Journal of Network and Computer Applications 100 (2017) 35–55

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Review

A survey: Hybrid SDN

Sandhya*, Yash Sinha, K. Haribabu

BITS, Pilani, Department of Computer Science and Information Systems, Pilani Campus, India

ARTICLE INFO

Keywords:
Hybrid SDN
Incremental deployment
Software-defined networking
OpenFlow
Network controller
Network Operating Systems
Network Hypervisor
Software-defined environments

ABSTRACT

A full deployment of Software Defined Networking (SDN) paradigm poses multi-dimensional challenges viz., technical, financial and business challenges. Technical challenges of scalability, fault tolerance, centralization guarantees exist. Financial challenges of budget constraints, non-availability of phased transition model exist. Business challenges like acceptability, building confidence among network operators etc. exist. Therefore, a direct and sudden transition from legacy networks to pure SDN seems unlikely. A hybrid deployment of SDN can be one of the plausible intermediate paths, primarily because it provides an environment where both legacy and SDN nodes can work together. Thus, an incremental deployment strategy can be developed. Further, hybrid SDN can enforce the benefits of both the traditional networks and SDN paradigm. Hybrid SDN deployment has many advantages including adaptability to budget constraints, central programmability of the network, fallback to time-tested legacy mechanisms and so on. But there are challenges specific to hybrid models, like added complexity of running multiple paradigms together, realizing cooperation between control planes, etc. We envision that more research work is needed to maximize the benefits and limit the drawbacks.

In this paper, we present a comprehensive survey of hybrid SDN models, techniques, inter-paradigm coexistence and interaction mechanisms. Firstly, we delineate an overview of hybrid SDN roots and consequently we discuss the definition, architectural pillars, benefits and limitations of hybrid SDN. Further, we categorize the different models under various headings, that can be used for deploying hybrid SDN. Next, we do a comparative analysis of each model. We discuss implementation approaches in each model and challenges that may arise in the deployment of hybrid SDN.

1. Introduction

Modern-day communication networks which are based on distributed control and network transport protocols pose a lot of complex operational issues (Levin et al., 2014; Casado et al., 2007, 2006; Qazi et al., 2013). Although the traditional[1] IP networks have been adopted widely, they are *complex and hard to manage* (Benson et al., 2009). A number of issues such as policy enforcement on wide variety of boxes (devices), high performance in terms of connectivity, robustness and fault tolerance, application and user aware routing, complex traffic isolation etc. have led to creation of many overlapping mechanisms at various network layers which make the management cumbersome and even prone to failure and security loopholes.

To add fuel to the fire, the forwarding and control mechanisms exist within the same network device and are tightly interwoven, known as *vertical integration* (Kreutz et al., 2015). Each device has vendor specific properties, due to which, deployment of another device in the network may lead to incompatible interfaces. Enabling interoperability requires reconfiguration of the existing devices, which often becomes buggy if done manually. This vertical integration and *vendor specificity* (Boucadair and Jacquenet, 2014) 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 (i.e., lack of configuration automation methods and response mechanisms) leads to a lot of management efforts, especially where the network changes are frequent (Jammal et al., 2014).

As the demand for real-time applications is increasing, it has become difficult to scale the existing networks and provide reliability and security without degradation of performance. Due to the presence of various technologies and stakeholders, generally, the approaches in the traditional network aim for better user experience for a few services or enhancing link utilization for a subset of networks. These approaches although aim to obtain optimal performance use local information, without cross-layer considerations, and thus lead to sub-optimal performance (Xia et al., 2015; Pathak et al., 2011). Network management issues are diverse, and the changes required in

* Corresponding author.
E-mail addresses: p2015007@pilani.bits-pilani.ac.in (Sandhya), h2016077@pilani.bits-pilani.ac.in (Y. Sinha), khari@pilani.bits-pilani.ac.in (K. Haribabu).
URL: http://universe.bits-pilani.ac.in/pilani/khari/profile (K. Haribabu).

[1] We use the terms legacy and traditional interchangeably.

http://dx.doi.org/10.1016/j.jnca.2017.10.003
Received 25 March 2017; Received in revised form 4 August 2017; Accepted 3 October 2017
Available online 06 October 2017

the network are unpredictable. For example, different topology and address spaces require data link layer or network layer services or even more complex L4-L7 services (transport, session, presentation and application layers). In such a dynamic environment, it is difficult to enforce the required policies (Kreutz et al., 2015).

The emerging "Software Defined Networks" paradigm has the potential to address these issues by simplifying the present state of network architecture (Kreutz et al., 2015). SDN provides a new architecture which stands on five pillars of (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 the SDN paradigm are only meant to forward the packets, whereas the network control logic is centralized at the *network controller/operating system*. This breaks the vertical integration. Secondly, the centralized controller can configure, manage, secure and optimize network resources dynamically (Kim and Feamster, 2013). Further, a centralized controller enables network programmability thus, network management can be automated via programs. With 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 exercises control over the forwarding elements via OpenFlow protocol. The OpenFlow protocol (Ben et al., 2012) for communication between the forwarding devices and the controller has gained popularity and deployments at different scales have been carried out.

The SDN paradigm despite having such benefits is facing multi-dimensional challenges viz., technical, financial and business which have affected the adoption of SDN as full deployments. Technically, there are many questions to be answered such as how scalable, resilient and robust can the centralized controller be without becoming the single point of failure. As the network grows, it may require more than one SDN controller. In terms of security, SDN controller is an attractive target for attack (Sezer et al., 2013). In the absence of a secure and robust controller, the attackers have the opportunities to change the behavior of underlying network by making changes in the controller code. A great focus on SDN security is required to make it acceptable. Till now, there are a few discussions on SDN security in industry and research community and potential vulnerabilities exist across the SDN platform. For example, authentication and authorization mechanisms have been questionable to enable multiple organizations to access network resources while providing the appropriate protection of these resources (Hartman et al., 2013). Among financial challenges, the huge budget investment required for SDN deployment is of concern which most of the organizations cannot afford in one go. This new paradigm requires the network administrator to define a new set of policies and provide the paradigm's implementation in the (perhaps new) hardware devices. Neither there are well tested and production grade strategies available as of now, that can be used for incremental deployment. As far as business challenges are concerned, the transition might cause a disruption in the services for the end users. Among the network operators, there is a need to build confidence for adoption of new paradigm over the well-running, time-tested traditional paradigm.

In this work, we survey how combining SDN and traditional architectures, known as hybrid SDN models, can propose a solution for a smooth transition. An environment, where both legacy[2] and SDN nodes can work together, has the potential to extract the benefits of both, while mitigating their drawbacks. In fact, many of the solutions developed traditionally can address challenges in SDN. For example, increased latency due to the communication delay between the switch and the controller can be mitigated by deferring decision making partially to the legacy control plane, reacting quickly in an emergency, such as failures, etc. On the other hand, the controller having the global view of the network can tweak the weights of local routing mechanisms to enhance the traditional way of managing things. An incremental deployment strategy can be developed to meet the financial and business needs of various organizations.

We begin by defining the contexts in which the term *hybrid* is used and what do we mean by a *hybrid SDN architecture*. We emphasize various pros and cons and then describe the different theoretical models that have been proposed to realize hybrid SDN. We discuss the work done in each approach, their advantages and disadvantages and do a comparative analysis. We further discuss the implementations and their evaluations carried out by the academia and the industry. We throw light on hybrid SDN specific issues and how they have been addressed by various researchers. To the best of our knowledge, this is the first survey paper on hybrid SDN.

2. The hybrid SDN paradigm

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*. For example, traditionally switches with their distributed algorithms such as IGP (Interior Gateway Protocol) 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.

The main pillars of hybrid SDN architecture are **3C's**:

1. *Coexistence*: As the name suggests, this implies a heterogeneity in the infrastructure either in the data plane or the control plane or both. Components of both SDN and the legacy paradigm stand together in the network, although they may or may not interact together (see Communication). Strategic placement of SDN nodes gives rise to various incentives for a transition. Different placement strategies form the attributes of this pillar.
   (a) Coexistence at the Data plane only, i.e., SDN and legacy nodes[3] together exist in the network, but managed by the distributed control plane of the legacy paradigm only. Although this setup is possible, it provides little benefit as managing SDN nodes with legacy control offers no advantage. Therefore, in Fig. 1 we provide no classification for "Co-existence in the Data Plane Only".
   (b) Coexistence at the Control plane only, i.e., centralized SDN control and decentralized legacy control both prevail in the network. For example, Caesar et al. (2005) propose a routing controller which provides a consistent assignment of routes for external traffic with the help of a global topology view. The routers pull the routes for external traffic from this controller, but the controller does not interact with the legacy control. This is coexistence without any communication. See 2b for coexistence with communication.
   (c) Coexistence at both data and control planes. Here, the network contains both SDN and legacy components, both in the control plane (SDN controller and distributed legacy control) as well as

[2] In this document, we refer to switches that can talk to the SDN controller via the 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 part of the traditional network are referred to as legacy nodes or SDN-incompatible devices. They contain the legacy distributed control plane as well as the data plane and understand the legacy protocols.

[3] A node refers to a data plane component, also known as forwarding element (FE) in SDN contexts such as a router or a switch.

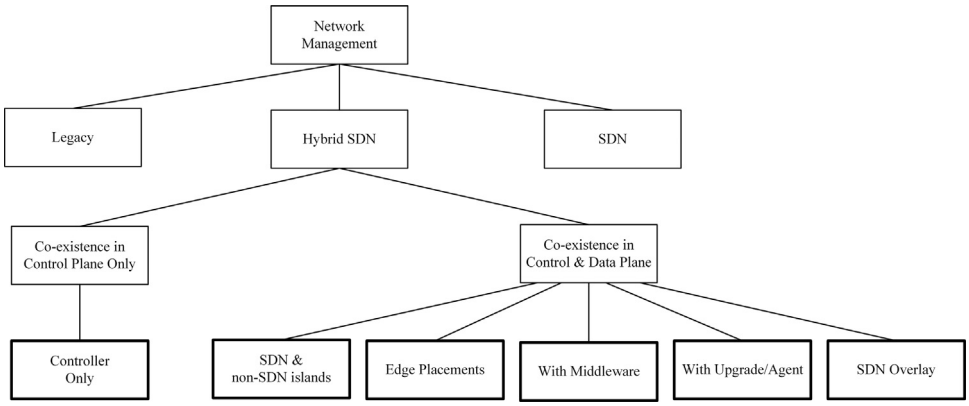

Fig. 1. Classification based on architecture and components.

the data plane (SDN and legacy nodes). For example, Telekinesis (Jin et al., 2015) introduces a mechanism to control the routing in legacy devices with OpenFlow protocol in a hybrid network.

2. *Communication*: Communication conveys the idea of inter-paradigm integration with mutual understanding and sharing & distributing of functionality among fundamentally heterogeneous components of the network. SDN and legacy components not only co-exist but also interact with each other and understand the interfaces and the protocols of each other to enhance each other. This involves a number of techniques like protocol translation, SDN nodes stealing the legacy (control) packet away to the controller, parsing packets, packet injection from the controller to the network and so on.

   For example, the SDN controller may help with the global topological view, whereas the local legacy control in the node may take fast local decisions even when the node-SDN controller link is congested. Sometimes communication is crucial. For example, in a network having a legacy and SDN switches, it is not possible to have a loop-free Layer-2 network unless both understand STP (spanning tree protocol) protocol. Hence, this is an important pillar. Communication can be realized at different planes:
   (a) Communication at the data plane only, i.e., SDN and legacy nodes exist together and they understand each other. For example, an SDN switch understands legacy protocols such as STP, RapidSTP, LLDP (Link layer discovery protocol) etc. used by legacy switches to function together. A setup only with this feature is of a little practical advantage unless there is some communication at the control plane.
   (b) Communication at the control plane only. Here the SDN controller understands the traditional distributed control such as advertisements of OSPF (open shortest path first) in order to discover network topology or to improve routing performance. For example, in Fibbing, Tilmans et al. (2016) introduce an SDN controller that functions together with the distributed control plane of the legacy paradigm to alter routing decisions with the help of fake OSPF LSAs (link-state advertisements).
   (c) Both data and control planes. In this scheme, both the control planes and data planes interact with each other with various degrees of protocol translation.
3. *Crossbreeding*: Crossbreeding involves intermixing different paradigms whose complementary attributes enhance the hybrid network. Here, crossbreeding indicates the degree of hybridization in terms of the following attributes that dictates architectural trade-offs. For example, there can be a trade-off between the number of features of a legacy protocol the SDN controller may parse and interpret versus the performance of the controller. Similarly, as the number of SDN nodes increase in the data plane, more traffic comes within SDN control, although this increases the budget for the organization (Levin et al., 2014). Therefore, these often act as parameters for a particular implementation to be chosen by an organization for deployment.
   (a) *Investment & budget constraints in transition*. This refers to the cost incurred in terms of hardware costs (for introducing a new SDN controller, SDN nodes, middleware, etc.), custom software creation and up-gradation etc. Often budget for purchasing new network hardware is constrained within an organization and therefore, a transition strategy which incurs less cost may be preferred.
   (b) *Transition smoothness in incremental deployment*. This is an indicator of the degree of disruption of services that happens when the transition is made from legacy to hybrid SDN. There are many factors which determine whether and for how much time the organization can afford to disrupt services. Some of them are the current setup of the network, incentive to transition, end-user experience, the loss incurred during downtime, security etc.
   (c) *Ease of automation*. The network is easy to automate if it is easy to carry out network-wide forwarding at the data plane for various kinds of traffic using software logic. With network programmability, in pure SDN it is easy to achieve this via central network logic. A hybrid network supports this quick, dynamic and automated packet forwarding up-to a limited degree. A complex configuration is less likely to be chosen as it is difficult to administer and debug. An organization may choose to enable network programmability for a particular service or traffic class initially, which will lead to a particular transition strategy.
   (d) *Policy expressiveness and enforcement*. A hybrid network supports flow abstraction and dynamic adjustment of network-wide traffic flow in a limited way because the added complexity in the data plane makes policy expression and enforcement difficult. Traffic steering[4] and middle-boxing dictate and limit how easy it is to express policy for a traffic flow and implement it by installing routing entries.
   (e) *Scalability and robustness*. This deals with the ability of the architecture to handle growing network size and failures which requires scaling of computing, storage, and network resources. The added complexity of multiple paradigms in hybrid SDN may make scalability a challenging aspect. It also becomes prone to conflicts and failures. For example, the number of services that SDN paradigm chooses to offer depends on the number and placement of SDN nodes. If more SDN nodes are placed at strategic locations, a service like load balancing can easily scale

[4] The traffic steering problem aims to find an optimized physical path when a flow has to be instantiated in the network. The aim is to optimally allocate a path in the network for the current request, that accommodates the maximum number of total requests in the long run (Cao et al., 2014).

up. Similarly, recovery mechanisms provided by the SDN controller can be more effective, if the SDN controller can monitor the network precisely. This is less likely to happen if only a few SDN nodes are placed to steal network packets to the SDN controller.

We explain coexistence and communication in detail in Section 3 and crossbreeding in Section 4 for specific SDN models.

2.2. Benefits it promises

The specific advantages of the approaches to the hybrid SDN paradigm have been mentioned in their corresponding sections. We present here an overview.

1. Hybrid SDN enables SDN-specific features (such as centralized control of the network) coupled with benefits of legacy (such as low deployment costs and time tested-maturity). Therefore, it can give *the best of both*. For example, in a traffic class-based hybridization model, the policy expression at a high level becomes easy. Hybrid SDN provides the feature to fallback to time-tested legacy mechanisms in case of SDN controller failure, which is not available in pure SDN paradigm.
2. There are areas where a *combination of centralized and decentralized mechanisms function well*. Update or installation of a large number of rules in the devices centrally could be a problem in pure SDN (due to control channel clogging, congestion, the processing capacity of controller etc.). Using both central and distributed control in the same environment, we can overcome this problem. If the communication with the controller is congested or the controller is unable to respond due to lots of loads, the switch can use distributed legacy routing mechanisms in the meantime to route crucial packets. By providing a central control over critical traffic only, overhead on the controller is reduced and the controller's scalability can be increased. On the contrary, in pure SDN there has been a lot of work going on to realize a hierarchical model of controllers to guarantee centralization with scalability for large networks Fu et al. (2014).
3. *Architectural tradeoffs* can be tuned to cater to the needs of an organization. For example, based on whether the organization wants to initially incentivize and accommodate the premium users or to enhance telecom billing, there can be different proportion 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.
4. There are *economic and business benefits* like gradual investment, building the confidence of network operators and end users etc.

2.3. Limitations

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

1. *Management of heterogeneous control plane* is difficult. Due to the interaction between multiple control planes, the network update procedures may not be safe (Vissicchio et al., 2014b). 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 traffic black holes. Establishing a communication session between the SDN controller and the legacy switch is challenging. Several alternatives to middleware, protocol translation, software upgrade, etc. exist which provide different advantages and disadvantages. For example, parsing IGP's advertisements such as OSPF LSAs in the controller for topology discovery takes huge overhead on the controller. The controller cannot support parsing of all legacy protocols as not all protocols can be translated into each other.
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 the 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, although more in hybrid SDN because of the overhead of interoperability. Thus, incremental deployment of SDN nodes requires more SDN controllers and this could increase the latency. They need to exchange information about the legacy devices and the program states using the east-west bound interfaces. Traffic Engineering (TE) optimization is more challenging in hybrid SDN as not all nodes support flow abstraction, OpenFlow, all packet matching and packet filtering mechanisms etc. Solutions using ACLs (access control list) or static routes provide limited functionalities He and Song (2015).

2.4. Probable contextomy

Within the context of SDN, the term “hybrid” has been used to indicate multiple meanings, such as heterogeneity, bilingualism and decision delegation. We have used the term “hybrid” to convey the hybrid SDN paradigm. We take this opportunity to clarify what we are proposing and **not** the following meanings.

2.4.1. Dual stack mode (hybrid switch)

A dual stack switch is a bilingual device that can support both OpenFlow protocol and legacy protocols within the same network device (Kandoi, 2015). 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. But in this document, we don't construe to this meaning.

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 (Ben et al., 2012), the controller can forward the packets to NORMAL port, delegating the forwarding decision to the traditional control plane.

3. Hybrid SDN models

In this section, we discuss different approaches to deployment of hybrid SDN and classify them. For each approach, we discuss the use cases that apply to them and the benefits and limitations of each approach.

3.1. Classification based on architecture and components

In Fig. 1, we categorize based on whether there is coexistence in the control plane or the data plane. Naturally, the case where neither there is coexistence in the data plane nor in the control plane is meaningless. Secondly, mere co-existence in the data plane without any co-existence in the control plane is hardly advantageous as explained in 2.1.a.

3.1.1. Co-existence in the control plane only

Looking back at the history of SDN, there were some attempts (Caesar et al., 2005; Balus et al., 2013; Atlas et al., 2013, ; Enns et al., 2011) that advocated introduction of a centralized system, within traditional IP networks for better configuration power and manageability. Inspired with those results, this model attempts to introduce a

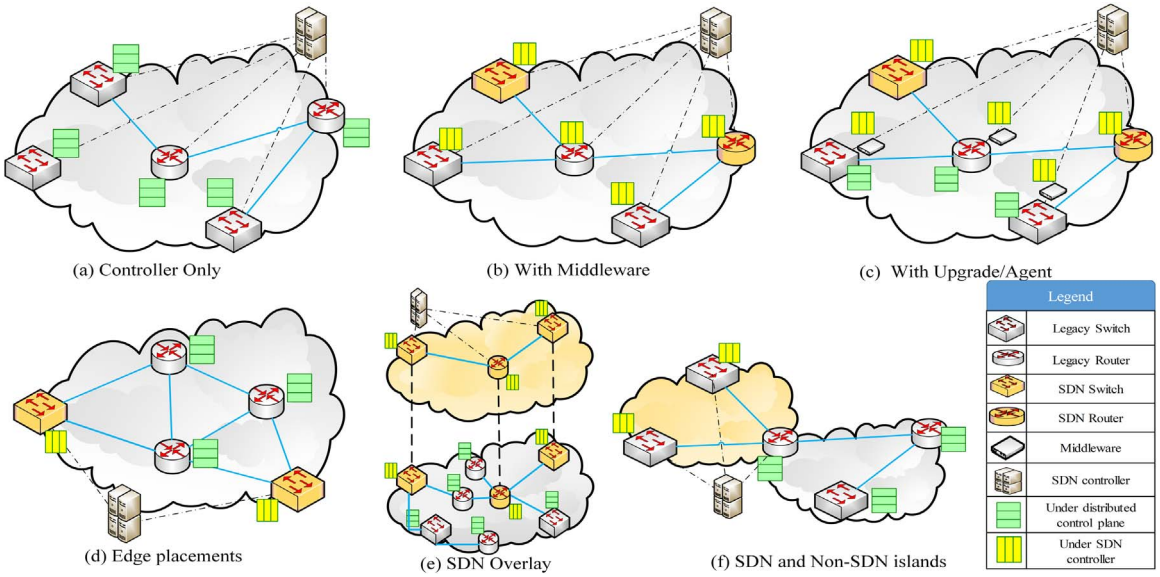

Fig. 2. Various hybrid SDN Models classified based on architecture and components.

central control to enhance the distributed control. Vissicchio et al. (2014a) refers to this approach as Integrated hybrid SDN.

Definition 3.1. Controller Only: As shown in Fig. 2a, the model introduces an SDN controller whereas the rest of the network remains unchanged. The idea is to enhance the distributed control plane with inputs from the centralized controller that has benefits of the global view of the network, fast convergence, flow abstraction, etc.

As Fig. 2a shows, the SDN controller may interact with the legacy nodes which can either be direct or deceptive. In *direct communication*, the controller itself understands the protocol and no protocol translation is involved. For example, in the Routing Control Platform Caesar et al. (2005), the controller acts as an IGP 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. Vanbever and Vissicchio (2014) propose a method to provide SDN control over the existing network, by introducing fake nodes in the network. In a strong form, the SDN controller may take responsibility for all the network services while using the legacy protocols as interfaces to communicate with the data plane devices.

Advantages.

1. This deployment strategy can be used by the network administrators to check, whether and to what extent there is a need for SDN deployment in their networks while building self-assurance about the 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. It is robust as it can always fallback to legacy distributed control, in case the controller goes down. In another case, the controller need not calculate temporary forwarding paths in case of failures as that can be taken by the legacy control plane automatically.
6. In the strong form, if the SDN controller has fully taken the responsibility of complete management, the organization can proceed to progressively replace and add new SDN data plane hardware.*Disadvantages*.

1. This approach incorporates limited SDN benefits to the traditional network.
2. The controller has the additional overhead of parsing a range of legacy protocol packets, discovering convergence of legacy routing protocols (like OSPF) etc. Thus, the controller may itself become vulnerable to failure.
3. A legacy protocol interface is much more complex than current proposals like OpenFlow and still, may not be complete in terms of functionality as compared to OpenFlow. Therefore, not all benefits of SDN can be realized (Fuentes et al., 2014).
4. Not all legacy nodes support all protocols, for example, a Layer 2 device such as the switch does not support Layer 3 protocols like OSPF. This necessitates that in order to be robust, the controller should have a hierarchical list of protocols (from complex ones that work in higher layers to simple ones that work in lower layers) to fallback to, in case the device fails to respond to other protocols. For example, a topology discovery service in a controller employs stealing and parsing OSPF Hello packets; but for a Layer 2 device, the controller must fallback to SNMP (Simple Network Management Protocol) or ARP (Address Resolution Protocol) depending on the type of device.
5. Security is yet 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.
6. Using this approach, we cannot modify packet fields (such as source/destination MAC (Media Access Control) addresses, SCTP (Stream Control Transmission Protocol), UDP (User Datagram Protocol), IPv4, IPv6 fields) which on the contrary, are supported in SDN enabled switches.

3.1.2. Co-existence in control & data planes

As the name suggests, the models where there is hybridization in both the control and data planes fall in this category. Consequently, these can be subcategorized based on whether the communication pillar exists.

3.1.2.1. SDN & non-SDN Islands. To begin the transition, an organization may choose a small part of its network to be upgraded to SDN whereas, the rest of the network continues to function unaltered. This leads to the formation of islands.

Definition 3.2. SDN & non-SDN Islands: In this approach (Fig. 2 f), the network is partitioned into different regions (i.e., SDN and non-SDN regions). The control in SDN region is centralized (i.e. provided by the SDN controller) and control in the non-SDN region is distributed. Dissimilar networks are connected via a gateway.

Vissicchio et al. (2014a) refer to this as Topology-based Hybrid SDN where there is a topological separation of the nodes controlled by each paradigm. In B4 (Jain et al., 2013), an SD-WAN (SDN in a Wide Area Network) is designed and implemented for connecting Googles 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, Hong et al. (2013) present SWAN, centrally controlling inter-data center network backbone with SDN and rest of the zones are legacy managed, authors present SWAN, centrally controlling inter-data center network backbone with SDN and rest of the zones are legacy managed.

Advantages.

1. 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.
2. Regions can be increased as the technology matures, expertise is cultivated, and a new budget is available.
3. Many enterprise networks are already separated into domains due to past merger/acquisitions, hierarchical separation for management, specific technical need etc. So in these cases, the organization may opt for this model.
4. Failure of SDN deployment has effect in the deployed region only, rest of the network has no effect.
5. Introduction of new mechanisms to communicate in between regions is easy. For example, solely by modifying the SDN control logic, safer (Le et al., 2010) and flexible (Wang et al., 2009) inter connection mechanisms can be deployed.

Disadvantages.

1. Deployment cost is high, as all devices in a given region are replaced with SDN nodes.
2. Cross-compatibility between islands may limit network functionality.
3. Long periods of service disruption in deployment phases. The network remains in a complex state until a full transition is complete.

3.1.2.2. Edge placements. Manzalini and Saracco (2013) propose that intelligence at the edge is going to be the trend as SDN adoption spreads. This would simplify management as separation of forwarding and separation of control will enable focus on solving different issues.

Definition 3.3. Edge placements: In this approach (Fig. 2d), SDN nodes are placed at the edge of the network. The SDN controller controls the forwarding decision at the edge nodes. For the controller, the topology is limited to the SDN devices only, i.e. it abstracts the existing network of legacy devices. The traffic in the core of the network uses the legacy protocols. The SDN paradigm is responsible for managing traffic that travels from the network to the outside world and vice-versa.

Casado et al. (2012) advocate this idea to separate the network edge from the core and highlight the key insights and benefits. For example, SDN nodes are stationed at the edges in a proposal (Mishra et al., 2016), which map the destination IP addresses of the incoming packets to unused IP addresses to enable customized routing through the legacy network.

Advantages.

1. Investment occurs only for the edge devices, while many SDN benefits can be realized.
2. Scalability (for the controller) is a function of a number of edge switches. Similar benefits can be realized as compared to a full SDN deployment of the same size. So, the load on the controller is less as compared to full deployment.
3. Separation of forwarding for edge and internal traffic simplifies the management and helps in independent evolution of fabric and edge (Casado et al., 2012).
4. Separation of control enables separate focus in solving two different problems. The internal fabric is responsible for packet transport across the network, while the edge provides more semantically rich services such as network security, isolation, and mobility (Casado et al., 2012).

Disadvantages.

1. SDN control is limited to, traffic which is passing through edge devices only.
2. The traffic generated within the network remains un-monitored.
3. There may be some conflicts between two paradigms while routing same packets.

3.1.2.3. With middleware. To provide a better SDN-like control, this approach aims to make the SDN controller understand legacy protocols with a specific software module, called the middleware, so as to enable it to interact with legacy nodes. This presents a two-step transition strategy. After transitioning from legacy to the Controller Only model, the second step involves changing the data plane progressively, adding SDN nodes and thus moving towards a full SDN deployment.

Definition 3.4. With middleware: As shown in Fig. 2b, the SDN controller uses a legacy protocol to the interface (middleware) and alter the legacy node configuration, whereas it controls the SDN switches in the standard way. The controller steals, parses and injects packets of legacy protocols to achieve the goal.

For incremental deployment in enterprise networks (Hong et al., 2016), the authors have proposed to forward messages flooded by legacy routing protocols as Packet-In to the SDN controller where they are parsed. Hand and Keller (2014) present a system of techniques for enabling SDN control over the existing legacy hardware, which is proprietary; to realize the fine grain control available in OpenFlow.

Advantages.

1. This model works even if there are no SDN nodes.
2. The configuration of the nodes in the network can be automated.
3. All OpenFlow features need not be supported. Only those can be supported which the organization requires.
4. Therefore, based on the need of an organization, only specific protocols are parsed, that not only provides the required SDN benefits, but also reduces the load on the SDN controller and enables fast performance.
5. Fallback to legacy mechanisms makes it robust, in case the SDN controller fails.

Disadvantages.

1. Not all legacy protocols can be translated at the controller. A set of protocols has to be chosen intelligently to cater to various Layer2/3 devices in the network and the needs of the organization.
2. Therefore, not all benefits of SDN can be enabled. For example, it can be difficult to collect network statistics from the nodes directly by issuing request messages from the controller.

3. Protocol translation at the controller incurs load on the controller and latency in communication.

3.1.2.4. With upgrade/agent. To provide a better SDN-like control, this approach aims to make the current devices understand OpenFlow protocol, so as to enable proper communication with the controller with the help of an OpenFlow agent.

Definition 3.5. With upgrade/agent: As shown in Fig. 2c, the existing nodes in the network are either upgraded or an agent is attached to translate and understand the OpenFlow protocol and hence communicate with the controller.

Others propose to introduce an agent for enabling interaction of legacy devices with the controller (Feng and Bi, 2015; Rostami et al., 2012; Tilmans and Vissicchio, 2014). The attempt is to answer, with minimal capital investment, what is the best SDN-like control possible so that the enterprises need not throw away their existing network deployments? There can be different degrees to which the functionality can be mimicked and hence the network performance can be improved.

Advantages.

1. Cost of deployment is less because for SDN like control it requires only a controller and a way to mimic OpenFlow like control (either by placement of an agent on top of legacy or by limited programming of the switches) for deployment.
2. One may not purchase SDN nodes necessarily.
3. Partial programmability and automation of tasks in the network can be achieved.
4. Not all OpenFlow features need to be supported. For example, if an organization wants to improve on network monitoring, the agent needs to add support only for messages like PortStatistics, FlowStatistics etc.
5. Consequently, based on the need of an organization, a specialized agent can be built, that not only provides the required SDN benefits and reduces deployment costs but can be tweaked for fast performance also.

Disadvantages.

1. Since existing devices are partially programmable, SDN deployment depends on their reconfiguration. Due to reconfiguration, there can be some disruption in the network.
2. Scalability can be an issue, because of the bottleneck of the intermediate agent.
3. The data plane becomes complex and supports both paradigms, this may give room to security loopholes. For example, if (unencrypted) remote logging feature of a legacy switch is used to report statistics to the controller (by assigning the controller IP as the remote logging system), the connection may not be secure enough.
4. Protocol translation incurs latency.

3.1.2.5. SDN overlay. With the incentive of introducing network automation feasible across different network infrastructures with the robustness of fallback to time-tested legacy underlay, this model aims to build an SDN overlay on the top of the existing legacy network.

Definition 3.6. SDN overlay: As illustrated in Fig. 2e, an SDN network is built as an overlay on the 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 sees the SDN overlay as the actual network. The overlay is composed of logical links which in turn consist of one or more legacy devices at the substrate.Big Switch Networks' Big Virtual Switch[5] 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, irrespective of whether it is compatible with OpenFlow. This is the basis for many of today's data center SDN products.

In Levin et al. (2014), Caria et al. (2015b) and Lu et al. (2013), only an abstracted view of the underlying topology is exposed to the SDN controller through which SDN benefits are realized.

Advantages.

1. 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.
2. Specific services that require virtualization can be implemented with ease.

Disadvantages.

1. Virtual and physical networks are separate entities with different attributes. This makes the network complex.
2. Gateways between the overlay network and nodes on the physical network may need to pass high volumes of traffic. For example, the frontiers can be a bottleneck for communication between SCTs in Panopticon (Levin et al. (2014)).

3.2. Classification based on functionality

This classification is based on what functional role is assigned to SDN and legacy paradigms within a hybrid SDN model. We can classify these models as service based and traffic based; refer (Fig. 3).

3.2.1. Service based

During the transition, only some SDN nodes are available initially. Strategic placement of these nodes can provide or improve some services such as load balancing, firewall, simplifying routing (Levin et al., 2014; Agarwal et al., 2013) etc.

Definition 3.7. Service Based: In this approach (Fig. 4a), the legacy network and SDN devices coexist, each providing different a set of services. Network-wide services such as packet forwarding can be controlled by two paradigms together by having entries in the forwarding table of each node. Some services can be controlled by solely SDN paradigm to provide services such as DNS (Domain Name System) resolution, load balancing, network function virtualization etc.

For example, services of network robustness (quick reaction to link failures & reduction in path stretch) can be realized (Tilmans and Vissicchio, 2014).

Advantages.

1. The network operator can choose any of the two paradigms to provide services. For example, SDN can be used for traffic engineering. Some services can be naturally managed with a central control such as Network Function Virtualization (NFV), and operators may be willing to keep the legacy hardware for services like MPLS (Multiprotocol Label Switching) Virtual Private Networks (VPNs).
2. There can be a need to improve certain services within an organization, which can be an incentive for operators to start the transition.
3. Services can be shifted to SDN progressively with incremental deployment phases.
4. A failure that occurs in SDN deployment, affects the services that are handled by SDN paradigm only, leaving the rest of the network services

[5] http://www.bigswitch.com/sites/default/files/sdnresources/bvsdatasheet.pdf.

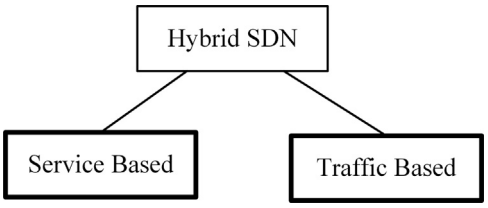

Fig. 3. Classification based on functionality.

unaffected. So it becomes easy to figure out the faculty services.

Disadvantages.

1. Investment is huge, although there can be an initial incentive to start the transition.
2. Services are disrupted for a considerable amount of time, particularly in the initial phase.
3. Some services or group of services cannot be necessarily separated to be controlled by the two different paradigms, for e.g., we cannot have load balancing done by the legacy control and routing done by SDN control.
4. Further, there can be challenges in separating centralized and decentralized control such as race conditions, latency due to the time taken for convergence of distributed mechanisms, path inconsistencies, etc. (Vissicchio et al., 2013).

3.2.2. Traffic-class based

With the incentive of ease of traffic management, in terms of fine-grained control, monitoring, security for various traffic classes this model can enhance applications (such as priority routing for premium users, and low delay guarantee for real-time applications) and simplify business cases (such as to provide VPN services while avoiding the limitations of MPLS RSVP-TE (Jain et al., 2013)).

Definition 3.8. Traffic-Class Based: In this approach (Fig. 4b), the traffic is partitioned into classes, some controlled by SDN and the remaining by legacy protocols. Although each paradigm controls a separate set of forwarding entries in each node, each paradigm takes care of all network services for the assigned traffic classes.

For example, the traffic is divided into traffic classes at the edge SDN switch which is then routed differently within the legacy core (Mishra et al., 2016).

Advantages.

1. This approach provides fine-grained control and helpful for real-time applications.
2. Easy and automated traffic management can be an initial incentive for taking up this model.
3. A lot of traffic-related optimizations such as traffic monitoring, traffic engineering, policy-based routing, firewall and middle-boxing are inherently supported via network programmability at the controller.
4. Reduces the overhead of the controller, as the controller's responsibility is to install rules only for a part of the traffic.
5. There is a separation of control for traffic classes, which helps in provisioning services such as security, isolation, application-based routing, etc.
6. Separation of control on various classes can help with aspects other than network management such as telecom billing, enabling services for premium users, etc.*Disadvantages.*

1. Strategies to separate legacy and SDN traffic have been proposed, such as mapping source and destination addresses of the packets to unused IP addressed in the network, VLAN (Virtual LAN) tagging, etc. This requires rewriting of a packet's header, installing packet forwarding entries for those IP addresses in SDN and non-SDN switches (Mishra et al., 2016), which leads to increased latency in the network and increased load on the switches as well as on the controller.
2. Incurs good investment for the organization.
3. Fine grained routing for different classes can be limited by the number of routing entries an SDN switch can support. This can affect scalability.

4. Comparative architectural analysis

As outlined in Table 1, in this section we provide a comparative analysis of all the models discussed in the previous section based on the pillars of hybrid SDN. This analysis helps the network operator to choose between different models, depending on requirements. We explain in each paragraph the incentive, investment required, transition smoothness, ease of automation, policy expressiveness & enforcement, scalability and robustness.

4.1. Classification based on architecture and components

4.1.1. Controller only

This model provides the advantage of introducing a central control, and progressively the control is shifted to the controller with a gradual maturity of technology and acquisition of expertise of the operators. Of all the hybrid SDN models, this model just requires the introduction of a controller, hence, practically free.

Disruption in the existing services is momentary because no physical reconfiguration of legacy nodes is required. Network-wide forwarding decisions are enhanced by the controller as it has a global view of the network, but it is limited by the capability of the controller 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 within a switch 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

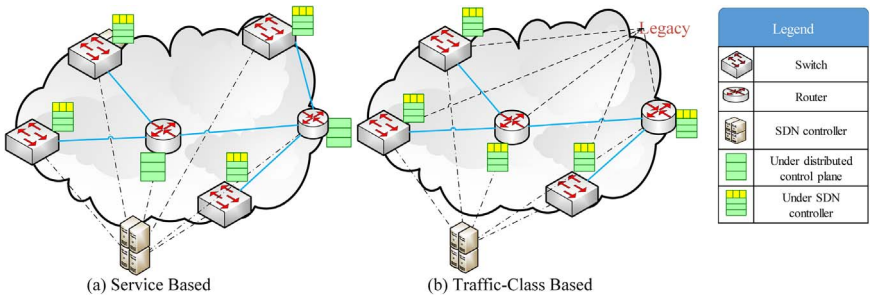

Fig. 4. Classification based on functionality.

Table 1
Comparison of different models in terms of hybrid SDN pillars.

Pillar	Coexistence		Communication	Crossbreeding				
	SDN components	Incentive	Inter-paradigm Integration	Investment	Transition smoothness	Ease of automation	Policy expressiveness & enforcement	Scalability and robustness
Legacy Network	None	None	None	None if, pre-deployed	Disruption free	Complex configurations	Firewall, VLANs, ACLs etc.	Legacy protocols
Controller only	Controller	Building initial confidence, central control	Protocol translation, Packet stealing & injection	SDN Controller, is Practically free	No physical reconfiguration, momentary disruption	Limited by protocol translation & #flow entries	Limited by protocol translation, not fine-grained	Limited by controller load, fallback to leagcy
SDN & non-SDN Islands	Controller with SDN islands	Region based transition	Via gateway, interconnection, mechanisms	Proportional to size of SDN islands	Possible disruption in replaced island	Programmable for SDN island only	Traffic within & through SDN island	Paradigm specific,
Edge Placements	Controller, edge placements of the SDN nodes	Intelligence at edge	Separation of control without communication	Proportional to #devices replaced at edge, expensive	Partial disruption for edge traffic due to reconfiguration	Enabled for edge & not for internal traffic	Enabled for edge traffic only	Scales gracefully, robust redundant nodes
With Middleware	Controller, SDN nodes	SDN control & nodes with min cost	Protocol translation, Packet stealing & injection	SDN Controller, SDN nodes	Partial disruption due to strategic placement of SDN nodes	SDN node: Programmable, Others: limited by protocol translation	Enabled for traffic passing at least one SDN node	Limited by controller load, recovery by controller
With Upgrade/ Agent	Controller, SDN nodes & agents	Minimal investment and hardware re-use	Protocol translation via agent	SDN Controller, SDN nodes and agent	Disruption: SDN nodes, software upgrade & agent	Programmable for SDN nodes & partially enabled for others	Enabled for all traffic, limited by agent support	Limited due to protocol translation, fallback to legacy
SDN Overlay	Controller, SDN nodes	Maximum benefits in hybrid	SDN nodes: VLAN gateways	SDN Controller, SDN nodes	High disruption due to network reconstruction	Overlay & SDN nodes only	Not enabled for un-monitored, underlay network	By network design, controller load, fallback on legacy underlay
Service Based	Controller, switches placed for the services	Introducing SDN based services like NFV	Data plane coordination for service division	Gradual, based on deployment	Possible disruption due to reconfiguration	Easier automation & programmability for SDN services	Support for middle-boxing for specific services	SDN services limited by load on the controller
Traffic- Class Based	Controller, traffic passes through one or more nodes	Easy traffic management	Shared control & conflict resolution	Gradual, based on deployment	Possible disruption due to reconfiguration	Programmable for SDN traffic, conflicts between traffic classes possible	Support for traffic monitoring & enggeering, policy based routing etc.	Limited by controller load & #flow entries in SDN nodes
Pure SDN	Controller, SDN nodes	SDN benefits	None	Maximum	Full disruption & re-construction	Fully programmable	Network-wide traffic	Concern Yeganeh et al. (2013)

sufficient computing power, but also depends on the distributed routing protocols, which the controller translates. In case of failure, the network can continue to function with legacy protocols as it used to work without the controller.

4.1.2. 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, may be prolonged in case existing region is replaced with SDN nodes. Reconfiguration of all the islands and installation of interconnection mechanisms or the gateway may take time. Network programmability is enabled only within an island. Policy expressiveness & enforcement is enabled in SDN island, although this may be extended to other traffic as well if it passes via an SDN island. Scalability is a function of the size of islands whereas robustness is managed by the paradigms separately.

4.1.3. 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 costs. With edge placements of SDN nodes, this model focuses to have SDN intelligence at the edge (Casado et al., 2012) and thus, a great incentive for SDN transition. The investment is proportional to the number of SDN nodes introduced and can be expensive. In this scenario, this architecture can maximize network performance without compromising security, mobility or survivability, while at the same time, minimizing both capital and operational expenditures.

Partial disruption occurs in the edge traffic due to the introduction and reconfiguration of SDN nodes. A reconfiguration for legacy nodes may be necessary. Programmability is possible for traffic passing via edge. Separation of forwarding for edge and internal traffic simplifies the management and helps in separating focus for packet transport across the network and edge services (edge security, isolation, mobility etc.). It also enables the independent evolution of fabric and edge. Policy expression and enforcement are possible only for edge traffic. Distributed traffic management frees up controllers to focus on large scale network and policy management as well as other services, resulting in a more efficient architecture. This scales up gracefully because if one needs to expand edge points, one requires to invest only in an SDN node, as compared to purchasing a single core device and later having to scrap it in favor of a new device that offers a little more capacity.

4.1.4. With middleware

The primary incentive for this model is to enable SDN control on the existing the data plane with minimum cost. After the first step of the transition from legacy to the Controller Only model (and thus building up confidence and expertise), the second step is to add SDN nodes and thus move towards a full SDN deployment. The investment scheme can be gradual if the placement of SDN nodes is chalked out based on budget constraints, performance enhancements.

Disruption of services occurs partially due to the introduction of SDN nodes. It may happen in each phase of the transition. The SDN nodes are programmable, whereas, for other nodes, programmability depends on the SDN controller's capabilities, its load, and support from the legacy devices (e.g., some packet matching fields may not be supported). Flow abstraction, and enhanced traffic management in terms of policy expression and enforcement are available for a flow under different schemes. For example, it may be required the flow must pass via at least one SDN node for benefits like monitoring, firewall, etc.; and it must pass via two SDN nodes, for fine-grained routing. The load on the SDN controller is the bottleneck for scaling up and the robustness is provided both by the SDN controller (e.g., routing traffic away from the congested area by introducing a fake node) and legacy protocols.

4.1.5. With upgrade/agent

If the legacy nodes can be tweaked to communicate with the controller either via a software upgrade or introducing an agent, we need to introduce only the 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 either due to introduction of SDN nodes, software upgrade or introduction of an agent; 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, if legacy nodes can closely act like SDN nodes with the help of an agent, even complex policy expression and enforcement along with middle-boxing & traffic steering functionalities may be realized. Scalability is limited as the introduction of agent incurs an increase in latency in communication. It is also limited by legacy protocols used. The model offers robustness in terms of fallback to legacy protocols.

4.1.6. SDN overlay

With the incentive of leveraging maximum SDN benefits, the model aims to build an SDN overlay on the top of legacy networks. 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, although un-monitored traffic in the underlay may not support this fully. Scalability is a function of network design and controller load. Failure recovery is provided by the SDN controller as well as the legacy protocols.

4.2. Classification based on functionality

4.2.1. Service based

With the aim of introducing service management using SDN, nodes are introduced at strategic locations in the network. Investments can be gradual based on the incremental deployment strategy of new nodes.

Every incremental stage can cause disruption of services due to the reconfiguration of nodes. A particular service which is getting migrated may face prolonged outage, whereas other services may resume. The services that SDN supports can be fully programmed using the SDN nodes. This may be partially available for services provided by legacy paradigm. There is inbuilt support for flow abstraction for traffic of services provided by SDN paradigm. Specific services like traffic-steering and middle-boxing can be enhanced. The scalability of these services is limited by the load on the controller.

4.2.2. Traffic-class based

SDN nodes are introduced at strategic locations in the network with the aim of monitoring maximum traffic under SDN. Investments are gradual, based on the incremental strategy used to cover traffic flows.

There can be a lot of disruption of services during reconfiguration, although once done, the traffic management can be fully programmable if it passes through an SDN switch. Not all nodes in the path are required to be SDN enabled. Traffic management is easy to implement because of inherent support for traffic monitoring, policy based routing, etc. The network is scalable for non-SDN traffic and can be made more scalable for SDN traffic if more SDN nodes are deployed, although it is limited by the number of flow entries possible in SDN nodes.

Table 2
Summarized overview of hybrid SDN.

S.No.	Plane	Problems and issues	Hybrid SDN
1.	Data plane	Controller-switch communication	Feng and Bi (2015), Parniewicz et al. (2014), Hand and Keller (2014), Caria et al. (2015b)
2.	Control plane	Traffic engineering	Vanbever and Vissicchio (2014), Caria et al. (2015b), Hong et al., (2016) , He and Song (2015), Agarwal et al. (2013), Chu et al. (2015), Guo et al. (2014), Caria and Jukan (2016a)
3.		Control conflict	Parniewicz et al. (2014), Vissicchio et al. (2013)
4.		Configuration	Lu et al. (2013), Hand and Keller (2014), Agarwal et al., (2015)
5.		Topology discovery	Hong et al. (2016), Agarwal et al. (2015), Caesar et al. (2005), Jmal and Fourati (2014), Pakzad et al. (2014), Ochoa Aday et al. (2015), Kandoi (2015), Pakzad et al. (2014), Pakzad et al. (2016), Sharma et al. (2011), Sharma et al. (2013), Saha et al. (2016)
6.		Fault tolerance	Chu et al. (2015), Sharma et al. (2011), Sharma et al. (2013), Caria and Jukan (2016a)
7.	Management Plane	Scalability	Fu et al. (2014), Dixit et al. (2013)
8.		Network Monitoring	Hong et al. (2016), Caria and Jukan (2016a)
9.		The placement problem	Caria et al. (2015a), Hong et al. (2016), Levin et al. (2014)

5. Implementation approaches of hybrid SDN

In this section, we discuss the approaches taken by the researchers to implement the hybrid SDN models. An overview of approaches in hybrid SDN implementation is explained in Table 3. We highlight the major contributions, specific deployment techniques used and limitations of the model.

5.1. Controller only

In this section, we discuss the different approaches used to achieve centralized control in the legacy network without the deployment of SDN nodes. For example, Caesar et al. (2005)achieve centralized control over the legacy devices by establishing an internal border gateway protocol (iBGP) session with each router in the network.

5.1.1. Session establishment

Caesar et al. (2005) 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. RCP provides consistent assignment of routes for external traffic, which provide reliability. RCP reaction is fast for link failures in the network. It is similar to SDN controller, but it is dealing with external traffic only. Using RCP for optimization of large ISPs (Internet Service Providers) could be difficult.

5.1.2. Injection of fake packets

Vanbever and Vissicchio (2014) propose the idea of achieving central control over distributed routing computation through fake nodes. In their next work (Vissicchio et al., 2014c), 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 of traditional routing protocol. The manipulation is done by introducing fake nodes in the network through 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 Fig. 5.

In their next work (Vissicchio et al., 2015), they show the evaluation of Fibbing controller along three axes viz., (i) load on router, (ii) topology augmentation and (iii) performance gain.

(i) 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 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 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. In the first phase, it injects an excessive number of fake nodes and computes the upper bound and lower bound for their respective costs. In the second phase, it 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 Tilmans et al. (2016) 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 limitations of Fibbing are, it can not manipulate the traffic based on ports because forwarding is done through IP address matching. Port based forwarding can be possible with the help of middlebox, but this incurs the cost for 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.

5.1.3. Translation of high-level goals into configurations

Hartert et al. (2015) 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 proposed “Declarative and Expressive Forward Optimizer” (DEFO) controller. The network operator can define certain goals, in terms of (i) redistribution of the load from heavily loaded links to less loaded links, (ii) traffic engineering e.g., bringing link utilization under a certain threshold, (iii) to enforce 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 optimization layer, it translates these goals into configurations. The forwarding rules generated by DEFO overwrites the forwarding rules generated by connectivity layer. DEFO maximizes the utility of input (for example, link capacity, expected traffic matrix etc.) for network optimization. In case of the controller failure, the forwarding rules are installed by the connectivity layer. Thus provides robustness.

Table 3
Summarized overview of hybrid SDN papers.

Reference	Year	Addressed issue	Proposed solution	Required	Limitations
Katiyar et al.	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	–
Hand et al.	2014	SDN like control over legacy devices	Proposed a system called ClosedFlow, which provides an SDN like control over legacy devices	Uses ACL, Route-Map and remote log in	Buffering of packets at devices is not supported
Jin et al.	2015	Providing control over legacy paths using OpenFlow in hybrid Environment	Proposed a hybrid network controller: Telekinesis, which modify the forwarding table of both legacy as well as SDN nodes using OpenFlow	POX OpenFlow Controller with Path verification and Path update	It can not update all paths with in the networks
Hong et al.	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 an SDN switch
Agarwal et al.	2013	Traffic Engineering in hybrid SDN	Solves the Dynamic Routing problem using Approximation method	Deploy some SDN devices in the Network	Legacy devices are not fully controlled
Chu et al.	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, install some SDN devices in existing Topology	Does not consider the case when no link failure occurs in the network
Levin et al.	2014	Policy enforcement in hybrid SDN	Proposed a way to provide abstraction of legacy network	Network is divided into cells and at the edges SDN nodes are placed	Cannot dynamically configure the legacy devices and the path through legacy devices in case of dynamic traffic demand
Tilmans et al.	2014	Packet loss	Proposed IBSDN architecture to provide backup path in case of link failure	Needs an agent deployment for the network nodes	Failure recovery is completely dependent on IGP
Vissicchio et al.	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 can not provide port based routing
Caria et al.	2016	How to distribute the load over the links in case of failures	Proposed an iterative heuristic algorithm to find near optimal spare capacity on each link	Network partitioning with SDN nodes	–
Vissicchio et al.	2014	How to provide centralized control in traditional network	"Fibbing", introducing fake nodes in the network	Inclusion of fake nodes in the traditional network	It can be vulnerable to security loopholes
Hartert et al.	2015	Optimization of large network to avoid congestion	DEFO controller	No modification is required	–
Parniewicz et al.	2014	How to provide unified view (i.e abstraction) of hybrid SDN network to controller	HAL architecture	No modification is required	–

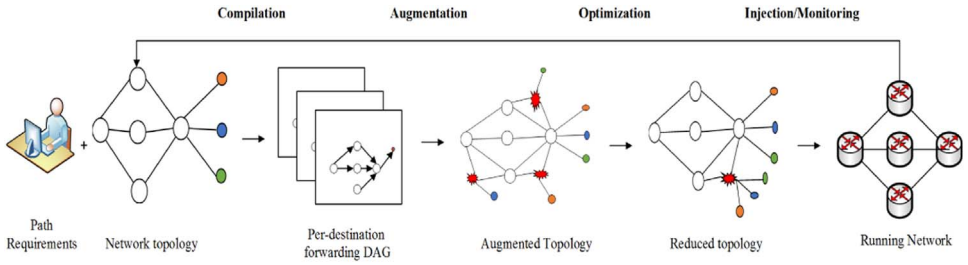

Fig. 5. Fibbing work-flow.

5.2. SDN and Non-SDN islands

With partitioning of the network into regions which are managed separately by either of the paradigms, along with an optional mechanism to implement cross-region services; this model has largely been using 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 (Jain et al., 2013). In SWAN (Hong et al., 2013), 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. One or more network agents intermediate between the controller and the switches

5.3. SDN overlay

The main incentive behind SDN overlay model is to build an SDN network overlay on the top 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.

5.3.1. Network virtualization

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.

Levin et al. (2014) give an architecture “panopticon” for incremental deployment of SDN 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 (SDNc) ports and legacy ports using VLAN IDs. The SDNc ports are controlled by the SDN controller whereas, legacy ports are not. Logical view for the controller consists of SDN nodes that contain their physical ports and SDNc ports (which may physically lie at a nearby legacy switch). To control legacy ports with the SDN controller and thus enable packet forwarding, it uses Solitary Confinement Trees (SCTs) and a VLAN ID assigned to each SCT to provide isolation. The constraint (named Waypoint Enforcement) is that the traffic between SDN-controlled ports must traverse through at least one SDN switch. Further, it may increase path-stretch for certain flows. Forwarding at non-SDN ports remains unaffected by panopticon. The authors provide a planner for incremental deployment of SDN devices in the existing network. This can be a long-term solution for incremental SDN deployment. Lu et al. (2013) 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 of multiple legacy nodes.

5.3.2. SDN partitioning

In Caria et al. (2015b), a single OSPF domain is divided into sub-domains with the help of deployment of 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 and it replies like an OSPF node. Once the HNM gets complete information about network topology, it provides optimal routing by altering the LSAs through changing the link weights. It does not affect the routing within the sub-domain.

In their next work (Caria and Jukan, 2015a), the authors propose a hybrid SDN/OSPF network control plane. The network is divided into sub-domains (Caria et al., 2015b) and an optical bypass[6] is set up between the SDN switches. The purpose of optical bypass in between border nodes is to offload traffic which transits among sub-domains. Therefore, it is easy to cope up 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.

The authors use a brute force mechanism to partition the network into sub-domains by the deployment of SDN nodes. In their recent work (Caria et al., 2016b), they propose an 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 placement of SDN nodes in such a way that their removal leaves the network unconnected. Further, they propose models for capacity planning, traffic engineering, and load balancing. In their recent work (Caria and Jukan, 2016a), they propose a heuristic algorithm for estimation of spare capacity required to deal with link failures in the network. They also provide an analysis, where spare capacity required in case of the SDN partitioning scheme is less than legacy and other hybrid models.

5.4. Edge placements

We present here some of the approaches used by different authors.

5.4.1. IP address mapping

Mishra et al. (2016) propose a framework for policy implementations in the network similar to those supported through OpenFlow. The framework allows for exploiting the benefits offered by SDN, using the legacy network devices and a minimal amount of SDN enabled switches. The design exploits the immense availability of unused IP addresses within networks. SDN nodes are stationed at the edges, which map the destination IP addresses of the incoming packets to unused IP address and thus, enable customized routing through the legacy network. The legacy nodes forward the packets based on destination IP address. The legacy network routes are controlled through static routes, while the entire network is managed by the SDN controller.

[6] An optical circuit, that consists of optical switches connected through an optical fiber cable.

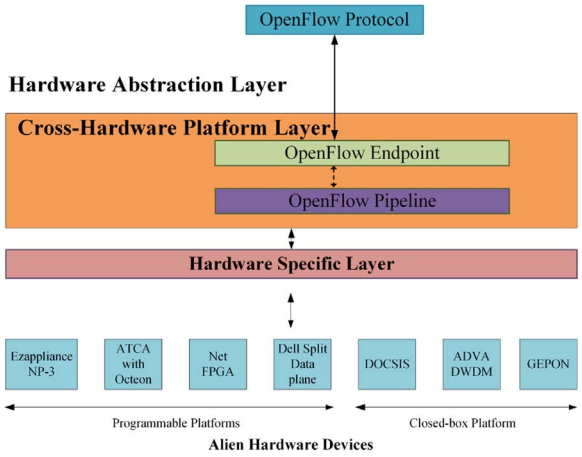

Fig. 6. HAL implementation framework over alien hardware devices.

5.4.2. Using virtual machines

Nakahodo et al. (2014)propose a way to reduce the congestion in the network using S-OSPF (Smart-OSPF) (Mishra and Sahoo, 2007). 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 the congestion in the network. The second VM (OvS switch) uses S-OSPF protocol to construct the forwarding table, whereas, first VM (quagga router) uses OSPF. These VMs are connected with a virtual link.

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 is as follows:

Lu et al. (2013) 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 the 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 nodes 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. The change in legacy nodes is performed by HybNET controller 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 (Hand and Keller, 2014). 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. The switch is configured to remote-log adjacency changes to the controller, although this establishes flow rules in the switch. OpenFlow's packet matching and apply actions are realized partially via the use of ACLs and Route maps. Modification in the packet header, such as changing port number and protocol specific modifications are unsupported by ClosedFlow. Although as per OpenFlow documentation, modify actions are considered optional for implementation of an OpenFlow switch/network, these are used heavily, hence a drawback for this mechanism. 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.

Parniewicz et al. (2014) propose an architecture "Hardware Abstraction Layer " (HAL), to transform the legacy nodes into OpenFlow nodes. HAL provides abstraction by hiding the underlying network topology as well as 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 Fig. 6. HSL collects the information about network topology and sends this information to CHPL. When a network packet comes, it is forwarded to CHPL module, which is processed by the OpenFlow pipeline. The flow entries and packet related actions are forwarded back to HSL, which translates these actions into devices dependent syntax. In their next work, Belter et al. (2014) further show the implementation of HAL on "Alien hardware devices"[7], e.g., "EZappliance" and "DOCSIS" alien devices.

5.6. With upgrade/agent

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

5.6.1. Configuration of agent in legacy devices

Tilmans and Vissicchio (2014) propose an "IGP-as-a-Backup SDN" (IBSDN) architecture. In the proposed architecture, an agent is configured for each SDN node to exchange the routing information in the network. In normal condition, the controller configures the route using OpenFlow protocol. Whereas, the configured agent builds backup paths using IGP protocol. Whenever a network failure happens in the network, the IGP agent quickly re-establishes the connectivity by using the local routing information.

5.6.2. Software upgradation

Feng and Bi (2015) propose an architecture called "OpenRouteFlow". OpenRouteFlow provides a 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 software platform. The OpenRouteFlow agent communicates the distributed routing information to the OpenRouteFlow controller. At the same time, the OpenRouteFlow agent receives the application oriented control instructions from the OpenRouteFlow controller, which are then transmitted to the legacy device.

5.7. Services based

Panopticon (Levin et al., 2014) provides fast fault tolerance by partial deployment of SDN switches. The SDN controller uses the STP to provide fast reconvergence. The SDN controller notices the update of STP through frontier *"F"*[8] and it takes the necessary action to restore the end-point connectivity in the network. This requires physical link redundancy within the SCTs. In the case of failure of SDN nodes or their incident links, recovery requires re-computation of forwarding state.

Tilmans and Vissicchio (2014) 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

[7] ⟨http://www.fp7-alien.eu/?Page_id=62⟩

[8] A switch common between two SCTs (Solitary Confinement Trees).

node, which collects the information about routing. It uses this information to build backup paths using distributed routing protocol. In a normal condition, the packets follow the rules installed by the SDN controller and in the case of link failure, the packet is forwarded according to the backup provided by the agent. IBSDN provides fast re-routing in case of link failure, which avoids packet loss. Chu et al. (2015) 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.

5.8. Traffic based

Mishra et al. (2016) propose a framework for class based hybridization using IP address re-mapping in a network. At the source SDN switch, the packet header is modified and the source & destination IP addresses are altered to un-used IP addresses using a scheme provided by the SDN controller. The routes are installed by the SDN controller for these new IP addresses in the path and thus the packets are re-routed. The packet headers are reset to the original values at the destination SDN switch. The model can re-provision resources dynamically based on policies expressed by the network administrator. However, it requires the source and destination switches in a path of a flow to be SDN enabled. The performance of their model has not been evaluated.

Jin et al. (2015) "Telekinesis", a path control mechanism for legacy switches implemented at Layer 2. The underlying assumptions require a switch to have active MAC learning and direct path available between the legacy switch and the OpenFlow-enabled switch. The SDN controller instructs the SDN switch to send seed packets to the source host with the *MAC address* of the destination host. This tricks the host to accept that the SDN switch is actually the destination. Thus, the host ends up sending the packet to the SDN switch (and not to the switch which actually possessed the true MAC address), and a packet is re-routed. The SDN switch receives the diverted packet, then forwards it towards the destination. Since the MAC addresses are reset periodically after some time, the SDN controller needs to send the seed packet time and again. Further, if the seed packet arrives late, path flipping may occur. To overcome this, the SDN controller sends the seeds packets quite frequently.

To overcome the problem of path flipping, in their next work, Jin et al. (2017) propose a unified controller called "Magneto". Magneto introduces the concept of magnet MAC addresses. These magnet addresses are a set of IP addresses that don't exist in the network. These MAC addresses are mapped to the actual IP addresses of hosts using gratuitous ARP messages. SDN switches can send seed 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 gratuitous ARP message, the source host has the magnet MAC address of the destination host instead of the real MAC. Hence, a packet from source host is routed towards the SDN switch. The SDN switch has necessary flow entries to fix the ethernet packet header by changing destination MAC address from magnet address to the real one and the source MAC address to a magnet address.

He and Song (2015) provide a formulation of the TE problem by considering the two hybrid modes, namely barrier mode and hybrid mode. In barrier mode, the SDN traffic and legacy traffic is routed in separate capacity spaces, whereas in the hybrid mode the link capacity is shared by both SDN and legacy traffic. Hong et al. (2016) propose an architecture to achieve TE in hybrid SDN network. They suggest two load balancing heuristics, according to which the TE module installs the rules in SDN devices and divert the packet to less loaded links.

6. Addressing challenges in hybrid SDN

In this section, we discuss the challenges that come in the deployment of SDN paradigm in the existing network to bring up the hybrid SDN network. For example, an organization may need to decide the locations where the existing nodes are replaced by SDN nodes. This decision may depend on traffic patterns, budget constraints, user requirements etc. We categorize and discuss the work done by researchers for each issue as summarised in Table 2.

6.1. Topology discovery

Topology discovery is a crucial component in SDN networks as network applications depend on this information to configure and manage the network such as making routing decisions. Furthermore, applications need to know about the complete network topology in order to make optimal routing decisions. Also, the controller requires an up-to-date real time discovery mechanism to detect events like link failures to respond quickly. Secondly, the controller load and performance is critical for the scalability of a Software Defined Network (Tootoonchian et al., 2012). 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 (Pakzad et al., 2014, 2016).

In hybrid networks, topology discovery becomes even more difficult due to a number of factors such as: presence of nodes supplied by different vendors; different protocols supported by different nodes to different degrees such as SNMP, OSPF, BGP etc.; inability of other nodes to understand proprietary protocols of other vendors; inability of non-SDN nodes to communicate with the controller; presence of firewalls, middleboxes etc.; dynamic changes in network such as link failure, node going down and many more. In this section, we attempt to address the main protocols and approaches to the topology discovery that can be leveraged for topology discovery service in a controller in both full SDN networks and hybrid SDN networks. Table 4 gives an overview of topology discovery.

6.1.1. Layer 2 protocols

6.1.1.1. LLDP (BDDP) based mechanisms (OFDP). Several researchers (Ochoa Aday et al.,; HP SDN[9], Pakzad et al., 2014, 2016; Fu et al., 2014; Kandoi, 2015,; Hong et al., 2016) have used various LLDP based mechanisms to discover end hosts and links. This has been referred as OFDP (OpenFlow Discovery Protocol) (OpenFlowDiscoveryProtocol; Pakzad et al., 2014, 2016). After switch features reply (i.e., after a TCP (Transmission Control Protocol), a connection has been formed between the controller and the switch. The controller periodically, say every 5 Sec, commands the SDN switch to flood LLDP and BDDP (Broadcast Domain Discovery Protocol) packets through all of its ports and send back the responses back to the controller to detect direct links (between switches) and the switches respectively. The LLDP packets are received by the controller from the legacy devices via the SDN switches. These packets have the PTOPO-MIB field (Physical Topology Management Information Base) which contains information about the SDN device to which it is connected. Thus, this link is discovered. Further, there is no controller identifier in the packet when the legacy device sends an LLDP packet to the SDN device. This property is used by the controller to infer that the SDN switch is connected to a legacy switch. The network application can use these bits of information to correctly construct the topology information.

6.1.1.2. OFDPv2-A, OFDPv2-B. Pakzad et al. (2014, 2016), propose that as opposed to OFDP, where we send a unique packet for each port

[9] https://community.arubanetworks.com/aruba/attachments/aruba/SDN/43/1/4AA5-6738ENW.PDF.

Table 4
Overview of topology discovery.

Reference	Layer: Protocols	Device discovered	Mechanism	Complexity/ Overhead	Benefits	Limitations
HP SDN	L2:BDDP, ARP, L3:DHCP	End hosts, link between SDN devices	Install four rules: (stealing BDDP, copying ARP, DHCP discover/request and DHCP offer/ACK packets)	Number of messages is proportional to the way controller injects BDDP packets into the network and pushes four rules to all controlled switches	Simplistic approach	Uses normal port although its support is optional in OpenFlow specification,Can not detect non-SDN links
Ochoa Aday et al. (2015)	L2:LLDP, OpenFlow (Communications protocol)	SDN nodes and links	OpenFlow Feature Request Message	Number of LLDP Packet-Out messages: Number of switches * Number of active ports Total packet-in: 2*Number of links	Zero latency	Only OpenFlow switches can be detected, fails, if intermediate switches are traditional
	L2:BDDP	Non-SDN links	BDDP packets with eth type 0 × 8999 and destination MAC address ff:ff:ff:ff:ff:ff are used	–	–	Cannot discover legacy switches, not a standard,
Pakzad et al. (2014, 2016)	L2:LLDP, OFDP, OFDPv2	Non-SDN links	Variant of Ochoa Aday et al. so that the number of control messages handled at the controller and the SDN switch gets significantly reduced	Total packet-in: 2*Number of links Number of LLDP Packet-Out messages: Number of switches	Upto 45% fewer messages than Ochoa Aday et al.	Number of LLDP Packet-Out messages is reduced to only one per switch. OFDPv2-B uses less TCAM memory
Saha et al. (2016)	L2:LLDP	Non-SDN links	Enhanced mechanism using adjacency table and loop table to find alternative, efficient paths	O(N^2) maintained at the controller, where N is the number of switches	proposes mechanism for failure recovery as well	A lot of control messages
Jmal and Fourati (2014)	L2:LLDP	SDN links	Standard implementation in POX using LLDP packets	–	–	Cannot detect legacy devices/ links
Sharma et al. (2011, 2013)	L2:STP	Failure recovery	Based on spanning tree's link change event trigger	O(number of paths calculated by STP)	Robust link change event trigger, few packets dropped in restoration	–
Kandoi (2015)	L2: LLDP, SNMP, L3: OSPF, L5: BGP-LS	Links and routers	A number of mechanisms suggested	Carry link state information gathered by the intra-domain protocols in BGP packets, Controller acts as a fake BGP route reflector	Interoperability for mechanisms that use OSPF	At experimental stage
Hong et al. (2016)	L2: BDDP, LLDP, SNMP, L3: OSPF	Links and routers	Parse OSPF LSAs and Hello messages, Catch SNMP traps	Fixed amount of messages exchanged periodically,No extra messages required in case of SNMP traps	Robust in failure detection, Support interoperability	Latency due to OSPF convergence time
Gredler et al.(2015)	L5: BGP-LS	Links	BGP carries link-state information gathered by the IGPs	Protocol in draft stage at IETF	Useful for BGP enabled nodes	Each domain must have at least one BGP speaker
SNMP4SDN:Beryllium**	SNMP, CLI, LLDP	SDN/NonSDN switches, routers and links	Proactively notify the controller of traditional switch's existence using SNMP trap	Switch automatically sends a trap, so no messages are required to be sent from the controller	Proprietary configuration of switches, VLAN configuration, Flow configuration, Statistic retrieving, Switch/port property retrieving, Multi-vendor support	Not all devices support SNMP. CLI is a time taking approach.

** http://docs.opendaylight.org/en/stable-boron/user-guide/snmp4sdn-user-guide.html.

for every switch; in OFDPv2-A, we send only one packet per switch. The source MAC address is used to identify ports, uniquely since the switch cannot rewrite Port ID of LLDP payload. Version 2-B is an enhancement to use less TCAM (Ternary Content Addressable Memory) memory. This approach reduces the number of control messages by almost 45% as compared to OFDP protocol. Thus, this can be effective where a switch to controller communication is likely to be clogged.

6.1.1.3. STP. The STP protocol is helpful for link discovery. In Kandoi, (2015) and Jmal and Fourati (2014), the authors suggest that STP can be beneficial, especially in the cases where dynamic and robust detection of link failures is the need of the hour. OpenFlow 1.3 (Ben et al., 2012) provides mechanisms to detect STP changes, configure STP options, port configurations, etc. An example has been explained in detail in Ryu book. As soon as an STP recalculation is triggered (either due to changes in path's weight or link/node failure) the packets can be copied to and parsed at the controller to detect changes quickly. The ports which have been blocked by STP can be used by the OpenFlow to transmit control traffic or load balancing, thereby improving overall network utilization.

6.1.1.4. Table driven based topology discovery. Saha et al. (2016) propose a set of steps for topology discovery, loop finding, and failure recovery. Initially, a procedure maps the hosts to the ports of the switches. Subsequent procedures detect links between the switches using LLDP packets. Consequently, the controller finds a path between every pair of switches. Lastly, the controller runs a procedure to find an alternative path in case of a link failure. The key idea is to maintain adjacency and loop tables to find alternatives. Although there is a feature of failure recovery a lot of control packets[10] are generated.

6.1.2. Layer 3 protocols

6.1.2.1. S-IS, OSPF. If the legacy nodes support IGP protocols such as OSPF and IS-IS, these packets can be sent to the controller to discover topology. The Link state advertisements and Hello messages of IGP protocols such as OSPF are redirected to the controller where it is parsed to generate the topology (Kandoi, 2015; Hong et al., 2016). Not only this procedure is good to discover topology and topology changes, but also it can detect nodes from different vendors. However, this method is useful only for Layer 3 devices.

6.1.2.2. ARP, DHCP. In HP SDN, the authors propose the idea to copy, steal, modify or redirect packets of ARP and DHCP (Dynamic Host Configuration Protocol) protocols to sniff topology details. As soon as a new switch registers itself at the controller, the controller installs four rules: steal BDDP packets from the network to the controller and copy ARP, DHCP discover/request as well as DHCP offer/ack packets from the network to the controller. While BDDP packets are used to detect links between SDN devices, ARP and DHCP packet sniffing is to learn about end hosts.

6.1.3. Application layer protocols

6.1.3.1. CLI. This approach involves the usage of command line interfaces (CLI), scripts that mimic the functioning of an administrator. The controller programmatically logs in and alters the configuration of the router/switch using commands like Telnet, etc. (Mishra et al., 2016; SNMP4SDN:Beryllium[11]). Python modules such as expect can mimic a network administrator using Telnet on the switch. This is a poor approach, but can be used as fallback mechanisms in case all other mechanisms fail to work. But there are certain features in this approach which are not available elsewhere. For example, if with the help of CLI, the switch is configured to remote log events to the controller, the controller can make efficient use of the traditional topology discovery mechanisms like OSPF convergence.

6.1.3.2. BGP-LS. In Kandoi, 2015 and Gredler et al. (2015), the authors refer to an extension to BGP enabling it to carry link-state information gathered by the IGPs. The controller as a BGP route reflector can discover the topology. One or more Link-State NLRIs (Network layer Reachability Information) are contained in a BGP message. A Node NLRI uniquely identifies the router, a Link NLRI uniquely identifies a link, and a Prefix NLRI uniquely identifies an IPv4 or IPv6 Prefix originated by the BGP speaker.

6.1.3.3. SNMP. SNMP has been traditionally used to detect device failures and node reboots. The same idea can be extended for detecting the events in hybrid SDN by setting the remote IP as the IP address of the controller to which the traps are sent. Pakzad et al. (2014, 2016), Kandoi, 2015, SNMP4SDN:Beryllium and Hong et al., 2016 suggest and analyze the benefits of this approach. There are a variety of traps which can help to detect a lot of events in the dynamic network such as the connection of a new switch or link failure.

6.2. Configuration

Network configuration is one of the important functions of network management. Network configuration management process organizes and maintains the information about all the components in the network. This information is used, when a network needs to be updated, repaired or expanded. 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 can not be configured directly by the SDN controller. The legacy devices are configured manually, which can lead to misconfiguration or errors. In hybrid SDN, the network administrator may have to use vendor specific network configuration and management tools. These tools work only with the products of a single vendor and meant to be used for a legacy network. This further complicates the issue. The controller can use CLI such as telnet to configure the legacy nodes, but the support is limited. For example, with telnet, we can not modify the IP address of a packet header. This issue is further magnified by multiple software versions, different vendors and limited support for protocol translation at the SDN controller.

Lu et al. (2013) propose HybNET, a framework to automate the network management of a hybrid SDN network. HybNET provides a 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 by translating configuration of the legacy devices into OpenFlow configuration.

ClosedFlow (Hand and Keller, 2014) provides an idea of making current networks centrally controllable by configuring each interface of

[10] Packets generated for network management such as DHCP packets, OpenFlow packets are called control packets.

[11] http://docs.opendaylight.org/en/stable-boron/user-guide/snmp4sdn-user-guide.html.

the router. But, the implementation of ClosedFlow is limited to the devices that support functionality like ACLs, Route maps.

Katiyar et al. (2015) propose a method for automated SDN deployment with a focus on automating the configuration of SDN switch installation in SDN/hybrid networks. The aim is to minimize the risk of errors in the manual configuration and reduce the operational cost. They propose components such as Locator and Configurator. 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 the DHCP option field) to AutoConf Server (ACS). The ACS calls an Intermediate Switch Configurator (ISC) component to configure both intermediate SDN and non-SDN switches to provide connectivity between the newly added SDN switch and the SDN controller. After configuration of intermediate switches, ACS sends the configuration parameters as a reply to the newly added SDN switch, which is then used in the controller registration process.

6.3. The placement problem

While we transit from the legacy network to SDN via a hybrid SDN model, we need to choose the subset of legacy nodes (number and location) to replace with SDN nodes based on factors such as budget & resource constraints, traffic matrix, network topology and performance benefits. This decision making can be viewed as an optimization problem with objectives of maximum link utilization, minimal disruption, maximum benefit to budget ratio etc. subject to constraints like hardware lifecycle management, long-term evaluation 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 placement.

Hong et al. (2016). formulate the problem of SDN deployment in the legacy network with a bilinear term of unknowns and solving the unknowns is NP-complete. Further, they provide heuristics for SDN node placement by replacing the existing nodes. The first heuristic picks up a legacy node with the highest degree in the network topology graph. The second heuristic uses K-shortest path algorithm to find the path between all source and destination pairs and selects the nodes which have the highest frequency of occurrence in these paths. In the third heuristic, the legacy nodes are selected for the replacement with higher traffic volume. Caria et al. (2015b) propose a method for SDN device deployment in Open Shortest Path First (OSPF) network. The existing network is partitioned into sub-domains by strategic placement of SDN nodes. The placement is done such that removal of SDN nodes partitions the network into disconnected components. In the case of failure of the sub-domain border node, OSPF provides the alternative path via other border nodes.

Levin et al. (2014) 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 whereas DEG replaces the legacy switch with a higher degree in the topology graph.

6.4. Conflicts in hybrid control planes

Hybrid networks are difficult to manage compared to legacy or pure SDN networks, because of the simultaneous presence of legacy and SDN control planes. Any update in the hybrid network can trigger a forwarding inconsistency, which may lead to disruption in traffic engineering 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 mutual understanding based on mechanisms such as protocol translation. For example, a controller may parse and inject packets of 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 choose to 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 by different researchers are the following.

Vissicchio et al. (2013) give 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 in the following manner. For each destination, GPIA iteratively creates a set of nodes that can be configured without any inconsistency in the network. GPIA picks up any node from the intersection of all computed sets until the intersection returns an empty set. At the end, if the computed sequence involves all the nodes that exist in the network, then the algorithm returns the sequence else it backtracks to the previous step and chooses a different node from the intersection. Finally, the algorithm returns the sequence of nodes for reconfiguration. However, the algorithm is dependent on recursion tree generation in the backtrack phase and limited to the co-existence of SDN and IGP only.

Parniewicz et al. (2014) 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.

6.5. Controller-switch communication

In the SDN architecture, the controller is typically a remote entity. The SDN nodes proactively establish a TCP connection to the SDN controller. This incurs latency in communication. Whereas, in the legacy networks the control functionality sits within the forwarding device itself, so no controller-switch communication delay occurs.

In SDN nodes, the forwarding decisions are taken based on the complete topology view. Whereas, in legacy nodes, the forwarding decisions are taken based on local information. Since the SDN nodes are controlled by the SDN controller using OpenFlow protocol and legacy nodes are designed to be controlled by distributed routing protocols like OSPF; in hybrid SDN, to provide communication between the SDN controller and legacy nodes is not trivial. This can be achieved with the help of a translator module, which can exist as a plugin in the SDN controller, like SNMP4SDN[12]. This translation can be full or limited. There is a trade-off between performance gain and a number of features enabled by the controller.

The communication can be in-band or out-of-band. In in-band communication, the control traffic is sent on the same link which is used to send the data traffic. Thus, there is no extra cost incurred for new link hardware. However, this can cause network congestion. In out-of-band communication, a dedicated link is used for control packet transmission. This decreases communication latency. It incurs a cost and requires management of out-of-band links between nodes in the network and the SDN controller.

Feng and Bi (2015) propose a system of an OpenRouter along with an OpenRouteFlow controller which by the means of software upgrade achieves the following: it makes the router OpenFlow compatible, enables dynamic push of routes using ACLs and improves the visualization of global routing and flow views. This can help to achieve a unified routing view in hybrid models and reduce the deployment costs by a significant margin.

Parniewicz et al. (2014) propose HAL, which provides an abstraction to the real network that consists of legacy and SDN nodes. HAL

[12] http://docs.opendaylight.org/en/stable-boron/user-guide/snmp4sdn-user-guide.html.

takes the network packet, processes it through the OpenFlow pipeline to apply the changes and the translator module translates these changes into commands specific to the platform of underlying physical devices in the network.

In ClosedFlow (Hand and Keller, 2014), the SDN controller uses SSH or telnet to install the flow entries in the forwarding table of switches. The switch uses remote logging to communicate the adjacency changes to the controller.

Caria et al. (2015b) propose "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 module gets all information about the 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. The SDN nodes distribute these routes further in the sub-domain by flooding.

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. Dixit et al. (2013) propose an ElstiCon architecture, where the controllers initially operate at pre-defined load window and as the load changes over the 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 lightly loaded controller. Fu et al. (2014) propose a hierarchical architecture for the control plane to provide scalability in SDN networks.

6.7. Traffic engineering

The main goal of traffic engineering is the optimization of network performance to facilitate the reliability of network operations (Awduche et al., 2002). This can be achieved by making the network fault tolerant and congestion-free by balancing the load on the links, etc. In hybrid SDN, it is difficult to provide TE, as the legacy devices are not under the control of the SDN controller fully. The rules which the SDN controller enforces to provide TE, are applicable only on SDN nodes in general, leaving the behavior of legacy devices un-altered.

6.7.1. Load balancing

Vanbever and Vissicchio, 2014 propose an architecture, which takes physical topology & path requirements as input and produce an augmented topology. The augmented topology considers all path requirements, which are needed to provide load balancing in order to avoid congestion (Vanbever and Vissicchio, 2014). Hong et al., (2016) propose a TE module for load balancing. Whenever a new flow comes, the TE module routes it on the least loaded path. The SDN controller uses meter table feature of OpenFlow 1.3 (Ben et al., 2012) to retrieve the link-load dynamically. However, this is possible only if the flow traverses at least one SDN node.

Caria et al. (2015b) propose a module called "Hybrid Network Manager" (HNM). The main functionality is to gather the information about the network viz., topology, traffic in the network, the position of SDN nodes and routing. In HNM, a module called *Traffic Engineering Engine* (TE engine) provides the optimal routing based on the information collected by HNM. The TE engine module is aware of the partitioning of the network. It provides load balancing by the optimal sub-domain routing for load balancing 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 by the SDN controller. He and Song (2015) provide the TE problem formulation for two cases. In the first case, legacy traffic and SDN traffic is routed in separate link capacities, whereas in the second case, the link capacity can be occupied by either SDN or legacy traffic.

Guo et al. (2014) propose an algorithm named SDN/OSPF Traffic Engineering (SOTE) to explore traffic engineering in 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 construction of DAG, the flows are split at SDN node by adding the outgoing link from SDN the node to the DAG. If a loop is formed by adding the link, that link is removed.

6.7.2. Fault tolerance

Chu et al. (2015) provide fault tolerance by redirecting the traffic to an SDN switch, in case a link-failure happens. It provides a backup IP tunnel, through SDN switch, for all the destinations from a router that is affected by a link failure.

Caria and Jukan (2016a) propose an idea of SDN partitioning, to partition the legacy network into sub-domains. In legacy networks, to deal with network failures a fraction of link capacity is preserved. This preserved link capacity is not used under normal condition. This paper shows that the proposed SDN partitioning mechanism requires less amount of spare capacity than a legacy network.

7. Related work

Nunes et al. (2014) present an overview of SDN, starting from its earlier ideas to the recent development. They also talk about challenges posed by the SDN and propose directions for future work. Kreutz et al. (2015), give a comprehensive survey on SDN. They also perform a comparative analysis of the SDN paradigm and traditional paradigm. Further, they explain the SDN layered architecture and cross-layer problems such as troubleshooting and debugging.

Xia et al. (2015) survey latest developments in SDN. They state the general definition of SDN followed by the SDN benefits and research challenges. Further, they explain the layered architecture of SDN and OpenFlow standardization and deployment.

Jarraya et al. (2014) delineate the details of SDN roots and the main components of the SDN architecture. They propose a taxonomy, where they classify the issues in SDN and research work done for each issue at each layer. They lay out the inter-layer issues as well.

Hu et al. (2014) provide the architectural details of SDN and OpenFlow. Next, they present a survey of work done by the researchers in SDN/OpenFlow. Gong et al. (2015) provide a background of SDN, SDN architecture and research work done on SDN components and applications.

All the survey papers till date discuss the latest developments and deployments related to the SDN paradigm. To the best of our knowledge, this is the first survey paper on hybrid SDN. In this paper, we have provided the various models suitable for transition to SDN from the traditional network and compared them on various characteristics such as investment, automation, traffic management and scalability. We present the different approaches taken by the researchers to deploy SDN devices in the industry and the academia. We portray the research challenges in hybridization of the two paradigms.

8. Conclusion

Although SDN has the potential to solve the present day complex operational problems of networking, there are various deployment issues. Especially for existing deployments, a smooth transition approach is required that meets the budget constraints, has a disruption-free phased transition model and can fallback safely to legacy mechan-

isms so as to build confidence. Therefore, researchers have proposed hybrid SDN models for various use cases and transition scenarios (Vissicchio et al., 2014a). Hybrid SDN can mitigate the challenges of both SDN and legacy networks and can provide models that take into account benefits of both. We present a detailed comparison of various models and techniques proposed and implemented currently by the research community and discuss challenges yet to be addressed.

References

Agarwal, A., Gupta, S., Talwar, A., 2015. A hybrid approach to networking: Integrating OpenFlow and legacy switches using OpenDayLight. URL: ⟨http://docplayer.net/32607513-Telecom-white-paper-a-hybrid-approach-to-networking-integrating-openflow-and-legacy-switches-using-opendaylight.html⟩).

Agarwal, S., Kodialam, M., Lakshman, T., 2013. Traffic engineering in software defined networks. In: INFOCOM, 2013 Proceedings IEEE, IEEE, pp. 2211–2219.

Atlas, A., Nadeau, T., Ward, D. 2013. Interface to the routing system framework, URL: ⟨https://www.ietf.org/archive/id/draft-ward-i2rs-framework-00.txt⟩).

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X., 2002. Overview and Principles of Internet Traffic Engineering. RFC, 3272, (Tech. rep.).

Balus, F., Bitar, N., Ogaki, K., Stiliadis, D., 2013. Federated sdn-based controllers for nvo3. URL: ⟨https://tools.ietf.org/html/draft-sb-nvo3-sdn-federation-01⟩.

Belter, B., Parniewicz, D., Ogrodowczyk, L., Binczewski, A., Stroiński, M., Fuentes, V., Matias, J., Huarte, M., Jacob, E., 2014. Hardware abstraction layer as an SDN-enabler for non-OpenFlow network equipment. In: 2014 Third European Workshop on Software Defined Networks, IEEE, pp. 117–118.

Benson, T., Akella, A., Maltz, D., 2009. Unraveling the complexity of network management. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, pp. 335–348. URL ⟨http://dl.acm.org/citation.cfm?Id=1558977.1559000⟩).

Boucadair, M., Jacquenet, C., 2014. Software-defined networking: A perspective from within a service provider environment. URL: <https://tools.ietf.org/html/rfc7149>

Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., van der Merwe, J., 2005. Design and implementation of a routing control platform. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, USENIX Association, pp. 15–28.

Cao, Z., Kodialam, M., Lakshman, T.V., 2014. Traffic steering in software defined networks: Planning and online routing. In: Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing, DCC '14, ACM, New York, NY, USA, pp. 65–70. ⟨http://dx.doi.org/10.1145/2627566.2627574⟩).

Caria, M., Jukan, A. 2016a. Link capacity planning for fault tolerant operation in hybrid SDN/OSPF networks. In: Global Communications Conference (GLOBECOM), IEEE, 2016, pp. 1–6.

Caria, M., Jukan, A., 2015a. The perfect match: Optical bypass and SDN partitioning, in: 2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR), IEEE, pp. 1–6.

Caria, M., Jukan, A., Hoffmann, M., 2016b. SDN partitioning: A centralized control plane for distributed routing protocols. In: IEEE Transactions on Network and Service Management, vol. 13, IEEE, pp. 381–393.

Caria, M., Das, T., Jukan, A., 2015b. Divide and conquer: Partitioning OSPF networks with SDN. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2015, pp. 467–474.

Casado, M., Garfinkel, T., Akella, A., Freedman, M.J., Boneh, D., McKeown, N., Shenker, S., 2006. Sane: A protection architecture for enterprise networks. In: Usenix Security.

Casado, M., Freedman, M.J., Pettit, J., Luo, J., McKeown, N., Shenker, S., 2007. Ethane: Taking control of the enterprise. In: ACM SIGCOMM Computer Communication Review, Vol. 37, ACM, pp. 1–12.

Casado, M., Koponen, T., Shenker, S., Tootoonchian, A., 2012. Fabric: a retrospective on evolving SDN. In: Proceedings of the first workshop on Hot topics in software defined networks, ACM, pp. 85–90.

Chu, C.-Y., Xi, K., Luo, M., Chao, H.J., 2015. Congestion-aware single link failure recovery in hybrid SDN networks. In: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, pp. 1086–1094.

Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R., 2013. Towards an elastic distributed SDN controller. In: ACM SIGCOMM Computer Communication Review, Vol. 43, ACM, pp. 7–12.

Enns, R., Bjorklund, M., Schoenwaelder, J. , 2011. Network configuration protocol (netconf), Network.

Feng, T., Bi, J., 2015. OpenRouteFlow: Enable legacy router as a software-defined routing service for hybrid SDN. In: 2015 24th International Conference on Computer Communication and Networks (ICCCN), IEEE, pp. 1–8.

Fu, Y., Bi, J., Gao, K., Chen, Z., Wu, J., Hao, B., 2014. Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks. In: 2014 IEEE 22nd International Conference on Network Protocols, IEEE, pp. 569–576.

Fuentes, V., Matias, J., Mendiola, A., Huarte, M., Unzilla, J., Jacob, E., 2014. Integrating complex legacy systems under OpenFlow control: The DOCSIS use case. In: 2014 Third European Workshop on Software Defined Networks, IEEE, pp. 37–42.

Gong, Y., Huang, W., Wang, W., Lei, Y., 2015. A survey on software defined networking and its applications. Front. Comput. Sci. 9 (6), 827–845.

Guo, Y., Wang, Z., Yin, X., Shi, X., Wu, J., 2014. Traffic engineering in sdn/ospf hybrid network. In: Network Protocols (ICNP), 2014 IEEE 22nd International Conference on, IEEE, pp. 563–568.

Hand, R., Keller, E., 2014. ClosedFlow: OpenFlowlike control over proprietary devices. In: Proceedings of the third workshop on Hot topics in software defined networking, ACM, pp. 7–12.

Hartert, R., Vissicchio, S., Schaus, P., Bonaventure, O., Filsfils, C., Telkamp, T., Francois, P., 2015. A declarative and expressive approach to control forwarding paths in carrier-grade networks. In: ACM SIGCOMM Computer Communication Review, Vol. 45, ACM, pp. 15–28.

Hartman, S., Wasserman, M., Zhang, D., 2013. Security requirements in the software defined networking model, IETF Draft (draft-hartman-sdnsec-requirements), URL: ⟨https://tools.ietf.org/html/draft-hartman-sdnsec-requirements-00⟩.

He, J., Song, W., 2015. Achieving near-optimal traffic engineering in hybrid software defined networks. In: IFIP Networking Conference (IFIP Networking), 2015, IEEE, pp. 1–9.

Hong, D.K., Ma, Y., Banerjee, S., Mao, Z.M., 2016. Incremental deployment of SDN in hybrid enterprise and ISP networks. In: Proceedings of the Symposium on SDN Research, 2016, ACM, pp. 1.

Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., Wattenhofer, R., 2013. Achieving high utilization with software-driven WAN. In: ACM SIGCOMM Computer Communication Review, Vol. 43, ACM, pp. 15–26.

Hu, F., Hao, Q., Bao, K., 2014. A survey on software-defined network and openflow: from concept to implementation. IEEE Commun. Surv. Tutor. 16 (4), 2181–2206.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., et al., 2013. B4: experience with a globally-deployed software defined wan. ACM SIGCOMM Comput. Commun. Rev. 43 (4), 3–14.

Jammal, M., Singh, T., Shami, A., Asal, R., Li, Y., 2014. Software defined networking: state of the art and research challenges. Comput. Netw. 72, 74–98.

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

Jin, C., Lumezanu, C., Xu, Q., Zhang, Z.-L., Jiang, G., 2015. Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks. In: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, ACM, p. 20.

Jin, C., Lumezanu, C., Xu, Q., Mekky, H., Zhang, Z.-L., Jiang, G. 2017 Magneto: Unified fine-grained path control in legacy and openflow hybrid networks. In: Proceedings of the Symposium on SDN Research, ACM, pp. 75–87.

Jmal, R., Fourati, L.C., 2014. Implementing shortest path routing mechanism using Openflow POX controller. In: The 2014 International Symposium on Networks, Computers and Communications, IEEE, pp. 1–6.

Kandoi, R., 2015. Deploying software-defined networks: a telco perspective.

Katiyar, R., Pawar, P., Gupta, A., Kataoka, K., 2015. Auto-configuration of SDN switches in sdn/non-sdn hybrid network. In: Proceedings of the Asian Internet Engineering Conference, ACM, pp. 48–53.

Kim, H., Feamster, N., 2013. Improving network management with software defined networking. IEEE Commun. Mag. 51 (2), 114–119.

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

Le, F., Xie, G.G., Zhang, H., 2010. Theory and new primitives for safely connecting routing protocol instances. ACM SIGCOMM Comput. Commun. Rev. 40 (4), 219–230.

Levin, D., Canini, M., Schmid, S., Schaffert, F., Feldmann. A., 2014. Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks. In: 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp. 333–345.

Lu, H., Arora, N., Zhang, H., Lumezanu, C., Rhee, J., Jiang, G., 2013. Hybnet: Network manager for a hybrid network infrastructure. In: Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference, ACM, p. 6.

Manzalini, A., Saracco, R., 2013. Software networks at the edge: A shift of paradigm, in: Future Networks and Services (SDN4FNS), 2013 IEEE SDN for, pp. 1–6. ⟨http://dx.doi.org/10.1109/SDN4FNS.2013.6702555⟩).

Gredler, H., Medved, J., Previdi, S,. Farrel, A., Ray, S., 2015. North-Bound distribution of link-state and traffic engineering (TE) information using BGP. URL ⟨https://tools.ietf.org/html/draft-ietf-idr-ls-distribution-10⟩).

Mishra, A.K., Sahoo, A., 2007. S-OSPF: A traffic engineering solution for OSPF based best effort networks. In: IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference, IEEE, pp. 1845–1849.

Mishra, A., Bansod, D., Haribabu, K., 2016. A framework for openflow-like policy-based routing in hybrid software defined networks. In: INC, pp. 97–102.

Nakahodo, Y., Naito, T., Oki, E., 2014. Implementation of smart-OSPF in hybrid software-defined network. In: 2014 4th IEEE International Conference on Network Infrastructure and Digital Content, IEEE, pp. 374–378.

Nunes, B.A.A., Mendonca, M., Nguyen, X.-N., Obraczka, K., Turletti, T., 2014. A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. 16 (3), 1617–1634.

Ochoa Aday, L., Cervelló Pastor, C., Fernández Fernández, A. Current trends of topology discovery in OpenFlow-based software defined networks.

OpenFlowDiscoveryProtocolGENI: geni. ⟨http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol⟩).

Pakzad, F., Portmann, M., Tan, W.L., Indulska, J., 2014. Efficient topology discovery in software defined networks. In: Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on, IEEE, pp. 1–8.

Pakzad, F., Portmann, M., Tan, W.L., Indulska, J., 2016. Efficient topology discovery in OpenFlow-based software defined networks. Comput. Commun. 77, 52–61.

Parniewicz, D., Doriguzzi Corin, R., Ogrodowczyk, L., Rashidi Fard, M., Matias, J., Gerola, M., Fuentes, V., Toseef, U., Zaalouk, A., Belter et al., B., 2014. Design and implementation of an OpenFlow hardware abstraction layer. In: Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing, ACM, pp.

71–76.
Pathak, A., Zhang, M., Hu, Y.C., Mahajan, R., Maltz, D., 2011. Latency inflation with mpls-based traffic engineering. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, ACM, pp. 463–472.
Qazi, Z.A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., Yu, M., 2013. Simple-fying middlebox policy enforcement using sdn. ACM SIGCOMM Comput. Commun. Rev. 43 (4), 27–38.
Rostami, A., Jungel, T., Koepsel, A., Woesner, H., Wolisz, A., 2012. Oran: Openflow routers for academic networks. In: IEEE 13th International Conference on High Performance Switchingand Routing, {HPSR} 2012, IEEE, pp. 216–222.
Saha, A.K., Sambyo, K., Bhunia, C., 2016. Topology discovery, loop finding and alternative path solution in POX controller. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 2, 2016 pp. 553–557.
Sezer, S., Scott-Hayward, S., Chouhan, P.K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N., 2013. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (7), 36–43.
Sharma, S., Staessens, D., Colle, D., Pickavet, M., Demeester, P., 2011. Enabling fast failure recovery in OpenFlow networks. In: Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the, IEEE, pp. 164–171.
Sharma, S., Staessens, D., Colle, D., Pickavet, M., Demeester, P., 2013. Fast failure recovery for in-band openflow networks. In: 9th international conference on the design of reliable communication networks (drcn), 2013, IEEE, pp. 52–59.
Pfaff, Ben, Lantz, B, Heller, Bea, et al., 2012. Openflow switch specification, version 1.3.0. Open Netw. Found..
Tilmans, O., Vissicchio, S., 2014. IGP-as-a-Backup for robust SDN networks. In: 10th International Conference on Network and Service Management (CNSM), IEEE, pp. 127–135.
Tilmans, O., Vissicchio, S., Vanbever, L., Rexford, J., 2016. Fibbing in action: On-demand load-balancing for better video delivery. In: Proceedings of the ACM SIGCOMM 2016 Conference, ACM, pp. 619–620.
Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R., 2012. On controller performance in software-defined networks. In: Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services.
Vanbever, L., Vissicchio, S., 2014. Enabling SDN in old school networks with software-controlled routing protocols. In: Presented as part of the Open Networking Summit 2014 (ONS 2014).
Vissicchio, S., Vanbever, L., Cittadin, L., Xie, G., Bonaventure, O., et al., 2013. Safe updates of hybrid SDN networks. Univ. catholique de Louvain, Tech. Rep..
Vissicchio, S., Vanbever, L., Bonaventure, O., 2014a. Opportunities and research challenges of hybrid software defined networks. ACM SIGCOMM Comput. Commun. Rev. 44 (2), 70–75.
Vissicchio, S., Vanbever, L., Cittadini, L., Xie, G.G., Bonaventure, O., 2014b. Safe routing reconfigurations with route redistribution. In: Proceedings of the IEEE INFOCOM, 2014, IEEE, pp. 199–207.
Vissicchio, S., Vanbever, L., Rexford, J., 2014c. Sweet little lies: Fake topologies for flexible routing. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, ACM, p. 3.
Vissicchio, S., Tilmans, O., Vanbever, L., Rexford, J., 2015. Central control over distributed routing. ACM SIGCOMM Comput. Commun. Rev. 45 (4), 43–56.
Wang, Y., Schapira, M., Rexford, J., 2009. Neighbor-specific bgp: more flexible routing policies while improving global stability. In: Proceedings of the Eleventh International Joint Conference on Measurementand Modeling of Computer Systems, SIGMETRICS/Performance 2009, Vol. 37, ACM, pp. 217–228.
Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H., 2015. A survey on software-defined networking. IEEE Commun. Surv. Tutor. 17 (1), 27–51.
Yeganeh, S.H., Tootoonchian, A., Ganjali, Y., 2013. On scalability of software-defined networking. IEEE Commun. Mag. 51 (2), 136–141.

Sandhya is pursuing Ph.D. as a full time scholar under the supervision of Dr. K. Haribabu, Assistant Professor, Department of Computer Science & Information Systems, Birla Institute of Technology and Science, Pilani, India. Her research interests are Computer Networks and Software Defined Networks.

Yash Sinha received his M.Sc.(Tech.) in Information Systems from Birla Institute of Technology & Science, Pilani, India in 2016. Currently, he is pursuing Masters of Engineering in Computer Science at Birla Institute of Technology & Science, Pilani, India. His research interests are Software Defined Networks, hybrid SDN, peer-to-peer networks and machine learning.

K Haribabu is currently working as Assistant Professor in the Department of Computer Science & Information Systems, at Birla Institute of Technology & Science, Pilani, India. He has completed his Ph.D. in 2012 from BITS Pilani. His areas of interests are P2P systems, Software Defined Networking and cyber-physical systems. He is a member of IEEE and ACM.