Part II

Waypoint Enforcement in Hybrid SDN

Chapter 6

Hybrid SDN Survey

6.1 Introduction

Modern-day communication networks which are based on distributed control and network transport protocols, pose a lot of complex operational issues [68] [165] [166] [167]. Although the traditional [2] IP networks have been adopted widely, they are *complex and hard to manage* [168]. A number of issues such as policy enforcement on a wide variety of boxes (devices), robustness and fault tolerance, application and user aware routing, and complex traffic isolation make the management of traditional networks cumbersome. To add fuel to the fire, forwarding and control mechanisms exist within the same network device and are tightly interwoven, known as *vertical integration* [12]. Each device has vendor-specific properties. The deployment of another device in the network may lead to incompatible interfaces or require reconfiguration of the existing devices, which often becomes buggy if done manually. This vertical integration [12] and *vendor specificity* [169] hinders flexibility and hampers innovation in network infrastructure evolution. The transition from IPv4 to IPv6 is a testimony to this. This inertia of current networks (lack of configuration automation methods and response mechanisms) leads to a lot of

- Sandhya, Yash Sinha, K Haribabu. *A survey: Hybrid SDN*, Journal of Network and Computer Applications pp. 35-55 Elsevier, vol. 100, 2017.

[2]We use the terms legacy and traditional interchangeably

management efforts where changes are frequent [170].

The emerging "Software Defined Networks" paradigm has the potential to address these issues by simplifying the present state of network architecture [12]. SDN provides a new architecture that stands on five pillars: (i) control and data plane separation, (ii) central control and manageability, (iii) network programmability, (iv) flexibility in terms of flow abstraction with agility, and (v) vendor neutrality. Firstly, the routing devices in SDN paradigm are only meant to forward the packets, whereas the network control logic is centralized. This breaks the vertical integration. Secondly, the centralized controller can configure, manage, secure, and optimize network resources dynamically [6]. Further, a centralized controller enables network programmability, thereby network management can be automated via programs. Using the global view of the network and flow abstraction, the network traffic can be dynamically adjusted. SDN advocates open standards and vendor neutrality, which further boosts innovation and adoption. The controller uses southbound API (Application Programming Interface) for communication with the forwarding devices. OpenFlow [18] and P4 [39] are the most popular southbound APIs in SDN.

Despite having many benefits, SDN faces multi-dimensional challenges viz., technical, financial, and business, which discourage full deployment of SDN. Technically, there are many questions to be answered, such as how scalable, resilient, and robust the centralized controller can be without becoming a single point of failure. As the network grows, it may require more than one SDN controller. In terms of security, the SDN controller is an attractive target for attackers[171]. In the absence of a secure and robust controller, the attackers have the opportunity to change the behavior of the underlying network. A great focus on SDN security is required to make it acceptable [172]. Among financial challenges, the concern is a huge investment for SDN deployment, which most of the organizations cannot afford in one go. Another challenge is related to business. SDN is not standardized yet; the SDN software and hardware may not be thoroughly tested. Thus, it is not easy to build confidence for adoption of a new paradigm over the well-running, time-tested traditional paradigm.

Hybrid SDN [173] provides an environment where both legacy and SDN devices can work together. In this chapter, we refer to the switches that can talk to the SDN controller via OpenFlow protocol as SDN switches or SDN nodes. These devices are forwarding elements responsible for processing and forwarding packets in the data plane. On the other hand, other network elements such as routers and switches that do not support OpenFlow and are part of the traditional network are referred to as legacy nodes or SDN-incompatible devices. Legacy nodes contain both the control plane as well as the data plane and operate on the legacy protocols. Hybrid SDN can extract the benefits of both while mitigating their drawbacks. Many of the solutions developed traditionally can address challenges in SDN. For example, increased latency due to communication delay between switches and the controller can be mitigated by deferring decision making partially to the legacy control plane, improving the reaction time in an emergency such as failures, etc. On the other hand, the controller has a global view of the network, and it can tweak the weights of local routing mechanisms to enhance the traditional ways of managing the network. Thus, an incremental deployment strategy can be developed to meet the financial and business needs of various organizations.

In this chapter, we discuss the Hybrid SDN paradigm with its benefits and limitations. We propose multiple models for SDN deployment. We categorize the works from literature based on the proposed models and discuss their advantages and disadvantages. We also provide the context in which the term *Hybrid* is used in this chapter and what we mean by *Hybrid SDN architecture*. We further discuss the implementations and the evaluations done in the literature in Hybrid SDN networks. We throw light on Hybrid SDN specific issues and how various researchers have addressed them.

6.2 The Hybrid SDN Paradigm

6.2.1 What is Hybrid SDN?

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.*

6.2.2 Benefits SDN Promises

The specific advantages of each Hybrid SDN models are discussed in Section 6.3. Here we present an overview.

1. Hybrid SDN provides economic and business benefits. It requires a huge budget to replace all the existing legacy devices. Hybrid SDN provides the flexibility to deploy the SDN device depending on the budget of an organisation. Incremental deployment of SDN devices helps in building the confidence of network operators.

2. Hybrid SDN enables SDN-specific features like centralized control on the underlying devices without full SDN deployment [174, 175, 176, 177]. Internet Service Provider (ISP) requires millions of forwarding rules, and SDN switches can only support tens of thousands of forwarding rules. Thus, ISP can use SDN switches at the distribution layer and legacy devices at the access layer to handle millions of forwarding rules while reaping SDN benefits.

3. There are areas where a combination of centralized and decentralized mechanisms function well. For example, update or installation of a large number of rules in the devices centrally could be a problem in pure SDN due to clogging of the control channel and limited processing capacity of the controller. Using both central and distributed control in the same environment, we can overcome this problem. If the controller's communication is congested and the controller is unable to respond, then the critical traffic can be forwarded through the legacy network. Thus, Hybrid SDN provides the advantage of falling back to the legacy network.

4. Based on whether an organization wants to incentivize and accommodate the premium users initially or enhance telecom billing, there can be different proportions in which the traffic can be controlled either by SDN or non-SDN paradigm. This can be tuned based on the specific needs of the organization.

6.2.3 Limitations

Different implementation approaches have different drawbacks, but in general, we list the following disadvantages of hybridisation.

1. Management of heterogeneous control plane is difficult. Due to multiple control plane interactions, the network update procedures may not be safe [178]. Anomalies may occur in the reconfiguration process. For example, due to control-plane conflicts, an update might trigger forwarding inconsistencies. This can further lead to forwarding loops and black holes. Establishing a communication session between the SDN controller and the legacy switch is challenging. Several alternatives such as middleware, protocol translation, software upgrade, etc., exist. This increases the controller-switch communication delay.

2. There is added complexity in the data plane. For example, realizing heterogeneity with a heterogeneity adaptation layer such as a middleware to translate legacy protocols back and forth increases latency and processing time. Further, introducing a middleware requires fixing security issues in the middleware, replication for failure guarantees, extra processing power etc. If the reconfiguration of legacy devices is done via manual intervention, that could lead to an inappropriate or error-prone deployment of a Hybrid SDN system, similar to legacy networks.

3. There are specific issues such as controller scalability, fault tolerance, traffic engineering, etc. The controller can only control a limited number of devices. Thus, incremental deployment of SDN nodes requires more SDN controllers, and this can increase the latency. Traffic Engineering (TE) optimization is more challenging in Hybrid SDN as not all nodes support flow abstraction, OpenFlow, packet matching and packet filtering mechanisms etc. Solutions using ACLs (Access Control List) or static routes provide limited functionalities[179].

6.2.4 Probable Contextomy

Within the SDN context, the term "Hybrid" has been used to indicate multiple meanings. We have used the term "Hybrid" to convey the Hybrid SDN paradigm, not the following meanings.

6.2.4.1 Dual Stack Mode (Hybrid Switch)

A dual stack switch is a device that can support both OpenFlow protocol and legacy protocols within the same network device [180]. In this context, the term "Hybrid" refers to a single switch with OpenFlow on one VLAN (virtual LAN) and legacy forwarding on another VLAN. These devices are known as "Hybrid devices" with "Hybrid Port Mode" e.g., Brocade MLXe Series switch. However, in this chapter, "Hybrid" is not construed in this manner.

6.2.4.2 OpenFlow Hybrid Mode

In general, the controller is responsible for the forwarding decisions. In OpenFlow-hybrid mode supported by OpenFlow 1.3 [32], the controller can forward the packets to the NORMAL port, delegating the forwarding decision to the traditional control plane.

6.3 Hybrid SDN Models

This section proposes different models for SDN deployment based on the architecture and component, and functionality. For each model, we discuss its benefits, and limitations.

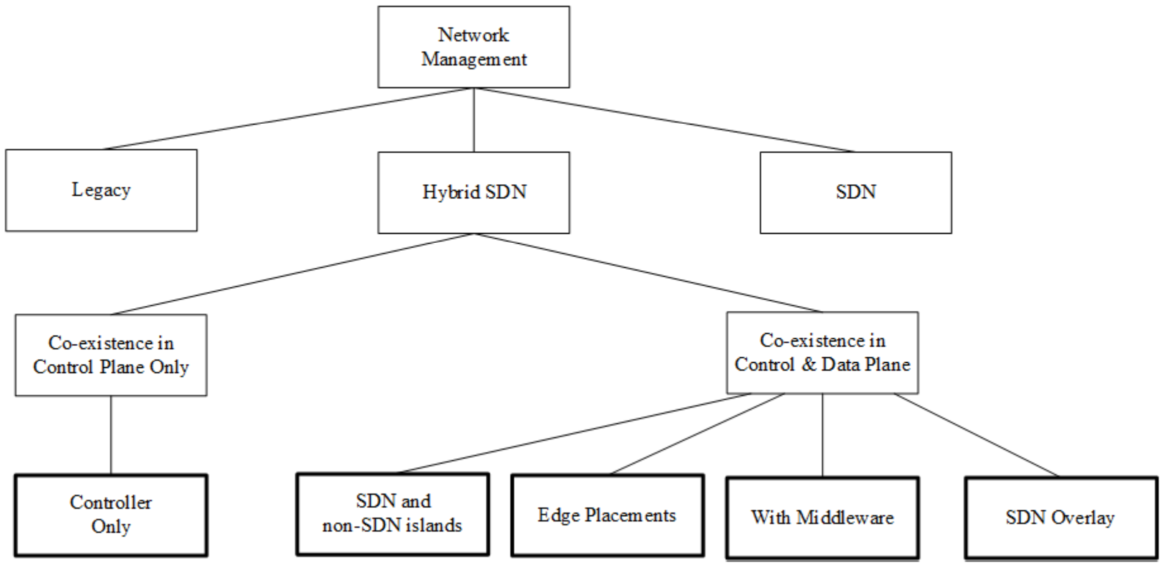

Figure 6.1: Classification based on architecture and components

Figure 6.1, shows the categorisation based on the co-existence in the control plane or data plane. The mere co-existence in the data plane without any co-existence in the control plane is hardly advantageous.

6.3.1 Co-existence in Control Plane Only

In literature, there are some works that attempt to introduce a centralized system within traditional IP networks for better configuration and management [181] [182] [183] [19]. Taking motivation from these works, we propose a model, controller only, for Hybrid SDN networks. In [173], the authors refer to this approach as Integrated Hybrid SDN.

As shown in Figure 6.2 (a), the model introduces an SDN controller, and the rest of the network remains unchanged (i.e., all devices in the underlying network are legacy). The idea is to enhance the distributed control plane with inputs from the centralized controller.

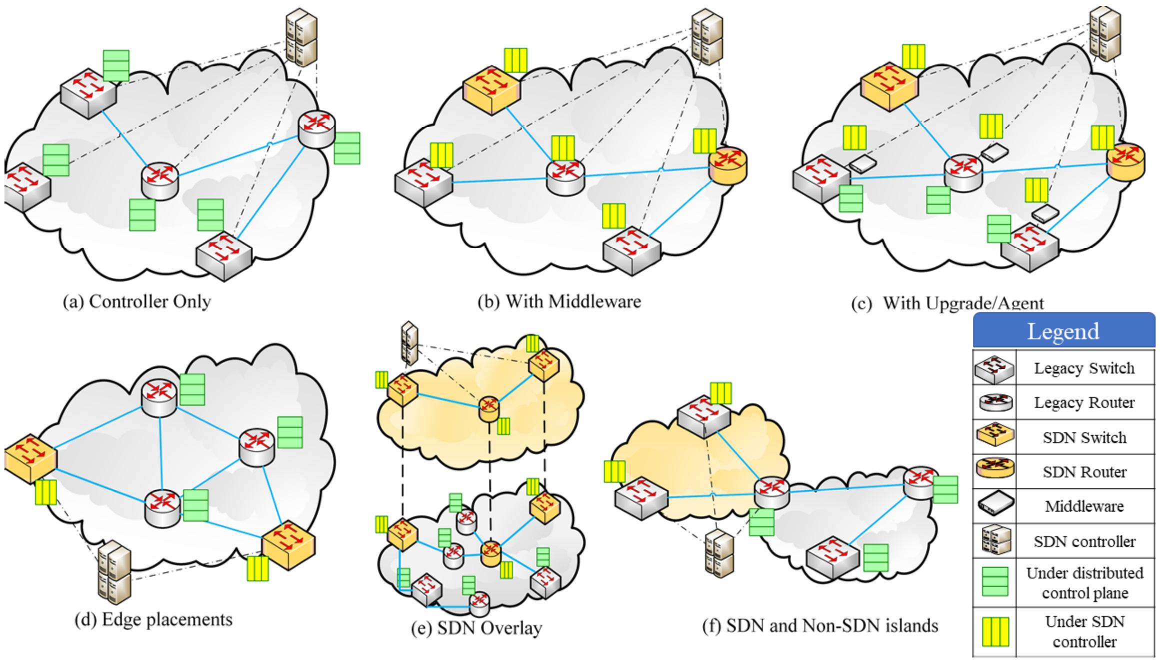

Figure 6.2: Various Hybrid SDN Models classified based on architecture and components

As shown in Figure 6.2 (a), the SDN controller may interact with the legacy nodes directly or deceptively. In direct communication, the controller itself understands the protocol, and no protocol translation is involved. For example, in the Routing Control Platform [181], the controller acts as an IGP (Interior Gateway Protocol) and BGP (Border Gateway Protocol) node to communicate back and forth. In deceptive communication, the controller interacts with the legacy nodes with the help of protocol translation, e.g., it injects packets of legacy protocols in the network to provide SDN like benefits. In [184], the authors propose a method to provide SDN control over the legacy network by introducing fake nodes in the network. In a substantial form, the SDN controller may take responsibility for all the network services while using the legacy protocols as interfaces to communicate with data plane devices.

Advantages.

1. The network administrator can use this model to check whether and to what extent there is a need for SDN deployment in their networks while building self-assurance about reliability and understanding of its operation.
2. The model does not require any SDN data plane hardware and consequently incurs the least investment.
3. Further, this deployment neither entails any change or disruption of previous services running in the network. Relying on legacy protocols, the administrators may choose this model as the first phase in an incremental deployment strategy.
4. No interoperability issues between heterogeneous devices need to be addressed.
5. Further, it is robust as in case of controller failure, it can always fall back to legacy distributed control.

Disadvantages.

1. This approach incorporates minimal SDN benefits to the traditional network.
2. A legacy protocol interface is much more complex than an SDN protocol like OpenFlow and still, may not be complete in terms of functionality compared to OpenFlow. Therefore, not all benefits of SDN can be realized [185].
3. Security is another issue in this approach because the centralized controller is exposed and prone to attacks. If the controller gets compromised, then fake control messages can enter the network.
4. Using this approach, we cannot modify packet fields, such as source/destination MAC (Media Access Control) addresses, UDP (User Datagram Protocol), IPv4, IPv6 fields. Which on the contrary, are modifiable in SDN enabled switches.

6.3.2 Co-existence in Control and Data Plane

As the name suggests, in this model legacy and SDN paradigm co-exists at both the planes. That is, a network consists of legacy devices, SDN switches, and SDN controller. However, we can further categorize this model as follows,

1. **SDN and non-SDN Islands:** To begin the transition, an organization may choose a small part of its network to upgrade to SDN, whereas the rest of the network continues to function as it is. This leads to the formation of islands. The network is partitioned into SDN and non-SDN islands, as shown in Figure 6.2 (b). The control in SDN islands is centralized, and control in the non-SDN islands is distributed.

   In B4 [58], an SD-WAN (SDN in a Wide Area Network) is designed and implemented for connecting Google's data centers across the planet. SDN is adopted in the backbone to maximize bandwidth utilization, whereas it connects to remote data centers, storage accesses with non-SDN protocols. Similarly, in [60], authors present SWAN, centrally controlling inter-data center network backbone with SDN, and the rest of the zones are managed traditionally.

   Advantages.

   (a) Naturally fits a transition strategy in which SDN is introduced on a per-region basis. This can be an incentive to begin the transition with a small region, build confidence, and move to the next.

   (b) Regions can be increased as the technology matures, expertise is cultivated, and a new budget is available.

   (c) Many enterprise networks are already separated into domains due to past mergers/acquisitions, hierarchical separation for management, specific technical need, etc. So in these cases, the organization may opt for this model.

   (d) Failure of SDN deployment has an effect in the deployed region only. The rest of the network has no effect.

   (e) Introduction of new mechanisms to communicate in between regions is easy. For example, solely by modifying the SDN control logic, safer [186] and flexible [187] interconnection mechanisms can be deployed.

Disadvantages.

(a) Deployment cost is high, as all devices in a given region are replaced with SDN nodes.

(b) Cross-compatibility between islands may limit network functionality.

(c) Long periods of service disruption are possible in deployment phases. The network remains in a complex state until a full transition is complete.

2. **Edge Placements:** In [188], the authors propose that intelligence at the edge is going to be the trend as SDN adoption spreads. In this approach, SDN nodes are placed at the edge of the network, as shown in Figure 6.2 (c). The SDN controller controls the forwarding decision at the edge SDN nodes. For the controller, the topology is limited to the SDN devices only, i.e., it abstracts the existing network of legacy devices. In this approach, the SDN paradigm is responsible for managing traffic going in and out of the network. The traffic in the core of the network is routed by the legacy devices only.

   Advantages.

   (a) Investment occurs only for the edge devices, while many SDN benefits can be realized.

   (b) Scalability of the SDN controller is a function of a number of SDN switches. The load on the controller is less as compared to full deployment.

   (c) Separation of forwarding for edge and internal traffic simplifies the management and helps in the independent evolution of fabric and edge [70].

   (d) Deployment of SDN switches at edges provides rich services such as network security, isolation, and mobility [70].

   Disadvantages.

   (a) SDN control is limited to traffic that is passing through edge devices only.

   (b) The traffic generated within the network remains unmonitored.

   (c) There may be some conflicts between two paradigms while routing the same packets.

3. **With Middleware:** To provide SDN control over legacy devices, middleware software can be built that helps in communication between the SDN controller and legacy devices. As shown in Figure 6.2 (d), the SDN controller uses middleware to interact with legacy nodes, whereas it controls the SDN switches in the standard way.

   Advantages.

   (a) This model works even if there are no SDN nodes.

   (b) Partial programmability and automation of tasks in the network can be achieved.

   (c) Based on the need of the organization, only specific protocols can be parsed for SDN control. Thus, it reduces the load on the SDN controller.

   (d) Fallback to legacy mechanisms makes it robust in case the SDN controller fails.

   Disadvantages.

   (a) Not all legacy protocols can be translated by the middleware. A set of protocols has to be chosen carefully.

   (b) This approach has a scalability issue. If load on the middleware increases, then it can become a bottleneck.

   (c) It does not provide all SDN benefits as it is not possible to collect network statistics from legacy nodes directly by issuing request messages from the controller.

   (d) Protocol translation at the controller incurs load on the controller and latency in communication.

4. **SDN Overlay:** As shown in Figure 6.2 (e), an SDN network is built as an overlay on top of the legacy network. Some of the chosen devices in the network are replaced with SDN devices in order to facilitate waypoint enforcement, better traffic management, etc. The controller can only see the SDN overlay as the actual network. The overlay is composed of logical links, which can consist of one or more legacy devices.

Big Switch Networks' Big Virtual Switch [189] is one example of an SDN overlay application. Big Virtual Switch makes it possible to run a software-defined network on top of any infrastructure. In [68] [190] [191], only an abstracted view of the underlying topology is exposed to the SDN controller through which SDN benefits are realised.

Advantages.

(a) Performance of both SDN and non-SDN network can be analyzed in the same network, and hence next deployment phases can be made more efficient using the results.

(b) Specific services that require virtualization can be implemented with ease.

Disadvantages.

(a) Virtual and physical networks are separate entities with different attributes. This makes the network complex.

(b) Gateways between the overlay network and nodes on the physical network may need to pass high traffic volume. For example, the frontiers can be a bottleneck for communication between SCTs in Panopticon [68].

6.4 Comparative Architectural Analysis

In this section, we provide a comparative analysis of all the models discussed in the previous section. Table 6.1 outlines the summary of the comparison. This analysis helps the network operator to choose between different models, depending on her requirements.

6.4.1 Controller Only

This model provides the advantage of introducing a central control, and progressively the control is shifted to the controller with gradual maturity of the technology and the operators' expertise. Among all the Hybrid SDN models, this model requires the deployment of a controller only. Hence, practically free.

Disruption in the existing services is momentary because no physical reconfiguration of the legacy nodes is required. The controller enhances Network-wide forwarding decisions as it has a global view of the network, but the capability of the controller limits it in terms of protocol translation (e.g., not all packet matches are supported) and pushing them as flow entries (e.g. number of route maps is limited). This provides partial programmability for the traffic. Policy expressiveness is limited by protocol translation (e.g., not all policies expressed by the administrator can be converted to flow entries), and enforcement is restricted (e.g., fine-grained routes may not be possible). The model can scale as long as the controller has sufficient computing power and depends on the distributed routing protocols, which the controller can translate. In case the controller fails, the network can continue to function with legacy protocols as it used to work without the controller.

6.4.2 SDN and Non-SDN islands

This model naturally fits a transition strategy in which SDN is introduced on a per-region basis. This can be an incentive to begin the transition with a small region, build confidence and expertise, and move to the next. The cost of investment depends on the number of SDN nodes deployed in an SDN island. Disruption of services occurs during SDN island creation and may prolong until the legacy devices are replaced with SDN nodes. Reconfiguration of all the islands and installation of interconnection mechanisms or gateway may take time. Network programmability is enabled only within an island. Policy expressiveness and enforcement are enabled in SDN island, although this may be extended to other traffic if it passes via an SDN island. Scalability is a function of the size of the islands.

6.4.3 Edge Placements

An architecture that provides more intelligent routing at the perimeter of the network than the centralized hub-and-spoke model is capable of optimizing traffic flow without compromising security or quality of service and driving up cost. In the edge placements model, the SDN devices are replaced or deployed at the edge of the network [70]. The investment is proportional to the number of SDN nodes introduced and can be expensive. Partial disruption occurs for the edge traffic during the deployment and configuration of SDN nodes. The traffic going through the edges is controlled by the centralized controller. Separation of forwarding for edge and internal traffic simplifies the network management. It also enables independent evolution of fabric and edge. Centralized policy expression and enforcement are possible only for edge traffic.

6.4.4 With Middleware

The primary incentive for this model is to enable SDN control on the existing data plane with minimum cost. If the legacy nodes can be tweaked to communicate with the controller either via software upgrade or introducing an agent, we need to introduce only an SDN controller in the network. This fosters maximum reuse of the existing hardware and thus minimizes investment. Disruption of services may occur for a short span of time due to deployment of SDN nodes, software upgrade, or deployment of middlewares; while reconfiguration is being done. Depending on the degree of protocol translation, the legacy nodes can be automatically configured and programmed. An organization may choose a specific set of protocol features only for translation to strike a trade-off between performance and protocol benefits. Consequently, legacy nodes can closely act like SDN nodes with the help of middleware. Scalability is limited as processing at middleware might increase the latency in communication. The model offers robustness in terms of fallback to legacy protocols.

6.4.5 SDN Overlay

With the incentive of leveraging maximum SDN benefits, the model aims to build an SDN overlay on the top of the legacy network. Investment is dependent on the design and implementation of the overlay. During the transition, this model faces prolonged disruption of services as it requires network reconstruction. Programmability, and policy enforcement and enhancement are fully enabled for the overlay network. Scalability is a function of network design and load on the controller. Failure recovery is provided by the SDN controller as well as the legacy protocols.

Table 6.1: Comparison of different models in terms of hybrid SDN pillars

	SDN components	**Incentive**	**Investment**	**Transition smoothness**	**Ease of automation**	**Policy enforcement**	**Scalability and robustness**
Legacy Network	None	None	None if, pre-deployed	Disruption free	Complex configurations	Firewall, VLANs, ACLs etc.	Legacy protocols
Controller only	Controller	Building initial confidence, central control	SDN Controller, is Practically free	No physical re-configuration, momentary disruption	Limited by protocol translation and number of flow entries	Limited by protocol translation, not fine-grained	Limited by controller load, fall-back to legacy
SDN and non-SDN Islands	Controller with SDN islands	Region based transition	Proportional to size of SDN islands	Possible disruption in replaced island	Programmable for SDN island only	Traffic within and through SDN island	Paradigm specific,
Edge Placements	Controller, edge placements of the SDN nodes	Intelligence at edge	Proportional to number of devices replaced at edge, expensive	Partial disruption for edge traffic due to reconfiguration	Enabled for edge and not for internal traffic	Enabled for edge traffic only	Scales gracefully, robust redundant nodes
With Middleware	Controller, SDN nodes and middleware	Minimal investment and hardware re-use	SDN Controller, SDN nodes and middleware	Partial disruption due to strategic placement of SDN nodes, software upgrade and agent	Programmable for SDN nodes and partially enabled for others	Enabled for all traffic, limited by middleware support	Limited due to protocol translation, fallback to legacy
SDN Overlay	Controller, SDN nodes	Maximum benefits in hybrid	SDN Controller, SDN nodes	High disruption due to network reconstruction	Overlay and SDN nodes only	Not enabled for unmonitored, underlay network	By network design, controller load, fallback on legacy underlay
Pure SDN	Controller, SDN nodes	SDN benefits	Maximum	Full disruption and reconstruction	Fully programmable	Network-wide traffic	Concern [192]

6.5 Implementation Approaches of Hybrid SDN Models

In this section, we discuss the approaches taken by the researchers to implement the Hybrid SDN models. An overview of Hybrid SDN implementation approaches is provided in Table 6.2. We highlight the major contributions, specific deployment techniques used, and limitations of works in the literature.

6.5.1 Controller Only

This section discusses the different approaches used to achieve centralized control in the legacy network without deploying SDN nodes. For example, in [181], the authors accomplish centralized control over the legacy devices by establishing an internal border gateway protocol (iBGP) session with each router in the network. In [181], the authors propose a centralized approach, "routing control platform" (RCP). It establishes an iBGP session with all the routers in the topology and uses a standard protocol to find a finely grained route to the destination on behalf of the router using the available routes and topology view [181]. RCP provides consistent assignment of routes for external traffic, which provides reliability. RCP reaction is fast for link failures in the network. It is similar to the SDN controller, but it is dealing with external traffic only. Using RCP for optimization of large ISPs (Internet Service Providers) could be difficult. In [184], Vanbever et al. propose the idea of achieving central control over distributed routing computation through fake nodes. In their next work [198], they propose a central controller called "Fibbing", which provides flexibility in network routing, like load balancing, traffic steering, and providing a backup path, by manipulating the input for traditional routing protocol. The manipulation is done by introducing fake nodes in the network by injecting fake LSAs. Fibbing takes the following as input, (i) path requirements from network operator (ii) network topology (iii) directed acyclic graph for each destination. Based on the path requirements, it injects fake LSAs in the network to introduce fake nodes in the network topology, announcing the reachability to a destination. The workflow of Fibbing is shown in Figure 6.3.

In their next work [197], they present the evaluation of Fibbing controller along three axes viz., (i) load on router, (ii) topology augmentation, and (iii) performance gain. (i)

Table 6.2: Summarized overview of hybrid SDN papers

Reference	Year	Addressed issue	Proposed solution	Required	Limitations
[193]	2015	Auto-configuration and adoption of new SDN switch in existing SDN/ traditional network	Provides different modules for configuration of intermediate switches/ routers and to locate the new SDN switch	DHCP/BOOTP discovery and offer message to carry the option number 222 to enable SDN configuration	-
[194]	2014	SDN like control over legacy devices	ClosedFlow, provides the SDN like control over legacy devices	Use ACL, route-map, and remote login	Buffering of packets at devices is not supported
[66]	2015	Providing control over legacy paths using OpenFlow in hybrid environment	Hybrid controller: Telekinesis, which modify the forwarding table of both legacy as well as SDN nodes using Open flow	POX OpenFlow controller with some additional components (path verification and path update)	It cannot update all paths with in the networks
[65]	2016	Traffic engineering and failure recovery in hybrid SDN	SDN deployment planner and TE module	Some of the legacy devices will be replaced by SDN	Not able to provide control for the flow, that does not traverse and SDN switch
[195]	2013	Traffic engineering in hybrid SDN	Solves the dynamic routing problem using approximation method	Deploy some SDN devices in the network	Non-SDN devices are controlled
[72]	2015	Cost-estimation and Fault tolerance for hybrid SDN	Proposed a heuristic algorithm for estimating the number of SDN devices required and method for load distribution	Depending on the topology, deploy a few SDN devices	Does not consider the case when no link failure in the network
[68]	2014	Policy enforcement in hybrid SDN	Proposed a way to achieve waypoint enforcement	Network is divided into cells and at the edges SDN nodes are placed	Cannot provide centralized control on $Non - SDN_c$ port's traffic
[196]	2014	Packet loss	Proposed IBSDN architecture to provide backup path in case of link failure	Needs an agent deployment over the network nodes	Failure recovery is completely dependent on IGP
[197]	2015	How to provide central control in distributed environment with minimum overhead	Proposed an architecture where the central controller participates in the distributed routing	No modification is required except some requirements are needed from network operator	It cannot provide port based routing
[69]	2016	How to distribute the load over the links in case of failures	Proposed an algorithm to find near optimal spare capacity link	Network partitioning with SDN nodes	-
[198]	2014	How to provide centralized control in traditional network	Introduces fake nodes in the network	Inclusion of fake nodes in the traditional network	It could be security vulnerable
[199]	2015	Congestion avoidance	DEFO controller	No modification is required	-
[200]	2014	Provide unified view of hybrid SDN network to controller	HAL architecture	No modification is required	-

Load on the router can be expressed in terms of CPU processing, memory usage, programmability for installation of forwarding entries and time taken in the convergence of routing protocol. Fibbing introduces very less CPU and memory overhead on routers even when a large number of fake nodes are injected into the network. The time taken by Fibbing to install thousands of entries in the network is approximately constant. Further, they show that the injection of fake nodes in the network does not have any visible impact on the time taken in the convergence of distributed routing protocols. (ii) For topology augmentation, they propose two augmentation algorithms, namely simple and merger. Simple algorithm introduces fake nodes for every destination. Whereas the merger algorithm works in phases. The first phase injects an excessive number of fake nodes and computes the upper bound and lower bound for their respective costs. The second phase merges the fake nodes based on the upper bound and the lower bound. They show that both algorithms augment the topology in time ranging from 0.5 ms to 8 ms. (iii) Fibbing controller injects the fake nodes on unused links, as a result of which the throughput gets doubled. In their recent work [201], Tilmans et al. assess the performance of the Fibbing controller for on-demand load balancing to enhance the video delivery. They illustrate that Fibbing provides better and fast load balancing in case of sudden congestion and provides a smooth video play to the end-users. The solution provided in [201] cannot manipulate the traffic based on ports because forwarding is done through IP address matching. Port based forwarding is possible with the help of middlebox, but this adds the cost of a middlebox. Fibbing can be vulnerable to security loopholes because any compromised router can send fake LSAs in the network. Fibbing is limited to destination-based routing.

In [199], Hartert et al. propose an architecture that consists of two layers, connectivity, and optimization layer, on top of the physical layer. The connectivity layer's responsibility is to provide default forwarding behavior to the underlying devices in the network. Whereas the optimization layer defines the forwarding rules with the help of the proposed "Declarative and Expressive Forward Optimizer" (DEFO) controller. The network operator can define certain goals in terms of (i) redistribution of load from heavily loaded links to less loaded links, (ii) traffic engineering, e.g., bringing link utilization under a certain threshold, (iii) enforcing constraints, e.g., forcing a traffic flow to pass through a firewall, etc. DEFO takes the goals defined by the network operator, and with the help of the optimization layer, it translates these goals into configurations. The forwarding rules generated by DEFO overwrites the forwarding rules generated by the connectivity layer. DEFO maximizes input utility (for example, link capacity, expected traffic matrix, etc.) for network optimization. In case of a controller failure, the forwarding rules are installed by the connectivity layer. Thus provides robustness.

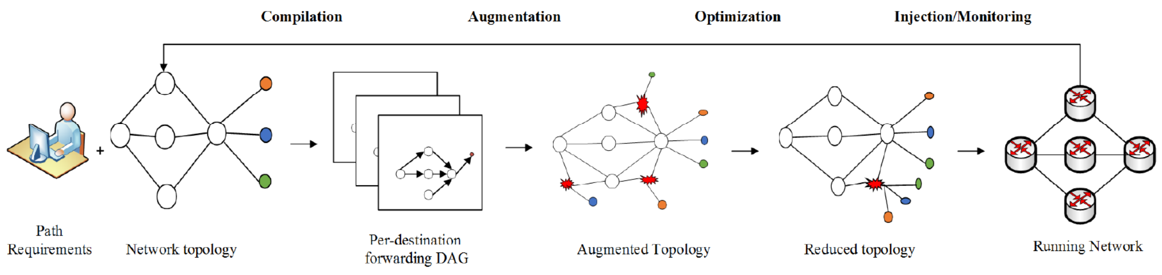

Figure 6.3: Fibbing Work-Flow

6.5.2 SDN and Non-SDN Islands

With partitioning of the network into regions that are managed separately by either of the paradigms, along with an optional mechanism to implement cross-region services; this model has largely been used to implement software defined WANs. In the concept of SD-WAN, the separation of control mechanism from its networking hardware leads to simplified management and operation. The *backbone* zone is SDN managed, whereas connection with the end-users, remote data centers, remote storage accesses, connected non-SDN WANs is supported based on existing distributed routing protocols of the legacy [58]. In SWAN [60], a logically centralized controller orchestrates all activity for the SDN controlled backbone. For each service, a broker aggregates demands from the hosts and apportions the allocated rate to them.

6.5.3 SDN Overlay

The main incentive behind SDN overlay model is to build an SDN network overlay on the top of the existing network irrespective of where the SDN nodes are placed. To build an SDN network overlay, the following approaches have been used by the researchers.

1. Network virtualization is the process of combining the software and hardware networking resources (e.g., switches, routers, etc.) to create a logical software-based view. The following works use network virtualization functionality to isolate the SDN and legacy network.

   In [68], Levin et al. give an architecture "panopticon" for incremental deployment of SDN nodes in a legacy network with minimum budget. In panopticon, the network is divided into cells, and these cells are connected through SDN switches. The switch ports are divided into SDN-controlled (SDN_c) ports and legacy ports. The SDN_c ports are controlled by the SDN controller and the logical view for the controller consists of SDN nodes that contain their physical ports and SDN_c ports. The traffic going from or to an SDN_c port is enforced to go through at least one SDN switch. They achieve this by assigning a VLAN ID to the path from an SDN_c port to an SDN switch. The proposed method is limited by the maximum number of VLAN IDs available. It cannot control the traffic going between two non-SDN_c ports. In [191], Lu et al. propose a controller called "HybNET" for Hybrid SDN network management. HybNET has a complete view of physical network topology. It provides an abstraction to the underlying network by offering a view of SDN nodes to the controller. The SDN nodes are connected by virtual links, where every virtual link may comprise multiple legacy nodes.

2. In [190], a single OSPF (Open Shortest Path First) domain is divided into sub-domains by deploying SDN nodes. They propose a network management module called "Hybrid Network Manager" (HNM). It runs on the top of the SDN controller, the information about the network topology and routing is forwarded to HNM using LSAs. During an initial phase, HNM does not alter any routing; it replies like an OSPF node. Once the HNM gets complete information about network topology, it provides optimal routing by altering the LSAs by changing the link weights. It does not affect the routing within the sub-domain.

   In their next work [71], the authors propose a Hybrid SDN/OSPF network control plane. The network is divided into sub-domains [190] and an optical bypass is set up between the SDN switches. The purpose of an optical bypass in between border nodes is to offload traffic that transits among sub-domains. Therefore, it is easy to cope with high traffic demands by over-provisioning link capacities. The authors claim that this solution is good enough, and full SDN migration may be skipped.

Caria et al. in their previous works [190], [71] use a brute force mechanism to partition the network into sub-domains by deploying SDN nodes. In their recent work [69], they propose an integer linear programming (ILP) module to partition the network into sub-domains. The ILP module is based on graph partitioning theory, according to which any node in the graph belongs to one and only one subgraph. ILP partitions the network by placing SDN nodes in such a way that their removal leaves the network disconnected. Further, they propose models for capacity planning, traffic engineering, and load balancing. In their recent work [202], they propose a heuristic algorithm for estimating spare capacity required to deal with link failures in the network. They also provide an analysis where spare capacity required in case of SDN partitioning scheme is less than legacy and other hybrid models.

6.5.4 Edge Placements

In [203], Mishra et al. propose a framework for policy implementations in the network similar to those supported through OpenFlow. The design exploits the immense availability of unused IP addresses within networks. SDN nodes are stationed at the edges, which map the incoming packets' destination IP addresses to unused IP addresses and enable customized routing through the legacy network. The legacy nodes forward the packets based on the destination IP address. The legacy network routes are controlled through static routes, while the SDN controller manages the entire network.

In [204], the authors propose a way to reduce congestion in the network using S-OSPF (Smart-OSPF) [205]. Hybridization is achieved using two VM (Virtual Machine) machines on the router, one VM for traditional devices like quagga and another VM for OVS (OpenvSwitch) switch. This hybrid router is deployed at the edge of the network for traffic distribution to reduce network congestion. The second VM (OVS switch) uses S-OSPF protocol to construct the forwarding table, whereas the first VM (quagga router) uses OSPF. These VMs are connected with a virtual link.

6.5.5 With Middleware

The following works are implementing different middlewares to provide translation between different paradigms. The working details of middleware in each work are as follows:

In [191], Lu et al. propose a HybNET controller to manage hybrid network infrastructure. HybNET works in two phases. In the first phase, it constructs the network topology and sets up RPC connection between the SDN controller and legacy nodes. In the second phase, the controller takes the network management request from the network operator and parses it. Based on the request, the HybNET controller computes the network operation and separates out the operations needed to be performed on legacy and SDN nodes. HybNET communicates the changes required in SDN nodes to the SDN controller via REST API calls. Further, these changes are communicated to SDN nodes via OpenFlow protocol. HybNET controller performs the change in legacy nodes via RPC callback functions. Any change in network infrastructure, like a change in physical topology, needs to be reported to HybNET controller.

Hand et al. propose a "ClosedFlow" model, which provides centralized control over the legacy devices by configuring the vendor-specific devices [194]. The step-by-step procedure involves enabling an in-band overlay control channel and remote access (Secure Shell, SSH, or telnet) to each switch for controlling flows, such as pushing new flow rules. OpenFlow's packet matching and apply actions are realized partially via ACLs and Route maps. Modification in the packet header, such as changing port number are not supported by ClosedFlow. The implementation of ClosedFlow is limited to the devices that support functionalities like ACLs, route maps, so it might not cover all types of flow entries as provided by OpenFlow.

In [200], Parniewicz et al. propose an architecture "Hardware Abstraction Layer " (HAL), to transform the legacy nodes into OpenFlow nodes. HAL provides abstraction by hiding the underlying network topology and vendor-specific features supported by each device from the SDN controller. This abstraction is provided by decoupling the management logic and hardware-specific logic of network devices. HAL is divided into sub-layers namely, Cross-Hardware Platform Layer (CHPL) and Hardware-Specific Layer (HSL) to achieve the aforementioned decoupling. The framework is shown in Figure 6.4. HSL collects the information about network topology and sends this information to CHPL. When a network packet comes, it is forwarded to the CHPL module and processed by the OpenFlow pipeline. The flow entries and packet related actions are forwarded back to HSL, which translates these action into device dependent syntax. In [206] Belter et al., the authors further show the implementation of HAL on "Alien hardware devices" [207], e.g., "EZappliance" and "DOCSIS" [185] alien devices.

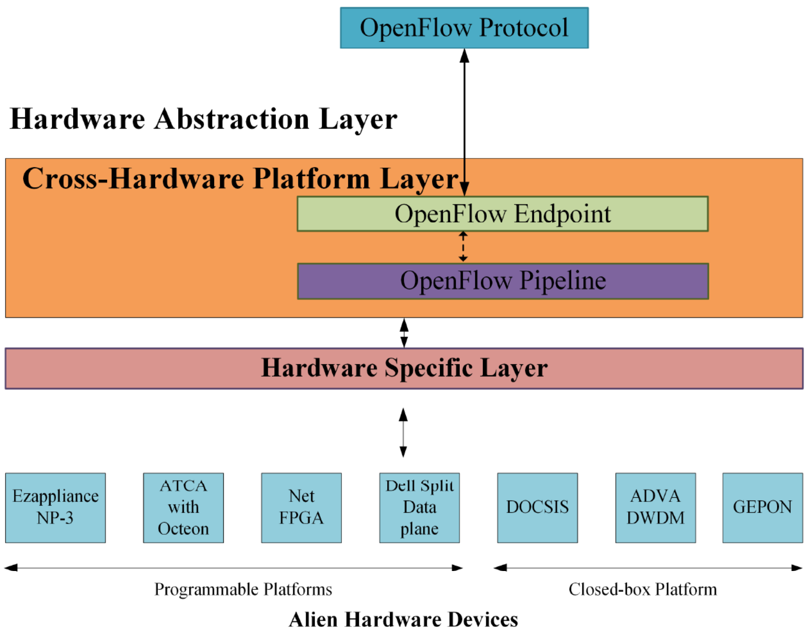

Figure 6.4: HAL implementation framework over Alien hardware devices

6.5.6 With Upgrade/Agent

In this section, we discuss various approaches proposed by researchers to upgrade the legacy devices to support centralized control.

In [196], Tilmans et al. propose an architecture "IGP-as-a-Backup" (IBSDN), which uses both distributed control as well as central control over the network. The SDN controller installs rules based on the primary policies provided by the network operator, which are called primary rules. In IBSDN, an agent is configured on each node, which collects the information about routing. It uses this information to build backup paths using distributed routing protocols. In a normal condition, the packets follow the rules installed by the SDN controller, and in the case of a link failure, the packet is forwarded according to the backup path provided by the agent. IBSDN provides fast re-routing in case of a link failure, which avoids packet loss. In [72], Chu et al. provide multiple backup paths by providing IP tunneling between each interface of router and SDN switch. Whenever a link fails, the packet is forwarded through an IP tunnel.

In [208], the authors propose an architecture called "OpenRouteFlow". OpenRouteFlow provides centralized control over legacy devices in Hybrid SDN by upgrading the legacy device's software. OpenRouteFlow architecture consists of "OpenRouteFlow" controller and "OpenRouter". The OpenRouter embeds an agent called "OpenRouteFlow" into the legacy devices. The OpenRouteFlow agent communicates the distributed routing information to the OpenRouteFlow controller. OpenRouteFlow agent receives the application oriented control instructions from the OpenRouteFlow controller, which are then transmitted to the legacy devices.

In [66], the authors propose "Telekinesis", a path control mechanism for legacy switches. The SDN controller instructs the SDN switch to send seed packets to the source switch with the *MAC address* of the destination host. This tricks the legacy switches to accept the SDN switch as the destination. Thus, the legacy switch sends the packet to the SDN switch (and not to the legacy switch, which actually possesses the true MAC address). The SDN switch receives the diverted packet and forwards the packet towards the destination. The MAC entry at source switch might revert back to the original MAC entry if the destination sends a packet to the source through the original legacy path. Thus, there exists path flipping problem. To overcome this problem, the SDN controller sends the seed packets quite frequently.

To overcome the problem of path flipping, in their next work, Jin et al. propose a unified controller called "Magneto" [67]. Magneto introduces the concept of magnet MAC addresses. These MAC addresses are mapped to hosts' actual IP addresses using gratuitous ARP (gARP) packets. SDN switches can send gARP packets with MAC address set as the magnet address of the destination host and send them to the source host. This causes a path to be set up between the source host and the SDN switch. Due to the gratuitous ARP packet, the source host has the destination host's magnet MAC address instead of the real MAC. Hence, a packet from the source host is routed towards the SDN switch. The SDN switch has necessary flow entries to fix the packet ethernet header by changing the destination MAC address from magnet address to real MAC address and the source MAC address to a magnet address. After making these changes in every packet, the SDN switch sends the packets towards the destination.

Table 6.3: Summarized overview of problems in Hybrid SDN and works done to solve them

S.No.	Plane	Problems and issues	Hybrid SDN
1.	Data plane	Controller-switch communication	[208][200][194][190]
2.		Traffic engineering	[184][190][65][179][195][72][209][202]
3.	Control plane	Control conflict	[200][176]
4.		Configuration	[191][194][210]
5.		Topology discovery	[65][210][181][211][212][213][180][212][214][215][216][217][218]
6.		Fault tolerance	[72][215][216][202]
7.		Scalability	[219][85]
8.	Management Plane	Network Monitoring	[65][202]
9.		The placement problem	[190][65][68]

6.6 Addressing Challenges in Hybrid SDN

In this section, we discuss the challenges that exist in Hybrid SDN. We categorize and discuss the works done by researchers for each challenge.

6.6.1 Topology Discovery

Topology discovery is a crucial component in SDN networks. The controller needs to know about the complete network topology to make optimal routing decisions. The controller also requires an up-to-date real-time topology discovery mechanism to detect events like link failures. In [213][212][214], the topology is discovered using LLDP (Link layer discovery protocol). The connection between the controller and an SDN switch is established using TCP (Transmission Control Protocol). Once the connection is established, the controller periodically (e.g., every 5 seconds) sends a command to SDN switches to flood LLDP. The controller also adds the necessary flow entry in the SDN switches to forward the LLDP packet to the controller as PacketIn. The LLDP packet has information about the switch, like port ID, TLV, and chassis ID. Using this information, the controller can infer the network topology.

The load on the controller is a critical aspect for an SDN network [76]. Since topology discovery is a service that typically runs continuously in the background on all the SDN controllers, it exerts considerable load. Moreover, the increased number of control messages used for topology discovery clogs switch to controller communication, especially in in-band networks, and load on the SDN switches also becomes high [212][214]. Thus, it is necessary to develop an optimal solution for topology discovery.

In hybrid networks, topology discovery becomes even more difficult due to a number of factors such as the co-existence of devices from different vendors, each device supports different protocols such as SNMP (Simple Network Management Protocol), OSPF, BGP, etc.

6.6.2 Configuration

Network configuration is one of the essential functions of network management. When a change happens in the network (e.g., topology change, node replacement, etc.), the SDN nodes are configured by the SDN controller using OpenFlow protocol. Since the legacy nodes cannot support OpenFlow protocol, they cannot be configured by the SDN controller. The legacy devices are configured manually, which can lead to configuration errors. In Hybrid SDN, the network administrator may have to use vendor-specific network configuration and management tools. The controller can use CLI, such as telnet, to configure the legacy nodes, but the support is limited. This issue is further magnified by multiple software versions, vendors, and little support for protocol translation at the SDN controller.

Lu et al. propose HybNET [191], a framework to automate the network management in Hybrid SDN networks. HybNET provides centralized control to the network operator by hiding the dissonance between legacy and SDN nodes. It provides a common configuration mechanism for both legacy and SDN nodes. ClosedFlow [194] provides an idea of making current networks centrally controllable by configuring each router interface. But, the implementation of ClosedFlow is limited to the devices that support functionality like

ACLs, Route maps.

In [193], Katiyar et al. propose a method to automate the configuration of SDN or Non-SDN devices to provide seamless service. The aim is to minimize the risk of errors in the manual configuration and reduce the operational cost. The authors propose two components, locator and configurator. The locator identifies a new SDN switch. The new SDN switch is referred as AutoConfClient (ACC), which acts as a DHCP client and sends DHCP discover message (with SDN option 222 added in DHCP option field) to AutoConf Server (ACS). The ACS calls an Intermediate Switch Configurator (ISC) component to configure intermediate SDN and non-SDN switches to provide connectivity between the newly added SDN switch and the SDN controller.

6.6.3 The Placement Problem

For Hybrid SDN deployment, we need to choose a subset of legacy nodes (number and location) to be replaced by SDN nodes based on budget and resource constraints, traffic matrix, network topology, and performance benefits. This decision making can be viewed as an optimization problem with objectives of maximum link utilisation, minimal disruption, maximum benefit to budget ratio, etc. However, this problem is difficult to solve. Therefore, many heuristics such as node degree, egress traffic volume, link weights have been studied by the researchers as criteria for replacement.

In [65], Hong et al. formulate the problem of SDN deployment in the legacy network. The formulation comes up with a bilinear term of unknowns, and solving the unknowns is NP-complete. Further, they provide heuristics for SDN node placement. The first heuristic picks up a legacy node with the highest degree in the network topology graph. The second heuristic uses a K-shortest path algorithm to find the path between each source and destination pair and then select the nodes that occur in most paths. In the third heuristic, the legacy nodes are selected for the replacement with higher traffic volume. In [190], Caria et al. propose a method for SDN device deployment in an OSPF network. The existing network is partitioned into sub-domains by strategic placement of SDN nodes. The placement is done such that the removal of SDN nodes partitions the network into disconnected components.

In [68], Levin et al. propose two heuristics for the placement of SDN nodes, namely VOL and DEG. VOL takes the volume of traffic passed through a switch as selection criteria for legacy device replacement. In contrast, DEG replaces the legacy switch with a higher degree in the topology graph.

6.6.4 Conflicts in Hybrid Control Planes

Hybrid networks are difficult to manage compared to legacy or pure SDN networks. Any update in the hybrid network can trigger forwarding inconsistencies, which may lead to disruption in traffic engineering policies and routing policies like bypassing a firewall or can lead to the formation of forwarding loops. This requires a conflict resolution mechanism.

Fundamentally there can be two approaches for conflict resolution. A straightforward way is to let the control planes interact with each other and resolve conflicts with a mutual understanding based on mechanisms such as protocol translation. For example, a controller may parse and inject packets of the legacy protocol to mutually understand, assist, and avoid possible routing conflicts. Another approach is to let the control planes manage separate entities like services, traffic classes, etc. For example, the different paradigms may extend different services (say DNS and routing) or control separate groups within the same service (separate traffic classes while routing or DHCP for hosts in different regions). Some of the solutions proposed in the literature are as follows.

In [176], Vissicchio et al. propose an algorithm, Generic Path Inconsistency Avoider (GPIA), to avoid inconsistencies in the forwarding entries. GPIA computes a sequence of nodes that can be configured without creating inconsistency in the network. GPIA iteratively creates a set of nodes for each destination that can be configured without any inconsistency in the network. The algorithm is dependent on recursion tree generation in the backtracking phase and is limited to the co-existence of SDN and IGP only.

In [200], Parniewicz et al. introduce "Hardware Abstraction Layer" (HAL) architecture that provides compatibility between current versions of OpenFlow protocol and network devices (both legacy and SDN). New packets in the network are forwarded to HAL, which parses these packets, provides a device-dependent configuration, and installs the flow entries in the underlying heterogeneous devices.

6.6.5 Controller-Switch Communication

In the SDN network, a centralized controller takes the forwarding decisions using the complete topology view. Whereas, in legacy nodes, the forwarding decisions are taken based on local information. The SDN nodes are controlled by the SDN controller using OpenFlow protocol, whereas distributed routing protocols like OSPF control the legacy nodes. In Hybrid SDN, enabling communication between the SDN controller and legacy nodes is not trivial. The communication between the SDN controller and legacy nodes can be achieved with the help of a translator module, which can exist as a plugin in the SDN controller, like SNMP4SDN [220]. This translation can be full, or it can be limited. There is a trade-off between performance gain and a number of features enabled by the controller.

Parniewicz et al. propose HAL [200], which provides an abstraction to the real network that consists of legacy and SDN nodes. HAL takes the network packets, processes them through the OpenFlow pipeline to apply the changes, and translates these changes into commands specific to the platform of underlying physical devices in the network. In ClosedFlow [194], the SDN controller uses SSH or telnet to install the flow entries in the forwarding table of legacy switches. Switches use remote logging to communicate the adjacency changes to the controller.

In [190], the authors propose a "Hybrid Network Manager" (HNM) module to provide traffic engineering in a Hybrid SDN network. The SDN nodes behave like traditional OSPF devices in the initialization phase. Once the HNM [190] module gets all the information about a network, the traffic engineering module in HNM computes the optimal route and sends these routes using tuned LSA through the SDN controller to SDN nodes. Further, SDN nodes distribute these routes in their sub-domains(a partition of the network) by flooding.

6.6.6 Scalability

With the incremental deployment of SDN nodes, the overhead on the SDN controller increases. To overcome this issue, more SDN controllers can be deployed in the network to distribute the load. Distributed controllers have some limitations. Firstly, this can stretch the path of a packet. Secondly, re-configuration and mapping between the SDN controller and devices are required. In [85] Dixit et al. propose an ElstiCon architecture, where the controllers initially operate at a pre-defined load window, and as the load changes over time, ElstiCon dynamically shifts the workload among the controllers. This is done by moving some of the SDN nodes from a heavily loaded controller to a lightly loaded controller. In [219], the authors propose a hierarchical architecture for the control plane to provide scalability in SDN networks.

6.6.7 Traffic Engineering

The main goal of traffic engineering is to optimize network performance to facilitate the reliability of network operations [221]. This can be achieved by making the network fault-tolerant, congestion-free by balancing the load on the links, etc. In Hybrid SDN, it is difficult to provide traffic engineering, as the legacy devices are not fully controlled by the SDN controller.

Vanbever et al. propose an architecture that takes physical topology and path requirements as input and produces an augmented topology. The augmented topology considers all path requirements, which are needed to provide load balancing in order to avoid congestion [184]. In [65], the authors propose a traffic engineering module for load balancing. Whenever a new flow comes, the traffic engineering module routes it on the least loaded path. The SDN controller uses the meter table feature of OpenFlow 1.3 [32] to retrieve the link-load dynamically. However, this is possible only if the flow traverses at least one SDN node.

In [190], the authors propose a module called "Hybrid Network Manager" (HNM). Its primary function is to gather information about the network viz., topology, traffic in the network, the position of SDN nodes, and routing. In HNM, the *Traffic Engineering Engine* module provides the optimal routing based on the information collected by HNM. The traffic engineering engine module is aware of the partitioning of the network. It provides load balancing using optimal sub-domain routing and computes the OSPF link metric accordingly. The metrics are then flooded as LSAs into the individual sub-domains. The computed routes are forwarded to the SDN controller, which is then forwarded to SDN nodes. In [179], the authors provide a formulation of the traffic engineering problem by considering the two-hybrid modes, namely barrier mode, and hybrid mode. In barrier mode, the SDN traffic and legacy traffic are routed in separate capacity spaces, whereas in hybrid mode, the link capacity is shared by both SDN and legacy traffic.

In [209], the authors propose an algorithm named "SOTE" (SDN/OSPF Traffic Engineering) to explore traffic engineering in a Hybrid SDN network. The goal of their work is to minimize the maximum link utilization. They run the SOTE algorithm and change the weights of the links in each iteration. After obtaining the weight settings, they construct a directed acyclic graph (DAG) by choosing a node and find the shortest path to all other nodes with respect to the chosen one. After the construction of DAG, the flows are split at an SDN node by adding the outgoing link from the SDN node to the DAG. If a loop is formed by adding the link, that link is removed. In [72], the authors provide fault tolerance by redirecting the traffic to an SDN switch in case of a link failure. It provides a backup IP tunnel through an SDN switch for all the destinations from a router that is affected by a link failure.

6.7 Summary

This chapter provides comprehensive survey of state-of-the-art in Hybrid SDN. We use different classification models to categorise the existing works and provide a detailed comparative architectural analysis of each model. We also discussed how various researchers addressed the challenges in Hybrid SDN.