A Survey of Deployment Solutions and Optimization Strategies for Hybrid SDN Networks

Xinli Huang, *Member, IEEE*, Shang Cheng, Kun Cao, Peijin Cong,
Tongquan Wei, *Member, IEEE*, and Shiyan Hu, *Senior Member, IEEE*

***Abstract*—A hybrid software defined networks (SDN) network contains both traditional and SDN network, which combines the robustness of traditional protocols with the flexibility of SDN while avoiding their limitations and incompatibility. However, a hybrid SDN network comes with its own set of challenges, including error-prone deployment processes, risks of inconsistency, and complex incremental deployment strategies. In this paper, we present a survey of the deployment solutions and optimization strategies for hybrid SDN networks. We systematically review solutions to control plane and data plane deployments, and describe typical use cases of hybrid SDN networks. We discuss and compare various optimization strategies from perspectives of traffic engineering, resource saving, network control capacity, and network security. This paper aims to provide insights to researchers into the future development of hybrid SDN networks and inspire more efforts in this area.**

***Index Terms*—Hybrid SDN, control plane deployment, data plane deployment, optimization strategy, traffic engineering.**

I. INTRODUCTION

THE DISTRIBUTED nature of traditional networks has multiple advantages such as scalability, reasonable convergence, resiliency and stability. Traditional equipment, however, has problems such as being vendor specific, offering a fixed set of features, requiring per-box management, cumbersome planning and deployment phases, and high chances of human error. As a result, it is difficult to implement innovative concepts such as network virtualization and on-demand provisioning of Network as a Service (NaaS) in traditional networks. Moreover, a fine-grained level of control is not offered in traditional networks [1].

Software defined networks (SDN) is an emerging network architecture which separates the control plane and the data plane [2]. Because of the centralized network control with global view, SDN provides flexible and reliable network management, support smart flow scheduling for the improvement of link utilization and network throughput. SDN is widely used in different scenarios such as Google's backbone network [3], Microsoft's public cloud [4], NTT's edge gateway [5], and optical (IP/WDM) network [6]. In addition, SDN has been proven successful in Network Function Virtualization service (NFV) which has made significant progress from trial evaluation to production deployment [7], [8]. Because of these significant benefits, enterprises and governments have strong motivations to deploy SDN.

There are several options for helping convert traditional networks directly into SDN networks [9], [10]. A ship in the night strategy is a straightforward approach to new networking architectures. In reality, however, the transition from the legacy network to an SDN network does not happen overnight. First, the transition requires significant deployment costs. In addition to the high price of SDN switches, companies also have to hire additional SDN programmers because an easy-to-use SDN configuration and management frameworks are still evolving [11], [12]. Second, SDN comes with its own limitations, the OpenFlow protocol [13] is not mature enough and commercial SDN switches and controllers are not completely stable and reliable [1]. Those factors slow down the SDN deployment step, thus promoting the birth of the hybrid SDN network [14], [15].

Different definitions about hybrid SDN networks are given in the literature. Generally speaking, hybrid SDN is a logical step in the process of transitioning from a traditional network to SDN. It combines the programmability of the centralized control with the robustness of the distributed routing. Operators can balance the load and manage the network more easily. Controllers in hybrid SDN networks can only focus on useful flow or traffic, while most packets are managed by the robustness traditional protocols. In this case, when deploying a network protector or a network optimizer, operators only need to migrate legacy access devices to SDN switches. However, SDN switches in hybrid SDN networks may be improperly deployed or an inefficient optimization strategy may be adopted, resulting in network performance degradation or inconsistency. Forwarding decisions made by the controller may conflict with traditional routing protocols due to the isolated control domains, which may lead to forwarding loops or black-holes. Therefore, it is of great significance to carry out the investigation into the deployment and optimization of hybrid SDN networks.

Manuscript received December 29, 2017; revised June 28, 2018 and August 5, 2018; accepted September 6, 2018. This work was supported in part by the Shanghai Municipal Natural Science Foundation under Grant 16ZR1409000, and in part by the Key Project of Shanghai Science and Technology Innovation Action Plan under Grant 18511103302. *(Corresponding author: Tongquan Wei.)*

X. Huang, S. Cheng, K. Cao, P. Cong, and T. Wei are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China (e-mail: tqwei@cs.ecnu.edu.cn).

S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA.

Digital Object Identifier 10.1109/COMST.2018.2871061

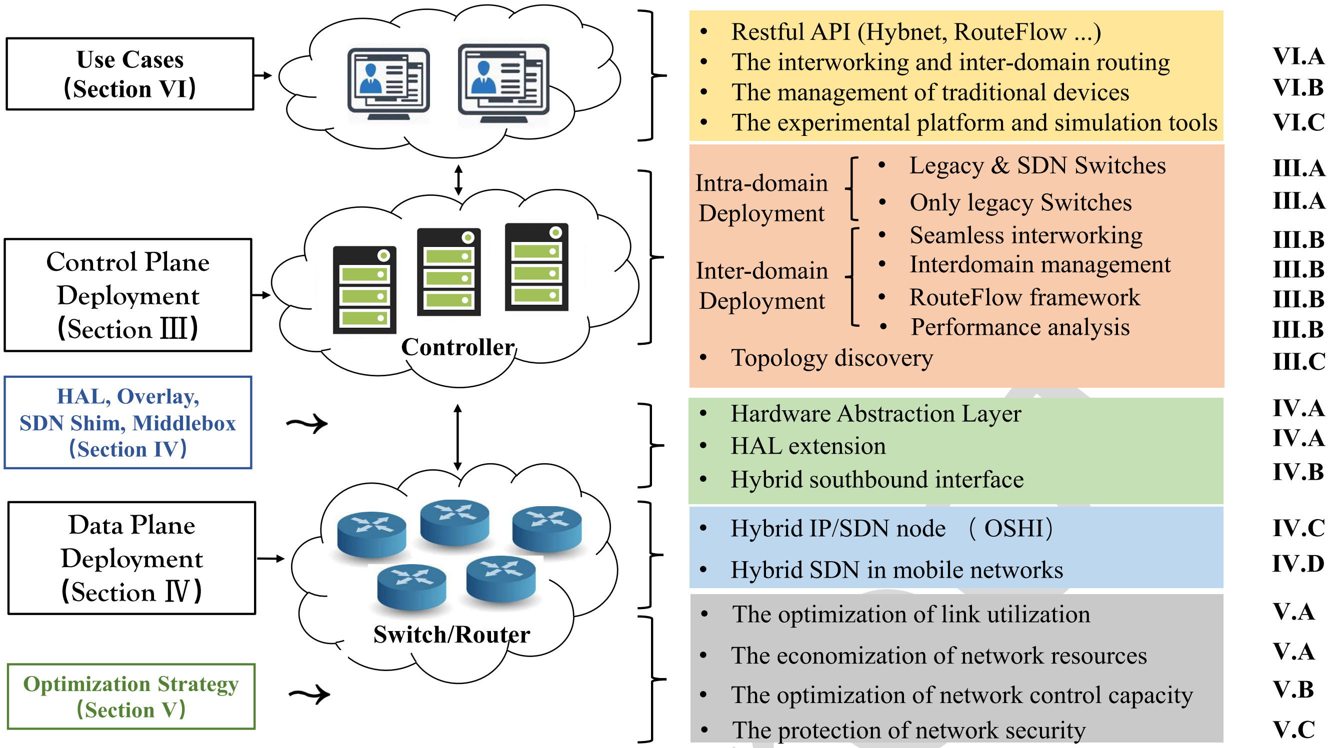


Fig. 1. The deployment solutions and optimization strategies for hybrid SDN networks.

There are many surveys about SDN networks, however, few of them review the state of the art of hybrid SDN networks. Kreutz *et al.* [16] and Thyagaturu *et al.* [6] briefly describe several representative hybrid SDN solutions in their SDN survey. They do not consider the investigation of hybrid SDN networks as a specific research field, but just regard it as a special case of SDN. Vissicchio *et al.* [1] discuss the opportunities and research challenges of hybrid SDN networks, and describe different use cases in each hybrid SDN model. However, no concrete techniques regarding to hybrid SDN networks are summarized. Rathee *et al.* [17] review some deployment solutions in hybrid SDN networks, and discuss the coexistence of traditional networks with SDN networks in detail. However, they do not consider the optimization strategies in hybrid SDN networks.

In this paper, we present a comprehensive survey of the research relating to deployment and optimization solutions in hybrid SDN networks that have been carried out to date. For the deployment techniques, we analyze different hybrid SDN models, summarize control plane and data plane solutions, and discuss several use cases. For the optimization solutions, we discuss and compare different mathematical models and corresponding optimization algorithms.

The structure and organization of the paper are shown in Fig. 1. Section II discusses the model and background of hybrid SDN networks. Sections III and IV describe how to seamlessly unify a traditional network with an SDN network from perspectives of the control plane and the data plane, respectively. We classify related deployment solutions from the perspective of the control plane and the data plane, which is similar to the pure SDN network. In addition, several common design principles are summarized, including the underlying protocols, topology discovery issues and the choice of hybrid network models, which can be used to reduce repetitive works in the deployment process. Section V compares some typical optimization strategies in hybrid SDN networks, including mathematical models, core algorithms, the evaluation and the use case of each strategy. Section VI summarizes some application scenarios where hybrid SDN networks have unique advantages. We summarize this survey and discuss the future research and development trend in Section VII. Table I lists the main abbreviations used in this paper.

II. WHY HYBRID SDN

A. The Concept of Pure SDN

Before discussing the hybrid SDN network, we need to have a brief review of the pure SDN network. The basic elements of a network are nodes (e.g., switches, routers, load-balancers), and interconnections (both physical links and protocol-dependent logical adjacencies) between nodes. In SDN, network nodes implement only the data-plane, while a separated architectural element, called SDN controller, realizes the control-plane. The SDN controller is a logically-centralized custom software, possibly corresponding to a distributed system. The independence between the controller and the nodes simplifies the development of a high-level management interface [18]. The SDN-based architecture is vertically divided into three layers, that is, SDN data plane, control plane and application layer, as shown in Fig. 2.

1) SDN Data Plane: The data plane (i.e., the forwarding plane) consists of distributed forwarding network elements that

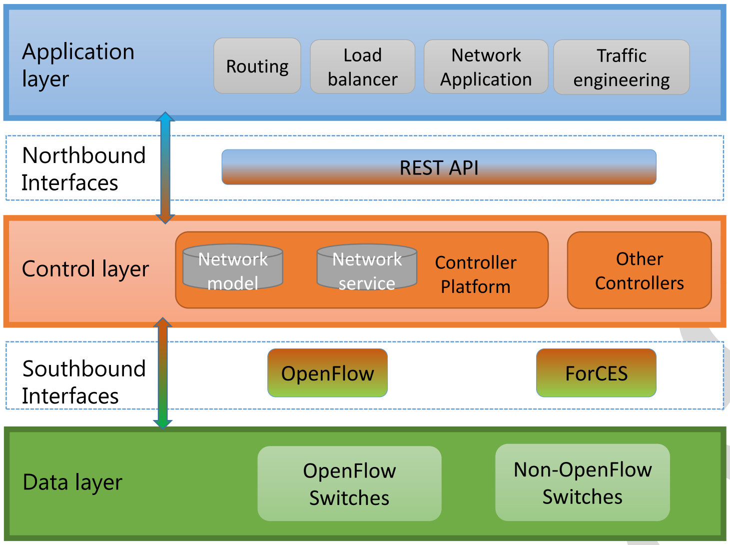


Fig. 2. A three-layer SDN architecture [18].

TABLE I
ABBREVIATIONS

Abbreviation	Explanation
AS	Autonomous System
ACL	Access Control List
BDDP	Broadcast Domain Discovery Protocol
CDN	Content Distribution Network
CLI	Command-line interface
FPTAS	Full Polynomial Time Approximation Algorithm
GMPLS	Generalized Multi-Protocol Label Switching
HAL	Hardware Abstraction Layer
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
ISP	Internet service provider
LFA	Link Flooding Attack
MPLS	Multi-protocol Label Switching
NaaS	Network as a Service
NFV	Network Function Virtualization
NOS	Network Operating System
OFELIA	OpenFlow in Europe Linking Infrastructure and Applications
OSHI	Open Source Hybrid IP/SDN
OVS	Open Virtual Switch
SDN	Software Defined Networks
SDX	Software Defined Network Switching Center
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
TCAM	Ternary Content Addressable Memory
VLAN	Virtual Local Area Network

forward data packets. In order to separate control and data plane, the data plane need to be remotely accessed through an open vendor-independent southbound interface. OpenFlow and ForCES [19] are well-known protocols for the southbound interface. They are responsible for splitting the forwarding plane and control plane in the network, and regulate the communication between the two planes.

2) SDN Control Plane: The SDN controller in the control plane is mainly responsible for two tasks. One is to translate SDN application layer requests to SDN datapath, and the other is to provide an abstract model of the underlying network for the SDN application layer. One SDN controller includes three parts: northbound interface agent, SDN control logic, and control plane interface driver. The SDN control layer is often referred to as the network operating system (NOS) because it supports network control logic and provides the application layer with an abstract view of the global network. The SDN control layer contains enough information to specify policies while hiding all implementation details.

3) SDN Application Plane: The application plane is composed of SDN applications. An SDN application can submit a request to the controller in a programmable manner. The SDN application includes a great amount of northbound interface drivers, which are responsible for driving the open northbound API provided by SDN controller. At the same time, the SDN application can abstract and encapsulate its own functions to provide a northbound proxy interface.

B. The Challenges of Pure SDN

This subsection discuss the challenges in pure SDN that can be mitigated in hybrid SDNs.

1) Deployment Challenges: Although SDN has such great advantages as improving network traffic control capability, reducing network resource consumption, and balancing link loads, it indeed has its own limitations. Deploying SDN in existing networks will incur economic, technical, and organizational challenges. First of all, SDN network can not ignore the initialization costs in the equipment transformation and professional training. Considering the scenario where SDN needs to make a huge change to the network model especially when it comes to architectural updates on the operators side, network managers need to learn how to design, update, debug and operate SDN networks, and companies need to hire experienced SDN network engineers [20]. Secondly, it takes a certain amount of time to produce SDN controllers that can be used at the production level. In order to ease

Key word	Legacy Network	Hybrid SDN	SDN Network
Protocols	OSFP,IGP,BGP…	OpenFlow and OSFP,IGP,BGP…	OpenFlow
Price	Very cheap	Cheap	High
Stability	High	High	Vulnerability
Robustness	Low	High	High
Scalability	Low	Depending on the deployment	High
Programmability	Low	Depending on the deployment	High
Forwarding speed	Low	High	High
Technical difficulty	No	High	Low
Forwarding behavior	Node Local Functions	Depending on the deployment	SDN Controlled Functions

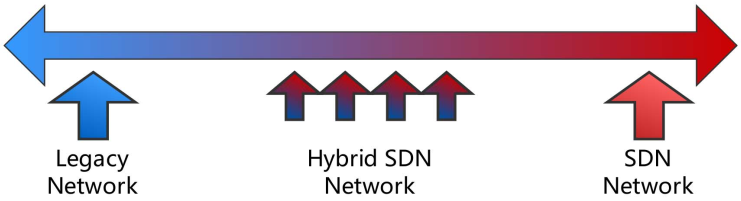


Fig. 3. The overview and comparison of legacy network, hybrid SDN, and pure SDN.

the inconsistent risks among nodes in the network, the controller must make quick decisions in various complicated situations. This will inevitably increase the complexity of network design and deployment, especially in large scale and high performance real-time networks. Thirdly, SDN technology has completely changed the original processing flow of the network chip, which is equivalent to the development of a new forwarding table. Finally, despite the successes so far, SDN implementation is still in an early stage of development, and various research institutes have proposed different design schemes, leading to the lack of uniform standards. The discrepancy among SDN standardization organizations also limits the development prospects of SDN to some extent.

2) Reliability and Security Challenges: First, new demands for mobility, server virtualization and cloud computing lead to increasing real-time requirements of applications such as video conferencing or Web browsing in a very short time range, especially when it comes to quality of service or security issues. To ensure reliability in pure SDN, a fast and expensive out-of-band wide area network between SDN controllers and switches will be needed in large networks [1]. For example, updating information about failures or new input streams would double overheads of network design and management. Second, SDN controllers are software that runs on Windows or Linux operating systems, which leaves the controller and operating system at the risk of being attacked. As long as the attacker can gain control over the SDN controller through continuous attacks, the entire network may be easily attacked. Even if there is no attack, the controller is extremely computationally intensive. Once the controller fails, the entire data center network is paralyzed and becomes uncontrollable.

In fact, the traditional network can naturally avoid the above deployment costs, and the reliability and security issues. This is because its forwarding behavior is realized in complicated hardware devices with no control planes. If there is already a solution to shortcomings of pure SDN in the traditional network, the architecture of the corresponding legacy network remains unchanged. Given this, we only need to focus on how to build a flexible and efficient network model that combines advantages of legacy and pure SDN networks. In other words, the hybrid SDN network can effectively alleviate the above challenges. Therefore, managing heterogeneous paradigms and ensuring profitable interaction between the two types of networks are of particular importance. Fig. 3. gives an overview and comparison of different network architectures.

C. The Models of Hybrid SDN

1) Brief Definition: Hybrid SDN refers to a networking architecture where both centralized and decentralized paradigms coexist and communicate together to different degrees to configure, control, change, and manage network behavior for optimizing network performance and user experience. For example, traditionally switches with their distributed algorithms try to control overall traffic routing whereas, in SDN, the controller routes traffic based on the global view. If these are combined, say a part of traffic is under traditional control and the remaining under the SDN controller, we get a hybrid SDN architecture.

In order to deploy hybrid SDN networks correctly and effectively, a suitable hybrid SDN network model is needed. In this section, we classify the modeling of hybrid SDN networks based on the functional division of networks, the combination of switches, and the network service and usage scenario.

2) Modeling Hybrid SDN Networks Based on the Combination of Switches: Based on the combination of switches, we divide the hybrid network into the three categories, including

- The network integrates SDN switches and traditional switches.
- Traditional switches utilize SDN framework for centralized control.
- SDN switches focus on the interconnection between SDN autonomous system (AS) and non-SDN AS.

For this scenario, deployment strategies can be summarized as

- SDN switch as a middlebox to send information and configuration to the entire network.
- The controller manages traditional switches indirectly by sending seed packets.
- Expanding the controller to control both SDN switches and legacy switches.
- Achieving the hardware abstraction layer or extending southbound interfaces without making any changes to the controller.

3) Modeling Hybrid SDN Networks Based on the Network Service and Usage Scenario: According to the network service and usage scenario, Vissicchio *et al.* [1] classify the hybrid SDN network as Topology-Based Hybrid SDN (TB hSDN), Service-Based hybrid SDN (SB hSDN), Class-Based Hybrid SDN (CB hSDN) and Integrated Hybrid SDN (Integrated hSDN). In the TB hSDN model, the network is partitioned into different zones, and each node or switch is within one zone. In this model, a zone is defined as a collection of interconnected nodes which are controlled by either SDN controllers or traditional protocols. It is required to select the appropriate locations to deploy SDN devices, or to divide the appropriate area as SDN deployment area. In the SB hSDN model, legacy and SDN framework provide different services. For the network-wide forwarding service, the two paradigms can control a different portion of the FIB of each node. In the CB hSDN model, the traffic is divided into two paradigms, one is CN-controlled (legacy), the other is SDN-controlled. Legacy and SDN framework typically span all the network devices, controlling a disjoint set of FIB entries on each switch. In the Integrated hSDN model, SDN has full control of the entire network, and the role of traditional protocol is to forward the control message to the forwarding table in all legacy switches.

After the identification of different types of hybrid SDN models, operators can consider concrete deployment plans. Building a hybrid SDN network requires deploying a port of the SDN switches in a traditional network. However, if no efforts are taken, traditional switches and SDN switches cannot communicate with each other. To solve this problem, many deployment solutions have been proposed, which can be divided into control plane deployment and data plane deployment solutions. In the control plane solutions, changes to the data plane are reduced as much as possible, so that the controller is responsible for unifying these complex underlying switches. In the data plane solutions, it is better to make the underlying network transparent to the controller, thereby minimizing changes in the control plane.

The control plane deployment and data plane deployment solutions are suited for different scenarios. For example, if a new hybrid SDN network is built from scratch, a control plane deployment scheme can be adotped to improve the network operating efficiency. If the original network is a pure SDN network, and the network has been running for a while, or some SDN switches are added to a traditional network by using existing SDN controllers (e.g., OpenDayLight [21], NOX [22]) with no further modifications, it is preferable to use a data plane deployment scheme. However, some data plane deployment schemes introduce an additional hardware abstraction device that may degrade the performance of the network [23].

When these deployment problems are solved, it is necessary to find optimization strategies to make better use of these few SDN switches in hybrid SDN networks. These optimization strategies can not only be implemented in SDN controllers (control plane), but also can affect the deployment order of SDN switches (data plane).

D. The Standardization of Hybrid SDNs

In order to promote the standardization process of hybrid SDN networks, some organizations and research groups have dedicated extensive research effort to hybrid SDN networks and published numerous technical documents and reports. Undoubtedly, the biggest beneficiary of the hybrid SDN network are network equipment enterprises. Therefore, several equipment manufacturers have developed some industry standards and technical guidance of hybrid SDN networks.

1) Standardization Efforts: NFV addresses the topic about "traditional networking coexistence", and discusses possible scenarios of hybrid SDN networks [24]. These scenarios are i) one set of ports (physical interfaces) being assigned to a traditionally controlled datapath whereas other ports (physical interfaces) are assigned to an SDN controlled datapath; ii) forwarding is controlled by traditional mechanisms, which can be used to carry the traffic for an SDN managed overlay network; iii) SDN managed classification operations and actions are used to implement value added processing (e.g., classification into categories for QoS purposes or firewalling) while traditional mechanisms continue to be used for forwarding; and iv) SDN performs major traffic classification and delegates partial forwarding to traditional forwarding elements. The Internet Engineering Task Force (IETF) [25] defines a set of southbound interfaces, which can be used in data plane deployment solutions (Section III-B).

2) Industry Efforts: NEC white paper (2014) [26] tries to avoid pitfalls of SDN by gradually introducing SDN framework to the area of an existing network where fine-grained control is required. The proposed hybrid models can be categorized into three types: add-on type, partial replacement type, and overlay type. In the white paper, NEC also shares the potential commercial value of hybrid SDN networks, including security gateway, DoS/DDoS attack countermeasures, optimization of inter-data center connections, virtualization of the server network, the migration of intra-data center network, and aggregating network management for multiple departments.

HP SDN white paper [27] describes how HP SDN solution uses SDN hybrid mode to achieve scalable, low-risk network deployments. The controller delegates some portion of the data plane forwarding decisions to the controlled switches. OpenFlow-hybrid switches are introduced in their framework to process all kinds of packets and receive/send instructions. Similar solution includes Huawei Enterprise Network Processor, Huawei Smart Network OpenFlow Controller [28], and OpenDayLight [21].

III. DEPLOYMENT SOLUTIONS IN CONTROL PLANE

In order to properly deploy a hybrid SDN network, it is necessary to solve two key problems. The first is how to make the SDN network and the traditional network unified in the view of the controller, and the second is how to get the correct status of the network so that the controller can make the forwarding decision. In this section, we focus on deployment methods in control plane, which are responsible for unifying different kinds of switches by adding extra components and modules. On this basis, depending on the scope of the deployment, these solutions are divided into Intra-domain and Inter-domain deployment methods, respectively. As complementary, we discuss the topology discovery issues and related protocols that need to be considered when implementing a real hybrid SDN network.

A. Intra-Domain Deployment

The intra-domain deployment is designed for the deployment of a hybrid SDN network within an AS. Controllers in these solutions need to find viable paths in the network for the seamless connection. The key to deployment is how to integrate the features of the SDN switches into the entire network. According to whether there is an SDN switch in the network or not, we divide the research into two categories. The first category is legacy switches and SDN switches coexist, and the second category is only legacy switches in the hybrid network.

1) Legacy Switches Coexist With SDN Switches: The coexistence of SDN switches and traditional switches is the most common situation in hybrid SDN networks. In this section, we summarize three types of deployment methods, including i) the management of legacy devices, ii) the waypoint enforcement of traffic, and iii) the hybrid extension of controllers.

The management of legacy devices: The most intuitive way is to force legacy switches to send packets to the controller without considering other strategies. Once the controller receives these packets, it will compute the network operations and announce the updates that need to be performed in the hybrid underlying network.

Jin *et al.* [29] implement a hybrid network controller, called Telekinesis, that provides SDN-enabled routing through legacy paths. The key is to forward the packet to the controller as soon as possible. For the purpose of updating routing entries in legacy switches, LegacyFlowMod integrates the concept of OpenFlow, legacy switch, traditional switch port and MAC address. Telekinesis calls LegacyFlowMod to instruct the SDN switches to send seed packets with the specific source MAC address to legacy switches. When the traditional switch receives the packet, it is believed that the node of the MAC address can be reached, hence, it modifies its IP-MAC mapping table. When a packet enters the network, the legacy switch will forward it to the nearest Openflow switch based on the IP-MAC mapping table. As shown in Fig. 4, the forwarding path s-LE2-LE5-d in Telekinesis includes an OpenFlow switch (OF6).

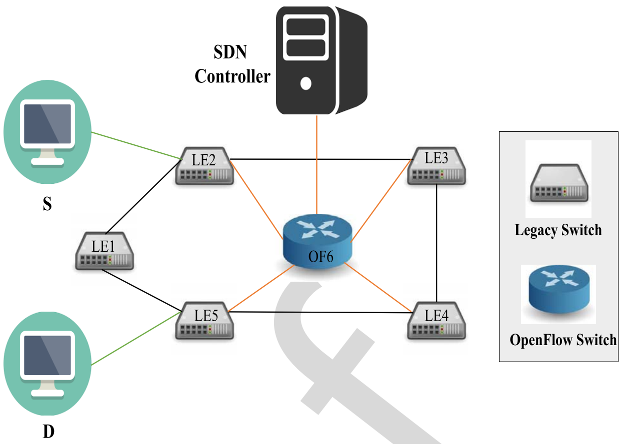


Fig. 4. Telekinesis [29].

The Telekinesis has two disadvantages: i) the controller only provides coarser-grained and destination-based control of legacy path, and ii) the new installed SDN-enabled path is vulnerable. In an SDN network, The controller will install more fine-grained paths because it can match packets based on both source and destination MAC addresses while the layer-2 routing in a legacy network is only destination-based. Besides, regardless of whether these incoming packets are seed packets, the MAC learning function in the traditional switch will react to all of them. When the switch relays a packet from a specific MAC address, a forwarding entry for this address may change, which may lead to the unstable path update. In order to solve the above restrictions, Jin *et al.* [30] further refine the controller, and introduce the concept of magnet address. By sending gratuitous ARP messages, the end hosts will update its IP to magnet address mapping table. The magnet address does not correspond to any real host in the network, which is a fake MAC address to obtain the network visibility and manage the forwarding behavior of end nodes and legacy switches. In this way, the packets from the same destination from different source hosts will go through different paths. According to the destination IP address, the last SDN switch on the new path will rewrite the magnet address to the real MAC address. The result shows that when only 20% of network switches are SDN-enabled, it can achieve full control over routing in the hybrid network.

Network virtualization is the process of combining hardware and software network functionality into a single software-based administrative entity. In order to properly deploy virtualization services, it is necessary to provide the network operator the centralized control over the whole hybrid network.

Lu *et al.* [31] propose a hybrid controller, named HybNET. In addition to the centralized control capability in the hybrid SDN network, a common API is supplied for operators to process transactions and configure hybrid network infrastructure across boundaries. As for the configuration of two types of switches, HybNET does not have the special needs of the network topology like Panopticon [32] and Fabric [33]. The topology and network status are entered manually by the administrator or acquired by a dynamic link discovery protocol (i.e., Link Layer Discovery Protocol (LLDP)). With respect to the manipulation of underlying devices, the framework requires that all switches in the network establish a connection with the controller. These features make it easy to control a traditional device as an SDN switch, and can be further utilized in the network virtualization deployment. For example, when an operator manages the network, the controller analyzes the specific configuration of the underlying network, divides the overall rules into OpenFlow rules and traditional configurations, and sends rules to the OpenFlow controller and traditional switches by SNMP and/or Network configuration protocol (NETCONF) [34].

The waypoint enforcement of traffic: Levin *et al.* [32] propose the Panopticon framework to abstract the transitional network into a logical SDN network. Panopticon is an incremental implementation method on the principle that the packet that traverses at least one SDN switch can obtain the end to end network control (e.g., access control). In this framework, some legacy ports are defined as SDN controlled (SDNc) ports and the traffic between any two SDNc ports must go through at least one OpenFlow switch. For each pair of ports that include at least one SDNc port, the controller will choose one SDN switch as the waypoint and compute the shortest end-to-end path that includes this waypoint. A traditional switch cluster (the set of connected components) is treated as a cell block, and the OpenFlow switch directly connected to the cell block is treated as the boundary node (frontier) of the cell block. In this solution, a per-VLAN spanning tree protocol is configured in each legacy switch to generate a secure path and independent VLAN ID is assigned to each spanning tree to restrict forwarding and guarantee waypoint enforcement, which is available at legacy switches. Legacy switches will forward the packet to the frontier based on MAC-learning and SDN switches act as VLAN gateways. Under special circumstances, when the path between two legacy ports only traverses the legacy switches, the forwarding is performed according to the traditional mechanisms and is unaffected by the partial SDN deployment. Panopticon is a common and high-efficiency deployment mechanism, which can deeply extend SDN capabilities into existing legacy networks.

Considering that the behavior of the edge devices and central devices are different, if the edge devices and central devices are treated equally, it may bring unnecessary complexity to the whole network. An acceptable solution in an SDN network is that the boundary devices are controlled by a fine-grained and service-oriented controller, while the remaining devices are controlled by a coarse-grained controller that focuses on high-speed forwarding. While in a traditional network, in order to save Ternary Content Addressable Memory (TCAM) resources, destination-based solution is responsible for the routing service if the packet belongs to the majority of node pairs, and the other traffic is routed by the complementary explicit routing [35]. Casado *et al.* [33] extend the idea to the hybrid SDN network. The authors suggest that boundary switches can be controlled by SDN controller, providing the advanced and innovative services. Non-boundary switches are controlled by traditional network protocols that provide basic packet transport function. This solution is suitable for network virtualization and SB hSDN model, which can be summarized as waypoint enforcement in edge switches. However, at the edge of an enterprise network, introducing the OpenFlow framework that is not accommodated by existing hardware involves replacing thousands of access switches. Furthermore, the solution limits the ability to apply forwarding policies within the network core, while Panopticon [32] focuses on traditional enterprise networks and the overall performance of the networks.

Caria *et al.* [36] adopt a partitioned solution that divides the entire network into separate subdomains and SDN switches are used to connect these subdomains. LSA altering received by the legacy router triggers a recomputation of the routing and forwarding table. In this solution, node and rule which update within each domain are learned by the OpenFlow switches at the boundary so that these messages can be passed to the SDN controller. When the controller receives the LSA from SDN switches, it will simply forward it to the proposed hybrid network manager, and vice versa. Furthermore, the manager will optimize the internal routing configurations for load balancing by computing the OSPF link metrics based on the partition. Finally, through the SDN switches in the corresponding boundary of sub-domains, these configurations will be flooded as LSAs and injected into the network. The best advantage of this method is that the impact of network fluctuations will be limited to the original area, because the SDN switches can isolate these changes. As for the specific partition strategies, the authors formulate the network partition problem as an ILP model and try to balance the size of each domain as much as possible.

Based on the concept of divide and conquer, the “optical bypass” framework is proposed in [37]. Similarly, traditional network domains are separated by SDN switches, and long-distance high-speed transmission between SDN switches is achieved over optical networks. In this case, heavy traffic is offloaded from the high-load link in the original OSPF domain while has no impact on the stability of the traditional network domain. The paper shows the use case in EU countries. The traffic between countries is forwarded through the optical network managed by the SDN controller, while internal traffic is still forwarded by traditional switches.

The hybrid extension of controllers: Modularization is one of the common ways to relief the complexity in software development, researchers can utilize this pattern to design an extended hybrid control plane and make it scalable. Hong *et al.* [38] implement a typical hybrid network controller by combining the optimized deployment scheme within the controller. The controller contains deployment planning module, global view module, traffic engineering module and failure

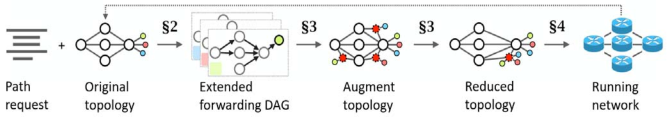


Fig. 5. The four-staged Fibbing workflow [41].

repairment module. When the new budget is approved, the deployment planning module adopts the deployment algorithm to select appropriate switches for incremental deployment. The global view module can obtain the link state of the entire network and pass the information to the traffic engineering module that implements the traffic engineering and fault-tolerance algorithms. The failover module is responsible for alleviating link congestion when failure happens and ensuring fast failure recovery.

OpenDaylight SAL [39] adds support for multiple southbound protocols. For example, Off-the-shelf commodity ethernet switches are commonly allowed to be configured by SNMP, so that an Ethernet switch can actively report its status to the administrative computer (e.g., OpenDaylight controller) using SNMP trap. Therefore, an SNMP southbound plugin (SNMP4SDN) is proposed to control and unify underlying devices supporting SNMP by using off-the-shelf commodity Ethernet switch. This plugin provides capabilities to manage configurations that can only be accessed via CLI.

2) Only Legacy Switches in the Network: In extreme cases, the network only contains controllers and legacy switches. Under this circumstance, the controller can exert a certain degree of control over these legacy switches.

Centralized control over distributed routing: Unlike the way that the OpenFlow messages are converted to legacy switch configurations through a hardware abstraction layer [40], Vissicchio *et al.* [41] implement a "Lie Definition Network" model, called Fibbing. The authors generate a false augmented topology in the data plane by passing false LSA messages to the legacy switches, causing the switch to mistakenly identify some false switches and links. For the reason that real forwarding decisions are all determined by legacy switches via the "tried and true" link state routing protocols. The key point is that the topology these switches find may includes fake nodes and fake links, so legacy switches can be controlled by these fake destination addresses and fake link weights. The fake routing generation process in the controller is regarded as a mathematical function. Specifically, the input parameters are these routing messages, the function is the routing protocols and algorithms in the network, and the output is target FIB entries on legacy switches. Output and function are given, the controller should automatically compute the input parameters. The main process of the program is shown in Fig. 5. After the generation of augmented topology, researchers added optimization steps to reduce the size of the topology. Fibbing is similar conceptually to Telekinesis [29]. The biggest difference between Telekinesis and Fibbing is that Telekinesis focuses on hybrid SDN networks where legacy switches and OpenFlow switches coexist, while Fibbing is defined in a layer-3 legacy network.

Vissicchio *et al.* [42] further increase the availability and scalability, add support for back-up links, and define an efficient demand expression language that supports high-level forwarding requirements. After operators enter the network demand, the controller automatically generates (or manually entered) the desired forwarding map. Then, the augmented path calculation module will design the extended topology based on the input parameters within milliseconds. Under the premise of retaining the forwarding paths, the module will further reduce the augmented topology because the original augmented topology can be very large. Finally, leveraging the Forwarding Address field of OSPF messages, the controller turns fake configuration into actual routing messages and injects them into the hybrid network. The framework is implemented in the Cisco and Juniper routers.

The direct control of legacy switches: The controller can achieve the direct control of the legacy switches to some degree. The establishment of a control system that basically meets the OpenFlow standard requires at least four basic attributes: i) the connection between the data plane and the control plane, ii) the controller can discover the underlying topology, iii) the controller can send instructions to the switch, and iv) the switches are able to send packet-in messages. Hand and Keller [43] propose a hybrid framework, named ClosedFlow, that targets to meet corresponding requirements by: i) establishing independent VLANs to establish a channel between the two planes; ii) sending remote log records from switches to the controller, which enables the controller to receive and store the topology status; iii) achieving an SDN-like control with routeMaps or access-control lists; iv) if a packet does not match any rules, the switch will send a meta-data to the controller or just forward entire packet to controller. This solution is relatively simple and intuitive compared to FIBBING's "spoofing" method, but it requires the switch to support layer 3 protocols and be preconfigured. ClosedFlow leads to a relatively low forwarding efficiency, which is not suitable for large-scale networks.

B. Inter-Domain Deployment

Inter-domain hybrid SDN deployment solutions enhance the cooperation capability of SDN domains in the large-scale network, and ensure the connectivity between different domains that based on different protocols. These solutions not only indirectly improve the efficiency of inter-domain routing, but also realize the innovation of routing services running across domains, most of which can be considered as long-term use cases.

1) Deployment and Management Solutions: In this section, we focus on the interworking and management between SDN and traditional BGP domains, and discuss RouteFlow [44], [45] framework that enables remote IP routing services in a centralized way.

Seamless interworking between SDN and BGP domains: Lin *et al.* [46] implement SDN-IP, which adopts the new SDN device to realize the interconnection between SDN domains and traditional BGP domains. In the SDN domain, there are no legacy routers while some specific SDN switches act as

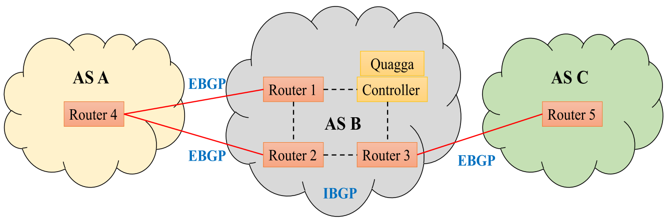


Fig. 6. A hybrid SDN network with BT-SDN framework [48].

BGP peers. To achieve peering, a BGP process is integrated in the network operating system (NOS) for the SDN AS, it exchanges routing updates and establishes the connection with BGP peers on the external IP network. The result is that the SDN AS can be regarded as a single router in the whole network. More specifically, the BGP process module generates BGP route updates, the BGP route module will synchronize these updates and store them in a local route information base. The proactive flow installer in the framework is responsible for calculating and installing the flow entries for Inter-domain traffic by using routes learned through BGP. The interconnection mechanisms between different domains can be further used in Software Defined Network Switching Center (SDX) [47], which allows the operators of participating ASes to deploy novel applications.

In a real scenario, border routers usually have a good performance with high price, so operators may be reluctant to discard them directly. Thus, Lin *et al.* [48] propose a practical framework, called BTSDN, which retains the border BGP routers. As shown in Fig. 6, in BTSDN, the BGP still works the same as in current Internet. The only difference is that the controller also runs IBGP protocol and acts as an IBGP router to learn the global inter-domain routing information. In order to synchronize routing information and set up a full mesh network topology, border routers in inter-domain run External Border Gateway Protocol (EBGP) and border routers in intra-domain run Internal Border Gateway Protocol (IBGP). Quagga [49], which is a routing software package that can provide TCP/IP routing services, talks to all the border routers. The remaining features are similar to those of SDN-IP [46].

The hybrid interdomain management layer: The slice of SDN is usually considered as vertical slicing, and the SDN network slices only isolate the network traffic between controllers while the hypervisor can see all nodes. When hosts and switches belonging to separate entities are integrated together, it becomes a security and privacy problem because all elements are visible to and controlled by the unknown hypervisor. To solve the problem, Thai and de Oliveira [50] propose an Interdomain Management Layer (IML) that is compatible with hybrid SDN networks. It allows network elements to be integrated together without revealing their exact topology and devices attributes. Resources are shared and AS boundaries are managed by the hypervisor through the tools provided by the hybrid policy control controller. IML has the following features: i) the support for flexible centralized policy configuration interface, ii) the preserve of end-to-end packet control of the SDN architecture, and iii) the support for hybrid SDN networks. In IML, SDN ASes are compatible with BGP-dependent ASes. Fig. 7 shows the basic components of the IML architecture in a hybrid SDN network which includes two SDN ASes and one traditional BGP AS. The proxy and bridge work together to allow an SDN AS to setup flows in peering ASes, and the policy controller is designed to avoid the interdomain policy misconfiguration. BGP aggregator intercepts any eBGP messages being sent to a controller from a border SDN switch.

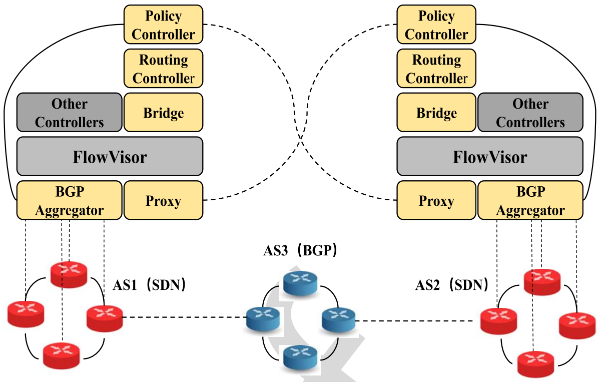


Fig. 7. Interdomain Management Layer [50].

Virtualized IP routing services on OpenFlow-enabled hardware: In some cases, operators may expect that OpenFlow-enabled nodes are aggregated as one single virtual router and exchange routes with traditional network devices. Nascimento *et al.* [44] and Vidal *et al.* [45] propose RouteFlow framework that is ideal for combining SDN switches with virtualization service. The RouteFlow framework is designed to provide virtualized IP routing services on OpenFlow-enabled devices, which builds a virtual L3 topology by mapping all OpenFlow-enabled switches to a Virtual Machine (VM) with a routing engine. The architecture consists of a slave daemon RF-slaved running on each VM, a routing engine RF-server and a controller which runs the route-flow daemon. The RF-server communicates with the VM through the RF-protocol and calculates the corresponding flow-mod commands. The controller uses these flow commands to configure the physical forwarding plane through OpenFlow. When the virtual router interacts with a traditional Layer 3 switch, messages initialized in the VM are passed to the physical OpenFlow data plane by the RF-server and the connected controller, respectively. On the contrary, routing messages from physical OpenFlow data plane is converted and transmitted to the VM by the controller and the RF-server.

RouteFlow creates a simulation framework that duplicates the physical network to the controller. Based on the work, Stringer *et al.* [51] create a distributed virtual router, named CARDIGAN, which adopts distributed protocols for interconnection. On the one hand, the simplified network structure provides shorter maintenance time and reduces the likelihood of misconfiguration. On the other hand, due to the tight coupling in the whole system, it will inevitably bring delays and corruption when installing a large number of rules. Future research could map more features in the traditional network to the underlying SDN architecture, which has good prospects in QoS, load balancing, and failover.

2) Convergence Time and Performance Analysis: Routing between domains is usually realized in a distributed way through BGP. Despite its global adoption, BGP has several shortcomings, such as slow convergence after routing changes, which may cause packet losses and even interrupt the communication. In this section, we review some works that focus on the inter-domain routing performance.

Caesar *et al.* [52] implement a Routing Control Platform (RCP) in traditional networks. RCP collects information about the external devices and internal topology, which can be utilized to select the BGP routes for each router in an AS. The centralized idea is considered as a modification of the traditional network. Integrating SDN to inter-domain routing can indeed improve the performance of BGP.

In order to solve the problems about: i) how much convergence time can be reduced by using the inter-domain hybrid method; ii) how many SDN domains are needed, and iii) how these domains need to be arranged in order to achieve maximum revenue, Sermpezis and Dimitropoulos [53] establish an inter-domain SDN model and propose a probabilistic approach that takes various parameters (i.e., topology, path, number of SDN switches) in the network as input. Based on the hybrid model, the authors derive upper and lower bounds for the time needed to achieve data-plane connectivity between two ASes, and exact expressions and approximations for the time till control-plane convergence over the entire network. For the purpose of minimizing the convergence time of the entire network, the model can be further utilized to evaluate the deployment of inter-domain hybrid SDN network or as an evaluation parameter for selecting the switches to join the SDN domain.

To show the difference intuitively, Gämperli *et al.* [54] construct a hybrid network simulation tool with multi-autonomous domains to study the running state of the hybrid SDN network. A multi-AS routing controller is implemented to outsource network functionality to external service providers, which is designed to address the slow convergence of BGP. POX [55] is used for the interaction with the OpenFlow switch cluster, and ExaBGP [56] is used to interface with external BGP routers. The controller maintains Switch Graph and AS graph for representing the core state. The results show that even at a very low penetration level of SDN switches, the inter-domain routing centralization can also dramatically shorten the convergence time. However, within a small-scale network, churn rates may slightly worse than the pure BGP network.

C. Topology Discovery

Topology discovery is a critical service provided by the controller, it is the basis for the normal operation of the network [57]. As for layer 2 discovery protocols, LLDP is always used in a pure OpenFlow network. In SDN network, the controller periodically commands SDN switches to flood LLDP messages, and SDN switches will forward them back to the controller as soon as they receive these messages. While in a hybrid OpenFlow network with traditional switches, traditional switches will drop these LLDP packets flooded by SDN switches. In this case, LLDP-based links discovery mechanism is not applicable and LLDP+BDDP (Broadcast Domain Discovery Protocol) is a specific solution for discovering multi-hop links in a hybrid OpenFlow network [58].

The broadcast address in the destination field of BDDP messages helps legacy switches to forward BDDP messages, which is adopted to find multi-hop links between OpenFlow switches. First, by encapsulating BDDP in the packet-out message, the controller sends BDDP messages to each OpenFlow switches. Then, if corresponding OpenFlow switch receive the packet-out message, it sends the BDDP message to the directly attached switches. Moreover, if a traditional switch receives the BDDP message, it matches the destination MAC address and floods this message to other active ports. Finally, an OpenFlow switch will receive the BDDP message, and forward it as a packet-in message to the controller.

As for layer 3 discovery protocols, IGP can be used to discover the interconnection between different devices. For example, SDN switches can intercept the OSPF link-state advertisement messages flooded by legacy routing protocol and forward it to the controller via a packet-in message. The controller should be extended to parse LSAs to topology information and detect links between legacy devices. Based on this principle, Hong *et al.* [38] implement a global topology module in HP Virtual Application Networks SDN controller [59]. The authors utilize OSPF Hello message to detect links between SDN switches and legacy switches.

As for application layer protocols, OpenDaylight adds support for SNMP [39], and we have discussed some BGP-based solutions in Section III-B.

IV. DEPLOYMENT SOLUTIONS IN DATA PLANE

In general, operators expect the controller just focuses on the original services while trying to avoid perceiving changes in the underlying network. Through the implementation of the hybrid data plane, we avoid the extra complexity in the control plane and give the controller complete control over the underlying network. In this section, we mainly concentrate on: i) hardware abstraction layer, ii) hybrid southbound interface, iii) hybrid IP/SDN node, and iv) hybrid SDN networks in mobile networks.

A. Hardware Abstraction Layer

When discussing the deployment solutions in the data plane, researchers aim to make little modifications in the control plane, while make it possible for the service running in the controller cannot feel the difference caused by the underlying networks. By adding an "overlay" or a "hardware abstraction layer" between the control plane and data plane, operators could extend their SDN-enabled services to legacy infrastructure, or seamlessly convert non-OpenFlow capable devices into modern OpenFlow switches.

1) HAL Architecture and Features: Hardware Abstraction Layer (HAL) defined in [40] is used to adjust different programmable network platforms. HAL is located between the

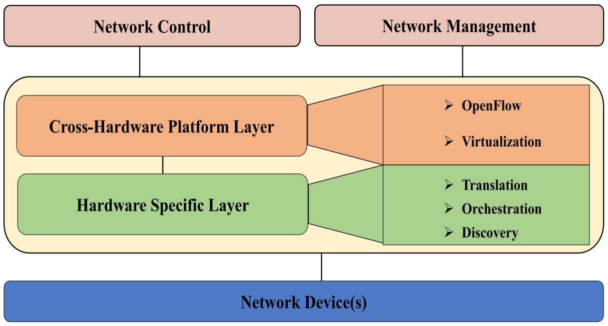


Fig. 8. Hardware Abstraction Layer (HAL) architecture [40].

OpenFlow controller and non-OpenFlow switches, which can be further extended according to different services. Through the management of HAL, the controller can manipulate all elements in the data plane. Belter *et al.* [60] implement HAL in different hardware groups (e.g., programmable network processors and point to multi-point equipment) to enable OpenFlow framework on different types of network equipment.

HAL mainly includes two layers, which are the Cross-Hardware Platform Layer (CHPL) and the Hardware-Specific Layer (HSL). CHPL is in charge of node abstraction, virtualization, and configuration. HSL is responsible for performing all required configurations for different hardware platforms via the hardware-specific modules. According to the type of network devices, the abstract forwarding API and the hardware pipeline API are used for the communication between the two sublayers. This approach is similar to some programming language design ideas (e.g., Java), which is divided into platform-independent and platform-related levels.

Fig. 8 shows the HAL architecture. CHPL is mainly composed of OpenFlow components and virtualization components. The OpenFlow component establishes a connection with the upper layer controller and receives/sends OpenFlow messages. Virtualization components provide platform support for virtualization and streaming-based partitioning, and enable different controllers to control different areas using different versions of the OpenFlow protocol. The operation of CHPL is manipulated by the network management system (NMS) to which the CHPL component is connected. HSL is mainly composed of network discovery module, rule translation module, and orchestration module. The network discovery module obtains information about a series of underlying devices, including the number of devices, models, features, and the underlying topology. The rule translation module is responsible for converting the control actions from the upper layer to the rules supported by the underlying device. The orchestration module sends the underlying network configurations to the controller to help it complete the initialization work and fix the underlying physical network failures.

2) HAL Extension and Comparison: In addition to HAL, some works have achieved the hybrid SDN network by introducing an additional level of abstraction.

Farias *et al.* [61] create a path to config OpenFlow operation and forward the configuration from the controller to legacy devices. The LegacyFlow datapath is the main feature of the proposal, which establishes a path between the controller and the legacy network. When the controller sends the action messages and configurations by secure channel using OpenFlow protocol to the LegacyFlow datapath, the action message is processed, and the information of the flow (i.e., Port in, VLAN ID, Ethertype) are extracted from OpenFlow message and finally sent to the switch control server. The server will build a correct setting according to the vendor of switches. This implementation is scalable, when new brands of switches join the network, operators only need to install adapters of corresponding switches. However, when a legacy switch receives a packet without specific rules, the switch can not pass necessary information to the controller. As a result, these packets can only be forwarded by legacy rules.

Casey and Mullins [23] create a plug-and-play and simple hardware device, named "SDN Shim", which realizes flow-level control from a legacy switch. This shim device provides SDN-like features on legacy switches to enable pre-sales testing and cost-effective infrastructure upgrade planning. The device presents itself to the controller as a regular OpenFlow switch, and it manages flows on the connected legacy switch in accordance with messages from the controller. In this solution, legacy switches should support VLAN tags, so each port can reside on its own VLAN. These legacy switches will pass the received message to the shim device without forwarding it immediately. After the "shim" device receives the unsolved packet, it sends back the result to the legacy switch via VLAN according to the already stored rules or uploads it to the controller via an OpenFlow packet-in message. The design is implemented on a commercial development board. Because of the constraint that a single shim device is hard to handle traffic on all the other ports at the same time, this solution is applicable for low-traffic environments.

Szalay *et al.* [62] present a VLAN-based hybrid framework, named HARMLESS. In the framework, each received packet will be tagged by the legacy switches with a unique VLAN id to identify the access port. Then, packets will be forwarded to the OpenFlow translator component which acts as a software switch. After that, the software switch outputs the packets to OpenFlow switch that is managed by the controller. After being processed by the controller, these packets tagged with a different VLAN id will be sent back to these legacy switches.

HAL framework could be extended to study the different speed of reconfiguration between legacy and SDN devices, as well as how those differences constrain the reality reconfigurability of the network. Sieber *et al.* [63] first present the measurement result that the difference in reconfiguration time between SDN and legacy devices can be a hundredfold. Then, the authors utilize the queuing theory to quantify and compare the maximum reconfiguration rate of the whole network based on the ratio of SDN and legacy devices. Finally, an intuitive metric is proposed to compare different network topologies in terms of their suitability for SDN deployment. In this work, the orchestrator adopts Network Services Abstraction Layer (NSAL) [64] to query the network topology and trigger reconfigurations. NSAL is a vendor and device-neutral abstraction layer, the reconfigurations are translated to device-specific configuration commands and placed in the queues of the devices. The simulation results proved that even a small number of inflexible legacy devices can severely reduce the maximum reconfiguration rate of the entire network.

Feng *et al.* [65] deploy an intra-AS Source Address Validation (SAV) protocol which is used to prevent spoofing attack in the China education and research network 2. Based on a common commercial router, the authors design and implement an OpenFlow-enable open router, called OpenRouter. OpenRouter adds lightweight control layer modules to set up a datapath and send sampling packets to an external OpenFlow controller. In this solution, the OpenFlow protocol is extended for routing notification and packets sampling. After processing, the OpenRouter receives the extended OpenFlow message from the controller. Finally, the control layer module analyzes this message and converts it into a command that supported by the underlying forwarding module.

B. Hybrid Southbound Interface

SDN southbound interface has a variety of standards [16], typically OpenFlow. OpenFlow1.1 defines OpenFlow hybrid switches with both OpenFlow and normal Layer 2 switch functionality. The OpenFlow specification supports "OpenFlow-hybrid mode" since OpenFlow1.3. It defines two classifications of switches, which are OpenFlow-only switches and OpenFlow-hybrid switches. Forwarding decisions on OpenFlow-only switches are completely determined by the controller, while OpenFlow-hybrid switches support both OpenFlow and legacy protocols (e.g., spanning-tree, OSPF) via a traditional networking pipeline. The OpenFlow1.3 specification adds support for "OpenFlow-hybrid mode" via the proposed "NORMAL port" action, which instructs switches to forward the packet based on the traditional networking pipelines. The drawback of this approach is obvious, these protocols are still for switches that at least support OpenFlow.

For making use of existing hardware where possible, the Internet Engineering Task Force (IETF) defines another set of southbound interface "I2RS" [67]. The purpose of I2RS is to integrate routing based on traffic, policy, application, time cost, network status and external events, while taking full advantage of the existing software. In the existing technology, the most common way to implement I2RS is to implement an agent in the user space of the operating system that is installed by the routing device. I2RS agent can communicate with one or more I2RS clients running on the application. It gathers information about the user and kernel space in the routing operating system and then forwards it to the external SDN controller.

Due to the heterogeneity of a hybrid SDN network, when deploying an SDN switch, manual configuration may introduce the risk of human errors and extra operational costs. Katiyar *et al.* [68] focus on automating SDN switch installation solutions. They propose a DHCP-SDN protocol as the extension of DHCP to provide the configuration of new SDN switches. The authors first observe that in a hybrid network, it is not always possible to ensure the reachability between the new SDN switches and the original controller. Then, the proposed Switch Locator will locate SDN switch in the network and the intermediate switches will be configured by the Intermediate Switch Configurator to connect with the new added SDN switch. Finally, the extended DHCP server can react to the DHCP discover message and configure the newly introduced SDN switches by DHCP-SDN.

C. Hybrid IP/SDN Node

Researchers can implement a SB hSDN network by adding support for legacy networks in some SDN switches.

Open Source Hybrid IP/SDN (OSHI) [69] node includes an SDN Capable Switch (Open vSwitch (OvS) [96]), an IP forwarding engine (Linux kernel IP networking) and an IP routing daemon (Quagga [49]). The IP forwarding engine is connected to a set of virtual ports of the SDN Capable Switch (SCS), and the SCS connects to the physical network interface that belongs to a hybrid SDN network. The SCS and the IP forwarding engine are connected by the internal virtual ports, which are implemented through the virtual port module in OvS. For the purpose that the IP routing engine only needs to calculate the routes according to the virtual ports without considering the physical ports, each physical port is connected to a corresponding virtual port of the network. The SCS classifies regular IP packets and fine-grained SDN-controlled packets. So that regular IP packets will be transmitted from the physical ports to the virtual ports, and these packets can be processed by the IP forwarding engine in IP routing daemon. In this solution, controllers and switches do not have to translate the IP routing table into SDN rules. The drawback is that the packet to be forwarded by IP routing will traverse the SCS switch twice, which may bring performance degradation.

Sharma *et al.* [70] propose an integrated network management and control system (iNMCS), and validate it by implementing four novel management use cases. iNMCS combines several legacy network management functions which implies that traditional network management tools and new SDN controllers can interact and operate on the same network. In this architecture, policy manager provides the interface for specifying network requirements and the control decision engine translates the policies specified by the network operator to various OpenFlow-based actions. The flows that need to adhere to service requirements are forwarded to the controller by the hybrid IP/SDN switches, whereas other flows are still handled by legacy protocols.

Hybrid switch that includes legacy functions (non-OpenFlow enabled switches) and an OpenFlow compatible data plane is now commercially available, called Dual Switch. These switches are OpenFlow hybrid switches that can make data plane forwarding decisions independently of the controller. For example, Huawei installs the Huawei Enterprise Network Processor (ENP) chip on their switches to support all the network protocols and SDNs and provide a large-sized OpenFlow table [28]. Accordingly, Huawei designs a Smart Network OpenFlow Controller (SOX) for controlling hybrid SDN networks. However, Dual Switch is a simple combination of legacy switch and SDN switch, where SDN mode and legacy mode cannot take effect at the same time.

TABLE II
COMPARISON OF RELATED WORK BY DEPLOYMENT WAYS, PARADIGM MODEL [1], SWITCH, USE CASE, EXPERIMENTAL APPROACH, KEY IDEA

Related work	**Deployment**	**Legacy switch**	**Model**	**Switch**	**Use Case**	**Experimental Approach**	**Key Idea**
Cheng et al. [29] [30]	CP Intra	C	TB hSDN	Legacy and SDN	Internediate step, Long-term plan	Simulation, Testbed	Magnet address, MAC learning
Levin et al. [32]	CP Intra	U	TB hSDN	Legacy and SDN	Internediate step, Long-term plan	Simulation	VLAN
Casado et al. [33]	CP+DP	U	SB hSDN	SDN or (Legacy and SDN)	Long-term plan		Guideline
Lu et al. [31]	CP Intra	C	TB hSDN	Legacy and SDN	NFV	Simulation, Testbed	VLAN, Virtual Links
Hong et al. [38]	CP Intra	U	TB hSDN	Legacy and SDN	Traffic engineering	Simulation	Extra algorithm
Caria et al. [36] [37]	CP Intra	U	TB hSDN	Legacy and SDN	Traffic engineering, Interediate step	Mathematical simulation	Partitioning
Vissicchio et al. [41] [42]	CP Intra	C	Integrated hSDN	Legacy	Traffic engineering, Long-term plan	Mathematical simulation	Augmenting topology
Hand et al. [43]	CP Intra	C	Integrated hSDN	Legacy	Traffic engineering, Interediate step	Simulation	Remote logging, VLAN
Lin et al [46] [48]	CP Inter		TB hSDN	SDN(SDN-IP)/ Legacy and SDN(BTSDN)	Traffic engineering, Long-term plan	Simulation	OSPF, Converter
Thai et al. [50]	CP Inter		TB hSDN	SDN	Isolation	Simulation	Proxy, Horizontal slicing
Vidal et al. [44] [45]	CP+DP Inter			SDN	Virtualization, CARDIGAN [51]	Simulation	Quagga, OpenFlow
Schlinker et al [66]	Emulation Platform		TB hSDN	Legacy and SDN	Emulation low-level API	Emulation	Quagga, OpenFlow
Gmperli et al. [54]	Emulation Platform		TB hSDN	Legacy and SDN	Emulation high-level API, Test convergence time	Emulation	Quagga, OpenFlow, ExaBGP
Parniewicz et al. [40] [60]	DP	C	Integrated hSDN	Legacy and SDN	Europe Linking Infrastructure and Applications (OFELIA)	Commercial use case	Hardware Abstraction Layer
Farias et al. [61]	DP Intra	C	SB hSDN	Legacy and SDN	Traffic engineering	Simulation	VLAN
Casey et al. [23]	DP Intra	C	Integrated hSDN	Legacy	Traffic engineering	Simulation	VLAN, Shim device
Szalay et al. [62]	DP Intra	C	Integrated hSDN	Legacy	Traffic engineering	Simulation	VLAN,
Sieber et al. [63] [64]	DP Intra	C	Integrated hSDN	Legacy and SDN	Traffic engineering, QoS, monitor	Simulation	Panoption, NSAL, Queuing theory
Tao et al. [65]	CP+DP Intra	C	Integrated hSDN	Legacy and SDN	Intra-AS source address validation (SAV) protocol	Simulation	OpenRouter (Abstraction layer)
Hares et al. [67]	DP	C	Integrated hSDN	Legacy	A new southbound interface		Routing System (I2RS) protocol
Katiyar et al. [68]	DP Intra	U	TB hSDN	Legacy and SDN	The configuration new SDN switches	Simulation	DHCP-SDN protocol
Salsano et al. [69]	DP Intra	C	CB hSDN	OSHI node	Flexibly configure, Back up, Traffic engineering	Simulation Commercial	Quagga, OvS
Sharma et al. [70]	CP+DP Intra	U	SDN	SDN	Network management and control system	Simulation	VLANs
Xu et al. [71]	CP+DP Intra	C	CB hSDN	Legacy and SDN	Traffic engineering	Numerical evaluation	Hybrid path
Poularakis et al. [72]	DP	C	CB hSDN	Smartphone	Traffic engineering	Prototype implementation	OvS
DP: Control plane deployment solution CP: Data plane deployment solution							
Intra: Intra-domain deployment Inter:Inter-domain deployment							
C(Controllable): Legacy switch can be controlled by controller. U(Uncontrollable): Legacy switch cannot be controlled by controller.							

Xu *et al.* [71] propose a way to implement a hybrid switch. When the switch receives a packet, it looks up the forwarding-table and switch/routing table in parallel. If a matching entry does not exists, the destination MAC address will be reported to the controller. Most flows follow the traditional paths, while large flows may be redirected by the controller. The framework decreases the overhead of TCAM resources by reducing the number of forwarding rules, which is more efficient than adopting wildcard rules or redesigning a novel datapath architecture in an SDN network [97].

D. Hybrid SDN in Mobile Networks

Existing mobile network protocols emphasize on the distributed deployment of network resources, while SDN framework concentrates on centralized control. Therefore, it is a challenge to apply SDN design into mobile networks.

Poularakis *et al.* [72] implement a hybrid SDN prototype in a smartphone. The authors propose two methods to integrate SDN and distributed control planes. The first way is the dynamic migration of control protocol by using a distributed routing protocol “as a backup”. When the network changes

TABLE III
COMPARISON OF OPTIMIZATION AND DEPLOYMENT STRATEGY

Related work	**Target**	**Selection**	**Forwarding Behavior**	**Budget**	**Key Idea(Solution)**
Agarwal et al. [73]	Minimize the maximum link utilization	YES	Only SDN	NO	Fully Polynomial Time Approximation Schem (FPTAS)
Guo et al. [74]	Minimize the maximum link utilization	YES	SDN and Legacy	NO	Change the link weight of the current network
Wang et al. [75]	Minimize the maximum link utilization and enhance the controllability	YES	SDN and Legacy	NO	Fully Polynomial Time Approximation Schem (FPTAS)
He et al. [76]	Minimize the maximum link utilization	NO	Only SDN	NO	Linear programming (LP) problem
Wang et al. [77]	Minimize the maximum link utilization	NO	Only SDN	NO	Linear programming (LP), Heuristic algorithm
Das et al. [78]	Maximize the total number of alternative path	YES		YES	Select the key node
Guo et al. [79] [80]	Minimize the maximum link utilization	YES	SDN and Legacy	NO	Genetic search algorithm, K-means algorithm
Xu et al. [81] [82]	Maximize the throughput	YES	Only SDN	YES	Depth-first-search, Randomized rounding
Wang et al. [83]	Find minimum-power network subsets in partially deployed SDN)	NO	Only SDN	NO	Create spanning trees for subsets of nodes
Wei et al. [84]	Turn off the idle links without traffic flow	NO	SDN and Legacy	NO	Neighboring region search
Jia et al. [85]	Turn off the idle links without traffic flow	NO	SDN and Legacy	NO	MPLS, OvS
Hu et al. [86]	Maximize controllable traffic	YES	Only SDN	NO	Fully Polynomial Time Approximation Schem (FPTAS)
Cheng et al. [87]	Select the waypoint for each flow and minimize the maximum link utilization	YES	SDN and Legacy	NO	Mixed integer programming (MIP) model
Jia et al. [14]	Maximize network control ability	YES	Only SDN	YES	Weighted Set Cover and Minimum Weighted Vertex Cover problem
Kar et al. [15]	Bring more coverage benefit with minimum deployment cost	YES	Only SDN	YES	Pcoverage and Hcoverage
Poularakis et al. [88]	Maximize controllable traffic and maximize the number of dynamically selectable routing paths	YES	Only SDN	YES	Submodular and supermodular functions
Vissicchio et al. [89] [90]	Safe update of hybrid SDN networks	NO	Only SDN	NO	A generic control-plane model [91]
Amin et al. [92]	Auto-Configuration of ACL Policy in Case of Topology Change	NO	SDN and Legacy	NO	Fully Polynomial Time Approximation Schema (FPTAS)
Chu et al. [93]	Single link failure recovery	YES	SDN and Legacy	YES	Heuristic algorithm
Markovitch et al. [94]	Fast failover	YES	Only SDN	NO	Spanning Tree Protocol (STP), BPDUs
Wang et al. [95]	Mitigate Link Flooding Attack	YES	SDN	NO	Traceroute

Selection: Whether select the appropriate location where the SDN switch should be deployed.

Forwarding behavior: Forwarding behavior defines which type of switches the algorithm needs to control or adjust.

such as link or node failures, mobile devices can automatically change their forwarding behavior from OpenFlow to distributed routing protocols. The second way is the cluster-based hierarchical control. By allowing a set of nodes to work together as one cluster and determining routes independently of other nodes, the SDN controller can only focus on the guidance of routing from one cluster to another. The proposed smartphone framework includes an OvS software [96] and a local software agent. With OvS, a smartphone becomes a virtual switch that is similar to an OpenFlow switch. The agent is designed to maintain the distributed protocol, synchronize network states with other nodes, and calculate routing paths.

In Table II, we provide a summary of all mentioned deployment methods and their characteristic proposed for each research work in hybrid SDN networks.

V. HYBRID SDN NETWORK DEPLOYMENT AND OPTIMIZATION STRATEGY

When considering traffic engineering solutions, a hybrid SDN network can adopt similar approaches proposed in a pure SDN network. In this section, we focus on optimization solutions and related deployment algorithms in hybrid SDN networks. In Table III, we provide a summary of the problem/goal and the solution proposed for each research work. Based on this table, we will discuss related works from the following aspects: the traffic engineering in hybrid SDN networks (i.e., the optimization of link utilization and load balancing, saving network resources), the optimization of network control capacity, and the protection of network security. Fig. 9. gives an overview of these optimization strategies.

A. The Traffic Engineering in Hybrid SDN Networks

The global centralized control and programmability of the network behavior provide effective features to support traffic engineering in a pure SDN network. In a hybrid SDN network, it is possible to apply some comprehensive traffic engineering solutions because the splitting ratio of the flows on SDN switches is arbitrary. Researchers need to choose appropriate migration sequence to maximize the profits of centralized control, which includes various optimization algorithms. These algorithms can be applied not only to the deployment of hybrid networks, but also to other network optimization strategies.

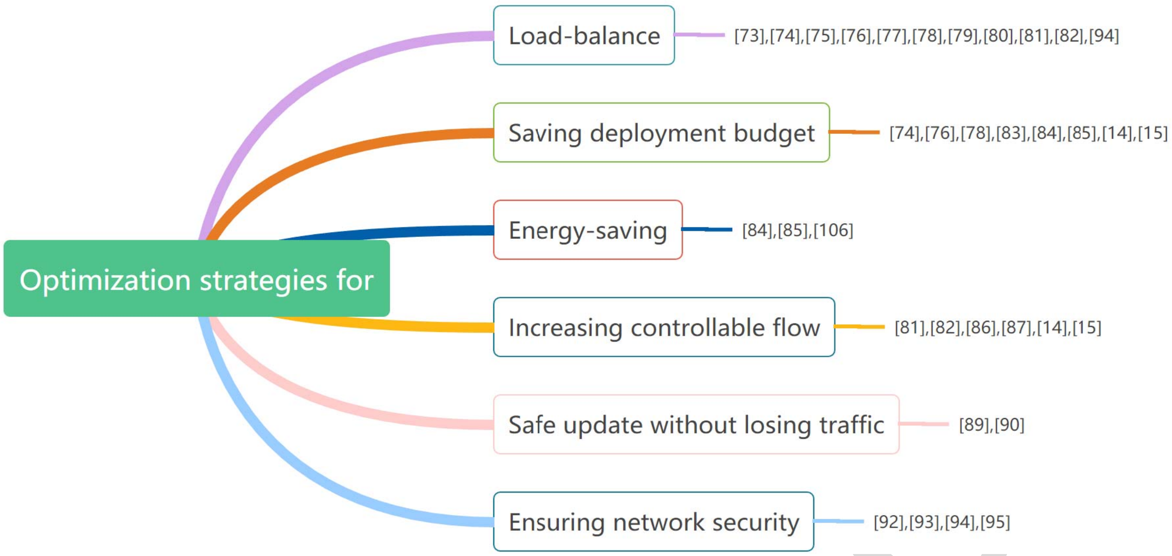


Fig. 9. The objectives of optimization strategies.

For example, the migration solutions could be extended to i) find appropriate locations for the middlebox in the traditional network to achieve fine-grained control over the network [98]; ii) solve the AP placement problem in the wireless networks to meet the energy requirements [99]; iii) improve the resource utilization in data centers by solving the VM replacement problem [100].

1) The Optimization of Link Utilization: In B4 network [3], by splitting flows among multiple paths, the centralized traffic engineering service brings the link up to nearly 100% utilization. The hybrid network is excepted to approach this utilization, so we summarize optimization algorithms according to different network models and topology parameters.

The minimization of the maximum link utilization: Agarwal *et al.* [73] establish a hybrid SDN network model and propose a related full polynomial time approximation algorithm (FPTAS). The purpose of the study is to develop an SDN deployment framework that can be used to minimize the maximum utilization of the links with different traffic patterns. For simplicity, the weight setting of links is fixed, the network topology and the locations of SDN switches have been determined in advance. The full-polynomial time approximation algorithm is superior to the traditional standard linear programming in time and space complexity. In order to obtain a better "given network state", the authors used random selection and greedy solutions to further determine the deployment location of the SDN switches.

Guo *et al.* [74] focus on how to optimize the OSPF weight setting to balance the traffic coming out of the legacy switches. The authors advocate that the splitting ratio of the SDN switches and the weight setting of the links can both be adjusted at the same time. The legacy switches that adopt OSPF protocol will forward the traffic from the specific port due to the change of the link weight. Therefore, for the goal of minimizing the maximum utilization of all links, the authors propose the SDN/OSPF Traffic Engineering (SOTE) algorithm. The algorithm dynamically changes the weight of each link and then calls the SDN node optimization function to calculate utilization in each iteration. At the end of all iterations, the link utilization, the weights of each link, and the behavior of SDN switches can be derived.

Caria *et al.* [78] propose a similar solution about changing link weights. In their work, SDN switches divide the initial OSPF domain into sub-domains, and LSA altering is adopted to change the routing between sub-domains. The authors observed that there are some limitations to the goal of minimizing the maximum link utilization. For example, in case of a heavy loaded link at the beginning, the optimization method may not yield a feasible solution. In their work, each link is associated with a cost based on its real utilization, and the objective of the proposed Integer Linear Programming (ILP) model is to minimize the total cost in the hybrid network. Comparing with SOTE, this evaluation strategy is more flexible than just focus on minimize the maximum utilization.

Wang *et al.* [75] propose a generic traffic engineering approach that complies with the forwarding characteristics and capabilities of SDN and distributed routing. They are the first to take into account the differences between legacy switches in order to minimize the maximum link utilization. The designed approach supports both traditional single path and multipath routing protocols in legacy switches. More specifically, this approach considers the flow-level and packet-level multipath forwarding on legacy switches, and describes their restrictions on forwarding. The traffic engineering algorithm in this paper is similar to [73] and [74], which is also based on FPTAS. The evaluation results confirm that, with 70% deployment of SDN nodes, the hybrid forwarding with traffic engineering could achieve as much throughput as a full SDN.

Existing hybrid traffic engineering approaches primarily focus on changing traffic splitting weights, Wang *et al.* [77] detect that the effectiveness of traffic engineering in hybrid SDN networks strongly depends on both the next-hops and traffic splitting ratios on SDN switches. They are innovative in considering the next-hops construction. In the research, the authors first construct forwarding graphs with consistent and potentially high throughput for effective traffic engineering, while maintaining forwarding consistency. Then, a heuristic forwarding graph algorithm that constructs forwarding graphs for flows is proposed to reduce the traffic engineering overhead. Finally, they calculate the traffic distribution based on the forwarding graphs with linear programming. Experimental results show that the algorithm achieves higher throughput and better load balancing than other forwarding graph construction method, especially during the early SDN upgrade period with less than 40% SDN deployment.

The traffic engineering in barrier mode: He and Song [76] are the first to propose traffic engineering in barrier mode, which is adopted to forward traditional traffic and SDN traffic in distinct overlay networks. In order to pass SDN traffic through legacy switches, a destination based forwarding and traffic aggregation routing protocol is defined in this paper, so the routing table in legacy switches is modified to support the programmable flow splitting. For the same purpose of minimizing the maximum link utilization, the problem is formulated as an ILP problem that can be solved by CPLEX or fast algorithms with approximate guarantee. Barrier mode is a conservative development method, because traditional network will not be influenced by new techniques, while the rough isolation may lead to the potential underutilization.

The selection of migration sequence: The above solutions optimize the utilization of the network when the hybrid network is basically deployed and the location of SDN switches are determined. Intuitively, if SDN switches are located at the edge of the network or only adjacent to few devices, this switch may only control and distribute little traffic. Hence, it is relatively difficult to bring more benefits to the entire network, and it is essential to decide the best location for SDN deployment.

Caria *et al.* [11] propose a novel two-stage migration scheduling algorithm to decide which switches should be replaced to increase as many alternative paths as possible. In the first stage, the algorithm first analyzes the network topology to find all paths that can be used for traffic engineering. For these paths, the algorithm then identifies the switches that must be SDN-enabled. In the second stage, for the purpose of maximizing the number of alternative path throughout the migration periods, the authors present an ILP model to determine the migration schedule. However, the model does not consider the traffic issues in the network.

Guo *et al.* [79], [80] combine the node selection strategy with the topology optimization algorithm to find the node update sequence that minimizes the maximum link utilization of the network. In order to accommodate uncertain traffic, the authors combine the off-line weight optimization for legacy switches with the on-line splitting ratio optimization for SDN switches. When considering migration sequence, if N nodes need to be updated, there are $N!$ update sequences. The search is exhaustive if the brute force algorithm is used. Therefore, the authors choose heuristic algorithms, that is, genetic algorithm searching (GAS) and greedy algorithm searching (GDS), for optimizing migration sequence. After each iteration, SOTE [74] algorithm is used to calculate the optimal link utilization of the network at this time. The result is used as the criterion to change the node update sequence to dynamically improve the network performance until the number of iterations is reached or the expected target is satisfied. Furthermore, in the pre-processing phase of the experimental data, the authors cluster historical traffic matrixs (TM) by using the k-means algorithm, and then computing the weight coefficient of every representative traffic matrix. When the algorithm optimizes the link weights, a larger weight means that its corresponding TM is more important. As a result, an expected TM can be obtained, which can display the average traffic in different traffic modes. This solution enables intra-domain routing to be robust to the changing traffic demands.

2) The Economization of Network Resources: The flexible control capability of the SDN switches enables multiple links that are previously not available for forwarding. However, the increase of available links may cause the pressure in network costs.

The analysis of deployment cost: The economic and budgetary implications need to be taken into account when seeking the best deployment and optimization options. As mentioned in [11], when a key-node node is deployed as an SDN switch, it provides some paths that will candidate for traffic engineering while generating different deployment costs. It could be considered as a $0-1$ knapsack problem. However, this model is too simple for large-scale and complex models, and more candidate links do not necessarily bring the best performance [82].

The reduction of energy consumption: For the purpose of reducing the unnecessary energy consumption, Wang *et al.* [83] aim to find minimum-power network subsets in a hybrid SDN network. They advocate that the power consumption of legacy devices cannot be altered and the controller can only manage the SDN switches. The power consumption of SDN switches plus the link connected to it is equal to the total power consumption. On this basis, the authors develop a new spanning tree algorithm to select the lowest-cost link subset, ensuring that each link does not exceed the load and the network reachability is not destroyed. However, when the load is too heavy, too many overload links are generated during the calculation of the spanning tree. Hence, the process of adjusting these links is complicated, which results in delays and packet loss problems. Similar algorithms exist in the study of CDN networks, that attempt to shut down idle devices during off-peak hours [101].

Wei *et al.* [84] propose the hybrid energy-aware traffic engineering (HEATE) algorithm, which aims to reduce power consumption by determining the optimal setting for the OSPF link weight and the splitting ratio of SDN switches. The authors assume that operators expect they could aggregate traffic flow onto partial links and then turn off underutilized links to save energy. The solution is similar to the heuristic algorithm in SOTE [74], which tries to delete the minimum-utilization link in each iteration, and then move traffic from low-utilization links to high-utilization links.

Jia *et al.* [85] propose a more viable energy saving solution. In their work, the SDN switches reroute packets based on multiple MPLS labels which take the forwarding ports of switches. The latest OpenFlow protocol supports MPLS technology, the Push MPLS header actions can push new MPLS headers onto the packet. When a new MPLS tag is pushed onto an IP packet, it becomes the outermost MPLS tag, and is inserted as a shim header immediately before any MPLS tags or immediately before the IP header. SDN switches encapsulate multiple MPLS labels for each packet that indicate the forwarding port numbers of switches. Energy saving requires fine-grained flow scheduling to shut down switches and links. An SDN switch achieves fine-grained flow scheduling by encapsulating the MPLS labels of the forwarding information. Thus, it can save energy by rerouting the flows to turn off the idle links and switches.

B. The Optimization of Network Control Capacity

When the traffic in the hybrid SDN network passes through an SDN switch, the controller may obtain the meta-data through the packet-in message. In general, the route of traffic is jointly decided by traditional distributed routing protocol running at non-SDN routers and the SDN controller. Considering the factors that may affect the percentage of controllable traffic, researchers mainly focus on the number of SDN switches, the location and forwarding behavior of these switches, the topology and the link weights of the entire network.

The maximum of controllable flow: In the case that the number and the location of SDN switches are determined in advance, Hu *et al.* [86] aim to find the maximum flow that can be controlled by only tuning the forwarding behaviors of the SDN devices. The authors formulate it as a linear optimization problem, and a full-polynomial time approximation algorithm is proposed in this paper. To ensure that the link does not exceed the maximum load, the SDN switches should direct traffic to the lower-load link while avoiding these traffic through other SDN switches. In each iteration, the algorithm computes the shortest controllable path between SDN switches and other nodes. The simulation result shows that when the network includes half of the SDN switches, all traffic is controllable.

Operators may expect the controller to be able to control all traffic in a hybrid SDN network, Ren *et al.* [87] advocate that each end-to-end flow can be forced to traverse at least one SDN switch. In their work, the forwarding process for each flow is divided into two parts: i) from the source node to a selected SDN switch, and ii) from this selected switch to the destination node. Under this constraint, the authors formulate the waypoint forwarding model and propose the flow routing and splitting (FRS) algorithm to find the maximum link utilization. FRS heuristically computes a most promising subset of all the available paths, and jointly determining an appropriate SDN switch as the waypoint for every flow. However, when the percentage of SDN switches is low, if each flow is still forced to traverse at least one SDN switch, it may lead to a high price, and the network performance is even worse than traditional networks. The result shows that the initial performance is poor compared to the method proposed in [74] that does not have the waypoint enforcement, while the performance is better when the proportion of migrated nodes exceeds 20%.

The migration sequence and budget analysis: Similarly, the migration sequence and the budget limitation of SDN switches are also important factors that may affect the proportion of network controllable traffic. Jia *et al.* [14] aim to maximize the network control ability with limited budgets, and minimize the cost of migration while achieving full control of the network. The authors formulate it as the Weighted Set Cover problem and Minimum Weighted Vertex Cover problem, respectively, and propose a unified heuristic algorithm. In each iteration, the algorithm greedily selects the optimal location based on the number of flows and the cost of SDN switches while ensuring that the deployment cost is lower than the budget.

Kar *et al.* [15] refine the network coverage model and propose two evaluation parameters, Pcoverage and Hcoverage. The definition of the former is as long as there is an SDN switch in a path, the path is P-covered, then Pcoverage is defined by the proportion of these paths in the entire network. If 20% of the switches in this path are SDN switches, then the Pcoverage is 20%. Hcoverage refers to the percentage of SDN switches within the P-covered path. The purpose of the study is to maximize the coverage with budget constraint or minimize the budget with the coverage constraint. Two corresponding heuristic solutions, maximum number of uncovered path first (MUcPF) and maximum number of minimum hop covered path first (MMHcPF), are proposed in this paper. Compared with other algorithms, MUcPF requires 5% to 15% less budget to achieve the assigned Hcoverage target. MMHcPF is a consistent algorithm, the difference between the maximum Hcoverage and the minimum Hcoverage in MMHcPF is only 20-30%.

Essentially, strategies in [14] and [15] only take the budget as a simple constraint and do not refine cost models. Poularakis *et al.* [88] propose a refined and complex cost composition function. Based on the general cost model, the authors focus on maximizing the programmable traffic that passes through at least one SDN switch and maximizing the flexibility by increasing the number of alternative paths. The theory of submodule and supermodule is used to design the algorithm with provable approximation ratios. The authors observe that the interplay between the two objectives in the experience is that if one objective is optimized, another objective will naturally obtain a benefit.

The above works mainly focus on the arbitrarily splittable flow routing in a hybrid network, which means the SDN switch can split the flow arbitrarily. However, when the size of the flow table is significantly less than the number of flows, the difficulty of flow table management will be increased due to the assumption of arbitrarily splittable flow routing. Xu *et al.* [81] add the h-splittable ($h \geq 1$) restriction in each flow and introduce a novel incremental SDN deployment scheme, named duplicated deployment. Duplicated deployment refers that new SDN equipment is placed “in addition” while not “instead of”, each SDN switches will be collocated with one legacy router. In this solution, anomalies in the SDN switches or SDN controller will not disrupt the basic connections in the traditional network. As for deployment strategies, the heuristic algorithm (MAX-k-SFD) is proposed to determine the location of SDN devices under the budget constraint. In each iteration, the algorithm selects the location to maximize the number of new controllable flows. After the deployment of SDN devices, the depth-first-search and randomized rounding based algorithm, named MRHS, is proposed to maximize the throughput, which has approximation ratio $O(\frac{1}{logN})$.

For the same purpose of throughput maximization in duplicated deployment scheme, Xu *et al.* [82] consider the budget constraint that the amount of additional bandwidth and the number of SDN switches should be limited. In order to re-route flows at an SDN switch, additional bandwidth is required on certain links. The authors formulate the problem of throughput maximization under budget constraints as a joint duplicated deployment and routing (DDR) problem, and an approximation algorithm based on the traffic mapping and randomized rounding methods is proposed. The algorithm first solves the relaxed DDR problem and get the fractional solution, then rounds it to an integer solution. It is proved that the approximation factor is O(log n) in the worst case and O(1) under most practical situations for link capacity and flow-table size constraints (n is the number of all switches).

C. The Protection of Network Security

In this section, we discuss some issues that may be encountered when considering the deployment of a hybrid SDN network, including: safe updates of hybrid SDN networks, the detection of network failures, the implement of traffic matrix and the avoidance of link flooding attack.

1) Safe Updates of Hybrid SDN Networks: Because of the potential conflicts between different control planes, updating a hybrid network may lead to numerous forwarding inconsistencies. To update hybrid networks without losing traffic or violating security policies, Vissicchio *et al.* [89] develop provably correct techniques that enable consistency in: i) the update of the SDN-controlled or the IGP-controlled forwarding paths, ii) the update when IGP-controlled flows become SDN-controlled or the SDN-controlled flows become IGP-controlled. During the hybrid network update, the authors first prove that any end-to-end connection can be guaranteed (e.g., black-holes and forwarding loops can always be avoided), while it is not always possible to guarantee the path consistency (i.e., violating some security policies). Then, the Generic Path Inconsistency Avoider (GPIA) algorithm is introduced to compute the longest consistent sequence of FIB replacements with no overhead.

Vissicchio *et al.* [90], [91] further extend the above method and theoretically prove that the new method can be used in topology-based hybrid networks. The router is defined with a model that is general enough to capture all kinds of control planes. This model is independent of the path calculation algorithm used for forwarding and the header field used for matching packets. On this basis, the control plane is divided into FIB-aware (FA) and FIB-unaware (FU) based on whether the forwarding entry is based on FIB content or not. Then, the algorithm [89] is proved that it can be adopted in the FU-only control plane, which may be unnecessarily complicated for a strongly consistent FU update and cannot be adopted in the FA-Existence network. Finally, the authors propose that safe updates for generic networks can be achieved by combining the replacement and duplication of FIB entries. The replacement phase consists of the GPIA algorithm that mentioned before. The duplication phase refers to duplicate the FIB entries that cannot be replaced without creating path inconsistencies.

2) The Detection of Network Failures: Any network may encounter the link failure problem. When a link failure occurs, the legacy switches will automatically select the standby link for transmission in a traditional OSPF network. The channel between a traditional IP router and an SDN switch can be established in hybrid SDN network, so that when a legacy switch detects a link failure, it can immediately redirect traffic to SDN switches. After that, SDN switches will identify which link is failed according to original and tunneled IP headers, and controllers can help these flows bypass the failed node or link. The selected recovery paths will be installed as rules in the SDN switches that are on the corresponding paths. Hence, a single link failure problem can be resolved.

Operators can find out the number of SDN switches they need when considering the single link failure problem in a hybrid SDN network. Chu *et al.* [93] formulate it as a binary linear programming problem and solve it through a heuristic algorithm with polynomial time complexity. Given the affected router and the destination hosts, the router should forward the packet to an SDN switch without using the failed link. For each failed link, the authors first find out all candidate locations for SDN switches. Then, the algorithm will periodically select the location according to the number of failures that can be covered in each iteration. The algorithm ends when each link failure is covered. Furthermore, because of the global view of the controller, given the affected routers and destination nodes, the algorithm is extended to minimize the max link utilization of the post-recovery network by choosing the optimal recovery path. Some related work [14], [88] try to find more selectable routing paths, which also make it possible to dynamically respond to link failures or link congestion.

Markovitch and Schmid [94] propose a network architecture, named SHEAR. In this architecture, the partially deployed OpenFlow switches divide the network into multiple loop-free domains. These switches located in loop-breaking locations are regarded as monitor points, which help the SHEAR controller to quickly detect and locate failures and provide traffic-engineering flexibilities. STP spanning trees are used by the controller so that no link failures are ignored. Specifically, SDN switches are responsible for each network domain and receive network updates through MSTP messages (BPDUs) . These updates will be forwarded to the controller so that the controller can locate the failure link and compute the affected traffic. The principle of the solution is that link failures will change the value of a root in the BPDU, and the controller can localize the failure between the expected root and the current root within the domain. Finally, the controller will reroute the traffic according to the spanning trees or just notify the network operator.

There are frequent link changes and new device additions in the network, and the network policies configured at the interfaces of switches may be violated. Amin *et al.* [92] propose an approach that automatically detects the network policies that are affected because of the topology changes, called Auto-PDTC. This approach simulates the network-wide and local policy at forwarding devices by using a three-tuple and a six-tuple, so that the controller can obtain link status information from all switches and then chart it. In the case of topology changes, the graph difference algorithm is used to auto-detect the changes, it constructs the search tree to verify policy violation either exist or not.

3) The Implementation of Traffic Matrix: Traffic matrix is widely used to monitor network status and prevent network anomalies. A traffic matrix requires a significant amount of monitoring equipment and network-wide configuration efforts, which is not readily available in legacy IP networks. While in an SDN network, SDN-enabled devices provide additional byte counters for all individual entries in their forwarding tables. Inspired by this feature, Medina *et al.* [102] aim to augment the SDN-based traffic statistics with SNMP-based throughput measurements, obtain and measure flows by temporarily offloading them on IP backup links. A backup link in addition to a regular IP link is easy to create and configure, allowing the measurement by regular SNMP link byte counters, which is vendor-independent and available in almost every router. More specifically, a separate physical port on a pair of IP routers is configured as a backup to an IP link. In addition, the framework defines a set of ACLs such that the flow in question can be distinguished from the remaining traffic. As complementary, to minimize the total cost, the authors propose a linear optimization model and a greedy heuristic algorithm to determine the optimal measurement locations for SDN switches and backup links.

4) The Avoidance of Link Flooding Attack: DDoS attack such as Link flooding attack (LFA) may degrade or even block network connectivity in the target area. The legitimate and low-density traffic in LFA can hardly be distinguished in traditional networks. Wang *et al.* [95] present a framework that can effectively mitigate LFA in hybrid SDN networks, named Woodpecker. After the optimal selection of upgrading switches based on the benefits (the amount of controllable traffic) of upgrading a certain switch, the key is to find out the congestion link and determine if LFA is happening. The detection module in Woodpecker is implemented to find the congested link. When the SDN switch finds that the traffic exceeds the threshold, an alarm message will be sent to the controller. The controller will install two flow-mod rules that match different ICMP messages to SDN switches. Then the controller will inject ICMP packet to the SDN switch via a packet-out message. The SDN switch will match and forward the ICMP packet based on the normal rules, while the legacy switch will forward the packet and decrease the TTL value or return the ICMP reply message according to the current TTL status. As soon as an SDN switch receives the ICMP reply packet, the packet will be sent back to the controller. Based on the received ICMP reply message from different SDN switches, the controller will locate the congested links. After that, a traffic engineering algorithm that aims to minimize the maximum utilization of all links is enforced to mitigate this attack. Finally, if some traffic is too heavy to handle, the controller will instruct switches to discard some packets if the IP address of these packets always appears on congested links.

D. Experiments and Simulations

Most of the optimization strategies first find the problems that can be optimized or urgently needed to be solved in the hybrid SDN network, then formulate the network optimization problem, and design optimization algorithms or heuristics to solve the problem. Due to the special structure of hybrid SDN networks, there is no simple network simulation tool such as Mininet (a popular simulation tool for pure SDN) [103]. Therefore, researchers have adopted different methods to validate the correctness of assumptions and verify the efficiency of algorithms, as is described below.

1) Simulation-Based Performance Measurement: Most works use numerical verification methods to prove the effectiveness of their algorithms. The commonly used implementation process are to i) extract the traffic data set of the traditional network, and ii) put the data set into the algorithms to verify the performance using statistical methods [73]. The data set used in these experiments is generally from website topology-zoo [104] or rocketfuel [105]. In the assumption of many schemes, the SDN switch allocate traffic according to customer designed algorithms, while the traditional switch performs data packet forwarding according to traditional routing protocols, which is difficult to implement in real-life network. This is why most research works adopt the static simulation approach for verification.

2) Real-Life Traffic-Based Performance Measurement: Some works do not assume the arbitrary allocation of traffic in SDN switches, hence use real-time traffic in their experiments [106]. In the experiments, Mininet [103] and SDN controller are the main components. In the Mininet, legacy switches are materialized as host nodes that run the Quagga software [49], while Open vSwitches act as SDN switches. The SDN controller is able to parse and respond to OSPF hello packets received and forwarded by the OvS switches [96] (through adequate OpenFlow rules installed in the SDN switches) and ensure the correct functioning of the adjacent OSPF routers.

VI. USE CASES IN HYBRID SDN NETWORKS

Vissicchio *et al.* [1] define four kinds of hybrid SDN models (i.e., TB hSDN, SB hSDN, CB hSDN and Integrated hSDN) according to the network service and usage scenario, each model has its potential transition use case and long-term design use case. As the extension to the deployment methods and optimization strategies that mentioned above, in this section, we summarize and analyze several representative applications and business cases related to hybrid SDN networks. Fig. 10. gives an overview of this section.

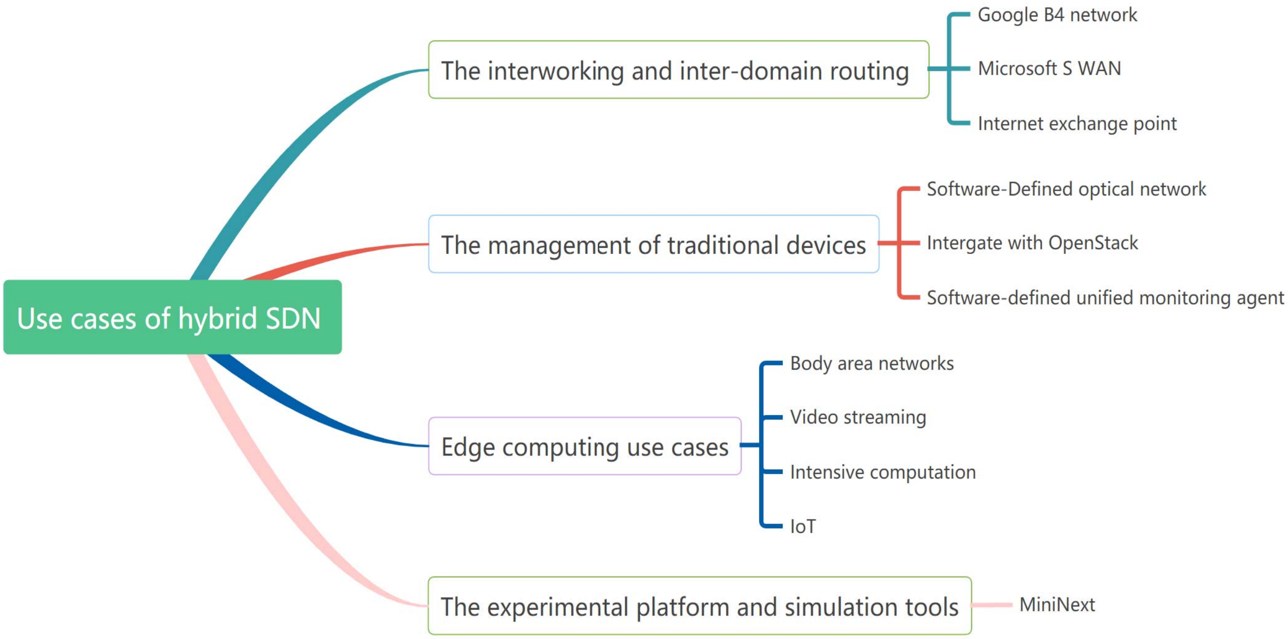


Fig. 10. Use cases in Hybrid SDN Networks.

A. The Interworking and Inter-Domain Routing

In the early stage of SDN, the most famous application should be Google's B4 network [3] and Microsoft's S WAN [4]. Specifically, Google selected SDN to transform the interconnected WAN network (G-scale Network) between the data centers, and has been fully transitioned to the OpenFlow network. Before the full deployment, B4 network experienced two hybrid deployment process. The first phase was completed in the spring of 2010, the OpenFlow switch was added to the network. However, the OpenFlow switch is the same as any other non-OpenFlow devices in the network, except that the network protocol is managed by the controller. The whole network was still like the traditional network. The second phase was completed by 2011, Google increased the size of the network and began to use the controller to manage the network, allowing the network to evolve to the SDN network. In fact, Google promoted the idea of agile development and put both SDN and traditional routing systems running in parallel. SDN has a higher priority than traditional routing, so that SDN can be gradually deployed to various data centers, allowing more and more traffic to be transferred from traditional routes to SDN framework. At the same time, if there is a problem with the SDN, the SDN framework in B4 can be turned off and return to the traditional routing approach. In this phase, SDN switches are allowed to interact with traditional routing protocols, and Google implements corresponding routing protocols as an SDN application. Even if the pure SDN framework is fully implemented, the data center is inevitably required to exchange information with external traditional networks.

Internet Exchange Point (IXP) is defined as a physical network access point where different ISP can connect their network and exchange BGP routes through this point. The SDX-L3 [47] is the abbreviation for Software Defined Internet Exchange Point - Layer 3. The traditional physical IXP routing and traffic forwarding is based on the IP prefix. The SDX supports rules that match multiple header fields, and each AS is allowed to adopt remote control over the traffic. Besides, the SDX integrates the virtual switch abstraction to ensure that ASes are not able to see or control interdomain routing outside of their purview. Furthermore, there are some optimization models that update rules as soon as a policy or BGP route changes. SDX makes IXP more flexible and reliable.

B. The Management of Traditional Devices

The advantages of the HAL are especially reflected in the Software-Defined Optical Network (SDON) [6]. Due to the presence of the abstraction layer, it is possible to make the optical switches that do not support the SDN framework become those switches that can be controlled by the SDN controller [107], [108]. OpenFlow in Europe Linking Infrastructure and Applications (OFELIA) is one of the important applications in optical networks. In order to allow optical switches to get rid of the shortcomings of not supporting OpenFlow, researchers assign agents for these optical switches. Similar to the deployment of HAL, the agent can make a connection with the controller, collect the state of the underlying layer and connect to the optical switches in the traditional way. Based on the OpenFlow agent, Hybrid GMPLS-OpenFlow [109] solution and Pure Extended OpenFlow [110] solution are presented in OFELIA. In the first solution, the standardized GMPLS control plane is reused to offload the OpenFlow controller from the complexity of circuit switching. In the second solution, the OpenFlow agent is used to exchange the configuration with the network elements and SDN controllers through the management interface and the extended OpenFlow protocol, respectively.

Hybrid SDN could be further utilized in the VNF and cloud computing service. HybNET [31] is suitable for virtualized network management service. If the operator needs to apply for a new VM, the configuration requirements along with the user information will be passed from the API to Hybnet. Researchers integrate HybNET with OpenStack. Specifically, it works in term with Neutron (the network service manager of OpenStack) to provide the hybrid network management function. Hybnet provides the tenants to modify, add, and delete virtual machines as well as achieve network isolation.

Choi *et al.* [111] implement a hybrid middlebox, named Software-defined Unified Monitoring Agent (SUMA). SUMA, as an intelligent switch-side inline middlebox, is located between OpenFlow switches and controllers. It provides management abstraction between SDN controllers, traditional NMS, and SDN switches by collecting traffic statistics in the background, monitoring network events, filtering and aggregating incoming packets. SUMA reduces the monitoring overhead of the controller, and the authors believe that it can be deployed as an important component of an efficient SDN deployment.

C. Edge Computing in Hybrid SDNs

Edge computing is a way to simplify traffic from IoT devices and provide real-time local data analysis. SDN concentrates the network intelligence at the controllers, thus avoiding edge devices performing complex network activities. Therefore, the control mechanism provided by SDNs can reduce the complexity of the edge computing architectures by bringing a novel approach to utilizing the available resources in a more efficient manner [112]. Hybrid SDN network can accelerate the process of the complexity reduction, because some edge computing service (i.e., video streaming, intensive computation) deployed in pure SDN can also be partially implemented in hybrid SDN network(cite). For example, a mobile user sends a service request to one of the cloudlets in the vicinity. Before the request is accomplished by the server, the user is authenticated to another network by changing its location. In hybrid SDN, the controller can track this movement with its ability to discover the topology and get the necessary information about the new location of the user, such as its recently assigned IP address. This allows service responses to be reached to the user by adding new flow rules to the switches on the path. During this entire process, the user is not aware of the operations occurring within the network, and the user experience is not interrupted.

D. The Experimental Platform and Simulation Tools

With a reliable a simulation platform, network operators can clone their network architecture into an emulated environment and then estimate the impact of changes in the network to its existing architecture. By installing Quagga [49] and running the corresponding routing protocol, operators could use a common PC simulation to support existing mainstream routing. Mininet [103] can simulate the network host, and support OpenFlow switches, controllers, links, suitable for simple network topology simulation. These two tools are the most commonly used simulation tools in traditional networks and SDN networks. MiniNext [66] combines Quagga [49] with Mininet to implement a tool that can build a simple hybrid SDN network simulation platform. This platform can simulate a hybrid network that includes traditional IGP and SDN technologies. In this way, even a laptop can simulate a hybrid SDN network with hundreds of nodes, and these nodes can be interconnected with real-world networks. Unlike the large-scale hybrid network simulation tool [54] that specializes in the creation of network graph, the measurement of convergence time and loss rates, and the visualization of routing changes, MiniNext focuses on simulating the operating environment and provides low-level APIs.

VII. CONCLUSION AND DISCUSSION

The purpose of this survey is to provide researchers who are active in or interested in the field of hybrid SDN issues with an overview of the state-of-the-art, including hybrid SDN models, deployment solutions, optimization strategies and different use cases. We pay special attention to control plane and data plane deployment solutions as well as optimization strategies that aim to improve the network performance and ensure consistency. We also summarize some common issues in the hybrid SDN network, including underlying protocols, topology discovery, and hybrid SDN models.

According to our understanding, there are some gray areas which need to be identified and properly addressed before hybrid SDN networks are commercially deployed, which constitute several future research directions, as presented below.

1) Security Issues in Deployment Solutions: Security is not considered as part of the initial design while it must be built as part of the long-term hybrid SDN network architecture. Researchers could pay more attention to migrate some security solutions to the hybrid SDN networks. For example, SDN data plane configuration checkers such as Anteater [113] and Header Space Analysis [114] can be extended to hybrid SDN networks, increasing the scalability of these deployment solutions.

2) Optimization Strategies for Real-Life Traffic: As for optimization strategies, most of the current optimization algorithms do not fully consider the real-life situations. For example, the selection of traffic data sets does not take into account the impact of different time periods (peak and off-peak hours), and the assumption of switch deployment costs is too simple. In the future, researchers can investigate complex budget models, add special constraints (i.e., some switches must migrate or can not migrate), adapt to multiple network environments, and adopt the neural network, Markov Approximation algorithms or data mining methods to solve complicated and real-time optimization problems [80].

Based on these optimization strategies, operators can provide some practical services in the future. For example, CDN is used to bring the content closer to the user to decrease latency and maximize throughput. During this process, CDN providers have to optimize the assignment of end-users and surrogates according to the load information in the network [115]. The hybrid SDN framework can accelerate the assignment for CDN providers by utilizing its global view and programmable interfaces of the whole network. This is because in hybrid SDN, the controller can redirect some traffic between client and server, that is, redirect the traffic of a given flow to an arbitrary node.

Using the traffic engineering strategies, researchers can study how the SDN controller guides more traffic and ensures load balancing. The CDN hybrid SDN service might need to focus on the correctness of TCP socket migration and the effective transfer of HTTP session in the complex network environments.

3) Virtualization Services in Hybrid SDN Networks: NFV is the best platform to reflect the commercial value of SDN, and there have been many NFV projects on pure SDN. For example, Flowvisor [116], a network slice service in pure SDN, enhances transparency and isolation between network slices by checking, rewriting, and managing OpenFlow messages as they pass through virtual network slices. Obviously, the AS domain controlled by hybrid SDN frameworks can also be part of these network slices. However, the combination of multiple switches may result in the degradation of network performance. Therefore, in addition to considering the unifying of different switches in the data plane, and the implementation of the extended Flowvisor controller in the control plane, researchers also need to consider the impact of isolation services on overall network flexibility.

4) Practical Simulation Tools in Hybrid SDN Networks: We discuss some simulation tools [53], [54] in Section III, but these simulation tools can only be used to evaluate network convergence time. We also summarize simulation solutions for optimization strategies in Section IV, however, these solutions are only suitable for specific experimental environments. Section V describes a common simulation tool for hybrid SDN (e.g., MiniNext [66]). However, it can only be used to test the network connectivity. In order to obtain typical performance metrics to verify if a hybrid SDN is successfully deployed, more practical simulation and emulation tools are expected. The challenges that need to be solved include i) these tools should obtain full control over all traditional switches such as SDN switches in the Mininet, and ii) evaluation criteria among deployment strategies are different, thus, new simulation tools need to provide reliable and unified data sets to adapt to various experimental environments.

REFERENCES

[1] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 70–75, 2014.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015.

[3] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[4] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.

[5] S. Natarajan, A. Ramaiah, and M. Mathen, "A software defined cloud-gateway automation system using OpenFlow," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, San Francisco, CA, USA, 2013, pp. 219–226.

[6] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2738–2786, Oct. 2016.

[7] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[8] S. Luo *et al.*, "Toward high available SDN/NFV-based virtual network service in multi-providers scenario," in *Proc. World Autom. Congr.*, Rio Grande, Puerto Rico, 2016, pp. 1–5.

[9] T. Benson, A. Akella, and D. A. Maltz, "Mining policies from enterprise network configuration," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, Chicago, IL, USA, 2009, pp. 136–142.

[10] T. Nelson, A. D. Ferguson, D. Yu, R. Fonseca, and S. Krishnamurthi, "Exodus: Toward automatic migration of enterprise network configurations to SDNs," in *Proc. ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, p. 13.

[11] M. Caria, A. Jukan, and M. Hoffmann, "A performance study of network migration to SDN-enabled traffic engineering," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Atlanta, GA, USA, 2013, pp. 1391–1396.

[12] B. Gu, M. Dong, C. Zhang, Z. Liu, and Y. Tanaka, "Real-time pricing for on-demand bandwidth reservation in SDN-enabled networks," in *Proc. IEEE Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, 2017, pp. 696–699.

[13] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[14] X. Jia, Y. Jiang, and Z. Guo, "Incremental switch deployment for hybrid software-defined networks," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, Dubai, UAE, 2016, pp. 571–574.

[15] B. Kar, E. H.-K. Wu, and Y.-D. Lin, "The budgeted maximum coverage problem in partially deployed software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 394–406, Sep. 2016.

[16] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[17] S. Rathee, Y. Sinha, and K. Haribabu, "A survey: Hybrid SDN," *J. Netw. Comput. Appl.*, vol. 100, pp. 35–55, Dec. 2017.

[18] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2017.

[19] G. Tarnaras, E. Haleplidis, and S. Denazis, "SDN and ForCES based optimal network topology discovery," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, London, U.K., 2015, pp. 1–6.

[20] R. Horvath, D. Nedbal, and M. Stieninger, "A literature review on challenges and effects of software defined networking," *Procedia Comput. Sci.*, vol. 64, pp. 552–561, Sep. 2015.

[21] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, Sydney, NSW, Australia, 2014, pp. 1–6.

[22] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[23] D. J. Casey and B. E. Mullins, "SDN shim: Controlling legacy devices," in *Proc. IEEE Local Comput. Netw. (LCN)*, 2015, pp. 169–172.

[24] (2018). *Framework for SDN: Scope and Requirements*. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Framework_for_SDN_-Scope_and_Requirements.pdf

[25] (2017). *Internet Engineering Task Force*. [Online]. Available: https://www.ietf.org/

[26] (2018). *SDN Component Stack and Hybrid Introduction Models*. [Online]. Available: https://www.necam.com/docs/?id=c2e5a040-cdf1-4fd7-b63e-6eea4b1f7a7b

[27] (2018). *HP SDN Hybrid Network Architecture*. [Online]. Available: https://community.arubanetworks.com/aruba/attachments/aruba/SDN/43/1/4AA5-6738ENW.PDF

[28] (2018). *SDN and Dual Control Planes on One Switch*. [Online]. Available: http://e.huawei.com/en/related-page/solutions/technical/agile-networking/technical-articles/Issue%205

[29] C. Jin, C. Lumezanu, Q. Xu, Z.-L. Zhang, and G. Jiang, "Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks," in *Proc. ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, p. 20.

[30] C. Jin *et al.*, "Magneto: Unified fine-grained path control in legacy and OpenFlow hybrid networks," in *Proc. ACM Symp. SDN Res.*, 2017, pp. 75–87.

[31] H. Lu *et al.*, "HybNET: Network manager for a hybrid network infrastructure," in *Proc. ACM Ind. Track 13th ACM/IFIP/USENIX Int. Middleware Conf.*, Beijing, China, 2013, p. 6.

[32] D. Levin *et al.*, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *Proc. USENIX Annu. Tech. Conf.*, Philadelphia, PA, USA, 2014, pp. 333–345.

[33] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving SDN," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 85–90.

[34] R. Enns, M. Bjorklund, and J. Schoenwaelder, "Network configuration protocol (NETCONF)," Internet Eng. Task Force, Fremont, CA, USA, RFC 6241, 2011.

[35] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Load balancing for multiple traffic matrices using SDN hybrid routing," in *Proc. IEEE Int. Conf. High Perform. Switching Routing (HPSR)*, Vancouver, BC, Canada, 2014, pp. 44–49.

[36] M. Caria, A. Jukan, and M. Hoffmann, "SDN partitioning: A centralized control plane for distributed routing protocols," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 381–393, Sep. 2016.

[37] M. Caria and A. Jukan, "The perfect match: Optical bypass and SDN partitioning," in *Proc. IEEE 16th Int. Conf. High Perform. Switching Routing (HPSR)*, Budapest, Hungary, 2015, pp. 1–6.

[38] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of SDN in hybrid enterprise and ISP networks," in *Proc. ACM Symp. SDN Res.*, Santa Clara, CA, USA, 2016, p. 1.

[39] (2013). *SAL Plugin for Supporting Generic Commodity Ethernet Switch*. [Online]. Available: https://wiki.opendaylight.org/view/Project_Proposals:SNMP4SDN

[40] D. Parniewicz *et al.*, "Design and implementation of an OpenFlow hardware abstraction layer," in *Proc. ACM SIGCOMM Workshop Distrib. Cloud Comput.*, Chicago, IL, USA, 2014, pp. 71–76.

[41] S. Vissicchio, L. Vanbever, and J. Rexford, "Sweet little lies: Fake topologies for flexible routing," in *Proc. ACM Workshop Hot Topics Netw.*, Los Angeles, CA, USA, 2014, p. 3.

[42] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 43–56, 2015.

[43] R. Hand and E. Keller, "ClosedFlow: OpenFlow-like control over proprietary devices," in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 7–12.

[44] M. R. Nascimento *et al.*, "Virtual routers as a service: The RouteFlow approach leveraging software-defined networks," in *Proc. ACM Int. Conf. Future Internet Technol.*, Seoul, South Korea, 2011, pp. 34–37.

[45] A. Vidal, F. Verdi, E. L. Fernandes, C. E. Rothenberg, and M. R. Salvador, "Building upon RouteFlow: A SDN development experience," in *Proc. 31st Simpósio Brasileiro De Redes De Computadores (SBRC)*, 2013, pp. 879–892.

[46] P. Lin *et al.*, "Seamless interworking of SDN and IP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 475–476, 2013.

[47] A. Gupta *et al.*, "SDX: A software defined Internet exchange," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, 2015.

[48] P. Lin, J. Bi, and H. Hu, "BTSDN: BGP-based transition for the existing networks to SDN," *Wireless Pers. Commun.*, vol. 86, no. 4, pp. 1829–1843, 2016.

[49] (2018). *Quagga Routing Suite*. [Online]. Available: http://www.nongnu.org/quagga/

[50] P. Thai and J. C. de Oliveira, "Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks," in *Proc. IEEE Comput. Commun. Netw. (ICCCN)*, Nassau, The Bahamas, 2013, pp. 1–6.

[51] J. P. Stringer, Q. Fu, C. Lorier, R. Nelson, and C. E. Rothenberg, "Cardigan: Deploying a distributed routing fabric," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 169–170.

[52] M. Caesar *et al.*, "Design and implementation of a routing control platform," in *Proc. USENIX Assoc. Conf. Symp. Netw. Syst. Design Implement.*, vol. 2, 2005, pp. 15–28.

[53] P. Sermpezis and X. Dimitropoulos. (2017). *Can SDN Accelerate BGP Convergence? A Performance Analysis of Inter-Domain Routing Centralization*. [Online]. Available: https://arxiv.org/pdf/1702.00188.pdf

[54] A. Gämperli, V. Kotronis, and X. Dimitropoulos, "Evaluating the effect of centralization on routing convergence on a hybrid BGP-SDN emulation framework," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 44. Chicago, IL, USA, 2014, pp. 369–370.

[55] (2017). *The POX Controller*. [Online]. Available: https://github.com/noxrepo/pox

[56] (2018). *The BGP Swiss Army Knife of Networking*. [Online]. Available: https://github.com/Exa-Networks/exabgp

[57] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 303–324, 1st Quart., 2017.

[58] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *Proc. IEEE Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Gold Coast, QLD, Australia, 2014, pp. 1–8.

[59] (2018). *HP Virtual Application Networks SDN Controller*. [Online]. Available: http://h17007.www1.hp.com/docs/networking/solutions/sdn/4AA4-8807ENW.PDF

[60] B. Belter *et al.*, "Hardware abstraction layer as an SDN-enabler for non-OpenFlow network equipment," in *Proc. IEEE Eur. Workshop Softw. Defined Netw. (EWSDN)*, London, U.K., 2014, pp. 117–118.

[61] F. Farias, J. Salvatti, P. Victor, and A. Abelém, "Integrating legacy forwarding environment to OpenFlow/SDN control plane," in *Proc. 15th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Hiroshima, Japan, 2013, pp. 1–3.

[62] M. Szalay *et al.*, "HARMLESS: Cost-effective transitioning to SDN," in *Proc. ACM SIGCOMM Posters Demos*, 2017, pp. 91–93.

[63] C. Sieber, R. Durner, and W. Kellerer, "How fast can you reconfigure your partially deployed SDN network?" in *Proc. IFIP Netw.*, Stockholm, Sweden, 2017, pp. 1–9.

[64] C. Sieber, A. Blenk, A. Basta, D. Hock, and W. Kellerer, "Towards a programmable management plane for SDN and legacy networks," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Seoul, South Korea, 2016, pp. 319–327.

[65] T. Feng, J. Bi, P. Xiao, and X. Zheng, "Hybrid SDN architecture to integrate with legacy control and management plane: An experiences-based study," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, 2015, pp. 754–757.

[66] B. Schlinker *et al.*, "Try before you buy: SDN emulation with (real) interdomain routing," in *Proc. ONS*, Santa Clara, CA, USA, 2014.

[67] S. Hares and R. White, "Software-defined networks and the interface to the routing system (I2RS)," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 84–88, Jul./Aug. 2013.

[68] R. Katiyar, P. Pawar, A. Gupta, and K. Kataoka, "Auto-configuration of SDN switches in SDN/non-SDN hybrid network," in *Proc. ACM Asian Internet Eng. Conf.*, Bangkok, Thailand, 2015, pp. 48–53.

[69] S. Salsano *et al.*, "OSHI—Open source hybrid IP/SDN networking (and its emulation on Mininet and on distributed SDN testbeds)," in *Proc. IEEE Eur. Workshop Softw. Defined Netw. (EWSDN)*, London, U.K., 2014, pp. 13–18.

[70] P. Sharma *et al.*, "Enhancing network management frameworks with SDN-like control," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, 2013, pp. 688–691.

[71] H. Xu, H. Huang, S. Chen, and G. Zhao, "Scalable software-defined networking through hybrid switching," in *Proc. IEEE Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

[72] K. Poularakis, Q. Qin, E. Nahum, M. Rio, and L. Tassiulas. (2017). *Bringing SDN to the Mobile Edge*. [Online]. Available: https://arxiv.org/pdf/1706.06001.pdf

[73] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2211–2219.

[74] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP)*, Raleigh, NC, USA, 2014, pp. 563–568.

[75] W. Wang, W. He, and J. Su, "Boosting the benefits of hybrid SDN," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, 2017, pp. 2165–2170.

[76] J. He and W. Song, "Achieving near-optimal traffic engineering in hybrid software defined networks," in *Proc. IEEE IFIP Netw. Conf. (IFIP Netw.)*, Toulouse, France, 2015, pp. 1–9.

[77] W. Wang, W. He, and J. Su, "Enhancing the effectiveness of traffic engineering in hybrid SDN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, 2017, pp. 1–6.

[78] M. Caria, T. Das, A. Jukan, and M. Hoffmann, "Divide and conquer: Partitioning OSPF networks with SDN," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, 2015, pp. 467–474.

[79] Y. Guo *et al.*, "Incremental deployment for traffic engineering in hybrid SDN network," in *Proc. IEEE Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nanjing, China, 2015, pp. 1–8.

[80] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in hybrid SDN networks with multiple traffic matrices," *Comput. Netw.*, vol. 126, pp. 187–199, Oct. 2017.

[81] H. Xu *et al.*, "Incremental deployment and throughput maximization routing for a hybrid SDN," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1861–1875, Jun. 2017.

[82] H. Xu, J. Fan, J. Wu, C. Qiao, and L. Huang, "Joint deployment and routing in hybrid SDNs," in *Proc. IEEE/ACM Int. Symp. Qual. Service (IWQoS)*, 2017, pp. 1–10.

[83] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu, "Saving energy in partially deployed software defined networks," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1578–1592, May 2016.

[84] Y. Wei, X. Zhang, L. Xie, and S. Leng, "Energy-aware traffic engineering in hybrid SDN/IP backbone networks," *J. Commun. Netw.*, vol. 18, no. 4, pp. 559–566, Aug. 2016.

[85] X. Jia, Y. Jiang, Z. Guo, G. Shen, and L. Wang, "Intelligent path control for energy-saving in hybrid SDN networks," *Comput. Netw.*, vol. 131, pp. 65–76, Feb. 2018.

[86] Y. Hu *et al.*, "Maximizing network utilization in hybrid software-defined networks," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, 2015, pp. 1–6.

[87] C. Ren *et al.*, "Enhancing traffic engineering performance and flow manageability in hybrid SDN," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Washington, DC, USA, 2016, pp. 1–7.

[88] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing SDN upgrades in ISP networks," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, 2017, pp. 1–9.

[89] S. Vissicchio, L. Vanbever, L. Cittadini, G. Xie, and O. Bonaventure, "Safe updates of hybrid SDN networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1649–1662, Jun. 2017.

[90] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, "Safe update of hybrid SDN networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1649–1662, Jun. 2017.

[91] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, and L. Vanbever, "On the co-existence of distributed and centralized routing control-planes," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, 2015, pp. 469–477.

[92] R. Amin, N. Shah, B. Shah, and O. Alfandi, "Auto-configuration of ACL policy in case of topology change in hybrid SDN," *IEEE Access*, vol. 4, pp. 9437–9450, 2016.

[93] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid SDN networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, 2015, pp. 1086–1094.

[94] M. Markovitch and S. Schmid, "SHEAR: A highly available and flexible network architecture marrying distributed and logically centralized control planes," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP)*, San Francisco, CA, USA, 2015, pp. 78–89.

[95] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental SDN deployment," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Messina, Italy, 2016, pp. 397–402.

[96] (2016). *OpenvSwitch*. [Online]. Available: http://openvswitch.org/

[97] H. Huang, S. Guo, J. Wu, and J. Li, "Green datapath for TCAM-based software-defined networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 194–201, Nov. 2016.

[98] T. Lukovszki, M. Rost, and S. Schmid, "It's a match!: Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, 2016.

[99] Z. Zheng, L. X. Cai, M. Dong, X. Shen, and H. V. Poor, "Constrained energy-aware AP placement with rate adaptation in WLAN mesh networks," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, Kathmandu, Nepal, 2011, pp. 1–5.

[100] G. Luo, Z. Qian, M. Dong, K. Ota, and S. Lu, "Improving performance by network-aware virtual machine clustering and consolidation," *J. Supercomput.*, 2017, pp. 1–19.

[101] C. Ge, Z. Sun, N. Wang, K. Xu, and J. Wu, "Energy management in cross-domain content delivery networks: A theoretical perspective," *IEEE Trans. Netw. Service Manag.*, vol. 11, no. 3, pp. 264–277, Sep. 2014.

[102] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 161–174, 2002.

[103] (2018). *Mininet: An Instant Virtual Network on Your Laptop*. [Online]. Available: http://mininet.org/

[104] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[105] (2003). *Rocketfuel-Topology*. [Online]. Available: http://research.cs.washington.edu/networking/rocketfuel/

[106] N. Huin *et al.*, "Bringing energy aware routing closer to reality with SDN hybrid networks," *IEEE Trans. Green Commun. Netw.*, to be published.

[107] R. Alvizu and G. Maier, "Can open flow make transport networks smarter and dynamic? An overview on transport SDN," in *Proc. IEEE Int. Conf. Smart Commun. Netw. Technol. (SaCoNeT)*, Vilanova i la Geltrú, Spain, 2014, pp. 1–6.

[108] M. Channegowda, R. Nejabati, and D. Simeonidou, "Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations [invited]," *J. Opt. Commun. Netw.*, vol. 5, no. 10, pp. A274–A282, 2013.

[109] S. Azodolmolky *et al.*, "Integrated OpenFlow—GMPLS control plane: An overlay model for software defined packet over optical networks," *Opt. Exp.*, vol. 19, no. 26, pp. B421–B428, 2011.

[110] M. Channegowda *et al.*, "Experimental evaluation of extended OpenFlow deployment for high-performance optical networks," in *Proc. IEEE Eur. Conf. Exhibit. Opt. Commun. (ECOC)*, Amsterdam, The Netherlands, 2012, pp. 1–3.

[111] T. Choi, S. Song, H. Park, S. Yoon, and S. Yang, "SUMA: Software-defined unified monitoring agent for SDN," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, 2014, pp. 1–5.

[112] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

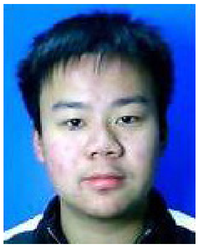
[113] H. Mai *et al.*, "Debugging the data plane with anteater," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 41. Toronto, ON, Canada, 2011, pp. 290–301.

[114] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proc. NSDI*, vol. 12. San Jose, CA, USA, 2012, pp. 113–126.

[115] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An SDN-based CDN/ISP collaboration architecture for managing high-volume flows," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 48–60, Mar. 2015.

[116] R. Sherwood *et al.* (2009). *FlowVisor: A Network Virtualization Layer*. [Online]. Available: https://pdfs.semanticscholar.org/64f3/a81fff495ac336dccdd63136d451852eb1c9.pdf

Xinli Huang (M'18) received the Ph.D. degree in computer science from Shanghai Jiao Tong University in 2007. He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University. His research interests are in the areas of internetworking, software defined networking, cloud data center networks, future networks, and network security.



Shang Cheng received the B.S. degree in computer science from Harbin Engineering University, China, in 2015. He is currently pursuing the M.E. degree with the Department of Computer Science and Technology, East China Normal University. His research interests include traffic engineering, routing optimization, and software defined networking.

Kun Cao is currently pursuing the Ph.D. degree in computer science with East China Normal University. His current research interests are in the areas of high performance computing, multiprocessor systems-on-chip, and cyber physical systems.

Peijin Cong received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016, where she is currently pursuing the master's degree. Her current research interest is in the areas of power management in mobile devices and edge computing.

Tongquan Wei (M'11) received the Ph.D. degree in electrical engineering from Michigan Technological University in 2009. He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University. His research interests are in the areas of Internet of Things, edge computing, cloud computing, and design automation of intelligent and CPS systems. He has been serving as a Regional Editor for the *Journal of Circuits, Systems, and Computers* since 2012.

Shiyan Hu (SM'10) received the Ph.D. degree in computer engineering from Texas A&M University in 2008. He is an Associate Professor with Michigan Tech, and he was a Visiting Associate Professor with Stanford University from 2015 to 2016. His research interests include cyber-physical systems (CPS), CPS security, data analytics, and computer-aided design of VLSI circuits. He has published over 100 refereed papers in the above areas. He was a recipient of the National Science Foundation CAREER Award. He is the Chair for IEEE Technical Committee on Cyber-Physical Systems. He is the Editor-in-Chief of *IET Cyber-Physical Systems: Theory & Applications*. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. He is also a Guest Editor for a number of IEEE/ACM Journals such as PROCEEDINGS OF THE IEEE and the IEEE TRANSACTIONS ON COMPUTERS. He has held chair positions in numerous IEEE/ACM conferences. He is an ACM Distinguished Speaker, an IEEE Systems Council Distinguished Lecturer, and an IEEE Computer Society Distinguished Visitor. He is a fellow of IET.