hierarchical control planes, the top-level master controller is also responsible for the slave controller, which has ensured smooth cross-domain communication. Typical controllers in current SDNs include NOX, Floodlight, Opendaylight, Ryu, Beacon, and ONOS [11]. These controllers are developed in different programming languages and are suitable for different network environments, such as the NOX, which is the first to support the OpenFlow protocol using C++ and python [30]. Floodlight is the first SDN controller developed on Java to meet the standards for commercial-grade applications [31].

OpenFlow Channel: The OpenFlow channel is a bridge between the control plane and the data plane, through which the two exchange information [5]. Specifically, the controller issues rules through this channel and completes switch configuration such as switch activation, identity confirmation, and time synchronization. The switch, in turn, uploads unprocessable events through this channel to the controller for decision-making. All OpenFlow channel messages must be formatted according to the OpenFlow protocol. OpenFlow channels are typically encrypted using Transport Layer Security (TLS), but can run on Transmission Control Protocol (TCP). An OpenFlow controller can manage multiple OpenFlow channels simultaneously, with each channel managing a different OpenFlow switch. An OpenFlow switch may also have one or more OpenFlow channels connected to a controller, each connected to a different controller. The creation of an OpenFlow channel is usually initiated by a switch. In special cases, some switches may allow the controller to initiate a connection, but at this point the switch needs to verify the connection authorization [32].

Since version 1.0.0 of the OpenFlow specification was introduced in 2009, improvements have been made to version 1.5.1 continuously, and the main differences between versions are shown in Table 1. Overall, developments in OpenFlow have focused on improving the processing performance of the underlying forwarding device, such as supporting more and more matching fields and larger flow entry bits for more fine-grained operations, introducing a more flexible pipeline with multiple tables to improve matching efficiency.

4. Low-latency transmission strategies in SDN

In this section, the SDN low-latency communication technology and research findings are reviewed and analyzed from aspects of traffic management, congestion control, load balancing and flow table management. Meanwhile, the contributions of some currently popular emerging network technologies are also listed for discussion at the end of this section.

4.1. Traffic management

Traffic management mainly includes traffic identification and traffic prediction. As a class of important network management function, it provides an efficient way for network administrators to manage the network at a fine-grain level by identifying or predicting the different categories to which the traffic belongs. In fact, for end-to-end latency reduction, traffic management plays a pre-processing role, i.e., such methods do not directly contribute to the reduction of latency time, but rather, by identifying the traffic entering the switching network, more convenient for subsequent traffic processing processes, including routing, transmission scheduling, anomaly detection. Because these measures need to operate when the traffic class is known [14,33].

Depending on the location of the management module, the overall architecture of SDN network traffic management consists

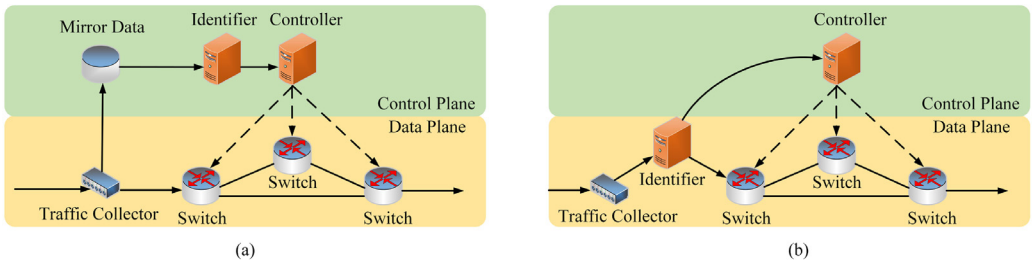

Fig. 3. Two common SDN traffic management architectures: (a) the traffic processing module is located in the control plane and (b) the traffic processing module is located in the data plane.

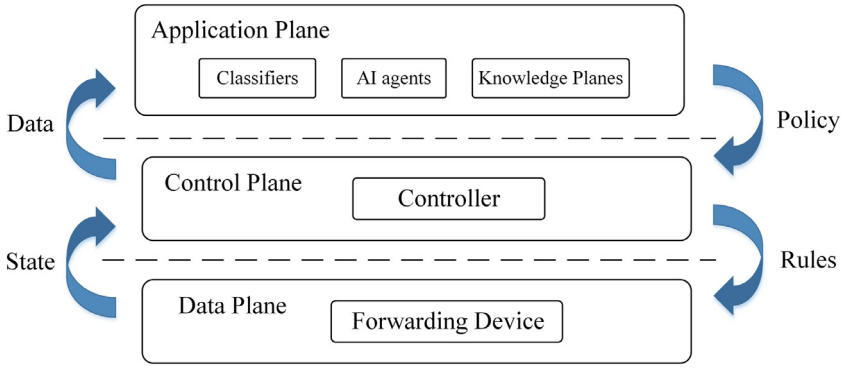

Fig. 4. The intelligent network architecture.

of two main types, as shown in Fig. 3. Where, as shown in Fig. 3(a), the management module is located in the control plane. When the traffic collector located in the data plane collects the target data according to predetermined rules, the packet is copied and passed to the management module for processing. After the processing is completed, the network application programs or adjusts the switching equipment according on the given flow control strategy to optimize the flow forwarding process. In contrast, in the architecture shown in Fig. 3(b), the management module is located in the data plane, specifically at the edge of the network or at the input port of the switching device (very few will be integrated in the switching device [34,35]). At this point, instead of replicating the collected traffic, the traffic is processed directly and the results are uploaded to the control plane for decision-making.

It should be noted that neither architecture is superior to the other. In practice, managers need to determine which architecture to use based on their own states, such as cost and security. For example, if it has high cost-tolerant capacity, distributed multi-controllers layout is the preferred solution. Network data can have multiple backups in different controllers to withstand a single point of failure. At this point, mirrored data is no longer needed, and overhead is most likely the primary concern for managers. Conversely, if the administrator can only afford single controller, mirrored data is often a better choice to prevent data loss.

4.1.1. Traffic identification

Traffic identification can be applied to both known and unknown traffic. Until now, there are three main types of traffic identification methods: (1) Port-based traffic identification method, (2) traffic identification method based on deep packet inspection (DPI) and (3) traffic identification method based on machine learning (ML).

The port-based traffic identification method identifies and tags the network applications corresponding to the port through the mapping table between the fixed port and the protocol [35]. For example, the port number corresponding to a Hyper Text Transfer Protocol (HTTP) application is 80, so when the detected traffic carries port information of 80, it is known that the type of traffic is HTTP. Similarly, the port number corresponding to the File Transfer Protocol (FTP) application is 20 and the port number corresponding to the Simple Message Transfer Protocol (SMTP) application is 25. However, due to the emergence of random ports, network address transformation, and proxy technology, port-based identification methods are gradually exiting.

The DPI-based flow identification method, also known as load-based flow classification, determines the traffic type by deep detection of the net load of each packet [36]. And if the feature field of a protocol can be found in the load, then the specific application type of the data flow can be identified. Many network equipment providers, including Cisco and Allot, have self-developed DPI solutions and modules [37,38]. Based on DPI, Li et al. [39] have proposed an application-aware traffic identification architecture integrated in SDN controller to reduce the end-to-end communication latency. This architecture maps traffic behavior to unique metadata (such as IP address) and adds these metadata to each packet as special fields to identify each traffic.

ML-based identification algorithms are currently the most popular method. It uses different behavior and statistical characteristics of different SDN network flows to identify traffic, avoiding port conflict and complexity of unpacking, and making the traffic identification process somewhat intelligent [18]. ML classifiers are usually deployed at the application layer, and use the global perspective controller to process the information collected from the data layer. The result after processing is turned into policy for decision-making. The structure diagram and operation process of intelligent network are shown in Fig. 4. At present, classifiers have been gradually improved and extended to form artificial intelligence (AI) agents [40] and even knowledge planes [41] for more powerful learning capabilities and deployment of more functions.

In [42], a SDN traffic management framework supporting distributed end-to-end QoS is proposed, which uses three deep

Table 2
Traffic management solutions in SDN.

Proposals	Objective	Method	Main Idea	Tools	Contributions
Ref. [36]	Traffic Identification	DPI	Analysis request is sent to the connected DPI if the header information is insufficient.	Mininet ONOS	The rate limit takes effect approximately 2.5 s after the DPI is implemented.
Ref. [39]	Traffic Identification	DPI	An application-aware identifier integrated in SDN controller is proposed.	Mininet Floodlight	Increases throughput while reducing the latency of end-to-end communications.
Ref. [40]	Traffic Identification	ML	NetworkAI, an intelligent architecture for self-learning control policies in SDN.	Gym Keras	The network load capacity is about double that of the benchmark algorithm.
Ref. [42]	Traffic Identification	ML	Three deep learning models, MLP, SAE and CNN, are used to detect encrypted traffic.	Keras Tensorflow	Identification accuracy is over 98%.
Ref. [43]	Traffic Identification	ML	An improved rectifier linear unit deep neural network to accomplish flow identification.	Tensorflow	Better recognition accuracy than traditional ML algorithms.
Ref. [44]	Traffic Identification	ML	Remove redundant features while ensuring ML algorithm identification performance.	Not Given	Reduced running time by 90% while maintaining overall accuracy.
Ref. [45]	Traffic Prediction	ML	A management plane is added to SDN incorporating AI technology.	OMNET++	Network latency was reduced by 84% and throughput increased by 51%.
Ref. [46]	Traffic Prediction	ML	AI agent is introduced in different layers of SDN to achieve service prediction.	Not Given	The operating efficiency, resource hit rate and user service quality have all been improved.
Ref. [47]	Traffic Prediction	ML+Fourier	A time-efficient trigonometric Fourier flow model is proposed.	SDNRoute	In some cases, the calculation time has been reduced by as much as 50 times.

learning models, multilayer perceptron (MLP), stacked autoencoder (SAE) and convolutional neural network (CNN) to detect encrypted traffic. Indira et al. [43] have modified the recognition model directly and proposed an improved rectified linear unit deep neural network to complete traffic recognition with better recognition accuracy than traditional ML and deep learning algorithms such as k-nearest neighbor and support vector machine.

Since traffic identification is indispensable for SDN network optimization, any chance to reduce latency is worth attention. Some researchers focus on how to speed up the process and improve the accuracy of traffic identification. For example, Zaki et al. [44] have introduced hybrid filter-wrapper feature selection algorithm, called FWFS, which removes redundant features in the SDN traffic recognition process, so as to reduce the computation cost and the traffic recognition time on the premise of ensuring the recognition performance of ML algorithm.

4.1.2. Traffic prediction

Compared with traffic identification, the function of traffic prediction is to predict the impact probability of overall traffic change or random sudden change traffic on key traffic service quality based on the actual operation of the current or recorded network. Critical traffic usually has a periodic generation pattern, and burst traffic can have a significant impact on the normal scheduling and transmission of critical traffic, making them unable to meet transmission latency requirements. Accurate traffic prediction results can enable network administrators or control systems to prepare ahead of time, reallocate network resources reasonably, and ensure the Qos and network stability for network users.

Mbous et al. [45] have added a management plane to the traditional SDN control plane and data plane, and proposed a three-layer architecture integrating AI technology. On the management plane, the prediction algorithm collects the spatial–temporal parameters of the input traffic and estimates the quality requirements based on Kalman Filter. Compared with the traditional scheme, network latency and throughput improved by 84% and 51%, respectively.

In view of the resource management difficulty caused by the large number of devices accessing a software-defined wireless network, Cao et al. [46] have introduced AI agent in different layers of SDN to a implement features such as network service prediction. Wherein, the user-side agent analyzes the historical data record and then predicts the user's resource request.

The time efficiency of system operation in high variable SDN load network is one of the most critical factors. Rzym et al. [47] have proposed a trigonometric Fourier flow model to solve the traffic prediction problem of time efficiency and combined it with a heuristic algorithm to obtain an optimal solution. The authors represent the load as a signal from which the continuous harmonic component is removed, and then compare the result with a predefined adaptive threshold (the threshold depends on the standard deviation of the signal) to determine whether the final model contains this harmonic. The results are 11x shorter than the typical Least absolute shrinkage and selection operator (Lasso) algorithm [48]. In Table 2, we list related traffic management solutions for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.1.3. Analysis and summary

- There is no definite solution for the location of traffic identification module in SDN structure, which needs to be determined according to the actual situation. While it is possible to get a better global view and faster network configuration when placed in the control plane, mirror traffic of the target and unregistered traffic will be sent to the control plane for decision making, which increase the communication overhead and latency of control information distribution. Therefore, decentralizing some of the identification functions to the data plane or even integrating them into the switch may have the benefit of reducing latency. Nevertheless, hardware overhead may be another concern [42].
- Most of the traffic identification methods proposed in the existing literature for SDN networks use AI technology including ML and deep learning to improve the accuracy and intelligence of identification. Due to the limited space, this section cannot introduce the application of AI algorithm in traffic identification in detail. For more information, readers can refer to [49–51]. However, the training process for such recognition algorithms is usually time costly, and the problem of conversion between old and new versions due to real-time updating of the model needs to be well addressed in the time dimension. In addition, the wide

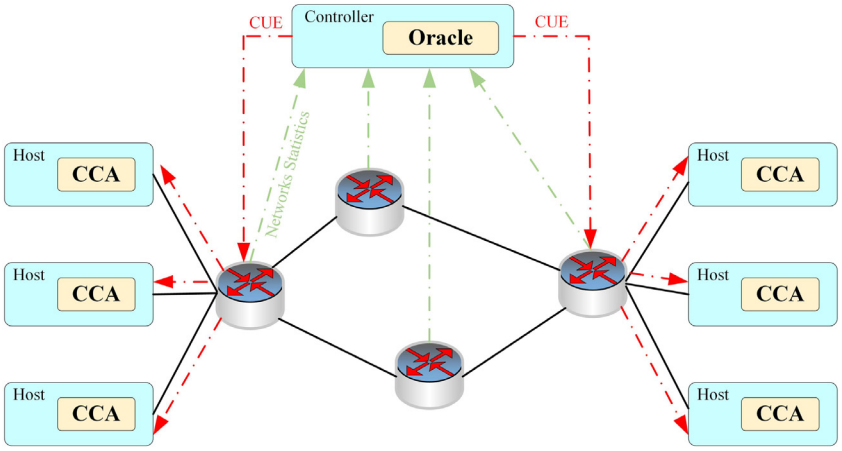

Fig. 5. The schematic view of the OpenTCP. The architecture mainly includes two main parts: (a) Oracle is a controller application that is responsible for collecting information about the underlying devices and finding suitable changes for TCP according to a predefined congestion control strategy. (b) The congestion control agent (CCA) is located at the end-host and receives the update information delivered by Oracle to modify the TCP stack.

variety of ML algorithms makes it a difficult task to choose the appropriate identification model and the optimal parameters for the different needs. Most seriously, we are unable to determine whether the classifier that uses the simulated data is equally applicable to the real environment. More questions about ML are presented in Section 5.1.

4.2. Congestion control

The separate network structure of control and forwarding in SDN makes it more convenient and free for network administrators to realize policy deployment and centralized control of the network, breaking the closed nature of the traditional network and improving the network's operability and scalability, so that the network can be privately customized according to different upper-level user needs. However, the deepening adoption of SDN in industrial control and commercial data centers has led to the same challenges as traditional networks, where the management complexity of large-scale and diverse network traffic interactions has overwhelmed existing management policies to meet performance requirements [52]. Especially in the case of limited resources and transmission of critical tasks, the network begins to generate frequent link congestion or even network paralysis.

Network congestion arises mainly because the network load at a given time is greater than the network resource capacity and the maximum processing capacity of the nodes. For example, burst traffic will cause a large number of traffic to rush to the switch from different terminals at the same time, resulting in rapid accumulation of port queues, buffer overload and packet loss. Similarly, long flow can occupy the buffer queue for long periods of time, resulting in the accumulation of time-sensitive data at the end of the queue and increasing the transmission latency greatly. In fact, SDN networks were originally designed as unconnected, best-effort packet-switched networks, which are only responsible for transmitting data to the best of their ability and are not sensitive or responsible for changes in network performance [5]. Therefore, targeted congestion control of SDN network transport links is necessary to ensure low latency delivery of critical data and smooth network operation.

Congestion control is to prevent the network equipment from exceeding the critical processing capacity that it can withstand by adopting certain mechanism, to avoid blockages in the transmission process, and to take effective control measures to the congestion phenomenon that has occurred [53]. In SDN networks, depending on where the congestion control policy is applied, there are three main types of approaches: host-based approach, queue-based approach and traffic shaping. The review in this section is divided along these three lines.

4.2.1. Host-based congestion control

As the source of data generation and transmission in the domain network, the end-host, with the assistance of relevant transmission protocols, sends user data to the switching network through different ports or paths at different transmission rates. When the user sends packets to the network at a lower rate than the network's processing speed, the intermediate device can quickly forward the packets without queuing. But queuing occurs in the network when users send packets to the network faster than the network can process them, and when queuing reaches a certain level, congestion occurs in the network. Therefore, when congestion occurs in the network, the most direct way to think about it is to reduce the initial sending rate of the stream by controlling the data sending parameters of the source host, thereby reducing the overall amount of data that exists in the network.

For SDN, a good global topology view of the controller can better support accurate host tuning. In the original architecture, either centralized controller design or distributed/hierarchical controller design, the southward interface usually existed only between the controller and the forwarding device. The controller does not issue instructions to the end-host directly [54]. For congestion control, by adding the terminal host to the controller's coverage area, the controller can feed back network status information collected from the switching device or destination host to the source host, completing the dynamic adjustment of the sending rate.

At present, the existing SDN congestion control mechanisms located in end-hosts are based on TCP and its extension mainly. TCP is a window-based control mechanism established at the sending end of the host to adjust the data sending rate by using data loss rate, average queue length, retransmission timeout, average packet delay, and standard deviation as a sign of network congestion [55]. Ref. [56] first proposed a TCP-based adaptive congestion control framework called OpenTCP in the SDN architecture, which installs a lightweight agent at the end-host that can change TCP sessions in an expressive manner and dynamically adjusts TCP parameters using the global network view that the controller has. Switches, in turn, are used to collect statistical information at the flow-level and link-level. Fig. 5 presents the schematic view of the OpenTCP. Inspired by this idea, a great deal of subsequent research began to focus on congestion control by feeding control parameters back to the sending end [57–59].

When data flows from end-hosts simultaneously compete for the same switch's output buffer queue, the queue will experience congestion and overflow, called TCP incast [60]. To solve this problem, Lu et al. [61,62] have proposed TCP congestion control based on SDN (TCCS) and improved it to apply in the IoT environment. The core idea of TCCS is to collect switch queue length information through OpenFlow protocol. When the queue length exceeds the threshold, congestion information is triggered to the controller. The controller then selects the elephant flow from the global TCP flow table and calculates its appropriate transmission rate, and sends the estimated acknowledgment character (ACK) message receiving window value of the elephant flow to the flow table. Finally, the switch matches the ACK message according to the new flow table automatically and modifies the receiving window size of the corresponding ACK message to reduce the elephant flow transmission rate. In contrast, the EQF mechanism [63]reduces the output rate at the sending end by attaching queue length information directly to the ACK packet. Similar ideas appear in [64–66] as well.

4.2.2. Queue-based congestion control

Host-based congestion control method does not achieve the goal once and for all. TCP and a series of improved control protocols do not perceive increases in queue length. Only when congestion in the switch node queue causes the network to overflow and subsequent packets arriving at the port queue are forced to be dropped does the source receive window show a significant acknowledgment latency or trigger a congestion message to the controller [57]. In fact, in more application scenarios, administrators will want to exercise some degree of preemptive control over queues that are frequently congested, or take appropriate proactive measures to control queues when congestion begins.

For SDN network, host-based congestion algorithm first upload the congestion signal from the switch to the controller, who then sends the control command to the source or modifies the confirmation signal by modifying the flow table, so as to adjust the flow transmission rate. The resulting control effective time will be prolonged, especially when the control signal is uploaded or congestion occurs at the core router (compared with the edge router, i.e. the switching device not directly connected with the end-host) [67–69]. What is more worrisome is that too many control signals will also suffer from blocking or aggravating blocking at the switching node [70]. As another type of congestion control, queue-based congestion control focuses on queuing the data streams and packets arriving at the exchange node to meet the time requirements of critical information transmission as much as possible with limited buffer space. The main methods include queue control and queue scheduling.

(a) Queue buffering control

The earliest way of queue buffering control is in the form of droptail, which is the most basic implementation of first-in first-out (FIFO) buffer, where when the queue buffer overflows, all subsequent incoming packets will be discarded, whether it is important or not. Because buffer overflow cause queuing latency for large numbers of packets and global synchronization of multiple TCP flows, droptail cannot provide satisfactory QoS in networks where large numbers of TCP streams exist. Active Queue Management (AQM) is a well-known idea in IP networks for effective congestion mitigation and QoS assurance [71,72]. Among them, the active buffer control mechanism relies on network nodes to manage buffers by actively sensing buffer occupancy. Before congestion occurs, packets are marked or discarded, which keeps the occupancy of buffer queue at a low level, provides more free space for burst traffic, and finally effectively reduces the waiting time of packets.

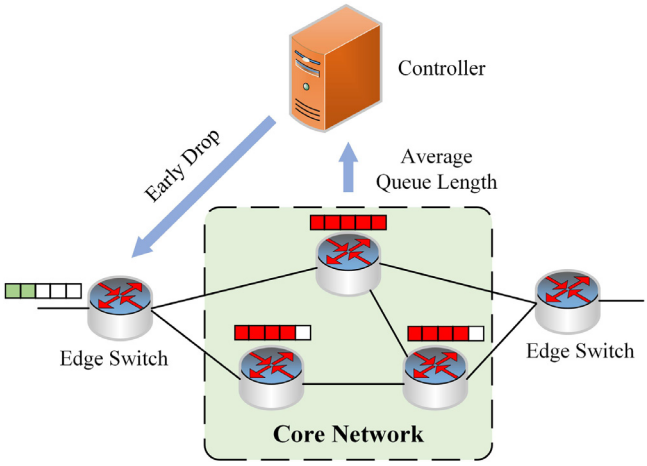

Fig. 6. The implementation process of the cooperative RED algorithm.

In terms of SDN architecture, the data plane focuses on packet forwarding, and the control and calculation functions are unified by the control plane. Therefore, the calculations associated with queue control, such as discard probability that were implemented by the switch in the AQM are also handed over to the controller for completion [69]. For queue control, OpenFlow version 1.3 has added Meter tables for metering and speed limiting, which enable rich QoS functionality by setting rules such as speed limiting for the flow. If the limit is exceeded, the packet is dropped [73]. On this basis, Gu et al. [74] have divide the calculation of drop probability into two stages. Stage one uses the network topology, round-trip times and the number of flows in the current queue perceived by the controller to calculate the baseline drop probability as a reference to the true drop probability in Stage two. In stage two, the controller calculates the applied drop probability dynamically according to the real-time queue size and the base drop probability and sends it to the switch.

Relying only on a single type of switch for congestion control may lead to inaccurate calculation of control parameters, especially in actual large-scale network where congestion is more likely to occur with core switches that are not directly connected to TCP terminals. Combining edge switch and core switch, Jeong et al. [75] have proposed a cooperative random early detection (RED) algorithm to avoid congestion on core switch. The schematic diagram of the operation process of the algorithm is shown in Fig. 6. RED is one of the most popular algorithms in AQM. It can drop packets before the queue is full, and the drop probability of packet increases with the length of the queue [76]. In this algorithm, the controller monitors the link state including the queue length of the core switch and informs the edge switch of the core network information. If a packet is sent to the bottleneck link, the edge router will discard the packet based on the drop probability. Lu et al. [77]have considered the congestion on edge switch and core switch at the same time. If congestion only occurs at the edge switches, the original RED algorithm is followed. If the congestion occurs at the core router, the packet drop probability is adjusted according to the congestion state, and the controller feeds parameters back to the edge router to adjust the data output rate.

Kundel et al. [78,79] have used P4 programmable equipment to implement the latest AQM algorithm Controlled Delay (CoDel) to enhance the anti-congestion ability of communication network. CoDel ensures that the queue drops packets at dynamic intervals in a high latency state by setting two parameters, TARGET, a delay threshold, and INTERVAL, a packet drop interval [80].

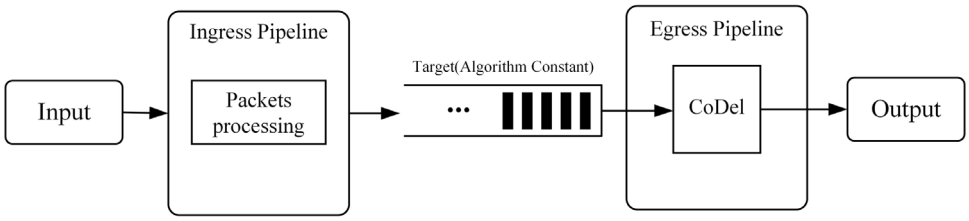

Fig. 7. CoDel algorithm integrated into the P4 reference pipeline.

As shown in Fig. 7, the author integrated the algorithm into the P4 reference pipeline. Wherein, the ingress pipeline can process the input packets and store them in the packet-buffering unit in any manner, and the egress pipeline is responsible for implementing CoDel. Experimental results show that the implementation of similar AQM algorithm in P4 data plane hardware can greatly reduce the latency in many use cases, such as the traffic shaping in Internet Service Provider (ISP) access network.

(b) Queue scheduling

The packet drop strategy used in AQM is not suitable for all network environments. At present, SDN architecture has been deeply applied in industrial IoT, IoV and other environments [81]. In industrial environment, the latency of closed-loop control needs to be less than 100 ms and maintain transmission reliability of 99.99% or more [82]. Similarly, the highest level of automated co-driving between vehicles to everything (V2X) -enabled vehicles requires very low latency of less than 10 ms and 99.99% high reliability. In advanced driving scenario, V2X-enabled vehicles have to meet even 3 ms of extremely low latency with 99.999% performance with very high reliability [83]. In these security-critical environments, any drop of a packet has the potential to cause serious damage and irreparable loss. And this kind of system needs high stability assurance, so that if packet dropping cannot be avoided, the lost information needs to be immediately given a new sending policy and resent.

Queue scheduling technology is an effective management of the network link bandwidth at the node of the network, which enables the node to assign the arrived packets to different queues or selectively forwards the packets from the queues according to the specified service principles, such as priority and latency requirements. Thus, the traffic flow at the nodes can share the output link bandwidth resources in a predetermined way, and each traffic flow can get differentiated services.

The most common technique used in SDN traffic scheduling is to isolate packets with different priority attributes using priority queues [73,84–86], so that high priority packets will be forwarded first to ensure end-to-end latency. The priority queue mentioned here refers to the priority between multiple queues, rather than the priority between elements in the queue. Queues that typically hold high-priority flows, such as low-latency transmission traffic flows, are often called high-priority queues, and packets within a single queue have the same priority. Since version 1.4 of OpenFlow specification, queue reference mechanism has been added to enable each OpenFlow switch to queue and isolate different traffic by using a separate queue on the output port, and continue to provide configurable traffic service rate [84].

Dynamic allocation of overall bandwidth among different queues is more conducive to alleviate congestion. Wang et al. [87] have achieved dynamic bandwidth allocation between different queues by setting three priority queues. Of these, the queue with the highest priority has the right to occupy all the bandwidth. The intermediate priority queue can borrow redundant bandwidth from the highest priority queue and the insufficient bandwidth must be obtained by occupying the lowest priority queue bandwidth. The lowest priority queue can only work based on the remaining total bandwidth capacity. Furthermore, Aujla et al. [88] have propose a combined queue migration scheme for SDN environment, including inter-queue migration (moving packets to queues on different switches) and intra-queue migration (moving packets between queues on the same switch).

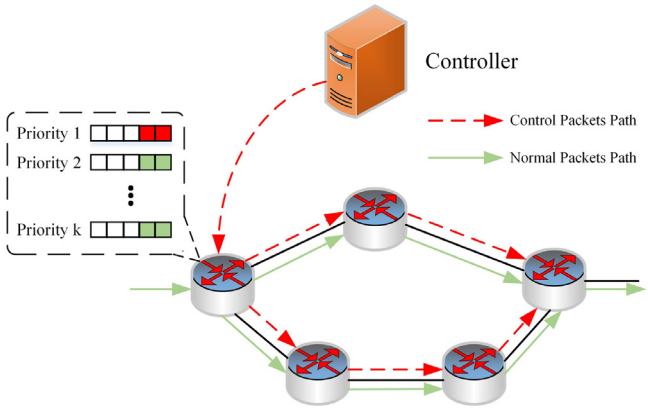

Fig. 8. The schematic diagram of the priority when control packets are transmitted simultaneously with normal packets in in-band control network.

Because SDN networks have the characteristic of separating the control plane from the forwarding plane, conflicts between control traffic and data traffic are inevitable during the routing process, especially in in-band control networks. Thus, in [89], the authors have investigates the problem of priority scheduling when control packets and normal packets are transmitted simultaneously. The authors argue that control packets carrying up-to-date configuration information, especially information related to switch configuration, have the highest priority needed to satisfy the transmission of control data first. The diagram of the priority when both are transmitted simultaneously is shown in Fig. 8. Ref. [90] also consider separating the controller flow from other traffic into different queues in the queue analysis of OpenFlow protocol to solve the problem of improper modeling, so as to ensure that the flow from the controller are not sent back to the controller again.

4.2.3. Traffic shaping

When the switching node forwards data, the rate and total amount of traffic flowing out of the node needs to be limited to some extent in the event of network congestion so that the traffic is sent out at a more appropriate and uniform rate. Traffic shaping is a measure that adjust the traffic output rate actively. Unlike host-based congestion control methods, traffic shaping primarily acts on the egress queue on the switching node, rather than operating at the terminal [91]. Because in most cases, the terminal parameters are set by the user according to their own and environmental needs and are difficult to standardize. Using traffic shaping technology can effectively control link congestion and prevent further deterioration of the network situation. Typical algorithms of traffic shaping include leaky bucket algorithm (LBA) and token bucket algorithm (TBA).

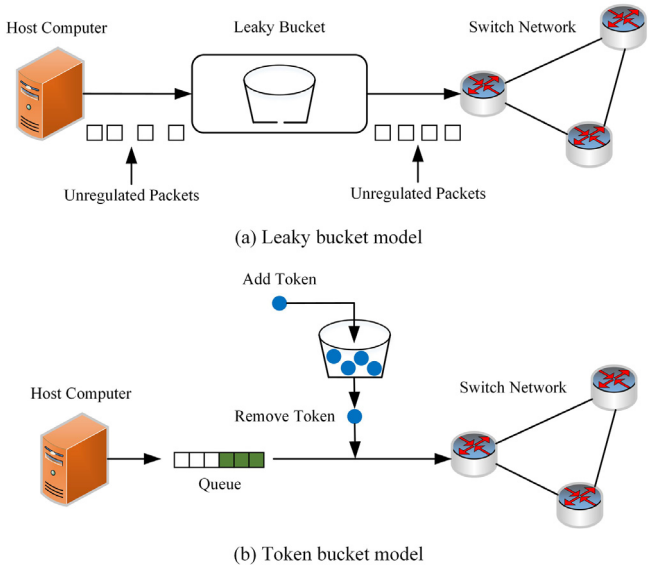

Fig. 9. The schematic diagram of leaky bucket model and token bucket model.

The LBA can be regarded as a customer–server queuing model, i.e. the LBA controls a fixed output rate regardless of the rate at which the traffic is input, thus smoothing the traffic rate [92]. The flow chart of the LBA model is shown in Fig. 9(a). When there is no traffic input, the LBA does not perform any operations. Leaky bucket can be regarded as a queue buffer with fixed capacity, similar to droptail in queue management mode, where when the bucket is filled, all subsequent incoming packets are discarded. In addition, the LBA can only output packets according to the preset rate. However, in the current SDN network environment, the link state is in dynamic flux, so network administrators want shaping algorithms that not only smooth the traffic, but also provide a faster response to bursts of data. At the same time, it is best to avoid losing data.

Currently, TBA [81,93–95] is widely used in SDN switching devices. As shown in Fig. 9(b), unlike the LBA, the token bucket stores tokens instead of packets. Packets arriving at the switching node are stored in a buffer outside the token bucket sequentially. The presence of tokens in the bucket indicates whether traffic can be sent, and the number of tokens determines how many packets can be sent. The output rate of token bucket is determined by rate at which tokens are issued, i.e. if the average sending rate configured by the user is A, a token will be added to the bucket every 1/A second, and the token will be automatically deleted after the packet is sent. The number of tokens in the token bucket can be accumulated when there is no packet input and the accumulated tokens can be used when there is a burst of traffic. Therefore, the token generation rate can be changed periodically according to the network statistics to achieve dynamic traffic shaping, improve the link utilization and alleviate congestion.

Based on the above theory, for broadband access network, Seddiki et al. [96] have proposed Per-Flow QoS, which allocates upstream and downstream bandwidth for different applications by installing traffic shaping rules. Where each application type is associated with a different queue, so each queue has different traffic adjustment strategies. At the same time, in order to ensure that the access links are fully utilized, the controller monitors the active links and dynamically configures traffic shaping parameters for them.

Bhaumik et al. [97]believe that while QoS requirements are the primary consideration in traffic shaping, it is also necessary

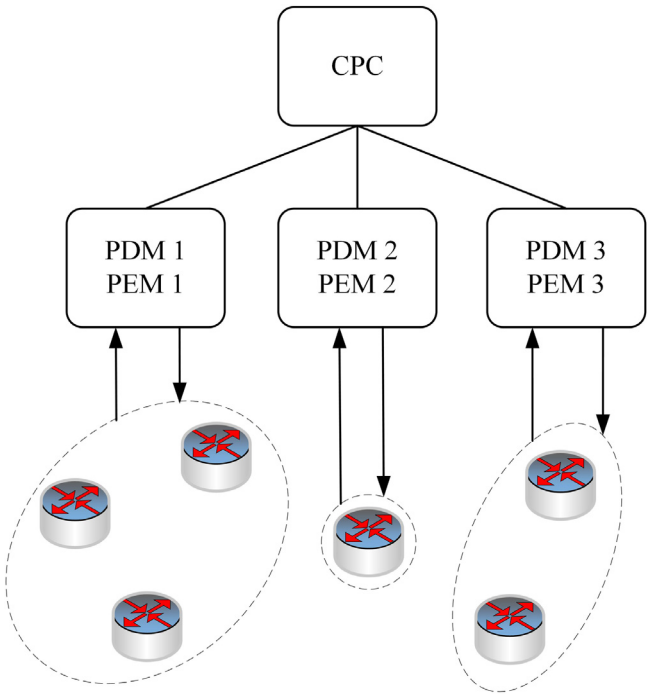

Fig. 10. The schematic diagram of the hierarchical two-dimensional queuing. PDM is responsible for monitoring and collecting the current status of network devices and generating bandwidth requirements for traffic shaping. PEM formulates shaping strategies and calculates specific parameters according to the requirements generated by PDM. CPC acts as a super controller to coordinate multiple PDM/PEM in different areas.

to ensure the fairness between traffic. Therefore, a traffic shaping architecture called hierarchical two-dimensional queuing was developed, as shown in Fig. 10. The architecture includes three modules: policy decision manager (PDM), policy enforcement manager (PEM) and central policy controller (CPC). The PDM is responsible for monitoring the traffic status and generating bandwidth requirements for traffic shaping. The PEM module generates traffic shaping parameters (such as queue parameters, token bucket size, token rate, etc.) according to the bandwidth requirements from PDM and installs them on the network equipment. The CPC acts as a super controller to coordinate PDM and PEM in different management areas, reflecting the principle of fairness.

In [98], a traffic shaping mechanism is proposed to reduce congestion in the multi-access edge computing scenario. Based on the traffic classification results and latency constraints, the deadline for each traffic with latency tolerance (DT) is set. When the network is congested, low-latency flows are sent first and the controller redirects DT traffic to the intermediate cloud server for buffering. If the deadline for DT content is about to expire, stop storing and sending it immediately. Otherwise, content will be retained until congestion is reduced to an acceptable level. In Table 3, we list related congestion control solutions for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.2.4. Analysis and summary

- With regard to cost and technology, the current network architecture has not completed the complete transformation from traditional network to SDN network. How to do congestion control in such a situation is a tricky thing. Because SDN switches and legacy switches are not similar in the way they operate and in their control configuration. For example, legacy switches use

Table 3
Congestion control solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [56]	Host-based	An adaptive congestion control framework based on TCP is proposed.	SciNet	Flow completion time is reduced by 62% compared to unmodified TCP.
Ref. [57]	Host-based	A new scalable congestion control protocol is proposed so that total utilization does not exceed link capacity.	Vswitch ns-3	The average queue completion time is less than 10ms, which is about 50%–67% lower than for typical algorithms.
Ref. [58]	Host-based	Calculate environment-specific congestion control parameters based on centrally available network properties.	Mininet	At the mean and 95th percentile, flow completion time is reduced by 12% and 31% respectively.
Ref. [59]	Host-based	Use the path visibility provided by OpenFlow to track changing bottlenecks.	Open vSwitch Ryu ns2	Actual throughput gains of 6X and 1.5X were achieved in the core and edge scenarios respectively.
Ref. [73]	Queue buffering	Introducing AQM to SDN without compromising the simplicity of the data plane.	ns-3	Improvements in average queue length, link utilization and Perflow Throughput.
Ref. [75]	Queue buffering	Packets to be dropped on the bottleneck link are dropped in the edge router in advance.	ns-3	Higher throughput can be achieved with the same load.
Ref. [78]	Queue buffering	Demonstrates how to implement the latest AQM algorithm using P4 programmable network devices.	P4-switch Mininet	The queue delay is less than 5 ms.
Ref. [83]	Priority Queues	Separate paths, Separate queues and Limited queues are proposed to enable the coexistence of congestion control mechanisms optimized for different targets.	Physical Testbed	Parallel operation of different congestion mechanisms is achieved.
Ref. [84]	Priority Queues	Use mechanisms that include global visibility and network management to ensure end-to-end latency for high critical flows.	Ryu Mininet Open vSwitch	The average round-trip latency are less than the 40us. The 99th-percentile delays were also much lower than when without mechanisms.
Ref. [85]	Priority Queues	A new dequeue scheduler is designed to transmit packets from multiple queues in approximate sorted order.	Cavium OCTEON mptcp-htsim	The average completion time for short flow is increased by 2–4x, and the 99th tail latency is increased by 4–8x.
Ref. [86]	Priority Queues	A new packet context-aware QoS model is proposed.	NOX Ofsoftswitch	This algorithm achieves higher video quality than DiffServ.
Ref. [87]	Dynamic bandwidth allocation	According to the QoS constraints of Per-flow and the performance guarantee provided by the subqueue, Per-flow is mapped to different subqueue at the egress port.	OpenDayLight Iperf	The low-priority queue sacrifices twice the latency to ensure that the sub-priority and high-priority queue latency are less than 80ms and 40ms, respectively.
Ref. [88]	Dynamic bandwidth allocation	A ensembled queue migration scheme for SDN is proposed, including inter-queue migration and in-queue migration.	Physical Testbed	The low-priority queue latency increases from 0 to 150ms, and the high-priority queue latency continues below 15ms.
Ref. [96]	Traffic Shaping	By installing traffic shaping rules for application flows, different applications are allocated upstream and downstream bandwidth.	Open vSwitch OpenWrt Raspberry Pi	Round-trip latency is reduced by approximately 50%, and jitter is reduced by approximately 87.5%.
Ref. [97]	Traffic Shaping	A traffic shaping structure called hierarchical two-dimensional queuing.	Rocketfuel Ryu	The average bandwidth occupancy is increased by about 16.7%–20%, and the average control overhead is reduced by about 60%-66.7%.
Ref. [98]	Traffic Shaping	A traffic shaping mechanism to reduce congestion in multi-access edge computing scenarios.	Not Given	The average latency of content delivery is reduced by up to about 61%.

allocation algorithms such as Internal Gateway Protocol (IGP) to control traffic passing through the internal, whereas in SDN networks, the controller are based on a global view [20]. A hybrid approach can be attempted, i.e., congestion control using multiple of the above strategies simultaneously. But whether there are conflicts between multiple strategies deserves further study.

• In the review, we found that the global view of the controller is widely used in SDN to collect the status information of the related links and underlying switches, and adjust the associated congestion control policies and parameters accordingly. However, information gathering, decision making, and rule issuance are also time consuming, especially when high-speed forwarding devices are used, and congestion is already irreversible as a large number of packets may be flooding the network by the time a new decision is made. Therefore, the issue of real-time decision-making needs further attention in future research.

• ML technology can also play a powerful role in congestion control, helping managers achieve smarter, more accurate perception and optimization. For example, neural network can approach any nonlinear function theoretically, and has the advantages of parallel computing and online learning, which has a good control effect on nonlinear system. Applying the latest ML theories and models to congestion prediction and control is interesting, and its benefits are worth looking forward to.

• Congestion control strategy can bring some inspiration for SDN security, especially for (Distributed Denial of Service) DDoS

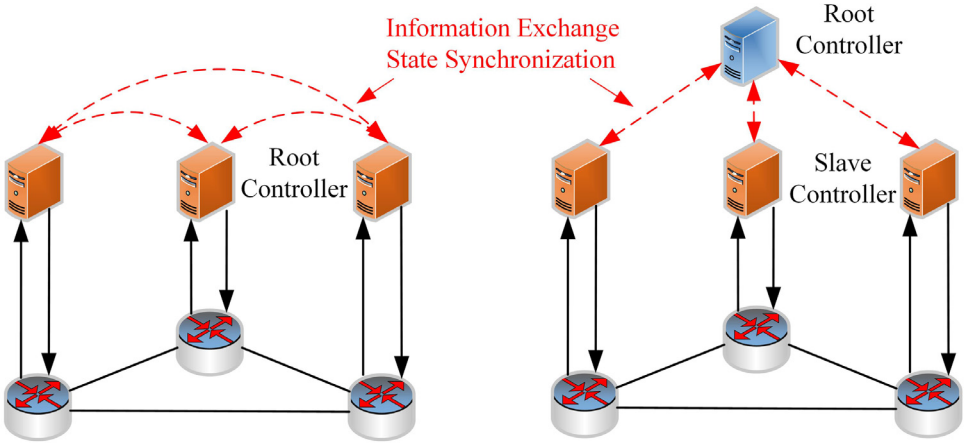

Fig. 11. Flat architecture and hierarchical architecture of SDN distributed control plane.

attack types such as the emerging link flooding attack (LFA) [99]. LFA is different from traditional DDoS attacks that target servers, but by attacking the main input and output link of the server group, which leads to the failure of the server to meet normal users requests. Therefore, the possibility of extracting ideas from congestion control strategies for similar attack defense could be a promising research direction.

4.3. Load balancing

In SDN, a controller often needs to be connected to multiple switching devices, especially in data center network (DCN) or connected to the backbone network directly. These hotspot controllers are responsible for receiving and processing incoming requests from the underlying devices and forming effective configuration parameters according to changes in network status to ensure the correct and orderly forwarding of various types of data packets. However, when the network faces with emergencies, such as the network traffic surge caused by the emergence of some hot events, the network equipment, which is already running at full capacity, will be under even greater pressure or even collapse, and user needs cannot be met immediately. In contrast, controllers or servers that are not in the center of a hotspot are always idle. This imbalance not only results in a great waste of some network resources and low network utilization, but also makes network performance difficult to be fully utilized.

Not only the controller, but also the underlying data transmission is plagued by this imbalance. First, the throughput of a single data link between network devices is often limited when it comes to cost and other constraints. Secondly, operators often use multiple switching devices of the same type within a given network, and these devices are dominated by a single controller. As a result, the controller makes the same decisions for similar forwarding requests, such as in Beacon controller where the default routing module is based on Dijkstras [100]. The combined effect of these two causes will result in multiple flows being sent to the same transmission link, even if the link is already congested, reaching or exceeding the link throughput limit eventually.

How to maximize the use of limited SDN resources and maintain balanced operation of the network has been a hot topic for researchers. Load balancing provides an inexpensive, efficient, and transparent way to scale performance metrics such as available bandwidth and true throughput of network devices and servers. It greatly reduces user response waiting time, enhances network data processing capability, and improves network flexibility and availability by distributing a large number of data streams to multiple operating units for simultaneous execution or parallel transmission over multiple paths in the network for concurrent or individual access. In SDN networks, load balancing operates primarily by manipulating controller load, data transmission paths, and stream size.

4.3.1. Controller-based load balancing

The classic SDN architecture relies on a single controller with a global view that needs to respond to all switch requests and flow table creation within the control area. However, when facing the wide area network (WAN) or large-scale network environment, it is difficult for a single controller to handle a large number of new flow requests in a timely manner, which leads to the bottleneck of network response and even single point of failure. The QoS and scalability of the network are seriously restricted. Statistical research have shown that in the worst case, a network with 100 switches may have nearly 10 million traffic arrivals per second [100]. Even though researchers have proposed a variety of SDN controllers that support multithreading, such as NOX-MT [101], Maestro [102], Floodlight [103], Beacon [100], they still face challenges in availability and applicability.

The distributed control plane composed of multiple controllers breaks the traditional centralized control center and becomes a widely concerned solution. This physically decentralized architecture divides the underlying data plane into multiple subnetworks, with each controller instance responsible for only one sub-region of switching devices. According to the difference of physical organization of controller, distributed architecture can be divided into flat architecture and hierarchical architecture as shown in Fig. 11 [104]. The flat structure means that the control plane has only one layer, and there is no membership relationship between each controller and only has a view of its own control domain. The information exchange and state synchronization between controllers are carried out autonomously and periodically. Typical architectures include Onix [105], Hyperflow [106], ONOS [107]. In contrast, in hierarchical architecture, there is often more than one control plane. The root controller on the top layer has control over the local controller on the lower layer, and has a global view of the network. The root controller provisions synchronization and communication between local controllers. Typical architectures include Kandoo [108], Orion [109].

The advantage of distributed architecture is that when any controller (except the root controller) or the southbound communication link failure, the forwarding device can transfer to other control domains within a tolerable time and overhead to continue normal operation. Although SDN multi-controller architecture brings more robust operation mode for enterprises and users, the selection of the number and location of controllers and how to

quickly and stably migrate switches between different controllers to ensure load balance have become new challenges.

(a) Controller placement

In the real network environment, one controller can only manage a limited number of forwarding devices and the number of controllers cannot be increased arbitrarily. At the same time, the controller placement problem is proved to be a NP hard problem [110]. On the one hand, if the placement of the controller is too centralized, the response time between the edge switch devices and the controllers will increase significantly. When a new flow enters the network or communicate across domains, the controller will not be able to collect the underlying information in time and plan the best transmission routing for the new flow, causing the flow to wait and pile up in the switch for a long time. Not only that, but also the flow table update process of switching devices will also be affected. On the other hand, if the controllers are placed too decentralized, the negotiation and synchronization latency between controllers will not be guaranteed, which will affect the logical consistency between controllers and resulting in conflicts in the cross-domain data forwarding process [111].

At present, most researches define the controller placement problem as a multi-objective combinatorial optimization problem, and the optimization goals and constraints depending on the environment and users' concerns. Si et al. [112]have proposed the SDN controller layout solution LiDy+ with switch traffic load and southbound communication latency as constraints. The solution can employ open and restricted search policies based on the characteristics of the deployment environment and adjust the number of controllers by predicting the switch traffic load. Tanha et al. [113] have elaborated on the problem of elastic controller deployment in WAN and propose a clique-based heuristic deployment location solution in graph theory. The scheme takes switch-to-controller latency, inter-controller latency, controller performance (quantified by the number of flows processed per second), and switch traffic load together as optimization targets. Hock et al. [114] have applies the elastic Pareto optimal control framework in SDN-based core networks. The framework models three metrics: the maximum latency from switch node to controller, the maximum number of switches a controller can own and the maximum number of controller-less nodes appearing for a certain placement, which can provide network operators with all Pareto-optimal locations. In addition, many placement methods have been proposed and validated in fog architecture [111], cellular network [115] and 5G communication [116], all of which take into account, to some extent, the impact of the application environment on optimization goals.

More research on controller placement can be found in [12, 104,117] , where authors provide detailed overviews of more control plane architectures and placement schemes, while considering optimal and feasible solutions under more constraints.

(b) Switch migration

The process of optimizing controller deployment for a particular SDN network is usually done before the network is put into operation or at an early stage of operation, based on feedback from previous network statistics or in reference to user requirements. In earlier versions of OpenFlow, switches were allowed to connect to only one controller. Regardless of whether switching devices and controllers can adapt to each other or not, part of the communication process is forced to take place, since the deployment optimization process is often not optimal. In addition, in the real network, the jitter of the traffic load caused by time and space factors will put the switching devices in different regions under different operating pressures, i.e., the load may transfer between controllers over time. Switching nodes with heavy traffic loads will communicate more frequently with the controllers responsible for which it is responsible, resulting in increased latency in controller response and further exacerbation of oscillations [118,119].

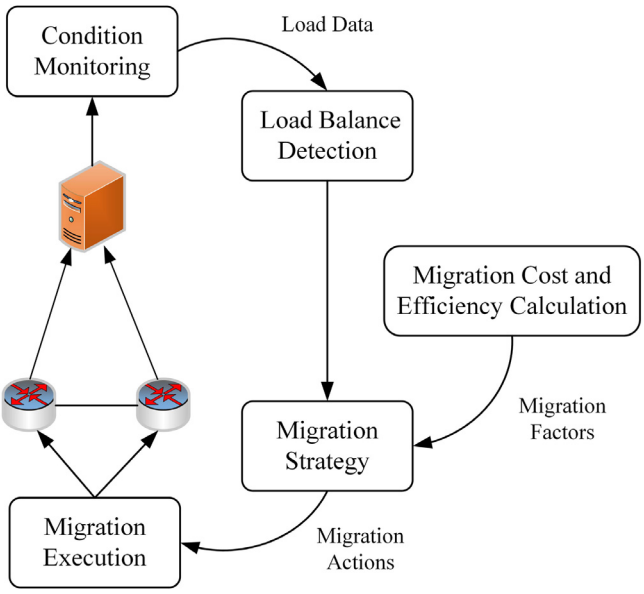

Fig. 12. Switch migration steps in the [124]. The monitoring module collects the underlying information, the load balancing module decides whether to migrate or not, the migration efficiency calculation module builds the migration efficiency model, the migration strategy formulation module creates the migration decision and the migration execution module executes the migration.

Switch migration, as a load balancing mechanism in the distributed controller architecture, breaks the fixed mapping relationship between controllers and switches. When a controller in the network is overloaded, it is possible to migrate some of the switches it manages to adjacent controllers to achieve partial load transfer [120]. On this basis, in [121], the authors have proposed a game-based switch-controller mapping method. Controllers compete to serve the migrated switches within their own processing capacity in order to obtain the maximum migration benefit, which is defined by the amount of traffic processed and their per-unit price. Al-Tam et al. [122] have retained the migration decisions of previous moments to the current moment for reference, thus achieving load balancing while emphasizing the stability of mappings between switch controllers, i.e., to minimize the number of links that need to be remapped. In order to improve the intelligence of migration, Min et al. [123] have used the feedback mechanism of Q-learning algorithm to learn the current network state to determine which switches need to be migrated and where to migrate. The algorithm can adjust the interval between two migration actions adaptively.

How to choose the time and conditions for switch migration is also a hot topic. The appropriate migration timing can ensure the stable operation of the overall SDN network, and reduce the time overhead for oscillation and stability caused by migration. Wang et al. [124] have quantified the conditions that trigger migration and establish a migration efficiency evaluation model to balance the migration cost and the load balancing effect before and after migration. The steps of migration are shown in Fig. 12, which mainly involves the monitoring module, load balancing detection module, migration efficiency calculation module, migration strategy formulation module and migration execution module.

Filali et al. [125] have used a multistep Autoregressive Integrated Moving Average (ARIMA) prediction model to predict the long-term load on the controller. When the predicted results indicate that the controller is overloaded, the management module makes advance arrangements for the migration operation to prevent the migration process from creating hysteresis.

Cui et al. [126] have suggested that more precise load migration could not be performed solely to determine whether a performance bottleneck was present based on the controller's load, but rather whether the controller's response time exceeded a set threshold. The experimental results verify that response time can be used to judge the changing trend of controller load more accurate. In Table 4, we list related controller-based load balancing solutions mentioned in this paper for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.3.2. Routing-based load balancing

For SDN, the forwarding device or slave controller needs to provide consistent action support to the top control module to avoid communication obstacles, which is guaranteed by the same protocol. Therefore, the built-in default settings or control algorithms of network devices often have a high degree of similarity, which leads to the switch device to make the same forwarding decision for incoming packets, regardless of the priority difference of packets and network status. Also, some transmission paths may be more attractive to packets than others, such as fewer forwarding hops or higher bandwidth. This similarity will cause an unbalanced distribution of packets over different paths, ultimately leading to hotspot routes that may be exhausted more frequently or quickly than other routes.

The optimization algorithm is based on a multi-constrained solution strategy that uses the global network link state and topology information to analyze the network performance in real time and provide global or local routing patterns that meet the QoS requirements (including metrics such as hop count, latency, bandwidth, and error rate) for end-to-end traffic delivery, especially for time-sensitive traffic. Based on the differences between the optimization algorithms proposed in the existing literature, we divide the following survey into two parts: static routing strategy and dynamic routing strategy.

(a) Static routing strategy

Here, we classify such an optimization mechanism as a static routing policy. That is, before the packet leaves the end-hosts or at the edge switch of an intra-domain network, the controller has calculated the comprehensive optimal solution under the constraints and planned a suitable path to connect the sending and receiving nodes for the packet in advance according to the current network state and quality of service requirements. The path is not changed again during subsequent transmissions, regardless of changes in the network state, until the packet is delivered to the destination and the path fails. It should be noted that the static state referred to here is not similar to the fixed routing algorithm in traditional network that does not take into account the network state, but specifically refers to the fact that the packet does not change its route during transmission.

The advantage of a static routing policy is that the controller only needs to configure the flow table once in the node through which the flow is passing, avoiding transmission jitter and conflicts caused by frequent routing adjustments, and reducing the time overhead of flow table distribution. Dinh et al. [127] have used Epstein algorithm to calculate K-shortest paths between source and destination simultaneously for new flows entering the network, and assign the flow to the path with the least load in the current state for transmission. Further, Yan et al. [128] argue that different types of flows require different transmission paths to meet the latency requirements. They first distinguish different types of services by identifying the IP address class of the source node, and then choose the optimal path to meet the QoS constraints from multiple paths between the source node and the target node. Song et al. [129] have determined link load by detecting switch utilization. When utilization exceeds 70%, the current path is deemed unsuitable for continuing and the controller plans a new path that can bypass this switch for subsequent flow. When the utilization drops to 50%, the original path is re-enabled because according to the routing algorithm, the original route is better than the bypass.

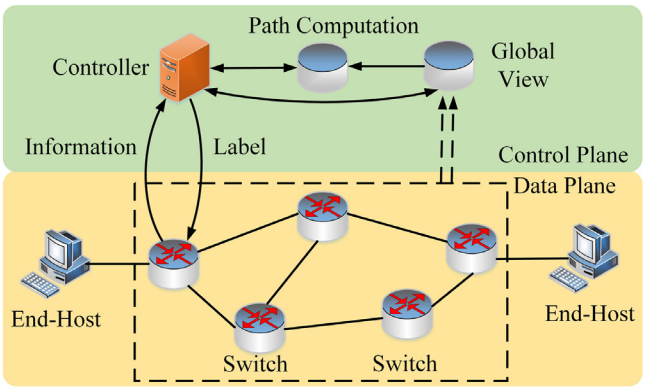

Fig. 13. The composition diagram of SR running in SDN architecture. Step. 1. The application communicates its requirements (such as latency, bandwidth, etc.) to the SDN controller. Step. 2. The controller collects the underlying traffic information and the overall state of the network, then calculates the specific network path and encodes it as a segment list. Step. 3. The SDN controller programs the packet header at the edge switch node of the network.

Multi-constrained path planning will no longer a simple problem when networks contain large-scale switching nodes. To improve the intelligence and optimization of path calculation, Cui et al. [130] have used Elman neural network model to obtain link data information that reflects the original data features. The fused feature data is then transmitted to the controller for fast load balancing of the sensor node links. In [131] and [132], the authors have considered the entire network as an image for route optimization. Specifically, the former combines different numerical features of the traffic pattern into a three-dimensional matrix input CNN, based on deep learning technology. Then, each CNN can represent a set of path combinations, and the output uses a two-dimensional vector to indicate whether the path set is crowded. The latter is based on graph neural networks to capture the path and link states generated by the network topology and routing configuration, thereby better understanding the complex relationship between topology, routing and input traffic, and achieving average per-packet latency prediction.

Segment routing (SR) technology, first proposed by Cisco and certified by the Internet Engineering Task Force (IETF) standardization, makes the SDN routing mechanism more concise and fast [133]. The core idea is based on source routing, i.e., the source node determines the data forwarding path. The SDN controller only needs to encode the end-to-end routing information into the packet header and send it to the network device as an ordered list of labels, does not need to add forwarding rule to each device on the path, nor does it need to reassign forwarding state in the associated switches when resending packets. The schematic diagram of the combination of segmented routing and SDN architecture is shown in Fig. 13. SR has been supported by a number of equipment vendors, including Inter, Cisco, and Juniper, while the network operator Google has focused on applying SR to its management needs [134]. Academic research on SR is ongoing and has produced many forward-looking research results [135–137].

(b) Dynamic routing strategy

When transient disturbances occur during network transmission, it is obviously a complete solution to recalculate routes based on the working sub-network state. However, the recomputation process may require more time overhead and pressure

Table 4
Controller-based load balancing solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [112]	Controller Placement	Selectively adopt open search strategy and limited search strategy, and adjust the number of controllers by predicting switch traffic load.	NOX-MT Matlab	The delay between the switch and the controller is kept within the maximum delay limit of 4ms.
Ref. [113]	Controller Placement	The optimization goals are the capacities of controllers, the traffic load of switches and switch-controller and inter-controller propagation latencies.	GUROBI	The average utilization rate of the controller reaches more than 80%, and the maximum switch-controller latency is less than 15ms.
Ref. [114]	Controller Placement	The optimization goals are the maximum latency from the switch node to the controller, the maximum number of switches per controller, and the maximum number of controller-less nodes.	POCO	Best trade-off between load balancing and delay is obtained under the given topology.
Ref. [121]	Switch Migration	Controllers compete with each other within their respective processing capabilities to maximize the benefits of switch migration.	Physical Testbed	The variation of response time is reduced by more than 90%, and the number of switch migrations is reduced by at least 50%.
Ref. [122]	Switch Migration	The migration decision at the previous moment is reserved for reference to minimize the number of links that need to be remapped.	Matlab	Increases the stability of the switch-controller assignment by 91%.
Ref. [124]	Switch Migration	A migration efficiency evaluation model was established to balance the cost of migration and the effect of load balancing before and after migration.	Beacon Mininet	The migration time is reduced by about 21.4%, and the average response time is reduced by about 15.8%.
Ref. [125]	Switch Migration	A multi-step ARIMA prediction model is used to predict the long-term load of the controller.	Matlab	Reduce the response time of the high-load controller by 13.7% on average and up to 29.6%.
Ref. [126]	Switch Migration	Use the response time to determine the changing trend of the controller load.	Floodlight Mininet	Load balancing time is reduced by 44.4%, and the number of switch migrations is reduced by 33.3%–50%.

on the flow table, and will force unaffected flows to abandon the currently adapted path and choose a new one. Therefore, overthrowing the existing operating mechanism of the entire network for replanning is not the best way to deal with transient disturbances.

Unlike the static routing strategy mentioned above, the advantage of dynamic routing strategy is that the path of the flow during transmission process can be adjusted at any time according to the latest state issued by the controller, so that part of the flows on the overloaded path can continue to be forwarded by dispatching to idle bypass. Note that the dynamic adjustment strategy is only responsible for local areas and does not need to re-plan the overall network, so it will not have a significant impact on normal flows.

Hao et al. [138] have described the rerouting problem as a multi-commodity flow problem for the elephant flow and uses the probability of success to characterize the uncongested probability of all available new paths. The success probability is calculated from the current utilization of the target link and the path criticality (referring to the probability that all switches on the path pass the elephant flow at the same time). Finally, the path with the highest probability is selected for traffic distribution.

Boero et al. [139] have inserted an AQM mechanism into the rerouting process. First, the data plane information collected by the improved Beacon controller is used to schedule the flow on the congestion queue in the switch to the idle queue. The controller then forms a new flow table to ensure that flows in different queues are sent to different transmission paths.

Shen et al. [140] have studied the dynamic routing problem in hybrid SDN architecture. The algorithm uses a tree structure to build all possible forwarding paths. The tree node is divided into SDN node and non-SDN node. When the crowdsourcing task passes non-SDN nodes, the Open Shortest Path First (OSPF) protocol is used as the default routing mechanism. When passing SDN nodes, the crowdsourcing is sorted according to the task load and a greedy algorithm is used to dynamically determine the next hop.

Redundant path reservation is another effective way of dynamic path planning. As shown in Fig. 14, before the data flow is sent from the host or the switching node, the controller has calculated multiple source–destination transmission paths (not necessarily equal cost paths). The data flow selects an optimal path in the current state, called the main path, for transmission. Moreover, during the transmission, the remaining paths will not be deleted, but one or more paths will be selected as backup paths until the flow is delivered to the terminal. When the main path fails, the switching node can quickly move all or part of the flow to the backup path for continued transmission, reducing the end-to-end latency caused by the failure.

Li et al. [141] have designed a local rerouting fast recovery mechanism to balance the network load for the case where the middlebox fails. This mechanism uses mixed-integer linear programming to calculate multiple redundant paths for middlebox agents that may fail. Considering the overhead caused by the number of redundant links, a backup path selection algorithm is proposed to preserve the most effective path among them. The authors believe that the difficulty in using backups is how to check whether the redundant paths have sufficient resources to accommodate the flows that need to be transferred.

Whether the backup path crosses the main path will affect the effectiveness of routing scheduling. When the overlap is too high, the backup path cannot resolve the failure of the main path. Therefore, Basit et al. [142] have determined the amount of overlap with the congested path among multiple backup paths by setting a link sharing threshold. If the number exceeds the threshold, i.e., no disjoint paths are available, the recalculation module is launched to adjust the path and recalculate the backup path.

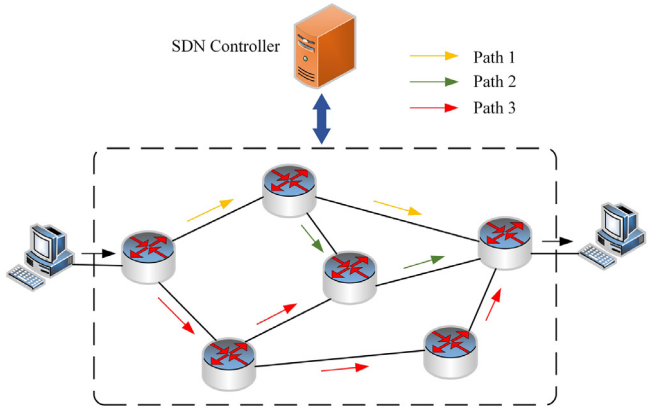

Fig. 14. End-to-end data transmission through redundant paths. When path 1 is selected for data transmission, path 2 or path 3 is reserved as a backup path.

The stream control transmission protocol (SCTP) developed and promulgated by the Signaling Transport Work Group (STWG) as the IETF standard tracking protocol has allowed end-to-end data transmission to take advantage of the host's multi-homing feature to set multi-paths to provide backup fault tolerance [143]. SCTP uses heartbeat to periodically detect whether the backup line is alive. When the main path is congested or interrupted, it can select one of multiple backup paths as the successor path for load balancing automatically. However, SCTP has not received much attention from equipment manufacturers and operators due to the difficulty and high cost of upgrading firmware (some may even need to be eliminated), and because the SCTP protocol message format has changed so much compared to TCP that it cannot pass intermediate devices such as firewalls.

In Table 5, we list related routing-based load balancing solutions mentioned in this paper for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.3.3. Flow-based load balancing

The traffic existing in the DCN can be divided into two types: elephant flow and mouse flow. Among them, elephant flows that only account for about 20% of the entire network traffic, such as file storage and data migration, but occupy nearly 80% of network bandwidth for a long time. In contrast, the mouse stream brought by Web services, search engines, e-commerce and other user-centric services are less than 10 KB in size and less than a few hundred milliseconds in duration [144]. If the elephant flow and the mouse flow are mixed and transmitted on the shared path, resource conflicts are frequent and serious, and the latency-sensitive mouse flow will be continuously delayed for delivery. Especially when the elephant flows are concentrated on a single link due to the monotonic scheduling algorithm, it will seriously occupy the network bandwidth, make the link become a hotspot path and lose effective communication capabilities gradually.

For such network environments, multi-path transmission technology has played a key role. This technology splits large flow into multiple independent subflows and uses multiple links between communication nodes to transmit flow fragments in parallel, avoiding network hotspots and single-link congestion, providing higher overall transmission rates and lower end-to-end latency. For SDN, the flow-oriented matching and forwarding operations make SDN have inherent support for multipath transmission, which can increase the utilization of network resources by performing per-flow operations. At the same time, as the most important link-layer communication protocol in SDN, OpenFlow has defined the use of group tables to support more advanced multi-path routing since version 1.1. Specifically, as shown in Fig. 15, the indication field of the flow entry in the flow table is used to refer to a specific group entry, where the group entry is defined as a structure composed of an identifier, a group type, a counter, and an action bucket. Each group consists of a set of operation buckets, and each bucket contains a set of weighted paths and operations applied to the matching flow.

Equal-Cost Multipath Routing (ECMP) finds and utilizes multiple paths with the same minimum cost in the DCN to implement flow segmentation. Note that the cost mentioned here (sometimes called weight) refers to the number of hops performed by sending a data packet from one host to another, and does not involve performance metrics such as link bandwidth. When an ECMP-enabled switching device finds that there are multiple equivalent optimal paths for the same destination address, it relies on the hash mapping of the TCP/IP 5-tuple of each packet flowing to obtain the output port and update the routing table [145]. In SDN, the equal-cost multipath discovery function is under the jurisdiction of the controller, and the path discovery process is still based on the Dijkstra algorithm. Rhamdani et al. [146] have validated ECMP in a software-defined DCN with a fat-tree topology. This solution uses a modified Dijkstra algorithm to ensure that multiple equivalent shortest paths are searched, and a modulo division hash function is used for hash mapping. The result of the modulus indicates which path to send the data packet. The experimental results show that when transmitting 500M data in the 20 Mbps background traffic scenario, using ECMP can save 81.2 s over the static routing strategy. In the 30 Mbps background traffic scenario, the difference in delivery time is a dramatic 774.98 s.

Current versions of various routing protocols, such as OSPF version 2, Intermediate system to intermediate system-optimized multipath (ISIS-OMP), Enhanced Interior Gateway Routing Protocol (EIGRP), and Border Gateway Protocol (BGP), allow switching devices to split flows for different subflows into multiple shortest paths of equal cost for routing. Google's global deployment of B4 systems also has a custom variant of ECMP hashing built in to achieve the necessary load balancing [52]. However, the disadvantage of ECMP is that on the one hand, ECMP will not split the flows if only a single shortest path is available at this time. On the other hand, because ECMP measures costs based on the number of hops rather than the performance of the link, there is always no guarantee that the uniform distribution of traffic is optimal. Load balancing can be very unsatisfactory especially when there is a large performance difference between paths.

ECMP defaults that the packets in the same TCP flow are transmitted to the destination node through the same path. As an extension of the TCP protocol, Multipath Transmission Control Protocol (MPTCP) allows a single application-level TCP connection to be split into multiple TCP subflows to implement multipath transmission. The protocol stack is shown in Fig. 16, where MPTCP is completely transparent to the original TCP-based applications. The subflow process is similar to traditional TCP in that connections are established with three-way handshakes and closed with four-way waves, so any application using standard TCP can apply the MPTCP protocol. It should be noted that both parties using MPTCP for multipath must also support the protocol at the same time. The sender will send a SYN packet with the MP_CAPABLE field added to initiate the link, and the SYN+ACK packet from the receiver also contains this field. If either party does not support the protocol, it reverts back to the standard TCP communication mode. In addition, MPTCP solves the problem of sequential reception of different traffic segments by increasing the packet sequence number of each subflow. Therefore, at the receiving end, MPTCP first uses the subflow sequence number to reorganize the packets in each subflow, and then uses the data sequence number to reorganize each subflow.

Table 5
Controller-based load balancing solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [127]	Static Routing	Dynamically select the best path among multiple paths to forward the incoming flow.	OpenDaylight Mininet	The average file download time in different scenarios is reduced by more than 33%, and the average network latency is reduced by more than 35%.
Ref. [128]	Static Routing	The use of multiple paths between the source and destination and queuing mechanism to ensure QoS.	Floodlight Mininet	Greatly reduces the data transmission latency and server response time, and improves the system throughput.
Ref. [129]	Static Routing	Calculate the link utilization in the SDN controller and apply the rerouting algorithm to the OpenFlow switch.	ONOS Mininet	Throughput increased by 11%, and Round Trip Time increased by 11.4 times.
Ref. [130]	Static Routing	The Elman neural network model is used to obtain the link information reflecting the original data to quickly balance the load on the sensor node link.	Matlab	The bandwidth utilization rate is maintained at around 1.
Ref. [131]	Static Routing	The CNN is used to intelligently calculate the path based on the input real-time traffic traces.	Not Given	The average packet delay is reduced by up to 69.4%, and the throughput is increased by 1.25 times.
Ref. [132]	Static Routing	Use graph neural network to capture the path and link status generated by network topology and routing configuration.	OMNeT++ RouteNet	The average latency is reduced by 23%, and the latency jitter is reduced by 34% on average.
Ref. [138]	Dynamic Routing	Describe the rerouting problem as a multi-commodity flow problem for elephant flows and use the probability of success to describe the non-blocking probability of all available new paths.	Ryu Mininet	The average latency is reduced by 87.95%, and the average throughput is increased by 56.1%.
Ref. [139]	Dynamic Routing	AQM mechanism is inserted during the rerouting process.	Mininet	The constraints and queue balance are guaranteed without modifying the OpenFlow specification.
Ref. [140]	Dynamic Routing	A tree structure is used to construct all possible forwarding paths.	VS2010	Improve the network throughput while reducing the probability of network congestion.
Ref. [141]	Redundant Path	Mixed-integer linear programming is used to calculate multiple redundant paths for middle box agents that may fail.	Mininet	An increase up to 26.4% of the throughput.
Ref. [142]	Redundant Path	Determine the degree of overlap between multiple backup paths by setting link sharing thresholds.	Ryu Mininet	As high as 54% increase in throughput

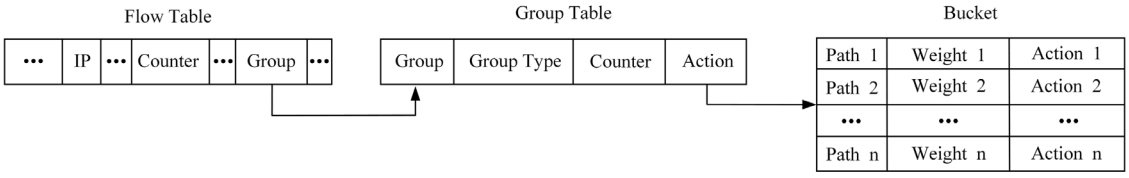

Fig. 15. Operational interaction between group tables and flow tables in OpenFlow.

At present, MPTCP has been deployed in many advanced engineering projects. For example, Apple has used MPTCP on iPhone, iPad, and Mac since the ISO 7 to switch from one wireless network to another seamlessly [147]. Many new versions of operating systems such as Solaris and FreeBSD also support MPTCP. Research on the application of MPTCP in SDN is also ongoing. Since MPTCP cannot control its own routing, Nakasan et al. [148] have proposed a simple multipath controller (Simple Multipath OpenFlow Controller) SMOC for MPTCP processes based on the open source OpenFlow framework POX to prevent contention for bandwidth on edge routing. Liu et al. [149] have designed a system that combines SDN and MPTCP to correct defects in MPTCP, such as randomized subflow routes. Herguner et al. [150] have used MPTCP for video streaming providers to ensure service quality. Among them, a mixed-integer linear programming model performs the determination of the number of MPTCP subflows and paths. In addition, MPTCP has also shown a forward-looking effect in ensuring the real-time and stability of software-defined Vehicle-to-infrastructure (SDV2I) communications [151].

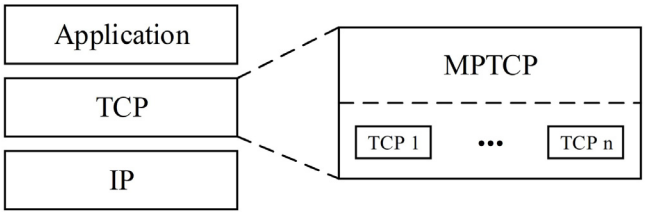

Fig. 16. The architecture of MPTCP.

In addition to ECMP and MPCTP, two commonly used multipath algorithms in SDN, there are other flexible multipath calculation algorithms and traffic segmentation algorithms that deserve our attention. A variant of OSPF, OSPF-OMP (optimized multipath), is implemented by exchanging special traffic load

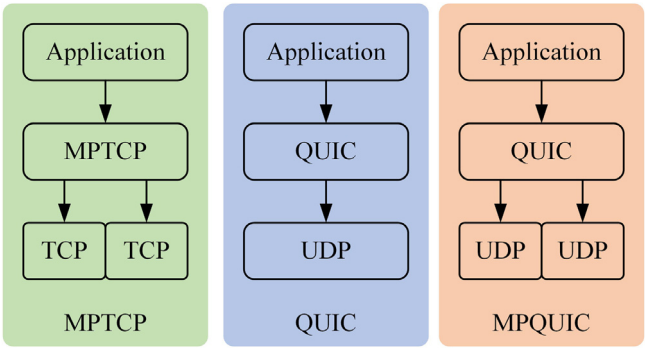

Fig. 17. The comparison between MPTCP, QUIC and MPQUIC. MPTCP decomposes a single flow into multiple TCP subflows, and QUIC multiplexes applications into one UDP flow for transmission. MPQUIC allocates applications over multiple UDP subflows.

control messages to allow traffic to be divided unevenly on equal-cost paths, and a hash function is used to generate the probability of each path [152,153].

Inspired by MPTCP, Viernickel et al. [154]extended and designed Multi-path Quick UDP Internet Connection (MPQUIC), which enables the QUIC protocol developed by Google to accelerate web traffic to support different paths, such as WiFi and LTE on smart devices, or IPv4 and IPv6 on hosts. The protocol can be deployed in user environments without changing operating systems like MPTCP and SCTP, and is superior to MPTCP in terms of download time and throughput. The comparison between MPTCP, QUIC and MPQUIC is shown in Fig. 17.

Aside from the above specific protocols, dynamic uneven traffic splitting according to the network status shows a stronger load balancing capability. Tomovic et al. [155] check in advance whether the multipath can meet the bandwidth requirements when performing traffic splitting, and adjusts the amount of traffic allocated to different paths based on the periodically-stated network status. Zhou et al. [156] have improved the even distribution of traffic in the ECMP algorithm, and proposed a weighted allocation scheme for link capacity to improve throughput. Wang et al. [157] directly have used the latency time as a measure and combines the pre-calculated k-minimum cost paths to achieve traffic splitting.

In Table 6, we list related flow-based load balancing solutions mentioned in this paper for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.3.4. Analysis and summary

- Regardless of load balancing methods are mentioned in this section, the core problem ultimately involved in most solutions comes down to the optimization problem. That is, the optimization solution is set by setting constraints that meet the target requirements to ensure the feasibility of the results. However, the excessive network scale and the limitation of computing resources make it difficult for the solution process to be completed in a short time, especially when there are many constraints. Therefore, as a challenging problem in the future, how to design a more efficient solution algorithm will make progress in reducing additional computing time and overall computing complexity. In addition, cross-domain solutions, such as intelligent optimization theory, may bring better inspiration.
- The dynamic routing update strategy can provide strong robustness to a certain extent to cope with changes in the network topology and the addition of new nodes, and even enhance the network's ability to resist failures. However, frequent updates of routing messages will cause routing oscillations and give rise to network instability, making it difficult to converge to a stable state. Therefore, how to achieve a balance between update frequency and network stability requires further in-depth thinking and exploration.
- In fact, the multi-path transmission technology is not an emerging technology, but for SDN, it shows a newer application scenario. Varieties of protocols have been proposed for multi-path transmission, operating from different parts of the Internet, from the application to the link layer and the physical layer. If readers want to know more about traffic splitting and multipath transmission, please refer to [59,60], the authors give a detailed introduction. And how to seamlessly connect these protocols with the SDN network becomes the next step worthy of attention.
- At the terminal, the sequential reorganization and delivery of traffic fragments is still a problem for multipath transmission, especially when the shortest and longest path lengths or hops are very different, there will be latency differences that cannot be ignored between parallel paths. To make matters worse, if managers deliberately pursue fine-grained flow control, this problem will be further exacerbated. Although per-packet will provide better load balancing than per-flow, the overall overhead needs to be considered.
- Controller placement and switch migration strategies can affect communication and negotiation between controllers to a certain extent, thereby affecting cross-domain communication latency and the interaction ability of different networks. Southbound interface protocol also need to be adjusted to varying degrees according to the deployment architecture and mapping relationship between the control plane and the data plane.

4.4. Flow table management

We have mentioned in Section 3.2 that each OpenFlow switch is configured with one or more flow tables to store the flow operation decisions issued by the controller. This rule cache-like operation not only enables the switching device to focus on manipulating packets, but also avoids the controller being accessed frequently. However, as the heart of the OpenFlow switch, the match–action mode of the flow table is facing severe challenges both internally and externally.

Internally, in order to support more fine-grained operations, the matching process of the SDN switch needs to check more header fields than the 5-tuples in traditional network switches. For example, OpenFlow has grown from the 12 match fields supported in version 1.0 to 44 in version 1.5, and the maximum number of bits required for a single flow entry has increased from 248 bits to 773 bits [158]. Multi-field processing requirements continue to increase the size of the flow table and the complexity of the matching process. And in the long run, as the number of new protocols continues to grow, the number of fields supported by OpenFlow does not show a downward trend. However, due to cost factors, the memory space used to store the flow table is struggling to keep up with. Even the most advanced OpenFlow switch can hold no more than 160K rules (OpenFlow v1.0 specification) [159].

Externally, the expansion of the network scale has caused the switching equipment to bear extremely challenging data pressure. For example, in DCNs, in order to support next-generation networks, packet processing needs to meet network throughput of 40–100 Gbps, and core switching equipment is even forced to provide processing speeds in excess of 400 Gbps. Such high intensity requirements will make the equipment run at full load all the time. What is worse, the separate of forwarding and control in SDN makes application development faster. Applications revise existing protocols or develop new ones to provide advanced functionality to meet a variety of user requirements, which will

Table 6
Flow-based load balancing solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [146]	ECMP	The ECMP scheme based on fat-tree topology is implemented in SDN.	POX Mininet	Up to 94.5% reduction in Packet Transmission Time and reduction in the number of lost packets 15.72%.
Ref. [148]	MPTCP	A multipath OpenFlow controller smoc was created that uses only the topology information of the network to avoid possible conflicts.	POX iperf	The average increase in throughput is 2.17 times in different structures.
Ref. [149]	MPTCP	The collected topology information is used to calculate key parameters for each subflow and to improve the routing and load balancing modules accordingly.	Floodlight Mininet	Reduces load balancing convergence time by 67.7%–85.5%.
Ref. [150]	MPTCP	Based on a mixed-integer linear planning model to determine the number of MPTCP subflows and the routing.	Mininet	Throughput has increased by 25%.
Ref. [155]	Dynamic Uneven Traffic Splitting	When splitting traffic, check in advance that the multiple paths meet the bandwidth requirements and adjust the traffic assigned to the different paths based on the periodically network state.	Floodlight Mininet	Better QOS request acceptance rates and lower route refactoring costs are achieved.
Ref. [156]	Dynamic Uneven Traffic Splitting	The uniform distribution of traffic in the ECMP is improved, and a weighted distribution scheme based on link capacity is proposed.	htsim	Reducing the variation in the flow bandwidths by as much as 25x and variation in flow bandwidth by 3x. The maximum number of table entries required has been reduced by 70%.
Ref. [157]	Dynamic Uneven Traffic Splitting	Traffic segmentation is achieved directly using delay time as a metric combined with a precomputed k minimum cost paths.	Mininet	Compared with ECMP, DFSM reduces 10% to 48% latency. Compared with label-switched tunneling, reduces the standard deviation of path latency from 10% to 7%. Compared with even traffic splitting, DFSM reduces the standard deviation of path latency from 14% to 7%.

bring more new header fields. Each field needs to be defined by a different syntax, such as range or prefix, and requires different matching methods, such as exact match (EM), range match (RM), or longest prefix match (LPM) [160]. This poses an unprecedented obstacles to existing rule storage architectures and find-match algorithms, as it is not possible to address all operating modes at once in a simple and unified way.

The conflict between the rich action requirements, rigid traffic operation modes, and device performance has resulted in the data processing process always failing to exert the best performance, and the latency boundaries for traffic delivery are constantly being raised. To this end, researchers have begun to look at two core aspects of packet classification and rule management to improve the processing speed of switching devices.

4.4.1. Packet classification

Packet classification has long been one of the key issues in the design of high-performance switching devices, requiring that incoming packet header fields be matched according to a set of predefined rules and that the corresponding actions be performed rooted in the matching results. For SDN, this problem is further highlighted due to the fine-grained operation supported by OpenFlow, and the idea that the traditional packet classification mainly focuses on fast lookup of tables is no longer applicable. So the improved method starts from both hardware and algorithm to solve classification complexity and time-consuming.

In terms of hardware, the vast majority of existing solutions store the flow table in Ternary Content Addressable Memory (TCAM) to ensure matching flexibility and high-speed classification performance. Indeed, mature commercial TCAM chips can perform 360 million searches on each table per second due to their parallel search capabilities and support for wildcard matching [161]. However, everything has two sides, and TCAM is also difficult to get rid of its limitations: (a) First, because the manufacturing process is not yet mature, the TCAM price remains high, and the single-bit cost is about 30 times that of Static Random Access Memory (SRAM) [162]. (b) The capacity of TCAM on a single device is very limited, with only a few thousand rules at most. (c) The operation of TCAM requires a lot of energy consumption. A 1MB TCAM chip will consume more than 15 W of power, accounting for about 25% of the total power consumption of the entire Application Specific Integrated Circuit (ASIC) [163].

Therefore, how to use the limited TCAM to maximize the classification performance, especially the lookup speed and time overhead, has become the focus of attention. Congdon et al. [163] have used temporal locality in network to predict the type of incoming packets, thus bypassing flow-based or TCAM-based lookups and speculate forwarding if conditions permit. By contrast, Chen et al. [164] have improved the lookup architecture and proposed a hybrid pipeline based on NAND-NOR TCAM and pipelined-TCAM to provide matching support for unequal word lengths.

In addition to TCAM, hardware-accelerated devices such as Field Programmable Gate Array (FPGA) [165,166] and GPU [167] are also in use. However, given the limitations of the hardware architecture itself, its performance is not expected to increase significantly in a short period. In some cases, such as cloud computing, the rule set size has grown faster than the evolutionary capacity of hardware, especially TCAM [168]. At the same time, considering the longer-term costs, software-based classification methods have begun to play an alternative role.

Hash table is one of the most commonly used data structures in many classification methods. It inserts and queries by mapping key values to the position calculated by a hash function in the table to speed up classification and improve efficiency. Typical applications such as Open vSwitch, a virtual switch supporting OpenFlow, use a variant of tuple space search (TSS), called priority sorting tuple space search (PSTSS) to implement the flow table as a single hash table to match the rules for the kernel and user space. PSTSS puts rules with the same non-wildcard and wildcard bits into the same tuple group to generate hash keys, avoiding

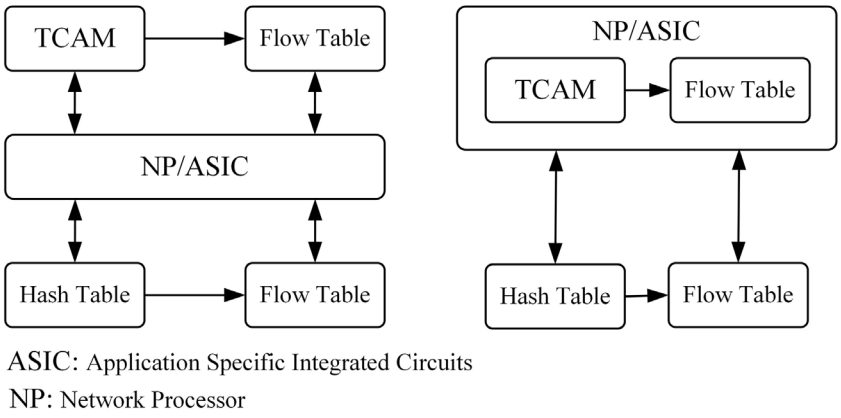

Fig. 18. Two typical Hash-TCAM mixed flow table lookup structures. The left side is the conventional architecture, and the right side is the search architecture when the NP or ASIC has a certain capacity of on-chip TCAM.

inserting all possible values for each rule [169]. In addition, Tuple Merge [170], an improved online matching algorithm based on TSS, and (Clustering-Based Packet Classification) CBPC [171], a packet classification algorithm based on ML to simplify the hash table process, both have made considerable progress in classification speed.

The drawback of hash table is that hash conflict occurs when multiple keywords are mapped to the same hash bucket through hash operation, and hash conflict is the main factor affecting hash lookup performance. Li et al. [172] have proposed a flow table lookup method, which combines multiple-cell hash table with TCAM. Hash table is the main measure for accurate matching and storage of flow table, and TCAM is used to deal with conflict overflow of hash table. Fig. 18 shows two typical Hash-TCAM hybrid flow table lookup structures.

Decision trees have also been actively studied for their high-speed classification capabilities for multi-field rules. The decision tree construction methods are mainly divided into equal-sized cutting and equal-dense splitting. The cutting-based scheme uses local optimization to partition the search space into many equal-sized subspaces [173–175]. In contrast, the splitting scheme aims at the same number of rules, and the subspace size after splitting are of different sizes [176–178]. Either way, the purpose of both is to decompose large rules into multiple small rule sets to speed up the classification process. Drawing on the advantages of both, Li et al. [177] have proposed an adaptive framework that combines cutting and splitting to solve the rule replication problem that both types of decision tree algorithms face simultaneously. The classification performance close to the hardware is achieved, and the number of memory accesses per unit time is tripled.

Decomposition-based technologies are attractive for low-latency network applications because they can take advantage of the parallelism provided by multicore processors to accelerate classification performance. Specifically, each field in the header is first searched over the entire rule set independently and then the search results are combined to obtain the final matching rule. Qu et al. [178] first used decomposition-based methods and optimization techniques for large-scale packet classification on multi-core processors. The author refines the decomposition process into a preprocessing stage, a searching stage, and a merging stage. In the preprocessing stage, a range-tree or hash table-based data structure is constructed for each field to reduce processing latency and improve throughput. In the search phase, the incoming header is divided into multiple fields, and each field is searched independently by a separate search thread. During the merge phase, parallel program threads are used to efficiently merge independent search results represented using bit vectors.

In addition to the three most commonly used classification algorithms mentioned above, ML including clustering algorithm [171] and neural network [43,179] have also been explored and used. The advantage of ML is that it is intelligent and does not need too much manual intervention. Especially for large-scale packet classification process, it can save a lot of time in the process of rule matching and lookup table building. However, as always, there are uncertainties and non-universalities in parameter adjustment.

In Table 7, we list related packet classification solutions mentioned in this paper for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.4.2. Rule management

Unlike traditional packet classification algorithms that use static rule lists, logically centralized controllers in SDN collect network state information to configure rules that need to be delivered to a switching device dynamically. As we mentioned earlier, due to the limited rule storage capacity in switching devices, the number of rules held simultaneously on each switch, whether SRAM or TCAM, is not sufficient to match the appropriate action to each incoming flow in a timely manner. To overcome this limitation, the switch can only meet the network throughput and latency requirements by maintaining high-frequency deletion, modification, and insertion of the flow table. This frequency is much higher than that of traditional networks and cannot be avoided in a short time. Therefore, under the premise of not affecting the performance of packet classification, fast and effective rule management measures are critical for maintaining the stability of the SDN architecture, especially high-performance networks. We review the SDN rule management into four parts: placement strategy, eviction strategy, compression strategy, and update strategy.

(a) Placement strategy

The first consideration in SDN rule management is which rules to deploy and where to deploy for maximum benefits, while at the same time meeting high-level policy and network constraints. First, not all applications need to react when the state of the network changes. Improper rule placement solution may cause the originally correct flow table to be discarded before use, which will force the controller to be tuned to handle unmatched flows frequently. Statistically, a data center with 100,000 new traffic per second requires an overall control channel bandwidth of at least 14 Gbps [180]. If bandwidth is not sufficient, it is obvious that latency will increase significantly. To make matters worse, the large number of applications that provide differentiated services to users, which will require specialized rule placement strategies. And since there may be switching devices with different software

Table 7
Flow-based load balancing solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [163]	TCAM-based	Use temporal locality in network to predict the type of incoming packets.	Not Given	Lookup process power consumption is reduced by 92%, while pass-through switching latency is reduced by 66%.
Ref. [164]	TCAM-based	A hybrid pipeline based on NAND-NOR TCAM and pipelined-TCAM to provide matching support for unequal word lengths.	Virtuoso HSPICE	Occupies up to 5% of the area to support long and variable word lengths, and reduces energy latency by 45% at the structural level.
Ref. [170]	Hash-based	An improved online matching algorithm based on TSS.	ClassBench MoonGen	34% faster for packet classification and 30% faster for rule updates.
Ref. [172]	Hash-based	A flow table lookup method that combines multiple-cell hash table with TCAM.	SystemC2.1 VC++6.0	More than 90% cost savings over TCAM alone.
Ref. [173]	Decision tree-based	Intelligently partition rule lists into multiple trees to support cutting methods and reduce rule duplication.	ClassBench	Packet classification is 58% faster.
Ref. [174]	Decision tree-based	A new cutting scheme and its corresponding decision tree algorithm are proposed.	ClassBench	The average throughput increase is 2.1 times.
Ref. [175]	Decision tree-based	Combines equal size cutting and pivot based cutting to increase packet classification efficiency.	ClassBench	The size of the flow table is reduced by almost 70%.
Ref. [176]	Decision tree-based	The algorithm converts rules into points and uses a clustering algorithm for efficient partitioning.	Physical Testbed	This is the first FPGA design that supports 10K rule sets while sustaining 100 Gbps throughput.
Ref. [177]	Decision tree-based	An adaptive framework that combines cutting and splitting to solve the rule replication problem.	ClassBench	10x reduction in memory requirements and 3x reduction in average number of memory accesses.
Ref. [178]	Decomposition-based	The decomposition process is refined into a pre-processing phase, a searching phase, and a merging phase.	openSUSE iperf	For a 32K rule set, a total throughput of 48 million packets per second and a processing latency of 2us per packet can be achieved.
Ref. [43]	ML-based	A modified linear unit deep neural network is used to process large rule sets.	TensorFlow	Classification accuracy improved by an average of 4.37% for the 1K rule set, 5.73% for the 5K rule set, and 6.1% for the 10K rule set.
Ref. [179]	ML-based	Self-organizing maps-based high-speed packet classification.	ClassBench	The number of searched rules is reduced significantly.

and hardware attributes, ignoring these differences when placing rules may face serious performance challenges.

The current rule placement strategy mainly starts from two aspects. On the one hand, the controller determines the optimal location for issuing rules according to the requirements of the packet forwarding path, with the goal of maximizing the number of flows accommodated in the network. On the other hand, considering the storage capacity of the switching equipment (especially TCAM) and the latency of issuing rules, the goal is to minimize the total number of rules issued.

For the former, relaxing strict routing policies is a good solution [181–183]. We also mentioned that the routing of most data packets in SDN is guided by rules issued by the controller. Thus, the relaxation of the routing strategy means that controllers will have more options to obtain a better rule deployment location. Better locations will yield better benefits in terms of latency and cost.

For the latter, reducing the number of rules issued has become a key concern. First, reducing the number of rules will prevent TCAM from being changed frequently. Due to the presence of wildcards, the rules in TCAM have a strict priority and the rules with the highest physical address are returned in the matching process. Therefore, when inserting a rule, TCAM needs to move all rules with higher priority to create space for the new rule. Similarly, after deleting a rule, all rules with higher priority need to move down [184]. An example of TCAM rule update is shown in Fig. 19. Secondly, the rule delivery latency will be effectively reduced. The test shows that when the number of switches that need rule update is reduced from 500 to 50, the setup time of single path will be reduced from 490 ms to 11ms [185].

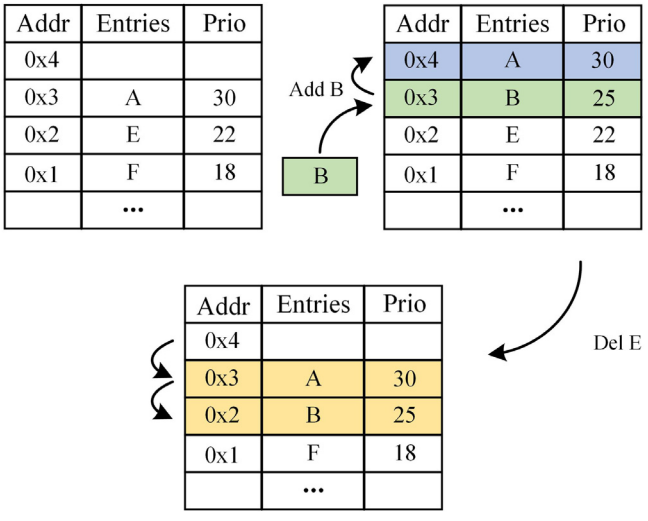

Fig. 19. An example flow entry insertion and deletion in a TCAM table. Before new entry B is inserted, entry A with a higher priority needs to be moved up. After entry E is deleted, entries A and B with higher priority need to move down.

For the case where the TCAM capacity is insufficient, the rule set can be partitioned to meet the policy and then used as a distributed storage method. The controller collects the remaining storage space and location information on the switch in real time. When any node in the control domain runs out of space, other nodes with large remaining storage space will assist in storage. This method is suitable for SDN-IOT networks [186,187].

(b) Eviction strategy

Due to the limitation of the storage capacity of the switch, the flow table will be filled up quickly and under high load for a long time in the big data environment. At this point, OpenFlow will send an error message to the controller to reject the insertion of a new entry. Therefore, after the placement position of the flow is determined, it is necessary to find and remove the entries that are already in the invalid state or low-priority entries in the flow table to free enough space to ensure that new entries can be accommodated. Current flow entry eviction strategies in SDN can be divided into passive timeout strategies and active deletion strategies.

Since OpenFlow v1.3, the controller can use both hard_timeout and idle_timeout mechanisms to maintain high utilization of the flow table [188]. Among them, the hard_timeout is calculated from the time the flow entry is issued, and the entry is cleared when it expires. The idle_timeout measures the idle period of the flow entry, that is, the unused time, and deletes the entry when it exceeds the threshold. When a new flow entry is issued, the idle_timeout or hard_timeout field value is set at the same time. If both exist and are valid, the time that expires first prevails. Starting with OpenFlow v1.4, the timeout mechanism can be operated by the switch independently of the controller according to the flow entry status and configuration [189].

However, the fixed timeout value cannot always maintain the efficiency of the flow table, as the network status and resource usage are always changing. Hence, based on the collected network information, such as link congestion and flow table utilization, adaptive timeout adjustment mechanisms is widely proposed and used [190–192]. The controller will continuously improve the timeout value set according to the network status. Note that a single timeout value does not apply to all flow entries. For example, the frequency of rule updating in time-dependent mouse flow will be much higher than that in elephant flow. If the timeout value is not set properly, it will lead to the accumulation of mouse flow rules or the early cleaning of elephant flow rules. Therefore, researchers began to set different timeout values for different rules according to the characteristics or priority differences between flows, so as to provide differentiated services [193–195].

The active deletion strategy no longer depends on the timeout threshold. Instead, the controller performs real-time deletion or replacement of the flow entry after calculating or inferring based on the network status uploaded by the underlying device, saving the waiting time required for the timeout value to take effect. At present, some simple active strategies such as random deletion, Least Recently Used deletion, and FIFO algorithms can be used in the OpenFlow seamlessly [180]. High-level measures use the idea of caching or detection based on specific indicators or fields. For example, Li et al. [196] have kept the timed-out entries in the spare storage space for continued retention. If the packets can arrive in time, the entries are recalled back to the table. Ying et al. [197] have determined whether the flow entry has value for continued retention by detecting the FIN field of the incoming packet. In addition, the idea of using AI algorithms to predict the activity of flow entries based on historical records has also achieved satisfactory results [198–200].

It should be noted that the active deletion policy and the timeout policy do not contradict each other. The former can interrupt the process of the latter at any time to realize the early eviction of flow entries, and improve the overall response sensitivity of the SDN network. The latter can be used as a bailout policy for the former to ensure that the performance does not exceed the minimum.

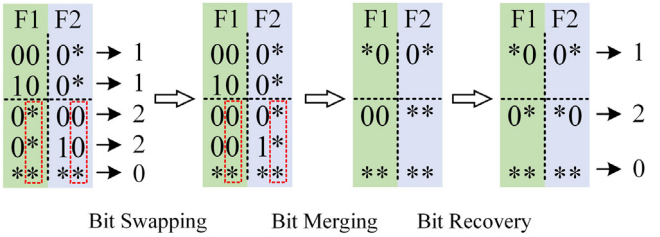

Fig. 20. Example of the bit weaving in [209].

(c) Compression strategy

The deletion strategy can only solve the space occupation of invalid entries and the redundancy, overlap, or inclusion relationship between flow entries is also one of the reasons for the low utilization of TCAM. The compression strategy eliminates the redundant part between the rules, so that one flow entry can be reused to process different flows. This type of strategy not only reduces the storage space occupancy rate, retain the ability to digest burst traffic, but also avoids the need for controllers to be frequently involved in switch requests and rule installation times [159]. Studies have shown that latency differences between devices will increase by three orders of magnitude due to the rules insertion [201].

Traditional prefix-based IP routing table compression technology [202,203] and Access Control List (ACL)-oriented rule optimization [204] are not directly portable to SDN: (1) SDN flow table contains multiple types of fields, and is not limited to prefix forms. And a rule can contains multiple matching conditions, not just for IP. (2) ACL generally involves only 2–4 matching operations, such as Deny or Permit, which cannot meet OpenFlow's requirements for fine-grained flow operations [205]. Even worse, the rules in the ACL prohibit the existence of conflicts strictly, while SDN allows rule conflicts to some extent. Currently, compression strategies in SDN are optimized from both rule-based and routing based.

The rule-based solution focuses on merging multiple entries so that a rule can be used to match multiple flows, the most direct method being to use wildcards based on the Do not care feature of TCAM [206–208]. Flow entries with the same action but different matching field values will be inserted as wildcards instead of exact field matching to further compress existing entries even if some of them already have wildcards. A non-prefix bit weaving scheme has proposed by [209] in which TCAM entries with the same operation and predicates differing by only one bit can be combined into one entry by replacing that bit with *. An example of this scheme is shown in Fig. 20, which mainly includes three steps of A, B, and C to achieve entry compression. Wildcards are supported in almost all fields of the current OpenFlow specification, and almost all flow entries contain wildcards. In particular, the OpenFlow V1.5 specification has started to support the use of wildcards to set the type of packet header and modify the value of the header.

Range expansion is another key issue that the compression process needs to face. The problem is caused by the port field being specified using a range, such as [1,65534] [210]. This means that TCAM's normal matching pattern cannot be used directly, but is forced to pay multiple entry to store the range fields. The traditional solution is mainly based on prefix extension for range representation, but the high expansion rate becomes a tricky

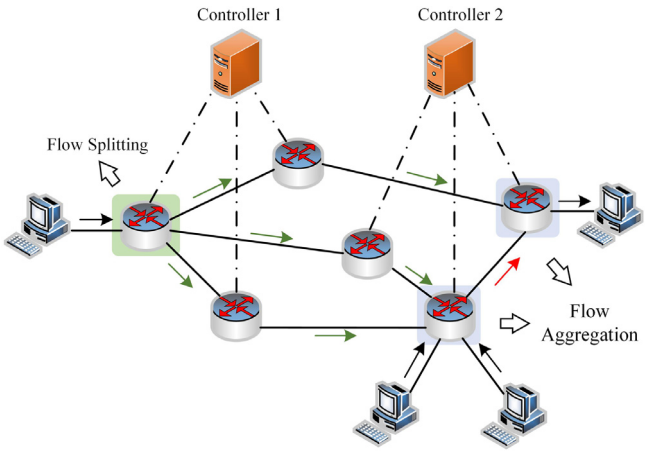

Fig. 21. Schematic diagram of flow splitting and flow aggregation.

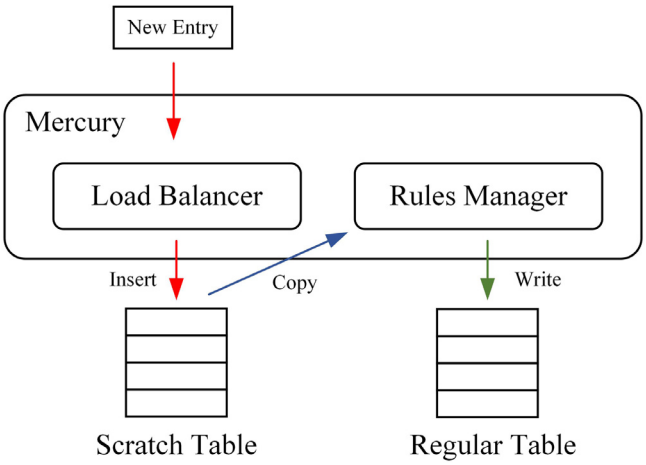

Fig. 22. Mercury Architecture proposed in [219]. The load manager is responsible for inserting new flow entries into the scratch table, and the rule manager regularly migrates the rules of the scratch table to the regular table.

factor [162]. The main core idea of current solutions to such problems is through coding [211–213]. In addition, the idea of implementing range matching by changing the TCAM hardware architecture has also been proposed by [214], but seems unlikely to be feasible in the foreseeable future, as TCAM is already entrenched.

As we also mentioned in Section 4.3., most traffic management methods can rely on default policies to operate, such as the Dijkstras-based ones in Opendaylight and Beacon controllers. These default policies usually have the lowest priority guarantee [180]. Therefore, under the premise of not affecting performance, using the default rules to operate on some flows with similar characteristics, such as filtering, routing, or buffering in the same queue, can also achieve a local minimization of the number of flow entries [181].

Different from flow entry reduction in rule-based compression, the routing-based method starts from the perspective of flow by scheduling the traffic so that multiple flows routed through multiple paths have the same forwarding requirements in a certain control domain. At this time, the controller only needs to install rules on the same set of switches that these flows pass to complete the matching and operation, reducing the number of switches that need to deploy rules. Flow aggregation is the most widely used technique currently [215–217]. Flow aggregation combines a group of independent flows (such as mouse flows) with the same forwarding and performance requirements into a whole, which can be treated as a single flow. This operation is equivalent to reducing the number of flows in the network and limiting the number of flow rules that need to be installed in the switch.

Actually, Flow aggregation is not contradictory to the flow splitting we described in Section 4.3. Flow splitting is for load balancing, while flow aggregation is for compressing the flow table space. As shown in Fig. 21, to a certain extent, the two can be converted at any time, i.e., the decision on which measure to use is based on changes in network state.

SR, which we mentioned and introduced in Section.4.3, can be regarded as a special routing-based flow table reduction method. Its contribution to flow table compression is to specify the packet forwarding path by encoding the routing information in the packet header. After receiving the packet, the intermediate node matches the link according to the label at the top of the stack and finds the corresponding forwarding exit, omitting the rule installation and matching process on the core switch.

(d) Update strategy

Supporting dynamic update of rules is the most significant difference between OpenFlow switches and traditional switches, which guarantees the switch's ability to handle unknown flows. However, rule updates (e.g., modification and insertion) have become the major bottlenecks that limit OpenFlow switches from meeting high throughput. In large-scale DCNs, latency associated with rule updates can accumulate thousands of packets waiting to be routed in a short period of time. In contrast, latency constraints for network reconfiguration and topology changes are limited to 25 ms [184]. Although the processing latency can be shortened by increasing the overall control plane processing power, such as using multiple controllers or switch migration (Section.IV.C), none of these approaches gets to the root of the problem. That is, the update latency is determined by the hardware write operation of the switch itself.

The speed of rule updates is a primary concern, especially for SDN that support fine-grained flow management. As we mentioned earlier, TCAM's the priority storage mechanism will result in a large number of rule moves for just one entry update if that entry needs to be inserted into the flow table. Therefore, under the premise that TCAM cannot be abandoned, changing the existing TCAM-based flow table design architecture [218–220] or optimizing the way of inserting entries will bring benefits in terms of update speed [184,221,222]. Typical design of the former such as the Mercury architecture proposed by [219] as shown in Fig. 22, implements a regular multi-step insertion by dividing a large TCAM table into two small tables. The latter focuses on minimizing the number of entry moves by using directed acyclic graphs or by minimizing the rule insertion steps.

In fact, the priority relationship that exists between rules is the most fundamental cause of updates latency. Even if no new rules are inserted, the rule priority adjustment caused by dynamic scheduling alone is sufficient to contribute most of the update operations. Tests have shown that even when OpenFlow switch rules are installed in active mode, the latency under the SDN architecture is still 27.95 times greater than that of traditional networks. The end-to-end latency caused by switch rule priority updates even exceeded 97% of the overall latency in extreme cases [223]. The existing methods have made some promising progress in addressing the latency caused by priority, including pre-queuing or priority adjustment outside the flow table, such as the controller side, to avoid multiple adjustments of the rule order within the flow table [224]. It is also a good option to avoid

unnecessary priority updates by optimizing the order of multiple flow table operations, such as deletion and replacement [225].

Most of the existing fast update algorithms default that the normal operations of the flow are not affected when the rules are updated. This is contrary to the facts. In fact, the table lookup leak rate due to frequent updates in SDN is as high as 25% [218]. Update latency caused by the rule modification will cause the interruption or loss of packet, if the rule being used or about to be used is modified during transmission process. Therefore, non-blocking updates have become an indispensable skill for SDN to implement fast dynamic configuration of rules and support fine-grained control. While allowing both old and new rules to exist in the same flow table is the most straightforward [226], decoupling the lookup and update processes is a more efficient and mainstream way to support non-blocking updates [227,228].

In Table 8, we list related rule management solutions mentioned in this paper for comparison, from the perspective of objectives, main ideas, testing tools, and contributions.

4.4.3. Analysis and summary

- Packet classification is not the same as the traffic classification we mentioned in Section 4.1. The differences are: (1) Traffic classification only requires the identification of the target traffic category. The purpose of packet classification is to distinguish each data packet in a fine-grained manner to ensure that each packet is matched to the next action accurately. (2) Different classification methods. The traffic classification process often involves only a specific field in the flow to complete the identification process, such as DSCP in IPV4. And this process can be classified using ML based on statistical characteristics. Packet classification requires multi-field matching for each data packet in the flow. (3) The packet classification process has strict priority restrictions, while the traffic classification process does not have this requirement.
- Although in this section we review the advantages that various packet classification methods and flow table management measures bring to low-latency communications, in practice, they still need to be applied selectively based on the actual network conditions. Because each method brings benefits with varying degrees of defects. For example, using a rule-based timeout policy can reduce the switch occupation, but at the cost of a 40% increase in control channel overhead. Wildcard-based matching schemes require fewer rules, yet they weaken the ability of OpenFlow switches to perform fine-grained matching and the application's visibility to the network, as the controller will not have access to richer underlying statistics [159].
- Consistency is still the most important point to consider in the rules management process. This includes not only consistency in updating states, but also semantic consistency. For flow table compression, ensuring the consistency of the rule set before and after compression is critical to ensuring correct flow action. For distributed switches, consistent update of rule status can avoid routing anomalies and deterioration of network performance. When the rule issuance and installation process cannot be completed on multiple switches simultaneously, the end-to-end latency will increase significantly [223].
- Not only updates, but also rule management, including delete and compression operations, will contribute more to low latency transfers if non-blocking operations are supported. However, there are still many challenges to truly achieve non-blocking management. For example, although non-blocking operations between rule management and flow matching have been partially implemented, non-blocking operations cannot be adversely affected due to the interdependency of operations, where multiple operations must be executed in a specific order to ensure the correct network state. Especially in distributed SDN network architecture, out-of-order execution of operations in different control domains will cause cross-domain routing to fail [228].

4.5. Other

This section reviews some emerging network technologies used in SDN, which can play a role in low-latency communication in SDN to varying degrees.

4.5.1. Information centric network

As a brand-new design concept of future network, information center network (ICN) transforms the end-to-end communication mode of traditional TCP/IP network into a content-centered information exchange model to cope with the increasing network traffic and the current drawbacks of IP network [229]. Through this reform, the network architecture shifts the attention from the host to the information that users really care about. Specifically, the new forwarding, naming and caching mechanism are the most fundamental driving force for ICN to move forward. First, the demand-driven forwarding mechanism severed the inherent connection between the data sender and the data receiver, so that the data location can be ignored. On this basis, ICN replaces the IP routing mechanism in traditional network with a packet-naming based routing mechanism that allows data to be further removed from a specific storage location and reused in the network. Finally, the node caching mechanism enables copies of content to be cached at the edge of the network close to the requester, in order to respond to user-requested content quickly.

For SDN, the emergence of ICN brings new development ideas: (1) ICN network nodes take advantage of content caching to ensure that user requirements can be met quickly, avoiding the time overhead and data loss associated with end-to-end long-distance transmission [230]. (2) Naming-based data representation eliminates the need for the controller to issue rules to process the forwarded content. This change not only frees the southbound communication channel from busy information interaction, but also allows the controller to focus more on network topology and route optimization. (3) The information-centric design concept provides SDN with the ability to identify content directly without the need for packet processing, such as DPI or traffic identification, to acquire content metadata to optimize network behavior [231].

At present, a variety of network structures, which combine SDN centralized control logic and ICN cache routing capability, have been proposed and good progress has been made in easing end-to-end latency and traffic engineering [230–233]. However, the combination of SDN and ICN still has many problems that need to be further solved in terms of scalability and communication protocol standardization. More detailed research content can be found in [234,235] if the reader has more interest.

4.5.2. Time sensitive network

In order to meet the more stringent latency-sensitive business requirements, the IEEE802.1 working group has upgraded the Ethernet audio/video bridge technology (AVB) to the time-sensitive network (TSN). In the AVB standard, the transmission latency of audio and video services is guaranteed by technologies such as clock synchronization (IEEE802.1AS), resource reservation (IEEE802.1Qat), and flow control (IEEE802.1Qav). TSN expands on the AVB protocol by revising or adding more control and scheduling standards to ensure that the traffic still has certain end-to-end low latency and jitter in high-occupancy networks. The main standardizations and functions of TSN are shown in Table 9 [236]. At the same time, the application scope of TSN has also expanded from AVB serving audio and video purely to vehicle application, industrial control and other fields, and has received strong support from various equipment manufacturers including Cisco, NXP, Broadcom, and H3C.

The purpose of SDN when it was proposed was to simplify the DCN, not to support the real-time communications required

Table 8
Rule management solutions in SDN.

Proposals	Method	Main Idea	Tools	Contributions
Ref. [181]	Placement Strategy	A resource constraint rule allocation model with relaxed routing policy.	Not Given	75%–90% of traffic forwarding needs can be met using 50% of the memory capacity.
Ref. [182]	Placement Strategy	Traffic rule distribution is based on three stages: forwarding path selection, flow rule placement, and rule redistribution.	POX Mininet	End-to-end latency and QoS conflicts are reduced by approximately 46% and 75%, respectively.
Ref. [183]	Placement Strategy	A mobility-aware adaptive flow rule layout scheme consisting of path estimation module and flow management module.	NS-3	The average latency is reduced by approximately 8%-28.12% and the cost of rule placement is reduced by approximately 38.25%.
Ref. [184]	Placement Strategy	Use techniques such as instruction type conversion, pseudo-deletion, and matching field distances to update flow rules.	Software TCAM Classbench	For a stream table containing 1k entries, the latency of each update is less than 12ms.
Ref. [185]	Placement Strategy	Minimize the number of switches that need to receive rule updates.	Floodlight Mininet	It reduces the median and 99-percentile latencies to 5.9ms and 7ms.
Ref. [191]	Eviction Strategy	Modify the idle timeout threshold to achieve a dynamic balance between the flow table write-in rate and the expired rate.	Matlab	The average number of active flow entries increased by approximately 40.5% and the match success rate increased by approximately 29%.
Ref. [192]	Eviction Strategy	Consider TCAM space, traffic classification, and link handover to optimize flow table management in software-defined satellite networks.	POX OpenStack	15.27%–24.34% reduction in flow table size, 8.2%–10.4% reduction in drop-flow rate, table-misses for high priority flows are reduced by 4.92% to 5.7%.
Ref. [193]	Eviction Strategy	Different dynamic timeout values are assigned to predictable and unpredictable flows.	Ryu Mininet	The miss ratio is reduced by about 0.3% to 11.5%.
Ref. [194]	Eviction Strategy	SDN network timeout mechanism based on hybrid Q-learning.	ClassBench	Table-hit rate increased from 97.6% to 99.4% while reducing the number of overflow by 83.8%.
Ref. [196]	Eviction Strategy	The timeout entry is stored in free storage and the entry is transferred back to the table if the packet can arrive in time.	Not Given	Table-hit rate increased by 60%.
Ref. [198]	Eviction Strategy	Learn from the history of the flow entry to predict when the flow entry is referenced.	Not Given	23% lower capacity misses and 2%–8% lower overhead.
Ref. [199]	Eviction Strategy	A novel reinforcement learning approach that automatically finds the values of the decision parameter.	Mininet	Performance improved by about 60% in reducing overhead and by about 14% in table-hit rate.
Ref. [206]	Compression Strategy	Efficiently route the mice flow by assigning internal pseudo-mac addresses to edge switches and hosts.	Ryu Mininet	Reduce the number of rules in edge switch, aggregation switch, and central switch by more than 90%.
Ref. [208]	Compression Strategy	Partition the field space into logical structures called buckets, and cache the bucket and all associated rules.	ClassBench	Reduces traffic setup requests by an order of magnitude, saving half the control bandwidth.
Ref. [209]	Compression Strategy	TCAM entries with the same operation and predicates differ by only one bit can be combined into a single entry by replacing the bit with *.	ClassBench Microsoft.Net	The average compression ratio reached 23.6%.
Ref. [211]	Compression Strategy	An effective multi-field range coding scheme.	ClassBench	For the 10K rule, up to 86% reduction in TCAM.
Ref. [213]	Compression Strategy	Combines extra bits and its fallback scheme for rule compression.	ClassBench	The redundancy of range rules can be reduced by approximately 36%.
Ref. [217]	Compression Strategy	An efficient flow tree structure is designed for flow aggregation.	POX Mininet	Reduction of flow entries by more than 90%.
Ref. [218]	Update Strategy	A dynamically updatable ternary segmented aging Bloom filter.	ClassBench	It saves 37% of space overhead and has an average update latency of less than 2.5 ns.
Ref. [219]	Update Strategy	Manage TCAM through smart partitioning.	Varys	Less than 10% overhead, less than 10ms insertion time, and 2X–5X improvement in application performance.
Ref. [220]	Update Strategy	TCAM is used to cache the most commonly used rules, and the software is used to handle a small number of cache-missing rules.	Ryu Physical Testbed	By caching less than 5% of the rules, the cache hit rate meets 90%.

(*continued on next page*)

in industrial and vehicle control applications. The OpenFlow protocol, which relies on TCP communication, also fails to meet the stringent latency requirements of industrial communication because it always requires complex handshaking and control

Table 8 (*continued*).

Proposals	Method	Main Idea	Tools	Contributions
Ref. [221]	Update Strategy	A greedy strategy to accelerate flow entry update.	Ryu Physical Testbed	100x improvement in update efficiency for 1k rule sets.
Ref. [222]	Update Strategy	Use dependency graphs as key abstraction to minimize update latency.	Ryu CoVisor ClassBench	The median end-to-end latency is less than 12ms and 90% is less than 15ms.
Ref. [224]	Update Strategy	Reduce the complexity of the flow modification process in SDN switches by temporarily adjusting the priority of flow rules.	Ryu Physical Testbed	Average TCP throughput improves by up to 9.4%.
Ref. [225]	Update Strategy	An algorithm that combines the efficiency of rule replacement and the applicability of rule addition to compute operational sequences.	Not Given	Memory overhead is reduced by 90%.
Ref. [227]	Update Strategy	Cache microflows in the hash table to create fast paths for packet.	Open vSwitch OpenMP	The search speed of 75 million packets per second on common PC, which is 3.4 times faster than the original Open VSwitch.
Ref. [228]	Update Strategy	Multiple operations on the same flow rule are simplified, and updates can be represented by a feasible sequence of operations.	Ryu Hedera	With an update arrival rate of only 1.6/s, update latency can be reduced by 16x and average link bandwidth can be reduced by 30%.

Table 9
The main standardization and applications of TSN.

Standardization	Full Title	Application
IEEE 802.1AS	Time Synchronization for Time-Sensitive Applications	Flow Synchronization
IEEE802.1Qcp	YANG Data Model	Flow Management
IEEE802.1Qat	Stream Reservation Protocol	Flow Management
IEEE802.1Qcc	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements	Flow Management
IEEE802.1CS	Link-Local Reservation	Flow Management
IEEE802.1Qav	Forwarding and Queuing of Time Sensitive Streams	Flow Control
IEEE802.1Qbv	Enhancements for Scheduled Traffic	Flow Control
IEEE802.3br	Interspersing Express Traffic	Flow Control
IEEE802.1Qbu	Frame Preemption	Flow Control
IEEE802.1Qch	Cyclic Queuing and Forwarding	Flow Control
IEEE802.1Qcr	Asynchronous Traffic Shaping	Flow Control
IEEE802.1CB	Frame Replication and Elimination	Flow Integrity
IEEE802.1Qca	Path Control and Reservation	Flow Integrity
IEEE802.1Qci	Path Control and Reservation	Flow Integrity

mechanisms. However, researchers are not willing to give up the industrial advantages brought about by SDN's revolutionary innovative ideas. In fact, SDN does not specify which technology should be used to implement the underlying data plane, which leaves the door open for the introduction of TSN. Time-sensitive software-defined networks(TSSDN) are beginning to gain attention as a new network architecture that combines the advantages of SDN and TSN. The structure of TSSDN retains the separation of the control plane and data plane in SDN: the control plane performs centralized network configuration and status collection to maintain the dynamics of the network structure. The data plane implements flow control and scheduling based on the TSN standards to minimize the traffic transmission latency, and thus realizing hard real-time guarantee for key information [237].

So far, rich tests and evaluations have verified the superiority of TSSDN in low-latency communications, and a series of prospective research results have been obtained [238–241]. After all, TSN and TSSDN are in the initial stage of research, so the challenges including protocol compatibility, reliability and interoperability need more detailed solutions. In any case, the benefits of TSSDN cannot be hidden and will eventually gain greater attention.

4.5.3. Edge computing

The SDN architecture manages the underlying network through centralized control to complete the traffic scheduling and transmission required by users. However, SDN is facing severe challenges when applied to distributed environments such as IoT and IoV. Most IoT applications need real-time computing and interaction support, such as medical data generated by human sensors that needs to be processed immediately in emergencies, or driverless cars that need to respond to pedestrian movements in real time [242]. This makes network latency a primary consideration, and the way of sending network terminal data to centralized server for processing is no longer applicable.

Edge computing aims to ensure the low-latency required by applications by placing data collection, calculation, and scheduling functions closer to the network edge nodes. At this point, the data generated by the sensor node or remote user terminal are no longer be sent to the centralized controller directly. Instead, they are sent to the nearest node with secondary management functions for fast service response [243]. Thus, edge computing complements the centralized cloud architecture in terms of latency and can be seen as the implementation of cloud computing on the user side.

In fact, both SDN and cloud computing share some similar properties in terms of architecture and operation mode, that is, their core services are provided to the lower layers through centralized computing and long-distance transmission. Therefore, the introduction of edge computing can also alleviate the slow response exists in the centralized control mode of large-scale SDN. Edge computing nodes with certain management functions to form an edge layer that can process data will replace the switching nodes at the edge of the SDN network.

However, it needs to be clear that SDN and edge computing are not hostile, but mutually beneficial. While edge computing provides low-latency guarantee for applications in SDN, it also

relies on maintaining periodic communication with the SDN top layer controller for network configuration optimization. Because the global view that the top-level controller has is the best place to perform optimization analysis [242,243].

4.5.4. Network function virtualization

Although SDN's revolutionary separation of control and forwarding has changed the traditional rigid network communication mode, SDN still has limitations in meeting service flexibility and reducing overhead. The fundamental obstacle lies in the tight coupling between the realization of software network functions and hardware execution devices, which means that SDN still needs to rely on specific infrastructure to implement the corresponding functions, such as cross-domain end-to-end services that require heterogeneous controllers to support [244].

As a new network architecture framework, the network function virtualization (NFV) technology realizes dynamic and flexible deployment by transferring the service functions based on dedicated hardware in the traditional network to software running on a common platform. Therefore, by taking advantage of NFV's openness and flexibility to virtualize SDN elements, it has a crucial auxiliary role in shortening the time required for the development of new functions and improving the quality of SDN services [245].

Currently, the integration of SDN and NFV into software-defined network function virtualization (SDNFV) to achieve network control has become a research hotspot and has been carried out in many ways. SDNFV integrates the advantages of both SDN and NFV, and can intelligently place network services across domains, so that SDN elements and network functions can be strengthened at the same time to adapt to the changing network status constantly. For example, Bu et al. [246] have proposed a dynamic deployment mechanism of network functions based on SDNFV to provide dedicated communication functions for different applications. Liu et al. [247] have used mixed integer programming to study the multi-stream fast update problem in the SDNFV system. In addition to academic research, the relevant standardization organizations are also actively promoting the integration process of SDN and NFV. Such as Open Innovation Pipeline released by ONF to guide the industrial development of SDN and NFV [248]. European Telecommunication Standards Institute (ETSI) also gives constructive suggestions for using SDN in the NFV framework [249].

5. Challenges and future directions

In this section, we point out the open challenges and future directions for implementing low-latency communication in SDN, thus providing more references that are valuable for interested researchers to accelerate their research.

5.1. Open challenges

5.1.1. Hybrid SDN Networks

SDN brings a simpler and more efficient way to manage network. However, when it comes to migration costs and the resulting time consumption, network operators and service provider organizations are reluctant to completely transition from traditional networks to pure SDN for a short period of time. The formation of hybrid SDN networks by deploying SDN devices in legacy networks selectively has been accepted by most organizations. While hybrid SDN offers a compromise solution at this stage, the architecture faces more complex challenges than traditional and pure SDN networks, including hybrid SDN device and controller deployment, network traffic engineering mechanisms, and security defenses.

For low-latency communication, the measures and schemes we mentioned in the previous section are not fully applicable to hybrid SDN. Communication protocols between different devices may not operate in a coordinated manner, e.g. traditional switches tend to use distributed algorithms, while SDN is based on a global centralized view [21]. The incompatibility between the two will result in a significant increase in the data transmission time per hop. In addition, controllers need to have different management policies for different underlying forwarding devices to keep the network up and running, especially for interoperability across domains [12]. Therefore, in hybrid SDN, the low-latency transmission strategies are no longer just for a single network architecture, but needs to take care of both the stability of legacy network equipment and the flexibility of SDN. In particular, how to develop the programming and policy languages required for low-latency transmission is also an issue that has been worth investigating in hybrid SDN [19].

5.1.2. Consistency

The large-scale promotion and horizontal expansion of SDN need to be presented in a distributed formation inevitably. As we mentioned in the review above, distributed architecture has a significant help for latency reduction, especially for controlling response latency. Because at this point the underlying load is taken care of by the nearest controller, the long distance transmission of information is avoided [250]. In addition, the distributed architecture has also demonstrated its robustness and reliability advantages in terms of protection against single point of failure and link congestion.

However, management in a large distributed SDN environment can be a daunting task, i.e., the distributed architecture faces challenges from state and policy consistency: (1) Controllers that are far apart cannot maintain the same view of the underlying device at the same time. (2) Devices in different domains may have significant differences in response to the same update requests. Studies have shown that inconsistencies between controllers or controllers and switches have a significant impact on SDN applications and QoS [251]. In particular, consistency issues will likewise pose a serious obstacle to low-latency communications. For example, data forwarding across domains will be delayed for delivery or discarded due to inconsistent matching rules stored within different devices. Controller state differences will result in the underlying request receiving a response with varying latency.

Not only that, but how to achieve a balance between consistency and other properties such as stability, availability, etc., is also an issue that requires in-depth study. On the one hand, many industrial-level or enterprise-level control applications require high stability as a guarantee. High stability necessarily entails a reduced frequency of conformance updates, i.e. at the expense of strong consistency. On the other hand, areas such as the IoV, the IoT and satellite communications require a high level of consistency as a security guarantee, where weak consistency does not meet demand [12]. Adaptive consistency, which many researchers are pursuing currently, may be a promising solution. This solution breaks the dilemma of choosing between strong and weak consistency, and instead adjusts the level of consistency between devices periodically based on dynamic changes in the network state to meet desired performance requirements [250, 251].

For distributed SDNs, the resolution of controller consistency relies heavily on the east–west interface protocol. However, no standardized east–west protocols are being proposed and widely recognized currently. Conversely, the northbound interface protocol and development language are too numerous, which makes

the interface under the distributed control plane difficult to manage and affects the formulation of the consistency strategy. Therefore, developing standardized northbound and east–west interface protocols similar to OpenFlow to maintain state and policy consistency has become an urgent task [11,13].

5.1.3. Scalability

Scalability is another key issue that needs to be addressed in the development of low-latency communication technologies in SDN. Scalability refers to the system's ability to adapt to change, i.e., to have a high degree of stability in system performance and management complexity without significant degradation as users and resources increase [252]. Systems and devices with high scalability can achieve linear growth in overall system capacity with few changes or even just additions to hardware equipment, thus meeting high throughput and low latency performance requirements.

Scalability in SDN stem primarily from the decoupling of control plane and data plane, and the logically centralized controller [12]. As we mentioned above, large-scale distributed architecture is the ultimate development trend of SDN. Regardless of the controller's functional complexity, excessive traffic growth will result in the controller not being able to handle all incoming requests simultaneously. In addition, the farther the forwarding device is physically from the control center, the more it will be affected by signaling delays and lose its ability to function properly. These factors all limit the size and radius of SDN usage severely [253].

While scalability concerns have been mitigated to some extent by SDN device performance enhancements, such as high-performance multi-core controllers [100–103], however, we believe that scalability is fundamentally different from mere performance enhancements. Scalability is the balance of high performance, stability and adaptiveness, and focuses on horizontal scaling of system performance. Performance optimization, however, is limited to the performance metrics of a single device and needs to be supported by hardware technology developments. In fact, the improvement of device performance has been unable to keep up with the increase in demand, and the cost is difficult to reap the benefits in a short period of time. Therefore, the key to achieving scalability is still to solve the problems of distributed SDN architecture, such as the lack of communication interfaces and incompatible devices.

5.1.4. ML

In our review, we found that the vast majority of low-latency communication schemes are attempting or have used ML algorithms for modeling to enhance the intelligence of the optimization process. While the experimental results are somewhat satisfactory, we argue that these experimental procedures are highly subjective and limited, and may not be consistent with the actual process. The fundamental reason is that ML algorithms are still a new thing with many challenges to solve:

(1) The training and testing process of ML models requires high quality datasets. However, the data acquisition and preprocessing process is resource-intensive, time-consuming, and highly time-sensitive. In addition, ML models have a strong dependence on the degree of balance of the datasets. Often the samples of interest account for a small proportion of the dataset due to their difficulty in collection and cannot be accurately detected.

(2) Different ML models have different operating principles and will yield different results when faced with different objectives. However, how to determine the best classification model based on environmental characteristics has been plaguing researchers. Moreover, the selection process of model parameters and classification features is also irregular and relies on constant attempts to obtain approximate optimal results.

(3) ML algorithms need to be updated to ensure the validity of the results constantly. However, the model update process is influenced by a variety of factors, including the frequency of updates and the manner in which they are made. How to strike a balance between multiple considerations is no easy task. In addition, ensuring performance consistency before and after model updating is also an important part.

(4) ML algorithms will fall into spatial complexity when faced with large-scale data learning. The recent rise of deep learning provides a good solution to the problematic thinking. Deep learning relies heavily on the nonlinear fitting ability of multilayered neural networks to exert strong classification performance. However, the training process of deep learning again faces the problem of training time complexity.

5.1.5. Interface protocols

For East/Westbound interfaces, due to differences in design schemes and operating principles between different types of SDN controllers, a unified interface protocol has not been formed to standardize communication processes such as handshake, routing, and synchronization. Similarly, since the northbound interface function depends on application requirements, the controller has no way to provide high-level abstractions to upper-level developers through a fixed interface language. To make matters worse, due to a long-term lack of unified interface specifications, developers or suppliers began to place dedicated high-level interfaces inside the controller to reduce operating costs, which undoubtedly further promoted the standardization process in the opposite direction [254]. For southbound interface, even though OpenFlow has become the popular and most widely accepted standardized protocol, however, many defects including scalability and compatibility have hindered the further development of OpenFlow. Especially when the SDN architecture is used in new fields, many new protocols, such as ForCES, OpFlex, NetConf, HUBsFlow, etc., have been proposed to supplement the gap left by OpenFlow or directly as a substitute [16]. Although ONF, IETF, Northbound Interface Work Group (NBIWG), Migration Work Group (MWG) and other organizations have been committed to the standardization of various interface protocols, it is still a challenging task to achieve the unified management of different protocols in a short time.

5.1.6. Model solution

The ability to achieve a fast solution to the network model can also significantly affect the transmission latency. Currently researchers are attempting to build different mathematical analysis models, such as network calculus [68] and queueing analysis [90], based on network statistics to characterize SDN operation scenarios and performance analysis. However, we believe that how to solve the model quickly and efficiently is more relevant and contributes more to low latency communication than the model building process. In fact, the computational complexity and time complexity of the model solution are not only proportional to the scale of the SDN network, but also depend more heavily on the choice of analysis model and parameter values. Even if the model solving process can use intelligent tools such as ML and bionic algorithm to find the optimal solution or suboptimal solution. However, due to the shortcomings of such intelligent algorithms that are easily trapped in the local minimum, the gradient disappears, and the difficulty of initial point selection, the model solution process still cannot avoid high time consumption, especially in the context of big data [18,49,255].

5.2. Future directions

5.2.1. Intelligent network

Although there are still many shortcomings in the implementation of intelligent networks, it is undeniable that the current immature and rigid SDN network management and policies still need intelligent solutions to respond to traffic and topology changes more autonomously. On the one hand, the knowledge base formed by AI through autonomous learning can provide SDN managers with more professional reference opinions. On the other hand, relying on existing communication protocols and modules, AI can be used to mine and analyze massive amounts of underlying data according to customer needs to achieve personalized services. For telecom operators and large companies, such as AT&T, Google, HP, Inter, Facebook, etc., the introduction of AI will free SDN network management from unnecessary manual intervention and reduce the resulting cost of resources [16,41]. In addition, AI has also entered the field of SDN security defense, where managers rely on intelligent means to filter risk factors to further enhance network defense capabilities and security levels, especially for the vulnerable control plane [9,10]. All things considered, we believe that the dilemma that AI is facing now will be slowly broken, and with neural network-led AI technology, it will bring a new change to SDN development.

5.2.2. Hardware switch

OpenFlow software switches are widely used in SDN due to their low cost, flexible configuration, and other advantages. However, software switches are less friendly and not commercially viable for deployment environments that seek high-speed forwarding performance. For example, traffic consisting of 64-byte Ethernet frames is transmitted at a link rate of 40Gbps, and the packet processing latency needs to be controlled to about 10 ns [256]. This poses a significant challenge to the software implementation. In contrast, switches or switching chips built on dedicated hardware such as Application Specific Integrated Circuit (ASIC), Graphic Processing Units (GPU), and FPGAs provide more powerful packet processing capabilities, which increases the switch's line rate as much as possible in a parallel way. At the same time, the programmability of hardware such as FPGAs eliminates the time required for forwarding device migration and new feature development, providing a low-cost and highly scalable solution. Studies have shown that hardware switching reduces latency by an average of 80% compared to software switching, achieving data forwarding speeds two orders of magnitude higher than CPUs and one order of magnitude higher than network processors, and that this trend will persist over time [257,258]. Currently, industry-leading programmable hardware vendors such as Xilinx, Altera, Lattice have introduced OpenFlow-enabled SOC platforms to enable high-speed packet processing, and large Internet companies such as Microsoft and IBM are using custom functional logic in their data centers to accelerate services [5,259,260]. In summary, the development and utilization of high-speed computing and processing power that comes with hardware can bring additional benefits to low-latency communications and deserves further attention in SDN.

5.2.3. Programmable data plane

The ever-expanding size of the network forces switching devices to perform many other tasks in addition to standard packet forwarding. However, SDN opens up only the control plane to programming by separating it from the data plane, for which the behavior remains fixed. The deployment of new protocols, whether for software or hardware switches, requires rewriting the protocol stack and packet processing logic without being totally protocol-oblivious [261]. If developers do not have the experience and skills to mature, the bar will be unimaginably high. Even if you turn to your equipment supplier, the update and commissioning process will be a long wait [262,263]. These limitations create significant barriers to rapid development and deployment of new features. As a result, there is an industry consensus that working to improve the programmability of the data plane is a promising and cost-effective endeavor. The programmable data plane decouples the protocol implementation or scheduler from the underlying switch architecture, i.e., makes the switch transparent to administrators and users and gives the switching device explicit field programmability capabilities to improve responsiveness to changes in packet processing behavior [264]. In addition, the development of the high-level programming language on which the programmable data plane depends is a critical step that cannot be ignored. Currently, languages such as P4 [79], POF [261], Domino [263], etc. have made a good example of programmable directions to support more general flexibility.

5.2.4. Testing tools and testbed

The performance testing tools and testbed used in SDN development have been analyzed in [19] and [265] in their outlook for the future, but we should point them out again here that good testbed can play a powerful supporting role for low-latency communication. Currently, the latest SDN theories are still being proposed. However, the contradiction between technology development and actual application hinders the smooth progress of commercial deployment. The root cause of the problem lies in the lack of testing tools and standardized testing solutions, which prevents technicians from accurately understanding and optimizing the performance of algorithms or devices before real deployments. Therefore, continuous attention to the following development directions is of key guiding significance for achieving efficient and accurate testing to access device performance and promote drive the pace of functional innovation: (a) Improving traditional testbed or proposing new ones to better fit the SDN environment or even more complex hybrid SDN. (b) Encouraging research institutions or vendors to break the privatization of existing testbed and solutions. (c) The absence of systematic and prescriptive test specifications has become a bottleneck constraining further development of SDN. (d) How to minimize the cost of measurement process and realize non-interference to the tested equipment

6. Conclusion

SDN's innovative architecture has brought about a near revolutionary evolution in networking technology. However, as industry needs and user experience become sensitive and rigorous to latency timeliness, the development trend of SDN has shifted the focus from the pursuit of stability and compatibility to the satisfaction of low-latency communications. In this paper, we specifically reviewed the contributions of related technologies to low-latency communication in SDN from four aspects: traffic management, congestion control, load balancing and flow table management. Some representative literature and ideas are presented for illustrative purposes. In addition, other emerging areas, including ICN, TSN, NFV, and edge computing, are also summarized for the revelations and payoffs of SDN low-latency communications to ensure the integrity of the survey. Finally, based on the state of the art, we emphasized the severe challenges still faced in low-latency communication and the promising future development directions to highlight their importance. It would be very inspiring as well as meaningful if our survey could bring guidance to interested researchers to advance their research process and reference to the development of related fields. In short, we believe that while there are still many obstacles in the way of achieving low-latency, this still be a promising and valuable research direction that will stay with SDN.

Declaration of competing interest

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

Acknowledgments

This work was supported by the The National Core Electronic Devices, China, High-end Generic Chips and Basic Software Major Projects of China (No. 2017ZX01030301) and the Industrial Internet Innovation Development Project of China — Universal Forwarding Chip of Time-Sensitive Network (No. TC190A446-2).

References

[1] J. Mei, K. Zheng, L. Zhao, Y. Teng, X. Wang, A latency and reliability guaranteed resource allocation scheme for LTE V2V communication systems, IEEE Trans. Wireless Commun. 17 (6) (2018) 3850–3860.
[2] D.J. Tyler, Restoring the human touch: Prosthetics imbued with haptics give their wearers fine motor control and a sense of connection, IEEE Spectr. 53 (5) (2016) 28–33.
[3] V.S. Devi, T. Ravi, S.B. Priya, Cluster based data aggregation scheme for latency and packet loss reduction in WSN, Comput. Commun. 149 (2020) 36–43.
[4] C.C. Moallemi, M. Sağlam, OR forum—The cost of latency in high-frequency trading, Oper. Res. 61 (5) (2013) 1070–1086.
[5] D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2014) 14–76.
[6] K. Benzekki, A. El Fergougui, A. Elbelrhiti Elalaoui, Software-defined networking (SDN): a survey, Secur. Commun. Netw. 9 (18) (2016) 5803–5833.
[7] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, M. Keshtgary, A survey on SDN, the future of networking, J. Adv. Comput. Sci. Technol. 3 (2) (2014) 232–248.
[8] W. Li, W. Meng, L. Kwok, A survey on openflow-based software defined networks: Security challenges and countermeasures, J. Netw. Comput. Appl. 68 (2016) 126–139.
[9] J. Benabbou, K. Elbaamrani, N. Idboufker, Security in OpenFlow-based SDN, opportunities and challenges, Photonic Netw. Commun. 37 (1) (2019) 1–23.
[10] I. Alsmadi, D. Xu, Security of software defined networks: A survey, Comput. Secur. 53 (2015) 79–108.
[11] Y. Zhang, L. Cui, W. Wang, Y. Zhang, A survey on software defined networking with multiple controllers, J. Netw. Comput. Appl. 103 (2018) 101–118.
[12] F. Bannour, S. Souihi, A. Mellouk, Distributed SDN control: Survey, taxonomy, and challenges, IEEE Commun. Surv. Tutor. 20 (1) (2018) 333–354.
[13] K. Foerster, S. Schmid, S. Vissicchio, Survey of consistent software-defined network updates, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1435–1461.
[14] I.F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, A roadmap for traffic engineering in SDN-OpenFlow networks, Comput. Netw. 71 (2014) 1–30.
[15] Z.N. Abdullah, I. Ahmad, I. Hussain, Segment routing in software defined networks: A survey, IEEE Commun. Surv. Tutor. 21 (1) (2018) 464–486.
[16] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, L. Yang, K. Cheng, X. Xiao, Fault management in software-defined networking: A survey, IEEE Commun. Surv. Tutor. 21 (1) (2018) 349–392.
[17] P. Tsai, C. Tsai, C. Hsu, C. Yang, Network monitoring in software-defined networking: A review, IEEE Syst. J. 12 (4) (2018) 3958–3969.
[18] J. Xie, F.R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, Y. Liu, A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges, IEEE Commun. Surv. Tutor. 21 (1) (2018) 393–430.
[19] R. Amin, M. Reisslein, N. Shah, Hybrid SDN networks: A survey of existing approaches, IEEE Commun. Surv. Tutor. 20 (4) (2018) 3259–3306.
[20] Y. Sinha, K. Haribabu, et al., A survey: hybrid SDN, J. Netw. Comput. Appl. 100 (2017) 35–55.
[21] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, S. Hu, A survey of deployment solutions and optimization strategies for hybrid SDN networks, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1483–1507.
[22] NSF NeTS FIND Initiative (FIND). http://www.nets-find.net.
[23] L. Yang, R. Dantu, T. Anderson, R. Gopal, Forwarding and Control Element Separation (ForCES) Framework, Technical Report, RFC 3746, April, 2004.
[24] H. Harai, Designing new-generation network-overview of AKARI architecture design, in: 2009 Asia Communications and Photonics Conference and Exhibition (ACP), IEEE, 2009, pp. 1–2.
[25] M. Casado, T. Garfinkel, A. Akella, M.J. Freedman, D. Boneh, N. McKeown, S. Shenker, SANE: A protection architecture for enterprise networks, in: USENIX Security Symposium, vol. 49, 2006, p. 50.
[26] M. Brunner, H. Abramowicz, N. Niebert, L.M. Correia, 4ward: a european perspective towards the future internet, IEICE Trans. Commun. 93 (3) (2010) 442–445.
[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.
[28] N. McKeown, Software-defined networking, INFOCOM Keynote Talk 17 (2) (2009) 30–32.
[29] R. Al-Haddad, E.S. Velazquez, A survey of quality of service (QoS) protocols and software-defined networks (SDN), in: Science and Information Conference, Springer, 2018, pp. 527–545.
[30] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, NOX: towards an operating system for networks, ACM SIGCOMM Comput. Commun. Rev. 38 (3) (2008) 105–110.
[31] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, R. Smeliansky, Advanced study of SDN/OpenFlow controllers, in: Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, 2013, pp. 1–6.
[32] A. Nygren, B. Pfaff, B. Lantz, B. Heller, C. Barker, C. Beckmann, D. Cohn, D. Malek, D. Talayco, D. Erickson, et al., Openflow Switch Specification Version 1.5. 1, Open Networking Foundation, Tech. Rep., 2015.
[33] M. Hayes, B. Ng, A. Pekar, W.K. Seah, Scalable architecture for SDN traffic classification, IEEE Syst. J. (99) (2017) 1–12.
[34] B. Wang, J. Su, A survey of elephant flow detection in SDN, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2018, pp. 1–6.
[35] S. Jeong, D. Lee, J. Choi, J. Li, J.W. Hong, Application-aware traffic management for OpenFlow networks, in: 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2016, pp. 1–5.
[36] P. Khandait, N. Hubballi, B. Mazumdar, Efficient keyword matching for deep packet inspection based network traffic classification, in: 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), IEEE, 2020, pp. 567–570.
[37] Enterprise Network Security-Encrypted Traffic Analytics. www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-security/.
[38] Encrypted Traffic Analytics. http://www.allot.com.cn/solution.php.
[39] G. Li, M. Dong, K. Ota, J. Wu, J. Li, T. Ye, Deep packet inspection based application-aware traffic control for software defined networks, in: 2016 IEEE Global Communications Conference (GLOBECOM), IEEE, 2016, pp. 1–6.
[40] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, Y. Liu, NetworkAI: AN intelligent network architecture for self-learning control strategies in software defined networks, IEEE Internet Things J. 5 (6) (2018) 4319–4327.
[41] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M.J. Hibbett, et al., Knowledge-defined networking, ACM SIGCOMM Comput. Commun. Rev. 47 (3) (2017) 2–10.
[42] P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: Deep learning based encrypted network traffic classification in sdn home gateway, IEEE Access 6 (2018) 55380–55391.
[43] B. Indira, K. Valarmathi, D. Devaraj, An approach to enhance packet classification performance of software-defined network using deep learning, Soft Comput. 23 (18) (2019) 8609–8619.
[44] F.A.M. Zaki, T.S. Chin, FWFS: Selecting robust features towards reliable and stable traffic classifier in SDN, IEEE Access 7 (2019) 166011–166020.
[45] J. Mbous, T. Jiang, M. Tang, S. Fu, D. Liu, Kalman filtering-based traffic prediction for software defined intra-data center networks, Trans. Internet Inf. Syst 13 (2019) 2964.
[46] Y. Cao, R. Wang, M. Chen, A. Barnawi, AI agent in software-defined network: Agent-based network service prediction and wireless resource scheduling optimization, IEEE Internet Things J. (2019).
[47] G. Rzym, P. Boryło, P. Chołda, A time-efficient shrinkage algorithm for the fourier-based prediction enabling proactive optimisation in software-defined networks, Int. J. Commun. Syst. (2019) e4448.

[48] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 267–288.
[49] M. Latah, L. Toker, Artificial intelligence enabled software-defined networking: a comprehensive overview, IET Netw. 8 (2) (2018) 79–99.
[50] T.N. Nguyen, The challenges in SDN/ML based network security: A survey, 2018, arXiv preprint arXiv:1804.03539.
[51] A.R. Mohammed, S.A. Mohammed, S. Shirmohammadi, Machine learning and deep learning based traffic classification and prediction in software defined networking, in: 2019 IEEE International Symposium on Measurements & Networking (M&N), IEEE, 2019, pp. 1–6.
[52] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al., B4: Experience with a globally-deployed software defined WAN, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 3–14.
[53] S.H. Low, F. Paganini, J.C. Doyle, Internet congestion control, IEEE Control Syst. Mag. 22 (1) (2002) 28–43.
[54] A. Lara, A. Kolasani, B. Ramamurthy, Network innovation using openflow: A survey, IEEE Commun. Surv. Tutor. 16 (1) (2013) 493–512.
[55] G. Hasegawa, M. Murata, Survey on fairness issues in TCP congestion control mechanisms, IEICE Trans. Commun. 84 (6) (2001) 1461–1472.
[56] M. Ghobadi, S.H. Yeganeh, Y. Ganjali, Rethinking end-to-end congestion control in software-defined networks, in: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, 2012, pp. 61–66.
[57] J. Hwang, J. Yoo, S. Lee, H. Jin, Scalable congestion control protocol based on SDN in data center networks, in: 2015 IEEE Global Communications Conference (GLOBECOM), IEEE, 2015, pp. 1–6.
[58] S. Jouet, C. Perkins, D. Pezaros, OTCP: SDN-managed congestion control for data center networks, in: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2016, pp. 171–179.
[59] A.Z. Khan, I.A. Qazi, RecFlow: SDN-based receiver-driven flow scheduling in datacenters, Cluster Comput. 23 (1) (2020) 289–306.
[60] Y. Chen, R. Griffith, J. Liu, R.H. Katz, A.D. Joseph, Understanding TCP incast throughput collapse in datacenter networks, in: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, 2009, pp. 73–82.
[61] Y. Lu, S. Zhu, SDN-based TCP congestion control in data center networks, in: 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), IEEE, 2015, pp. 1–7.
[62] Y. Lu, Z. Ling, S. Zhu, L. Tang, SDTCP: Towards datacenter TCP congestion control with SDN for IoT applications, Sensors 17 (1) (2017) 109.
[63] Y. Lu, X. Fan, L. Qian, EQF: An explicit queue-length feedback for TCP congestion control in datacenter networks, in: 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), IEEE, 2017, pp. 69–74.
[64] A.M. Abdelmoniem, B. Bensaou, A.J. Abu, Mitigating incast-tcp congestion in data centers with SDN, Ann. Telecommun. 73 (3–4) (2018) 263–277.
[65] J. Bao, J. Wang, Q. Qi, J. Liao, ECTCP: An explicit centralized congestion avoidance for TCP in SDN-based data center, in: 2018 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2018, pp. 00347–00353.
[66] A.M. Abdelmoniem, B. Bensaou, Enforcing transport-agnostic congestion control in sdn-based data centers, in: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), IEEE, 2017, pp. 128–136.
[67] D. Singh, B. Ng, Y. Lai, Y. Lin, W.K. Seah, Modelling software-defined networking: Switch design with finite buffer and priority queueing, in: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), IEEE, 2017, pp. 567–570.
[68] A.K. Koohanestani, A.G. Osgouei, H. Saidi, A. Fanian, An analytical model for delay bound of openflow based SDN using network calculus, J. Netw. Comput. Appl. 96 (2017) 31–38.
[69] H. Chibana, M. Yoshino, M. Tadokoro, D. Murayama, K. Suzuki, R. Kubo, Disturbance-observer-based active queue management with time delay using software-defined networking controller, in: IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2015, pp. 001049–001054.
[70] P. Sköldström, K. Yedavalli, Network virtualization and resource allocation in openflow-based wide area networks, in: 2012 IEEE International Conference on Communications (ICC), IEEE, 2012, pp. 6622–6626.
[71] R. Adams, Active queue management: A survey, IEEE Commun. Surv. Tutor. 15 (3) (2012) 1425–1476.
[72] M. Karakus, A. Durresi, Quality of service (QoS) in software defined networking (SDN): A survey, J. Netw. Comput. Appl. 80 (2017) 200–218.
[73] M. Hock, R. Bless, M. Zitterbart, Toward coexistence of different congestion control mechanisms, in: 2016 IEEE 41st Conference on Local Computer Networks (LCN), IEEE, 2016, pp. 567–570.
[74] S. Gu, J. Kim, Y. Kim, C. Moon, I. Yeom, Controlled queue management in software-defined networks, in: 2015 5th International Conference on IT Convergence and Security (ICITCS), IEEE, 2015, pp. 1–3.
[75] S. Jeong, Y. Kim, D. An, I. Yeom, CoopRED: Cooperative RED for software defined networks, in: 2015 International Conference on Information Networking (ICOIN), IEEE, 2015, pp. 307–308.
[76] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413.
[77] L. Lu, Y. Xiao, H. Du, Openflow control for cooperating AQM scheme, in: IEEE 10th International Conference on Signal Processing Proceedings, IEEE, 2010, pp. 2560–2563.
[78] R. Kundel, J. Blendin, T. Viernickel, B. Koldehofe, R. Steinmetz, P4-CoDel: Active queue management in programmable data planes, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2018, pp. 1–4.
[79] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al., P4: Programming protocol-independent packet processors, ACM SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 87–95.
[80] D.M. Raghuvanshi, B. Annappa, M.P. Tahiliani, On the effectiveness of codel for active queue management, in: 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), IEEE, 2013, pp. 107–114.
[81] D.N. Anantha, B. Ramamurthy, B. Bockelman, D. Swanson, Differentiated network services for data-intensive science using application-aware SDN, in: 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), IEEE, 2017, pp. 1–6.
[82] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, IEEE Trans. Ind. Inf. 14 (11) (2018) 4724–4734.
[83] Z. Lu, W. Chen, J. Wei, H. Yu, Current situation and prospect of V2x with ultra-reliable and low-latency, J. Signal Process. 35 (11) (2019) 1773–1783.
[84] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, R.B. Bobba, End-to-end network delay guarantees for real-time systems using sdn, in: 2017 IEEE Real-Time Systems Symposium (RTSS), IEEE, 2017, pp. 231–242.
[85] N.K. Sharma, M. Liu, K. Atreya, A. Krishnamurthy, Approximating fair queueing on reconfigurable switches, in: 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18), 2018, pp. 1–16.
[86] W. Wang, Q. Qi, X. Gong, Y. Hu, X. Que, Autonomic QoS management mechanism in software defined network, China Commun. 11 (7) (2014) 13–23.
[87] Y. Wang, Y. Zhang, J. Chen, Pursuing differentiated services in a sdn-based iot-oriented pub/sub system, in: 2017 IEEE International Conference on Web Services (ICWS), IEEE, 2017, pp. 906–909.
[88] G.S. Aujla, R. Chaudhary, N. Kumar, R. Kumar, J.J. Rodrigues, An ensembled scheme for QoS-aware traffic flow management in software defined networks, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–7.
[89] S. Sharma, D. Staessens, D. Colle, M. Pickavet, P. Demeester, In-band control, queuing, and failure recovery functionalities for openflow, IEEE Netw. 30 (1) (2016) 106–112.
[90] Y. Goto, B. Ng, W.K. Seah, Y. Takahashi, Queueing analysis of software defined network with realistic openflow–based switch model, Comput. Netw. 164 (2019) 106892.
[91] W.M. Zuberek, D. Strzeciwilk, Modeling traffic shaping and traffic policing in packet-switched networks, J. Comput. Sci. Appl. 6 (2) (2018) 75–81.
[92] M. Swarna, S. Ravi, M. Anand, Leaky bucket algorithm for congestion control, Int. J. Appl. Eng. Res. 11 (5) (2016) 3155–3159.
[93] S. Ren, Q. Feng, Y. Wang, W. Dou, A service curve of hierarchical token bucket queue discipline on soft-ware defined networks based on deterministic network calculus: An analysis and simulation, J. Adv. Comput. Netw. 5 (1) (2017).
[94] C.B. Hauser, S.R. P., Dynamic network scheduler for cloud data centres with sdn, in: Proceedings of The10th International Conference on Utility and Cloud Computing, 2017, pp. 29–38.
[95] R.M. Abuteir, A. Fladenmuller, O. Fourmaux, SDN based architecture to improve video streaming in home networks, in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2016, pp. 220–226.

[96] M.S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, Y. Song, Flowqos: Per-Flow Quality of Service for Broadband Access Networks, Technical Report, Georgia Institute of Technology, 2015.
[97] S. Bhaumik, S. Chakraborty, Hierarchical two dimensional queuing: A scalable approach for traffic shaping using software defined networking, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 150–158.
[98] M. Nasimi, M.A. Habibi, B. Han, H.D. Schotten, Edge-assisted congestion control mechanism for 5G network using software-defined networking, in: 2018 15th International Symposium on Wireless Communication Systems (ISWCS), IEEE, 2018, pp. 1–5.
[99] L. Xue, X. Luo, E.W. Chan, X. Zhan, Towards detecting target link flooding attack, in: 28th Large Installation System Administration Conference (LISA14), 2014, pp. 90–105.
[100] D. Erickson, The beacon openflow controller, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 13–18.
[101] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: 2nd {USENIX} Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12), 2012.
[102] Z. Cai, A.L. Cox, T.S. Ng, Maestro: A System for Scalable Openflow Control, Technical Report, 2010.
[103] F.O. Controller, Project floodlight, 2019, URL: http://www.projectfloodlight.org\/floodlight/. (Accessed 29 May 2019).
[104] O. Blial, M. Ben Mamoun, R. Benaini, An overview on SDN architectures with multiple controllers, J. Comput. Netw. Commun. 2016 (2016).
[105] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al., Onix: A distributed control platform for large-scale production networks., in: OSDI, vol. 10, 2010, pp. 1–6.
[106] A. Tootoonchian, Y. Ganjali, Hyperflow: A distributed control plane for openflow, in: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, vol. 3, 2010.
[107] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, et al., ONOS: towards an open, distributed SDN OS, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, 2014, pp. 1–6.
[108] S. Hassas Yeganeh, Y. Ganjali, Kandoo: a framework for efficient and scalable offloading of control applications, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012, pp. 19–24.
[109] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, B. Hao, Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks, in: 2014 IEEE 22nd International Conference on Network Protocols, IEEE, 2014, pp. 569–576.
[110] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, ACM SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 473–478.
[111] Q. Qin, K. Poularakis, G. Iosifidis, L. Tassiulas, SDN controller placement at the edge: Optimizing delay and overheads, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 684–692.
[112] M.T. ul Huque, W. Si, G. Jourjon, V. Gramoli, Large-scale dynamic controller placement, IEEE Trans. Netw. Serv. Manag. 14 (1) (2017) 63–76.
[113] M. Tanha, D. Sajjadi, R. Ruby, J. Pan, Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs, IEEE Trans. Netw. Serv. Manag. 15 (3) (2018) 991–1005.
[114] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, Pareto-optimal resilient controller placement in SDN-based core networks, in: Proceedings of the 2013 25th International Teletraffic Congress (ITC), IEEE, 2013, pp. 1–9.
[115] M.J. Abdel-Rahman, E.A. Mazied, K. Teague, A.B. MacKenzie, S.F. Midkiff, Robust controller placement and assignment in software-defined cellular networks, in: 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2017, pp. 1–9.
[116] A. Ksentini, M. Bagaa, T. Taleb, On using SDN in 5G: The controller placement problem, in: 2016 IEEE Global Communications Conference (GLOBECOM), IEEE, 2016, pp. 1–6.
[117] Y.E. Oktian, S. Lee, H. Lee, J. Lam, Distributed SDN controller system: A survey on design choice, Comput. Netw. 121 (2017) 100–111.
[118] M. Cello, Y. Xu, A. Walid, G. Wilfong, H.J. Chao, M. Marchese, BalCon: A distributed elastic SDN control via efficient switch migration, in: 2017 IEEE International Conference on Cloud Engineering (IC2E), IEEE, 2017, pp. 40–50.
[119] Y. Xu, M. Cello, I. Wang, A. Walid, G. Wilfong, C.H.-P. Wen, M. Marchese, H.J. Chao, Dynamic switch migration in distributed software-defined networks to achieve controller load balance, IEEE J. Sel. Areas Commun. 37 (3) (2019) 515–529.
[120] A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, R.R. Kompella, ElastiCon; an elastic distributed SDN controller, in: 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), IEEE, 2014, pp. 17–27.
[121] J.G. Mohanasundaram, T. Truong-Huu, M. Gurusamy, Game theoretic switch-controller mapping with traffic variations in software defined networks, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.
[122] F. Al-Tam, N. Correia, Fractional switch migration in multi-controller software-defined networking, Comput. Netw. 157 (2019) 1–10.
[123] Z. Min, Q. Hua, Z. Jihong, Dynamic switch migration algorithm with Q-learning towards scalable sdn control plane, in: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), IEEE, 2017, pp. 1–4.
[124] C. Wang, B. Hu, S. Chen, D. Li, B. Liu, A switch migration-based decision-making scheme for balancing load in SDN, IEEE Access 5 (2017) 4537–4544.
[125] A. Filali, S. Cherkaoui, A. Kobbane, Prediction-based switch migration scheduling for SDN load balancing, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.
[126] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, A load-balancing mechanism for distributed SDN control plane using response time, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1197–1206.
[127] K. Dinh, S. Kukliński, W. Kujawa, M. Ulaski, Msdn-te: Multipath based traffic engineering for sdn, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2016, pp. 630–639.
[128] J. Yan, H. Zhang, Q. Shuai, B. Liu, X. Guo, HiQoS: An SDN-based multipath QoS solution, China Commun. 12 (5) (2015) 123–133.
[129] S. Song, J. Lee, K. Son, H. Jung, J. Lee, A congestion avoidance algorithm in SDN environment, in: 2016 International Conference on Information Networking (ICOIN), IEEE, 2016, pp. 420–423.
[130] X. Cui, X. Huang, Y. Ma, Q. Meng, A load balancing routing mechanism based on SDWSN in smart city, Electronics 8 (3) (2019) 273.
[131] B. Mao, F. Tang, Z.M. Fadlullah, N. Kato, An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems, IEEE Trans. Emerg. Top. Comput. (2019).
[132] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, A. Cabellos-Aparicio, RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN, 2019, arXiv preprint arXiv:1910.01508.
[133] C. Filsfils, N.K. Nainar, C. Pignataro, J.C. Cardona, P. Francois, The segment routing architecture, in: 2015 IEEE Global Communications Conference (GLOBECOM), IEEE, 2015, pp. 1–6.
[134] S. Gay, R. Hartert, S. Vissicchio, Expect the unexpected: Sub-second optimization for segment routing, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.
[135] J. Sheu, Y. Chen, A scalable and bandwidth-efficient multicast algorithm based on segment routing in software-defined networking, in: 2017 IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.
[136] A.A. Barakabitze, I. Mkwawa, L. Sun, E. Ifeachor, Qualitysdn: improving video quality using MPTCP and segment routing in SDN/NFV, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 182–186.
[137] A.A. Barakabitze, L. Sun, I. Mkwawa, E. Ifeachor, Multipath protections and dynamic link recoveryin softwarized 5G networks using segment routing, in: 2019 IEEE Globecom Workshops (GC Wkshps), IEEE, 2019, pp. 1–6.
[138] J. Hao, Y. Shi, H. Sun, M. Sheng, J. Li, Rerouting based congestion control in data center networks, in: 2019 IEEE International Conference on Communications Workshops (ICC Workshops), IEEE, 2019, pp. 1–6.
[139] L. Boero, M. Cello, C. Garibotto, M. Marchese, M. Mongelli, BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch, Comput. Netw. 106 (2016) 161–170.
[140] D. Shen, W. Yan, Y. Peng, Y. Fu, Q. Deng, Congestion control and traffic scheduling for collaborative crowdsourcing in SDN enabled mobile wireless networks, Wirel. Commun. Mob. Comput. 2018 (2018).
[141] X. Li, H. Wu, D. Gruenbacher, C. Scoglio, T. Anjali, Efficient routing for middlebox policy enforcement in software-defined networking, Comput. Netw. 110 (2016) 243–252.

[142] A. Basit, S. Qaisar, S.H. Rasool, M. Ali, Sdn orchestration for next generation inter-networking: A multipath forwarding approach, IEEE Access 5 (2017) 13077–13089.
[143] J.R. Iyengar, P.D. Amer, R. Stewart, Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths, IEEE/ACM Trans. Netw. 14 (5) (2006) 951–964.
[144] T. Benson, A. Akella, D.A. Maltz, Network traffic characteristics of data centers in the wild, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010, pp. 267–280.
[145] M. Chiesa, M. Kindler, Traffic engineering with equal-cost-multipath: An algorithmic perspective, IEEE/ACM Trans. Netw. 25 (2) (2016) 779–792.
[146] F. Rhamdani, N.A. Suwastika, M.A. Nugroho, Equal-cost multipath routing in data center network based on software defined network, in: 2018 6th International Conference on Information and Communication Technology (ICoICT), IEEE, 2018, pp. 222–226.
[147] S.K. Singh, T. Das, A. Jukan, A survey on internet multipath routing and provisioning, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2157–2175.
[148] C. Nakasan, K. Ichikawa, H. Iida, P. Uthayopas, A simple multipath openflow controller using topology-based algorithm for multipath TCP, Concurr. Comput.: Pract. Exper. 29 (13) (2017) e4134.
[149] Y. Liu, X. Qin, T. Zhu, X. Chen, G. Wei, Improve MPTCP with SDN: From the perspective of resource pooling, J. Netw. Comput. Appl. 141 (2019) 73–85.
[150] K. Herguner, R.S. Kalan, C. Cetinkaya, M. Sayit, Towards QoS-aware routing for DASH utilizing MPTCP over SDN, in: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2017, pp. 1–6.
[151] P.K. Singh, S. Sharma, S.K. Nandi, S. Nandi, Multipath TCP for V2i communication in SDN controlled small cell deployment of smart city, Veh. Commun. 15 (2019) 1–15.
[152] G.M. Schneider, T. Nemeth, A simulation study of the OSPF-OMP routing algorithm, Comput. Netw. 39 (4) (2002) 457–468.
[153] X. Sun, Z. Jia, M. Zhao, Z. Zhang, Multipath load balancing in SDN/OSPF hybrid network, in: IFIP International Conference on Network and Parallel Computing, Springer, 2016, pp. 93–100.
[154] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, R. Steinmetz, Multipath QUIC: A deployable multipath transport protocol, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–7.
[155] S. Tomovic, I. Radusinovic, RO-RO: Routing optimality-reconfiguration overhead balance in software-defined ISP networks, IEEE J. Sel. Areas Commun. 37 (5) (2019) 997–1011.
[156] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, A. Vahdat, WCMP: Weighted cost multipathing for improved fairness in data centers, in: Proceedings of the Ninth European Conference on Computer Systems, 2014, pp. 1–14.
[157] Y. Wang, Y. Lin, G. Chang, SDN-based dynamic multipath forwarding for inter–data center networking, Int. J. Commun. Syst. 32 (1) (2019) e3843.
[158] S. Jouet, R. Cziva, D.P. Pezaros, Arbitrary packet matching in OpenFlow, in: 2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR), IEEE, 2015, pp. 1–6.
[159] C. Yu, C. Lumezanu, H.V. Madhyastha, G. Jiang, Characterizing rule compression mechanisms in software-defined networks, in: International Conference on Passive and Active Network Measurement, Springer, 2016, pp. 302–315.
[160] K.G. Perez, X. Yang, S. Scott-Hayward, S. Sezer, A configurable packet classification architecture for software-defined networking, in: 2014 27th IEEE International System-on-Chip Conference (SOCC), IEEE, 2014, pp. 353–358.
[161] H. Huang, S. Guo, J. Wu, J. Li, Green datapath for TCAM-based software-defined networks, IEEE Commun. Mag. 54 (11) (2016) 194–201.
[162] D.E. Taylor, Survey and taxonomy of packet classification techniques, ACM Comput. Surv. 37 (3) (2005) 238–275.
[163] P.T. Congdon, P. Mohapatra, M. Farrens, V. Akella, Simultaneously reducing latency and power consumption in openflow switches, IEEE/ACM Trans. Netw. 22 (3) (2013) 1007–1020.
[164] T. Chen, D. Lee, T. Liu, A. Wu, Dynamic reconfigurable ternary content addressable memory for OpenFlow-compliant low-power packet processing, IEEE Trans. Circuits Syst. I. Regul. Pap. 63 (10) (2016) 1661–1672.
[165] Y.R. Qu, V.K. Prasanna, High-performance and dynamically updatable packet classification engine on FPGA, IEEE Trans. Parallel Distrib. Syst. 27 (1) (2015) 197–209.
[166] A.V. Alyushin, S.A. Alyushin, V.G. Arkhangelsky, Bit-vector pattern matching systems on the base of high bandwidth FPGA memory, in: 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE, 2018, pp. 1342–1347.
[167] J. Tseng, R. Wang, J. Tsai, Y. Wang, T.C. Tai, Accelerating open vSwitch with integrated GPU, in: Proceedings of the Workshop on Kernel-Bypass Networks, 2017, pp. 7–12.
[168] P. He, G. Xie, K. Salamatian, L. Mathy, Meta-algorithms for software-based packet classification, in: 2014 IEEE 22nd International Conference on Network Protocols, IEEE, 2014, pp. 308–319.
[169] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, et al., The design and implementation of open vswitch, in: 12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15), 2015, pp. 117–130.
[170] J. Daly, V. Bruschi, L. Linguaglossa, S. Pontarelli, D. Rossi, J. Tollet, E. Torng, A. Yourtchenko, Tuplemerge: Fast software packet processing for online packet classification, IEEE/ACM Trans. Netw. 27 (4) (2019) 1417–1431.
[171] H. Alimohammadi, M. Ahmadi, Clustering-based many-field packet classification in software-defined networking, J. Netw. Comput. Appl. 147 (2019) 102428.
[172] C. Li, Y. Dong, G. Wu, Openflow table lookup scheme integrating multiple-cell hash table with TCAM, J. Commun. 37 (10) (2016) 128–140.
[173] J. Daly, E. Torng, Bytecuts: Fast packet classification by interior bit extraction, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 2654–2662.
[174] Z. Liu, S. Sun, H. Zhu, J. Gao, J. Li, BitCuts: A fast packet classification algorithm using bit-level cutting, Comput. Commun. 109 (2017) 38–52.
[175] J. Wee, W. Pak, Fast packet classification based on hybrid cutting, IEEE Commun. Lett. 21 (5) (2017) 1011–1014.
[176] J. Fong, X. Wang, Y. Qi, J. Li, W. Jiang, ParaSplit: A scalable architecture on FPGA for terabit packet classification, in: 2012 IEEE 20th Annual Symposium on High-Performance Interconnects, IEEE, 2012, pp. 1–8.
[177] W. Li, X. Li, H. Li, G. Xie, Cutsplit: A decision-tree combining cutting and splitting for scalable packet classification, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 2645–2653.
[178] Y. Qu, S. Zhou, V.K. Prasanna, A decomposition-based approach for scalable many-field packet classification on multi-core processors, Int. J. Parallel Program. 43 (6) (2015) 965–987.
[179] S. Hung, N. Iliev, B. Vamanan, A.R. Trivedi, Self-organizing maps-based flexible and high-speed packet classification in software defined networking, in: 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), IEEE, 2019, pp. 545–546.
[180] X. Nguyen, D. Saucez, C. Barakat, T. Turletti, Rules placement problem in openflow networks: A survey, IEEE Commun. Surv. Tutor. 18 (2) (2015) 1273–1286.
[181] X. Nguyen, D. Saucez, C. Barakat, T. Turletti, OFFICER: A general optimization framework for OpenFlow rule allocation and endpoint policy enforcement, in: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, 2015, pp. 478–486.
[182] S. Bera, S. Misra, A. Jamalipour, Flowstat: Adaptive flow-rule placement for per-flow statistics in SDN, IEEE J. Sel. Areas Commun. 37 (3) (2019) 530–539.
[183] S. Bera, S. Misra, M.S. Obaidat, Mobi-flow: Mobility-aware adaptive flow-rule placement in software-defined access network, IEEE Trans. Mob. Comput. 18 (8) (2018) 1831–1842.
[184] B. Zhao, J. Zhao, X. Wang, T. Wolf, Ruletailor: Optimizing flow table updates in OpenFlow switches with rule transformations, IEEE Trans. Netw. Serv. Manag. 16 (4) (2019) 1581–1594.
[185] R. Khalili, Z. Despotovic, A. Hecker, Flow setup latency in SDN networks, IEEE J. Sel. Areas Commun. 36 (12) (2018) 2631–2639.
[186] H. Ma, Y. Yang, Z. Mi, A distributed storage framework of flowtable in software defined network, Comput. Electr. Eng. 43 (2015) 155–168.
[187] W. Ren, Y. Sun, T. Wu, M.S. Obaidat, A hash-based distributed storage strategy of flowtables in sdn-iot networks, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[188] Open Networking Foundation, Openflow switch specification, version 1.3.0, 2012.
[189] Openflow switch specification-version 1.4. 0, 2013.
[190] Y. Liu, B. Tang, D. Yuan, J. Ran, H. Hu, A dynamic adaptive timeout approach for SDN switch, in: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), IEEE, 2016, pp. 2577–2582.

[191] L. Xie, Z. Zhao, Y. Zhou, G. Wang, Q. Ying, H. Zhang, An adaptive scheme for data forwarding in software defined network, in: 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), IEEE, 2014, pp. 1–5.
[192] T. Li, H. Zhou, H. Luo, I. You, Q. Xu, SAT-FLOW: multi-strategy flow table management for software defined satellite networks, IEEE Access 5 (2017) 14952–14965.
[193] A. Panda, S.S. Samal, A.K. Turuk, A. Panda, V.C. Venkatesh, Dynamic hard timeout based flow table management in openflow enabled SDN, in: 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), IEEE, 2019, pp. 1–6.
[194] Q. Li, N. Huang, D. Wang, X. Li, Y. Jiang, Z. Song, HQTimer: A hybrid Q-learning based timeout mechanism in software-defined networks, IEEE Trans. Netw. Serv. Manag. 16 (1) (2019) 153–166.
[195] H. Zhu, X. Fan, Y. Jin, Intelligent timeout master: Dynamic timeout for sdn-based data centers, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2015, pp. 734–737.
[196] X. Li, Y. Huang, A flow table with two-stage timeout mechanism for SDN switches, in: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2019, pp. 1804–1809.
[197] R. Ying, W. Jia, Y. Zheng, Y. Wu, Fast invalid TCP flow removal scheme for improving SDN scalability, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–5.
[198] H. Yang, G.F. Riley, Machine learning based proactive flow entry deletion for openflow, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.
[199] T. Mu, A. Al-Fuqaha, K. Shuaib, F.M. Sallabi, J. Qadir, SDN flow entry management using reinforcement learning, ACM Trans. Auton. Adapt. Syst. (TAAS) 13 (2) (2018) 1–23.
[200] G. Huang, H.Y. Youn, Proactive eviction of flow entry for SDN based on hidden Markov model, Front. Comput. Sci. 14 (4) (2020) 1–10.
[201] F.M. Mazzola, D.S. Marcon, M.C. Neves, M.P. Barcellos, It's About Time: Analyzing Flow Table Update Latency in SDN Switch Architectures.
[202] R.P. Draves, C. King, S. Venkatachary, B.D. Zill, Constructing optimal IP routing tables, in: IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. the Future Is now (Cat. No. 99CH36320), vol. 1, IEEE, 1999, pp. 88–97.
[203] Y. Liu, B. Zhang, L. Wang, FIFA: Fast incremental FIB aggregation, in: 2013 Proceedings IEEE INFOCOM, IEEE, 2013, pp. 1–9.
[204] R. McGeer, P. Yalagandula, Minimizing rulesets for TCAM implementation, in: IEEE INFOCOM 2009, IEEE, 2009, pp. 1314–1322.
[205] S. Luo, H. Yu, et al., Fast incremental flow table aggregation in SDN, in: 2014 23rd International Conference on Computer Communication and Networks (ICCCN), IEEE, 2014, pp. 1–8.
[206] F. Amezquita-Suarez, F. Estrada-Solano, N.L. da Fonseca, O.M.C. Rendon, An efficient mice flow routing algorithm for data centers based on software-defined networking, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.
[207] S. Shirali-Shahreza, Y. Ganjali, Rewiflow: Restricted wildcard openflow rules, ACM SIGCOMM Comput. Commun. Rev. 45 (5) (2015) 29–35.
[208] B. Yan, Y. Xu, H. Xing, K. Xi, H.J. Chao, Cab: A reactive wildcard rule caching system for software-defined networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, 2014, pp. 163–168.
[209] C.R. Meiners, A.X. Liu, E. Torng, Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs, IEEE/ACM Trans. Netw. 20 (2) (2011) 488–500.
[210] Q. Dai, H. Li, An advanced TCAM-sram architecture for ranges towards minimizing packet classifiers, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2018, pp. 158–163.
[211] Y. Chang, C. Lee, C. Su, Multi-field range encoding for packet classification in TCAM, in: 2011 Proceedings IEEE INFOCOM, IEEE, 2011, pp. 196–200.
[212] O. Rottenstreich, I. Keslassy, A. Hassidim, H. Kaplan, E. Porat, On finding an optimal TCAM encoding scheme for packet classification, in: 2013 Proceedings IEEE INFOCOM, IEEE, 2013, pp. 2049–2057.
[213] P. Sun, J. Lan, P. Wang, T. Ma, RFC: range feature code for TCAM-based packet classification, Comput. Netw. 118 (2017) 54–61.
[214] E. Spitznagel, D. Taylor, J. Turner, Packet classification using extended TCAMs, in: 11th IEEE International Conference on Network Protocols, 2003. Proceedings, IEEE, 2003, pp. 120–131.
[215] T. Kosugiyama, K. Tanabe, H. Nakayama, T. Hayashi, K. Yamaoka, A flow aggregation method based on end-to-end delay in SDN, in: 2017 IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.
[216] J. Huang, Y. He, Q. Duan, Q. Yang, W. Wang, Admission control with flow aggregation for QoS provisioning in software-defined network, in: 2014 IEEE Global Communications Conference, IEEE, 2014, pp. 1182–1186.
[217] Q.T. Minh, A. Van Le, T.K. Dang, T. Nam, T. Kitahara, An effective flow aggregation for SDN-based background and foreground traffic control, in: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2018, pp. 1–4.
[218] S. Kao, D. Lee, T. Chen, A. Wu, Dynamically updatable ternary segmented aging bloom filter for openflow-compliant low-power packet processing, IEEE/ACM Trans. Netw. 26 (2) (2018) 1004–1017.
[219] H. Chen, T. Benson, The case for making tight control plane latency guarantees in SDN switches, in: Proceedings of the Symposium on SDN Research, 2017, pp. 150–156.
[220] N. Katta, O. Alipourfard, J. Rexford, D. Walker, Cacheflow: Dependency-aware rule-caching for software-defined networks, in: Proceedings of the Symposium on SDN Research, 2016, pp. 1–12.
[221] K. Qiu, J. Yuan, J. Zhao, X. Wang, S. Secci, X. Fu, FastRule: EFficient flow entry updates for TCAM-based openflow switches, IEEE J. Sel. Areas Commun. 37 (3) (2019) 484–498.
[222] X. Wen, B. Yang, Y. Chen, L.E. Li, K. Bu, P. Zheng, Y. Yang, C. Hu, RuleTris: Minimizing rule update latency for TCAM-based SDN switches, in: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2016, pp. 179–188.
[223] T. Zhang, B. Liu, Exposing end-to-end delay in software-defined networking, Inte. J. Reconfigurable Comput. 2019 (2019).
[224] B. Oh, S. Vural, N. Wang, R. Tafazolli, Priority-based flow control for dynamic and reliable flow management in SDN, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1720–1732.
[225] S. Vissicchio, L. Cittadini, Safe, efficient, and robust SDN updates by combining rule replacements and additions, IEEE/ACM Trans. Netw. 25 (5) (2017) 3102–3115.
[226] T. Liu, C.H. Liu, W. Wang, X. Gong, X. Que, S. Cheng, USA: Faster update for SDN-based internet of things sensory environments, Comput. Commun. 120 (2018) 80–92.
[227] Y. Wang, D. Tai, T. Zhang, B. Xu, L. Jin, H. Dai, B. Liu, X. Wu, Flowshadow: a fast path for uninterrupted packet processing in SDN switches, in: 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), IEEE, 2015, pp. 205–206.
[228] G. Li, Y.R. Yang, F. Le, Y. Lim, J. Wang, Update algebra: Toward continuous, non-blocking composition of network updates in SDN, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 1081–1089.
[229] A.V. Vasilakos, Z. Li, G. Simon, W. You, Information centric network: Research challenges and opportunities, J. Netw. Comput. Appl. 52 (2015) 1–10.
[230] N. van Adrichem, F.A. Kuipers, NDNFlow: Software-defined named data networking, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), IEEE, 2015, pp. 1–5.
[231] H. Luo, Y. Xu, W. Xie, Z. Chen, J. Li, H. Zhang, H. Chao, A framework for integrating content characteristics into the future internet architecture, IEEE Netw. 31 (3) (2017) 22–28.
[232] R. Jmal, L.C. Fourati, An OpenFlow architecture for managing content-centric-network (OFAM-CCN) based on popularity caching strategy, Comput. Stand. Interfaces 51 (2017) 22–29.
[233] A. Mahmood, C. Casetti, C.F. Chiasserini, P. Giaccone, J. Härri, Efficient caching through stateful SDN in named data networking, Trans. Emerg. Telecommun. Technol. 29 (1) (2018) e3271.
[234] R. Jmal, L.C. Fourati, Content-centric networking management based on software defined networks: survey, IEEE Trans. Netw. Serv. Manag. 14 (4) (2017) 1128–1142.
[235] Q. Zhang, X. Wang, M. Huang, K. Li, S.K. Das, Software defined networking meets information centric networking: A survey, IEEE Access 6 (2018) 39547–39563.
[236] A. Nasrallah, A.S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, H. ElBakoury, Ultra-low latency (ULL) networks: The IEEE TSN and IETF detnet standards and related 5g ULL research, IEEE Commun. Surv. Tutor. 21 (1) (2018) 88–145.

[237] N.K. Haur, T.S. Chin, Challenges and future direction of time-sensitive software-defined networking (TSSDN) in automation industry, in: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Springer, 2019, pp. 309–324.
[238] N.G. Nayak, F. Dürr, K. Rothermel, Incremental flow scheduling and routing in time-sensitive software-defined networks, IEEE Trans. Ind. Inf. 14 (5) (2017) 2066–2075.
[239] T. Hackel, P. Meyer, F. Korf, T.C. Schmidt, Software-defined networks supporting time-sensitive in-vehicular communication, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, 2019, pp. 1–5.
[240] N.G. Nayak, F. Dürr, K. Rothermel, Time-sensitive software-defined network (TSSDN) for real-time applications, in: Proceedings of the 24th International Conference on Real-Time Networks and Systems, 2016, pp. 193–202.
[241] M. Boehm, J. Ohms, M. Kumar, O. Gebauer, D. Wermser, Time-sensitive software-defined networking: A unified control-plane for TSN and SDN, in: Mobile Communication-Technologies and Applications; 24. ITG-Symposium, VDE, 2019, pp. 1–6.
[242] A.C. Baktir, A. Ozgovde, C. Ersoy, How can edge computing benefit from software-defined networking: A survey, use cases, and future directions, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2359–2391.
[243] S. Tomovic, K. Yoshigoe, I. Maljevic, I. Radusinovic, Software-defined fog network architecture for IoT, Wirel. Pers. Commun. 92 (1) (2017) 181–196.
[244] J. Wang, X. Zhang, q. Li, J. Wu, y. Jiang, Network function virtualization technology: A survey, Chinese J. Comput. 42 (2) (2019) 185–206.
[245] T. Wood, K. Ramakrishnan, J. Hwang, G. Liu, W. Zhang, Toward a software-based network: integrating software defined networking and network function virtualization, IEEE Netw. 29 (3) (2015) 36–41.
[246] C. Bu, X. Wang, M. Huang, K. Li, SDNFV-based dynamic network function deployment: Model and mechanism, IEEE Commun. Lett. 22 (1) (2017) 93–96.
[247] Y. Liu, Y. Li, M. Canini, Y. Wang, J. Yuan, Scheduling multi-flow network updates in software-defined nfv systems, in: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2016, pp. 548–553.
[248] ONF lays out innovation roadmap for SDN and NFV. https://www.computerweekly.com/news/.
[249] NFV-EVE005: SDN Usage in NFV Architectural Framework. https://joinup.ec.europa.eu/solution/.
[250] M. Aslan, A. Matrawy, A clustering-based consistency adaptation strategy for distributed SDN controllers, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 441–448.
[251] M. Aslan, A. Matrawy, Adaptive consistency for distributed SDN controllers, in: 2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks), IEEE, 2016, pp. 150–157.
[252] B.C. ord Neuman, Scale in distributed systems, ISI/USC (1994) 68.
[253] S.H. Yeganeh, A. Tootoonchian, Y. Ganjali, On scalability of software-defined networking, IEEE Commun. Mag. 51 (2) (2013) 136–141.
[254] Z. Latif, K. Sharif, F. Li, M.M. Karim, S. Biswas, Y. Wang, A comprehensive survey of interface protocols for software defined networks, J. Netw. Comput. Appl. 156 (2020) 102563.
[255] A. Akbar Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, A. Rezaee, Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network, Int. J. Commun. Syst. 32 (4) (2019) e3875.
[256] S. Hager, D. Bendyk, B. Scheuermann, Partial reconfiguration and specialized circuitry for flexible FPGA-based packet processing, in: 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig), IEEE, 2015, pp. 1–6.
[257] P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, M. Horowitz, Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 99–110.
[258] D. Singh, B. Ng, Y. Lai, Y. Lin, W. Seah, Modelling software-defined networking: Software and hardware switches, J. Netw. Comput. Appl. 122 (2018) 24–36.
[259] J. Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, Enabling FPGAs in hyperscale data centers, in: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), IEEE, 2015, pp. 1078–1086.
[260] A. Putnam, A.M. Caulfield, E.S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G.P. Gopal, J. Gray, et al., A reconfigurable fabric for accelerating large-scale datacenter services, in: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), IEEE, 2014, pp. 13–24.
[261] H. Song, Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 127–132.
[262] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, J. Rexford, Pisces: A programmable, protocol-independent software switch, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 525–538.
[263] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, S. Licking, Packet transactions: High-level programming for line-rate switches, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 15–28.
[264] A. Sivaraman, C. Kim, R. Krishnamoorthy, A. Dixit, M. Budiu, Dc. p4: Programming the forwarding plane of a data-center switch, in: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–8.
[265] V. Nguyen, A. Brunstrom, K. Grinnemo, J. Taheri, SDN/NFV-based mobile packet core network architectures: A survey, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1567–1602.