Opportunities and Research Challenges of Hybrid Software Defined Networks

Stefano Vissicchio
Universite catholique de Louvain

Laurent Vanbever
Princeton University

Olivier Bonaventure
Universite catholique de Louvain

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

Software Defined Networking (SDN) promises to ease design, operation and management of communication networks. However, SDN comes with its own set of challenges, including incremental deployability, robustness, and scalability. Those challenges make a full SDN deployment unlikely in the short-term and possibly inconvenient in the longer-term.

In this paper, we explore hybrid SDN models that combine SDN with a more traditional networking approach based on distributed protocols. We show a number of use cases in which hybrid models can mitigate the respective limitations of traditional and SDN approaches, providing incentives to (partially) transition to SDN. Further, we expose the qualitatively diverse tradeoffs that are naturally achieved in hybrid models, making them convenient for different transition strategies and long-term network designs. For those reasons, we argue that hybrid SDN architectures deserve more attention from the scientific community.

1. INTRODUCTION

Judging from the increasing attention of both industry and academia to Software Defined Networking (SDN), communication networks seem on the verge of a paradigm shift. For years, networks have been relying on devices that are realized (and locked) by vendors, and can be only configured by operators. SDN defines a new architecture in which a custom software SDN controller exerts a logically-centralized control of the whole network. This new architecture promises to simplify network management, spur innovation, and make networks more powerful and flexible [1].

SDN comes with its own set of challenges and limitations, ranging from (business, economic and technical) deployment obstacles to concerns on logic centralization guarantees, e.g., in terms of resilience, robustness and scalability. Those factors are currently contributing to limit SDN deployments to a very small number of cases in which SDN is adopted in specific subnetworks (see, e.g., [2]). In the future, they can also make a full SDN adoption inconvenient in many cases.

In this paper, we show that combining SDN and traditional architectures in *hybrid SDN* models has the potential to sum their benefits while mitigating their respective challenges. Indeed, protocols and techniques developed in the traditional networking approach can provide working solutions for some technical difficulties of SDN. For example, a per-device control-plane naturally helps to i) quickly react to failures, e.g., by relying on local decisions; ii) update the control-plane, e.g., performing per-device changes after having diverted the traffic from the currently updated device; and iii) improve scalability, e.g., by spreading control-plane decisions on multiple devices.

In the following, we review the traditional and SDN architectures, and we identify main inertial factors for SDN deployment (§2). Those inertial factors highlight the practical need for supporting retro-compatibility and SDN incremental deployment, as also testified by the recent creation of industrial working groups on hybrid SDN devices (see, e.g., [3]). Then, we describe different hybrid SDN models (§3). For each model, (i) we define the interaction between the SDN controller and the distributed control-plane built by non-SDN devices, (ii) we identify related works, and (iii) we discuss use cases for both short-term (e.g., transition to SDN) and long-term (e.g., network design) goals. We also provide a qualitative comparison of the analyzed models with respect to several dimensions, including expressiveness, robustness, scalability and management complexity (§4). Our comparison sheds light on the variety of tradeoffs that are natively achieved by the different hybrid models. As such, each model looks especially suitable for a specific transition strategy and given architectural needs of a certain class of network operators. Our use cases and tradeoff analysis further suggest that hybrid SDN models can represent a valid complement or even an alternative to recent SDN research efforts, including extension of SDN protocols with more and more features (e.g., OpenFlow v1.3 [4]) and implementation of SDN controllers as distributed systems (e.g., for resiliency [5] or for partial decentralization of control-plane decisions [6]). Finally, we discuss basic research challenges and open problems in hybrid SDN networks (§5).

2. SDN: INCENTIVES AND INERTIA

A communication network can be seen a system with few architectural elements, assembled together to provide network *services*, like best-effort packet delivery, access control, tunnelling or data encryption. The basic elements of a network are *nodes* (e.g., switches, routers, load-balancers), and *interconnections* (both physical links and protocol-dependent logical adjacencies) between nodes. Nodes interact on two logically different planes. On the *data-plane*, nodes forward data packets (possibly after modification) to their respective next-hops, relying on a data structure called Forwarding Information Base (FIB). Higher-level decisions on edge-to-edge paths and traffic flow forwarding are taken at the *control-plane* level, which is typically realized in software. The control-plane controls the data-plane by updating the FIB of each node.

For years, most networks have been designed, deployed

	CN	SDN
required nodes	COTS with local control-plane	(possibly virtual) hardware with FIB
interface to node FIBs	node control-plane configuration	programmatic FIB interface
control-plane computation	distributed on each node	logically-centralized controllers
control-plane software	proprietary, closed	custom, open

Table 1: Main differences between CN and SDN paradigms.

and managed as COTS-based systems. A COTS (Commercial Off-The-Shelf) is a product realized, supplied and evolved by a vendor while being used without modification by its acquirer. The acquirer is then responsible for the integration of possibly heterogeneous proprietary COTSes. In the traditional network architecture, nodes are COTSes whose hardware realizes a local data-plane while proprietary software implements a portion of the network control-plane. Although their implementation cannot be modified, the behavior of COTS nodes can be influenced by operators by tweaking node configurations. Each node configuration defines the network protocols activated on that node. Moreover, it specifies protocol parameters to fine-tune information exchanges and criteria according to which nodes modify and forward data packets. Despite the huge amount of research work in network management, node configurations are still long text files written in obscure vendor-specific languages. Further, supported protocols and parameters that can be configured on commercial nodes are decided by vendors. In the following, we refer to this network design and management model as *COTS Networking (CN)*.

Recently, *SDN (Software Defined Networking)* has been proposed as an alternative to CN [1]. The main differences between CN and SDN are summarized in Table 1. Primarily, SDN is predicated around the separation between control-plane and data-plane. In SDN, network nodes implement only the data-plane, while a separated architectural element, called *SDN controller*, realizes the control-plane. The SDN controller is a logically-centralized custom software, possibly corresponding to a distributed system (see, e.g., [5, 6]). The independence between the controller and the nodes simplifies the development of a high-level management interface, e.g., based on declarative languages (see, e.g., [7]). Moreover, while nodes need to be complete products in CN, SDN can rely on simpler hardware that exposes a programmatic interface to the FIB (e.g., OpenFlow switches [4]) or even on virtual devices (e.g., virtual switches running on servers [8]). In the following, we distinguish between CN and SDN nodes, depending on their ability to support SDN. In particular, we consider a switch as an SDN node if it supports SDN protocols (e.g., OpenFlow [4]), and as a CN node otherwise. Moreover, given their intrinsic architectural differences, we refer to SDN and CN as *paradigms*.

By centralizing and customizing the control-plane, SDN promises to ease network design and management. However, discussions with operators highlighted major concerns for a wide adoption of SDN. They encompass both the transition from CN to SDN, and the SDN paradigm itself.

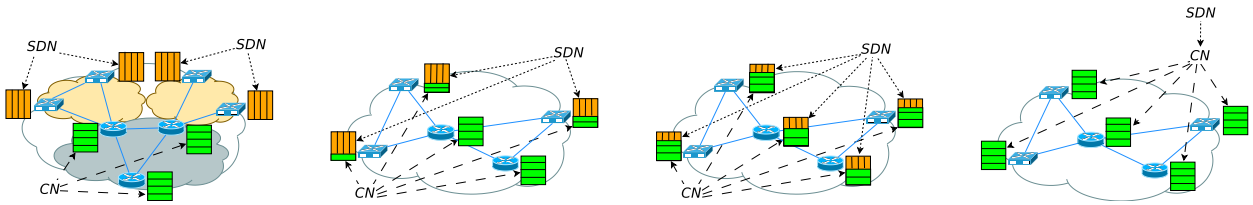
Regarding the transition, SDN deployment in existing networks poses economical, organizational and technical challenges. First, SDN has non-negligible initial deployment costs, in terms of equipment renovation and lack of expertise. To amortize huge investments, operators are generally reluctant to dismiss expensive CN nodes just to enable a full SDN deployment. Moreover, since SDN requires a radical change in their mental model, operators will need training to design, update, debug and operate SDN networks, or will need to be flanked by a new generation of network programmers. Second, production-level SDN controllers still seem hard to realize. Despite recent efforts to provide high-level network programming languages (e.g., [7]) and powerful abstractions (e.g., [9]), provably effective methodologies and tools to build a reliable SDN controller enforcing complex network policies are still not available. Hence, operators may feel puzzled on how to implement (or interconnect third-party) SDN controllers that will likely need to guarantee strict objectives, like security policy enforcement, extremely high availability and the lowest possible delay (at least for critical traffic), for any possible set of input packets. Those problems threaten to make the SDN transition very long, and to hamper full SDN deployments.

About the SDN paradigm itself, while mitigating the risk of local inconsistencies between nodes, logic centralization exacerbates architectural concerns, like reliability, robustness and scalability. Some examples follow. To ensure reliability, fast and expensive out-of-band wide area networks between SDN controllers and nodes will be needed in large networks, e.g., to have updated information on failures or on new incoming flows. This would double network design and management. Indeed, current out-of-band networks are typically simple local area networks with loose requirements, as they are used for non-critical tasks (e.g., periodical SNMP trap gathering) and in the few cases of major in-band connection disruptions. Logic centralization can also complicate control-plane management tasks like its own upgrade, e.g., to deploy a new network application or to install a new version of the controller software. During such an upgrade, no SDN control-plane is available, e.g., for failure recovery or new flow processing, likely leading to wide network unavailability [10]. Finally, scalability concerns apply to both hardware and software of SDN controllers, especially in large or highly-dynamic networks where controllers have to take quick decisions for all the network nodes upon a huge variety (and possibly frequency) of events, including failures, traffic demand changes, and new incoming flows.

3. HYBRID SDN MODELS

In this section, we define hybrid SDN models, and we describe use cases to which each of them naturally applies. The proposed models differ in the role respectively assigned to CN and SDN. In particular, we classify hybrid models on the basis of which paradigm provides which network service, by controlling FIB entries on which nodes.

Fig. 1 collects examples of the different hybrid SDN models. In the figure, blue icons, segments and rectangles represent nodes, physical links, and node FIBs, respectively. The horizontal and vertical filling patterns identify the portions of the FIBs respectively controlled by CN and SDN. For instance, a horizontally filled rectangle indicates that CN is responsible for all the entries in that FIB. The individual examples in the figure are described in more details in the following subsections.



(a) Topology-based. (b) Service-based. (c) Class-based. (d) Integrated.

Figure 1: Examples of hybrid SDN models.

3.1 Topology-Based Hybrid SDN

The topology-based hybrid SDN (TB hSDN) model relies on a topological separation of the nodes controlled by each paradigm. More precisely, the network is partitioned in zones, so that each node belongs to only one zone. A zone is a set of interconnected nodes controlled by the same paradigm. We refer to any zone in which nodes are controlled by SDN (resp., CN) as SDN zone (resp., CN zone). In SDN (resp., CN) zones, all the services are provided by SDN (resp., CN). A composition of the services realized by CN and SDN in different zones is then needed to implement cross-zone services, e.g., forwarding between any pair of source and destination in the network.

An example of TB hSDN is depicted in Fig. 1a. The figure represents a network divided in three zones, namely, two SDN zones (i.e., the yellow clouds) and one CN zone (i.e., the gray cloud). The example matches a case in which SDN is adopted in specific subnetworks, e.g., only in the backbone in order to maximize bandwidth utilization as in [2, 11]. A pair of general use cases for TB hSDN follow.

Transition use case. TB hSDN naturally fits a transition strategy in which SDN is introduced on a per-region basis, e.g., initially limiting SDN to specific countries and progressively extending its deployment to others. During this process, the portion of traffic managed by the SDN zones can be progressively increased in parallel to i) maturity of the technology ii) SDN node deployment, and iii) acquisition of expertise from operators. In particular, experimental deployments in which a controller (or one of its modules) manages non-critical traffic are easily accommodated. For instance, the example in Fig. 1a can represent an intermediate step during the SDN transition in which data centers are controlled by SDN, while the backbone is managed by CN for reliability and compliance with operator expertise.

Long-term design use case. Many networks, especially in enterprises, are already divided in several routing domains [12]. Reasons vary from business (e.g., merger and acquisitions) and organizational (e.g., Network Operation Centers with different expertise) to technical (e.g., possibility to leverage specific features of different protocols in subnetworks with different requirements) ones. By confining SDN and CN in disjoint network portions, the TB hSDN model generalizes this design. Moreover, this architecture eases the introduction of mechanisms to exchange information between SDN zones and between SDN and CN zones, and their replacement with new ones. For example, interconnection mechanisms safer (e.g., [13]) and more flexible (e.g., [14]) than current ones (i.e., route redistribution and BGP) can be implemented by relying on custom SDN controllers that propagate information between different zones.

3.2 Service-Based Hybrid SDN

In the service-based hybrid SDN (SB hSDN) model, CN and SDN provide different services. To implement some services, like network-wide forwarding, the two paradigms can span an overlapping set of nodes, controlling a different portion of the FIB of each node. Nevertheless, some nodes can be exclusively controlled by a single paradigm, e.g., to realize services like load-balancing or edge-to-edge tunneling.

An example of SB hSDN network is reported in Fig. 1b. In this example, SDN fills most FIB entries of nodes at the border of the network, while CN has an exclusive control of the entire FIB of internal nodes. As such, network-wide services like forwarding are delegated to CN, while edge-to-edge services like traffic engineering and access policies [8] or those requiring full traffic visibility like monitoring [15] are assigned to SDN. More generic use cases follow.

Transition use case. During a progressive transition to SDN, only some SDN nodes will generally be available. While they may not be used to implement the full set of services, SDN nodes can be strategically placed so that they can provide or improve a subset of services. Hence, SB hSDN enables a service-based transition, in which the SDN controller realizes more and more services at each step, by handling an increasing fraction of nodes (or node FIBs). For instance, few SDN nodes can be initially deployed to improve load balancing and overcome rigidities of CN protocols (e.g., simplifying local violations of shortest path routing) as studied in [16, 17]. By improving traffic engineering and easing declarative reaction to demand changes, this initial SDN deployment can be an incentive for operators to start the transition. The role assigned to the SDN controller can be progressively extended when other SDN nodes will be added.

Long-term design use case. SDN promises to support new network services, like Network Function Virtualization. However, operators may be willing to keep proved CN protocols and technologies for some services, like MPLS Virtual Private Networks (VPNs) and IPSec encrypted connections, instead of relying on new software to be integrated in their SDN controllers. In this case, the division of services may represent a long-term design choice, in which the SDN paradigm is used for services that CN solutions cannot satisfactorily provide.

3.3 Class-Based Hybrid SDN

The class-based hybrid SDN (CB hSDN) model is centered around the partition of traffic in classes, and the division of those classes into CN-controlled and SDN-controlled. Contrary to TB hSDN, CN and SDN typically span all the network nodes in this model, controlling a disjoint set of FIB entries on each node. Moreover, in contrast with the SB hSDN, each paradigm realizes all the network services

for the traffic classes assigned to it.

An example of CB hSDN is illustrated in Fig. 1c, where SDN (resp., CN) fills the FIB entries on each node to control a small (resp., big) portion of traffic. The SDN-controlled classes can be defined in terms of TCP flows (e.g., to implement fine-grained load-balancing for flows attracting most traffic), applications (e.g., to guarantee low delay on interactive applications), business cases (e.g., to provide VPN services while avoiding limitations of MPLS RSVP-TE [2]), or a composition of them. More general use cases follow.

Transition use case. A traffic-based transition strategy consists in installing an SDN controller and progressively delegating the handling of more traffic to it. This strategy is especially effective if many nodes are SDN-enabled (e.g., as if commercial switches or routers will support OpenFlow). Further, it is helpful for both testing controllers on non-critical traffic classes and limiting scalability problems at the beginning of the transition. Throughout such a transition, deployed CN protocols can work as a backup, e.g., in case of failure or unexpected behavior of the SDN controller. Also, depending on the number and position of SDN nodes, SDN-specific features (e.g., added flexibility via easy match of any packet field) can be leveraged on SDN-controlled classes.

Long-term design use case. The CB hSDN model may be adopted as a long-term design, especially if SDN protocols will be supported by CN nodes, e.g., through a software upgrade. In this design, operators can naturally leverage the typical disproportion between the number and the importance of traffic classes. For example, few flows typically attract most of the traffic (see, e.g., [18]). Those highly-attractive flows can be grouped in SDN-controlled classes, in order for the SDN controller to flexibly configure load balancing and dynamically change forwarding paths without the need for CN configuration tweaking. Alternatively, SDN-controlled classes can gather traffic for which security (e.g., access policies) or premium (e.g., optimal routing) services have to be offered. In the latter case, the SDN controller can reserve some forwarding paths, apply arbitrary traffic engineering or assign higher priority to traffic in premium classes, while CN protocols can deliver best-effort services for the remaining packets. Note that the assignment of classes to SDN can be decided by the operator in a declarative way, e.g., by specifying values of packet fields (e.g., all traffic towards TCP port 80) or a threshold for the amount of traffic corresponding to SDN-controlled classes.

3.4 Integrated Hybrid SDN

In all the previous hybrid models, CN and SDN complement each other by controlling disjoint parts of node FIBs. The Integrated hybrid SDN (or Integrated hSDN) breaks this symmetry. In this model, SDN is responsible for all the network services, and uses CN protocols as an interface to node FIBs. For example, it can control forwarding paths by injecting carefully-selected routes into a routing system or adjusting protocol settings (e.g., IGP weights). In practice, the FIB of any node is maintained by the CN paradigm, which is in turn controlled by an SDN controller.

An example of Integrated hSDN is in Fig. 1d. The figure actually depicts a general architecture that can be realized in different ways, ranging from BGP-based [19] or MPLS-based [20] controllers to systems leveraging unified interfaces to the routing system [21, 22]. The example also highlights that SDN nodes are unnecessary in this model, making Integrated hSDN suitable for the following use cases.

Transition use case. The independence of the Integrated hSDN model from the presence of SDN nodes makes it fitting a control-plane based transition strategy. This strategy includes two macro-steps. In the first step, only the control-plane is moved from CN to SDN, generally reducing the costs and the disruption risks of the initial SDN deployment. Indeed, equipment addition or replacement are unnecessary at this step, hence operators can acquire confidence in SDN while relying on traditional well-known protocols. In the second step, the data-plane is (progressively) changed, adding SDN nodes and updating the SDN controller in parallel to the SDN deployment.

Long-term design use case. From an SDN controller perspective, a CN protocol represents an interface to the node FIBs, in this model. Such an interface is more complex than current SDN proposals (e.g., OpenFlow) but it also provides different primitives. On one hand, Integrated hSDN controllers will need to manage protocol-specific aspects (e.g., message format) and mechanisms (e.g., convergence algorithms). On the other hand, they can be offloaded from complex tasks, like the computation of temporary forwarding paths to ensure connectivity in case of failures, that can be directly managed by the CN configuration. The latter feature can be the main driver for the adoption of Integrated hSDN as a long-term architecture.

4. TRADEOFF ANALYSIS

We now provide a more in-depth comparative analysis of the presented hybrid SDN models. We consider the following dimensions: i) *expressiveness* and *management simplicity*, in terms of easiness of non IP-based forwarding and enforcement of middlebox policies; ii) *robustness* and *scalability*, as architectural concerns; iii) *deployment costs*, in terms of hardware upgrade costs, custom software to be produced, and needed expertise; iv) *flexibility* and internal complexity, especially as related to the coexistence of multiple paradigms. This set of dimensions does not pretend to be complete. Rather, it is intended to show that diverse tradeoffs can be achieved by hybrid models on practically relevant networking aspects. Such a tradeoff diversity makes hybrid SDN networks potentially suitable for more use cases than those described in the previous section. We expect that a similar analysis can be useful for network operators to clarify which architecture (CN, SDN, or which hybrid one) can better fit her specific needs.

The results of our analysis are summarized in Table 2. In the table, comparison dimensions and network architectures are respectively disposed on rows and columns. Rows are divided in three groups, corresponding to the hardest challenges to deal with in CN (first group), SDN (second group), and hybrid SDN (third group) models.

General hybridization benefits: With respect to CN networks, hybrid models enable flexibility (e.g., easy match on any packet field for middleboxing) and SDN-specific features (e.g., declarative management interface). At the same time, they partially inherit robustness, scalability, technological maturity and low deployment costs from the CN paradigm. For instance, in TB hSDN, the full expressiveness and high manageability of SDN can be leveraged in the SDN zones, while re-routing tasks and improved scalability can be delegated to distributed protocols in CN zones. Similarly, in CB hSDNs, SDN features are available for critical

	CN	TB hSDN	SB hSDN	CB hSDN	Integrated	SDN
non IP-based forwarding	hard, complex configuration	programmable in SDN zones	programmable for SDN services	programmable for SDN traffic	very hard (e.g., BGP FlowSpec)	programmable
traffic steering, middleboxing	hard (e.g., box replication)	programmable in SDN zones	programmable for SDN services	programmable for SDN traffic	programmable by the controller	programmable
scalability and robustness	by CN protocols	by CN protocols in CN zones	by CN protocols for CN services	by CN protocols for CN traffic	possibly, by CN protocols	SDN controller concern
required custom software	none	controllers of SDN zones	controllers for SDN services	controllers for SDN flows	SDN controller	SDN controller
upgrade costs (hw, sw, expert)	none	partial, progressive renovation	partial, progressive renovation	partial, progressive renovation	none	global renovation
paradigm interaction	none	control-plane collaboration	data-plane visibility	control-plane coordination	control-plane integration	none

Table 2: Tradeoff comparison.

traffic classes only, but the controller can ignore the impact of events (e.g., failures) on the rest of the traffic. In SB hSDN, the SDN controller can implement only the services that would require complex CN configurations. This would reduce the number of events that SDN has to handle, which in turn would improve scalability of the controller and reduce hardware requirements, e.g., on the out-of-band network connecting controller and nodes. More in general, hybrid SDN enables simplification of both CN configuration and SDN software, and adoption of each paradigm to tackle challenges for which it is intrinsically more suitable.

Integrated hSDN represents an exception with respect to the other hybrid models, mainly because of the asymmetric roles assigned to CN and SDN. Since only CN protocols are used to access node FIBs, SDN-specific features (like programmatic and direct access to node FIBs) cannot be included in the Integrated model. However, for cases in which destination-based forwarding is sufficient to provide all the network services, Integrated hSDN has some advantages with respect to the other hybrid models, including centralized declaration and computation of all network service and absence of equipment upgrade costs.

Tunable tradeoffs: Architectural tradeoffs of each model can be fine-tuned according to some parameters.

Among them, the number and the topological position of the deployed SDN nodes play a crucial role. On one hand, the deployment of SDN nodes influences the possibility to realize SDN-specific features (e.g., direct declarative FIB control) for given traffic. For example, arbitrary SDN-decided tags can be added to packets only if the corresponding forwarding path traverses appropriately-configured SDN nodes. On the other hand, deploying SDN nodes has a cost, e.g., in terms of hardware addition (or software upgrades if SDN protocols will be supported by commercial CN nodes), topology modification and know-how acquisition.

Another important parameter for fine-grained tradeoffs is represented by the fraction of traffic controlled by SDN. Indeed, a higher amount of SDN-controlled traffic corresponds to the possibility of leveraging SDN-specific features (e.g., declarative interface) more widely, but it also increases architectural challenges (e.g., scalability) for the SDN controller. For example, larger SDN zones in TB hSDN and more SDN-controlled classes in CB hSDN translate to more network events to be managed by the SDN controller.

Again, Integrated hSDN is an exception, since the expressivity of the model does not depend on deployed SDN nodes and all the traffic is SDN-controlled, by model definition.

Hybridization drawbacks: While combining CN and SDN enables new fine-tunable tradeoffs, hybrid models have their own peculiar drawbacks. Among them, the need for managing heterogeneous paradigms and ensuring profitable interaction between them is particularly relevant, since it affects the realizability of network-wide services. The impact of such heterogeneity actually depends on the model. It is rather low in CB hSDN, where the paradigm interaction is restricted to coordination for specific operations, like moving traffic flows from a CN-controlled class to an SDN-controlled one, or vice versa. The interaction remains quite loose in SB hSDN, where each paradigm may need no more than visibility on the FIB entries configured by the other paradigm (e.g., to prevent conflicting data-plane decisions). Control-plane coordination is also needed if an SDN-controlled service has to become CN-controlled, and vice versa. In contrast, TB hSDN and Integrated hSDN require stronger forms of paradigm interaction. In TB hSDN, control-plane collaboration is needed to realize cross-zone services, e.g., network-wide forwarding with no loops. In the Integrated hSDN, the two paradigms are tightly coupled by definition of the model, as SDN relies on CN protocols to program node FIBs. This asks for control-plane integration.

Combination of hybrid models: A wider range of tradeoffs can be obtained by combining hybrid models together. For example, an operator may adopt a TB hSDN model that splits the whole network in a CN zone (e.g., that supports legacy nodes), a SB hSDN zone (e.g., to provide premium services to some customers), and a pure SDN zone (e.g., in its data centers). Similarly, the combination of hybrid models can lead to more articulated transition strategies. For instance, an operator might nest a service-based transition into an Integrated hSDN global strategy. In this case, an SDN controller realizing an Integrated hSDN model would be installed in a first macro-step. A second macro-step would consist in deploy the SDN data-plane by progressively assigning new services to the SDN controller in parallel to the deployment of new SDN nodes.

5. RESEARCH CHALLENGES

Our tradeoff analysis suggests that the combination of centralized and distributed paradigms can provide mutual benefits. Future work is needed to devise techniques and interaction mechanisms that maximize such benefits while limiting the added complexity of the paradigm coexistence.

We now discuss some open research challenges.

First, we envision that services hardly implementable with a single paradigm can be realized by conveniently combining CN and SDN. The identification, characterization and implementation (e.g., via cross-paradigm techniques) of those services need to be studied.

Second, a (partial) control-plane redesign would be needed to allow effective cooperation between architectural elements of different paradigms, like an SDN controller and a CN control-plane. To this end, both current network theory and control-plane protocols likely need to be extended, e.g., to take into account the simultaneous and collaborative presence of centralized and decentralized routing systems, and multiple forms of access control (e.g., commercial firewall rules and OpenFlow switch entries).

Third, the added complexity of having multiple paradigms can hamper the development of SDN abstractions in hybrid networks. Two examples are represented by safe network updates and declarative networking. In the first case, safe update techniques tailored to either CN (e.g., [23]) or SDN (e.g., [9]) networks will have to be extended, or new techniques to be proposed. Indeed, while complicating the problem, the presence of paradigm interaction mechanisms may not prevent provably-safe update procedures to be devised [24]. In the second case, a system integrating SDN programming languages with CN management primitives into a unified declarative interface is missing. Such an integrated management system will probably have to include a heterogeneity adaptation layer, e.g., to match a desired output to the actual capabilities of deployed CN and SDN nodes.

The challenges pointed out so far apply to all the hybrid models. Nevertheless, other challenges depend on the specific (combination of) hybrid models. Few examples follow. In the TB hSDN model, the most critical problem is probably the design of a safe control-plane interaction mechanism that allows to maximize the flexibility of SDN in SDN zones while preserving network-wide correctness properties (e.g., absence of forwarding loops). The design of such a paradigm interaction mechanism is likely to be challenging, as testified by CN protocol interconnection primitives (e.g., [13]). Cross-service and cross-paradigm techniques will need to be defined in the SB hSDN model, so that FIB entries can be optimized for all the services rather than on a per-service basis. In the CB hSDN model, algorithms and mechanisms to transfer traffic class control from one paradigm to another are still missing. Moreover, the SDN controller needs to interact with possible CN nodes present in the network, e.g., by relying on configuration protocols (e.g., [21]) or on commercial device SDKs (e.g., [25]). Finally, the practicality of the Integrated hSDN model depends on the possibility to characterize primitives and functions that each traditional protocol (e.g., in typical and safe configurations) can offer to an SDN controller.

More generally, the increasing relevance and frequency of partial deployment problems (e.g., SDN and IPv6 rollout) suggests the need for a still missing theory for mixed paradigm deployments in networking. While tailored to SDN partial deployment, our hybrid models seem a good starting point in this direction.

6. REFERENCES

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

[2] S. Jain et al., "B4: experience with a globally-deployed software defined wan," in *SIGCOMM*, 2013.

[3] Open Networking Foundation, "Outcomes of the Hybrid Working Group," 2013.

[4] ——, "ONF Web site," https://www.opennetworking.org.

[5] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks," in *OSDI*, 2010.

[6] A. Ferguson et al., "Hierarchical policies for software defined networks," in *HotSDN*, 2012.

[7] N. Foster et al., "Frenetic: a network programming language," *SIGPLAN Not.*, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[8] VMware Inc., "VMware NSX The Platform For Network Virtualization," Datasheet, 2013.

[9] M. Reitblatt et al., "Abstractions for network update," in *SIGCOMM*, 2012.

[10] L. Vanbever et al., "Hotswap: correct and efficient controller upgrades for software-defined networks," in *HotSDN*, 2013.

[11] C.-Y. Hong et al., "Achieving high utilization with software-driven wan," in *SIGCOMM*, 2013.

[12] F. Le et al., "Shedding light on the glue logic of the internet routing architecture," in *SIGCOMM*, 2008.

[13] ——, "Theory and new primitives for safely connecting routing protocol instances," in *SIGCOMM*, 2010.

[14] Y. Wang et al., "Neighbor-specific bgp: more flexible routing policies while improving global stability," in *SIGMETRICS*, 2009.

[15] Big Switch Networks, "Open SDN for Network Visibility," solution guide, 2013.

[16] D. Levin et al., "Panopticon: Reaping the Benefits of Partial SDN Deployment in Enterprise Networks," TU Berlin / T-Labs, Tech. Rep., 2013.

[17] S. Agarwal et al., "Traffic engineering in software defined networks," in *INFOCOM*, 2013.

[18] N. Sarrar et al., "Leveraging zipf's law for traffic offloading," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 1, pp. 16–22, 2012.

[19] M. Caesar et al., "Design and implementation of a routing control platform," in *NSDI*, 2005.

[20] W. Henderickx et al., "Federated SDN-based Controllers for NVO3," Internet Draft, 2013.

[21] R. Enns et al., "Network Configuration Protocol (NETCONF)," RFC 4741, 2011.

[22] A. Atlas et al., "Interface to the routing system framework," Internet Draft, 2013.

[23] L. Vanbever et al., "Lossless Migrations of Link-State IGPs," *IEEE/ACM Trans. on Netw.*, vol. 20, no. 6, pp. 1842–1855, 2012.

[24] S. Vissicchio et al., "Safe Routing Reconfigurations with Route Redistribution," in *INFOCOM*, 2014.

[25] J. Kelly et al., "Rapid Service Creation Using the JUNOS SDK," in *PRESTO*, 2009.