ACCEPTED FROM OPEN CALL

Interfaces, Attributes, and Use Cases: A Compass for SDN

Michael Jarschel, Thomas Zinner, Tobias Hoßfeld, Phuoc Tran-Gia, and Wolfgang Kellerer

ABSTRACT

The term Software Defined Networking (SDN) is prevalent in today's discussion about future communication networks. As with any new term or paradigm, however, no consistent definition regarding this technology has formed. The fragmented view on SDN results in legacy products being passed off by equipment vendors as SDN, academics mixing up the attributes of SDN with those of network virtualization, and users not fully understanding the benefits. Therefore, establishing SDN as a widely adopted technology beyond laboratories and insular deployments requires a compass to navigate the multitude of ideas and concepts that make up SDN today.

The contribution of this article represents an important step toward such an instrument. It gives a thorough definition of SDN and its interfaces as well as a list of its key attributes. Furthermore, a mapping of interfaces and attributes to SDN use cases is provided, highlighting the relevance of the interfaces and attributes for each scenario. This compass gives guidance to a potential adopter of SDN on whether SDN is in fact the right technology for a specific use case.

PRINCIPLES OF SOFTWARE DEFINED NETWORKING

How networks are currently structured and operated poses a significant financial issue to Internet service providers and, in fact, has become a handicap for progress in the cloud and service provider space. SDN [1] enables a programmable network control and offers a solution to a variety of use cases. The success stories of these bottom-up SDN solutions have led to a shift in the way operators and vendors perceive the network. In the following we define four basic principles of SDN. Each of these principles is mandatory for classifying a technology as SDN.

SEPARATION OF CONTROL- AND DATA PLANE

The physical separation of the control- and forwarding- or data plane is the best-known principle of SDN [1, 2]. It postulates the externalization of the control plane from a network device to an external control plane entity often called the "controller." In particular this means that an internal software control plane, while it may still exist, is not enough to brand a device or technology as "Software Defined Networking." The external controller has to have the ability to change the forwarding behavior of the network element directly. This enables several key benefits of SDN. The control- and data plane can be developed separately from each other, which lowers the entry-to-market hurdle, as a company no longer has to have expert knowledge in both areas. Moreover, the externalization of a software-based controller produces pressure on established hardware switch vendors, which are reduced to providing forwarding hardware only. This has already introduced new and disruptive start-ups to the market that have sped up innovation in the network. Even the market leader Cisco has reacted to this trend by introducing its own flavor of SDN with the Application Centric Infrastructure concept developed at the Spin-In company "Insieme." Customers are also enabled to "mix-and-match" products of different vendors and thus increase competition further. The switch vendors have reacted to the growing interest in SDN, forming the OpenDaylight project for an open SDN software platform.

Challenges in this area are to find the appropriate control protocol for the specific scenario out of different protocols and protocol versions, and the appropriate forwarding elements that support this protocol.

LOGICALLY CENTRALIZED CONTROL

The controller of an SDN network is a logically centralized entity, that is, it can consist of multiple physical or virtual instances, but behaves like a single component. The global network information such a central controller possesses enables it to adapt its network policy with respect to routing and forwarding much better and faster than a system of traditional routers could.

The realization of a logically centralized controller is challenging with respect to scalability depending on the specific scenario and network or virtual network size. Scalability can be achieved by implementing a centralized controller as a distributed system where the contained information has to be maintained consistently.

Michael Jarschel, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia are with University of Würzburg.

Wolfgang Kellerer is with Technische Universität München.

0163-6804/14/$25.00 © 2014 IEEE

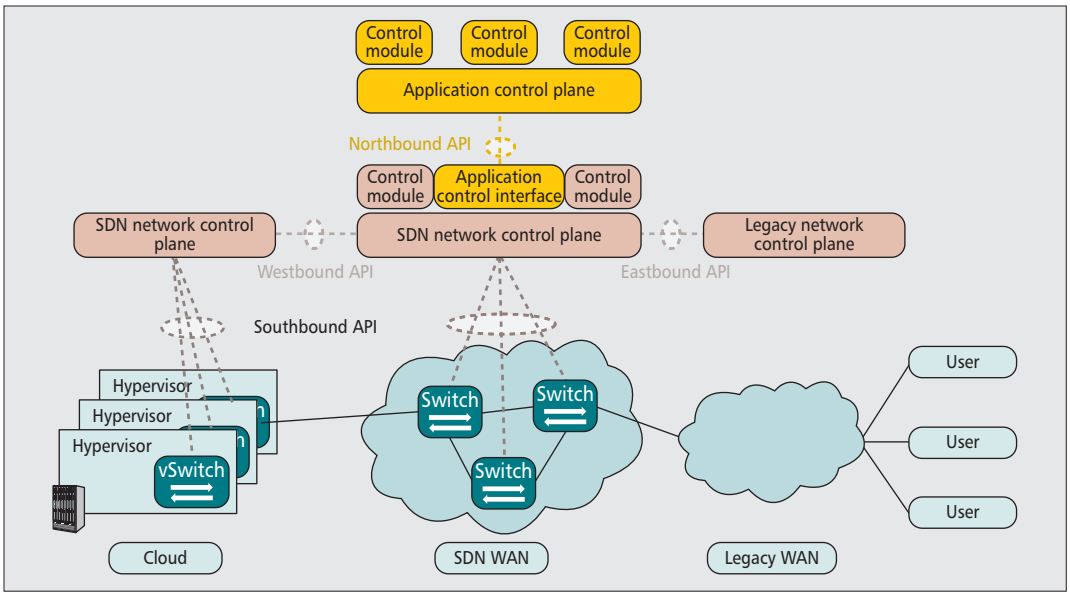

Figure 1. Example: interfaces of a software defined network.

To maintain open interfaces might be challenging since vendors try to introduce proprietary interfaces or to bypass proprietary information via the open interface. This could generate additional value if entities of the same vendor are used, but also lead to deadlocks and performance bottlenecks in mixed operation.

Open Interfaces

For SDN to reach its full potential in terms of flexibility and adaptability, it is fundamental that its interfaces are and remain open. A closed or proprietary interface limits component exchangeability and innovation. This is especially true for the interface between control- and data plane (Southbound Interface). In the absence of a standard open interface, one of the main SDN advantages — the interchangeability of network devices and control planes — would be taken away. This is also true for the remaining interfaces, which are discussed in more detail in the next section.

To maintain open interfaces might be challenging since vendors try to introduce proprietary interfaces or to bypass proprietary information via the open interface. This could generate additional value if entities of the same vendor are used, but also lead to deadlocks and performance bottlenecks in mixed operation.

Programmability

The fundamental paradigm shift in networking SDN has initiated is represented by programmability. It is enabled by the external software controller and open interfaces. The programmability principle is not limited to introducing new network features to the control plane, but rather represents the ability to treat the network as a single programmable entity instead of an accumulation of devices that have to be configured individually. SDN can thus be regarded as a very suitable complement to network virtualization, providing the control plane for an easy operation ("programming") of, for example, virtual networks in network substrates or to control specific flows within a virtual network as possible applications.

Here it is essential to find the appropriate abstraction level, which determines on the one hand the ease-of-use for network programmers, and on the other hand the abstraction overhead and therewith a possible performance degradation.

Key Interfaces and Features of Software Defined Networking

Key components of the SDN compass are its interfaces and features.

Definition and Significance of SDN Interfaces

The four key interfaces of Software Defined Networking are illustrated in Fig. 1 for a generic, example network, consisting of three autonomous systems (AS): a conventional IP or legacy access network at the user end, an SDN-based transit-WAN, and an SDN-enabled data center network (cloud).

Southbound-API — The Southbound-API represents the interface between control- and data plane. It is the enabler for the externalization of the control plane and therefore key to the corresponding SDN principle [2, 3]. Its realization is a standardized instruction set for the networking hardware. Implementation examples are the IETF ForCES Protocol [4] and most notably the OpenFlow protocol [5].

Northbound-API — SDN enables the exchange of information with applications running on top of the network. This information exchange is performed via the Northbound-API between the SDN controller and an "application control plane" [2, 3]. A universal, standardized Northbound API does not exist. Further, as the kind of information exchanged, its form and frequency depends on the targeted application and network, such universal API is not useful. Standardization of this interface only makes sense for common scenarios, provided that all implementations are kept open. While the SDN controller can directly adapt the behavior of the network, the application controller adapts the behavior of the application using the network. It can be implemented

Programmability is not only a principle but also the key feature of SDN and drives most SDN use cases. This opens the control plane to innovation using conventional software development methods, in turn enabling the customization of the network according to a specific setup or scenario.

as part of a single application instance to a central entity for the entire network responsible for all applications.

Westbound-API — The Westbound-API serves as an information conduit between SDN control planes of different network domains [6]. It allows the exchange of network state information to influence routing decisions of each controller, but at the same time enables the seamless setup of network flows across multiple domains. For the information exchange, standard inter-domain routing protocols like BGP could be used.

Eastbound-API — Communication with the control planes of non-SDN domains, for example, a Multi-Protocol Label Switching (MPLS) control plane, uses the Eastbound-API [7]. The implementation of this interface depends on the technology used in the non-SDN domain. Essentially, a translation module between SDN and the legacy technology is required. In this way, both domains should ideally appear to be fully compatible to each other. For example, the SDN domain should be able to use the routing protocol deployed between non-SDN domains or be able to react to Path Computation Element Protocol (PCEP) messages requesting path setups from an MPLS domain.

Definition of SDN Features

The combination of these four open interfaces together with the core features we outline in the following makes SDN a very flexible and powerful tool for network control and operation. Later we show how matching SDN's unique features to use cases can help a potential adopter of SDN to determine whether SDN is the right technology for that use case.

Programmability — Programmability is not only a principle but also the key feature of SDN and drives most SDN use cases. This opens the control plane to innovation using conventional software development methods, in turn enabling the customization of the network according to a specific setup or scenario.

Example: Based on one or more external information resources (e.g. cloud orchestration) the routing in a network is adapted automatically to optimize resource utilization. Google uses such a mechanism to optimize the bandwidth usage on links between the company's data centers. It achieves this by leveraging information from the traffic sources and grouping application traffic into flow groups with different priorities [6].

Protocol Independence — Protocol independence enables SDN to control or run in conjunction with a large variety of networking technologies and protocols on different network layers. This feature enables migration strategies from old to new technologies and supports the possibility to even run a different network protocol stack tailored for each application.

Example: In order to enable the migration from IPv4 to IPv6 a network operator decides to run both versions of IP in parallel. This is usually done using tunnels and encapsulation. The authors in [8] propose to use SDN-enabled forwarding elements with a centralized control plane to dynamically set up the tunnels at the end points.

Ability to Dynamically Modify Network Parameters — The ability to actively modify network parameters in a dynamic manner that is close to real time defines this SDN feature. Dynamic re-configuration is feasible in different time-scales. This covers wide area networks where only a few change operations are required per day, to data center networks where the constant instantiation or migration of virtual machines and their network connectivity has to happen in minutes or even seconds.

Example: In case of an overloaded link between two network elements with multiple routes, priority traffic is identified through application information and rerouted with minimal delay [9]. In this case the SDN controller receives information about individual flows from the application control plane to determine whether a certain application actually needs more resources and allocates a higher priority to its flows.

Granularity — Networking spans different protocol layers and also levels of data flow aggregates. SDN enables the control of traffic flows with a different granularity on both the aggregate level and the protocol layers. This can range from large MPLS tunnels in core networks to a single TCP connection in a home LAN. This is a necessary feature to ensure scalability and enable the control plane to work on different levels.

Example: In [10] the SDN controller operates on the granularity of individual flows to optimize the user experience for the user of one particular session. This high granularity is feasible in networks with a low total number of flows, for example, home or access networks. In [8] SDN is used to interconnect a virtualized access network to a legacy MPLS core operating on tunnels only. Each tunnel can contain a multitude of flows.

Elasticity — The elasticity feature of SDN describes the ability of the SDN network control plane to increase and decrease its resource consumption based on the required capacity. As controllers run in software, they can be flexibly instantiated and synchronized using a distributed or hierarchical approach on multiple physical or virtual hosts. This enables the control plane to react to variations in traffic mix and volume.

Example: Due to a temporarily increased amount of control traffic in a data center network, the SDN controller can no longer be hosted by a single physical device and has to be distributed among several machines. However, when the situation resolves itself, the control plane can again be relocated to its original host in order to conserve resources. There are several approaches to achieve this kind of distributed SDN control plane. The Onix [11] realization is based on synchronizing the network information base, that is, the global state of the network, across a cluster of servers. Each server directly manages a subset of network elements and exchanges information with the rest of the network controller instances via the shared network state.

Use Cases for Software Defined Networking

This section introduces several use cases that we have selected to derive and to illustrate a method for classifying SDN in terms of the above features and interfaces.

Cloud Orchestration

Over the last decade cloud services have developed at a rapid pace. However, the innovation in this field was mainly confined to server and data center technologies as well as distributed applications. This has led to networks becoming a hindrance for cloud operations. A major reason for this is the fact that networks and servers were traditionally managed separately. For cloud applications to be provisioned and operated quickly and in an automated manner, the management of both network and cloud framework needs to be integrated.

SDN is a viable way to achieve this integration, as the SDN controller as well as the cloud orchestration framework is software, and a (standardized) interface between both worlds is therefore easily attainable. This interface can then, for example, be used to notify the network controller of an imminent virtual machine migration or to notify the cloud orchestration that a link is overloaded and the server load should be moved to a different location.

In [11] the benefits of such an interface are shown. The cloud orchestration software OpenNebula is used to orchestrate virtual servers across multiple hosts and show that a short advance notification from the cloud orchestration to the SDN controller before a virtual machine migration was sufficient to maintain the user sessions of a video streaming service during the migration.

Load Balancing

Another service required for the successful operation of online services that are hosted in data centers is load balancing. Online services, e.g. search engines and web portals, are often replicated on multiple hosts in a data center for efficiency and availability reasons. Here a load balancer dispatches client requests to a selected service replica based on certain metrics such as server load. In general, a load balancer is typically a separately deployed function in a network that distributes the load among network and data center elements in its scope according to a certain optimization metric such as minimum average load or link cost.

Today's solutions for load balancers are effective but have limited flexibility in terms of customization. Being a proprietary middlebox function, such solutions also come at a high cost. When using SDN technologies, load balancing can be integrated within any forwarding element in the network, e.g. OpenFlow switch, avoiding the need for separate devices. Furthermore, SDN allows load balancing to operate on any flow granularity.

In [11] a use case for a data center load balancer is described and a solution based on OpenFlow is proposed. Instead of using a traditional middlebox solution the functionality is realized at the OpenFlow controller and enforced by setting aggregate flow rules using wildcards in the network elements. In this way the need for a dedicated balancer device is no longer existent. Current research tries to provide an abstract language that allows programmers to directly control the network and mechanisms such as load balancing [12].

Routing

The API between data plane forwarding and a centralized control plane in SDN provides ample opportunities for a routing protocol adaptation, which is very difficult in existing decentralized routing schemes implemented on closed box network elements. Routing services that can be realized by the SDN concept, for example, through programming modules on OpenFlow controllers directing OpenFlow Switches, include path selection for traffic optimization, multi-homing, secure routing, path protection, and migration between protocol versions, that is, IPv6.

In [13] the authors propose a hybrid SDN/BGP control plane that on the one hand leverages the new possibilities in a simplified centralized routing approach, and other hand benefits from the compatibility with legacy networks.

Monitoring and Measurement

SDN provides the network the ability to perform certain network monitoring operations and measurements without any additional equipment or overhead. The concept was introduced in [14] and is based on the fact that an SDN inherently collects information about the network to maintain a global network state at the logically centralized controller. This information can then be processed in software to obtain a subset of monitoring parameters. Furthermore, active measurements are enabled by selectively mirroring specific production traffic flows to the control plane or an external measurement device without the need to introduce artificial and potentially disruptive measurement probe traffic into the network. For example, by mirroring the traffic for a phone call at ingress and egress point of the network, the network administrator can determine the delay and quality of service for a particular call at a certain time.

Network Management

Today's network management policies are usually decided upon by the network operator and then configured once in each network element by an administrator. The larger the network, the higher the required configuration effort becomes. Hence, an established policy is seldom modified. This leads to an often very inefficient network operation. The fact that traffic patterns continually change cannot be taken into account this way.

In order to change this, the network needs to be able to adapt policies dynamically and automatically based on a range of information. This calls for a more general specification of network policies that are subsequently translated into specific rules for each device in the network using a policy engine. The logically centralized control plane of SDN offers itself as a very suitable way to enable such an approach, as it has all information about the network available.

Routing services that can be realized by the SDN concept, for example, through programming modules on OpenFlow controllers directing OpenFlow Switches, include path selection for traffic optimization, multi-homing, secure routing, path protection, and migration between protocol versions, that is, IPv6.

Use case \ Interface	Southbound interface	Northbound interface	Eastbound interface	Westbound interface
Cloud orchestration	✓	✓	X	X**
Load balancing	✓	✓	X	✓*
Routing	✓	X	✓	✓
Monitoring and measurement	✓	✓	✓	✓
Network management	X	✓	✓	X
Application-awareness	X	✓	X	X

• The Westbound interface is not used in cases where the load balancer is only responsible for a single SDN domain, e.g. a single data center.
** The Westbound interface is used when parts of the cloud are connected via a non-SDN controlled network, e.g. MPLS PCE.

Table 1. Mapping of use cases to SDN interfaces. Reliance on an interface is checked, whereas non-reliance is marked with an X.

For example, a high-level network policy dictates the prioritization of VoIP traffic inside an Enterprise network. The SDN controller can then identify corresponding network flows and assign them to a high priority level in each device. This is dynamic on the one hand as VoIP flows are set up and terminated with each phone call, and on the other hand it is automated as the devices are configured without the need for physical access and any human intervention. In fact, the administrator does not have to know the topology of the network or the devices involved in order to achieve the policy's goal. Such an approach has been implemented prototypically in [15].

Application-Awareness

Using network resources efficiently and optimizing traffic flows toward high end-user Quality of Experience (QoE) is an often cited goal for next generation networks. However, it is difficult to realize when nothing is known about the kind of applications that are run on the network and their state. Existing approaches in this direction often rely on Deep Packet Inspection to identify the applications. This, however, is not a very accurate technique and does not take the application state or QoE into account at all [16].

With the Northbound-API of the SDN controller, the application itself can inform the network about its properties and state. In this way, the network controller can direct traffic flows to complement rather than disrupt each other [9,17]. Furthermore, a previously made forwarding decision can be revised in light of changing situations in the network and a different application state. The other way around, if the network can no longer sustain a certain service level for the application due to lack of resources, it can notify the application to modify its behavior. For example, due to its architecture, SDN easily allows cross-layer optimization between applications and their demands and the network capabilities. Thus, a better use of the network resources with respect to more generic constraints such as user-centrality [9] or energy-efficiency [11] is possible.

A Use Case Based Analysis of SDN Interfaces and Features

To analyze the importance of SDN in terms of its interfaces and features for different use cases, we map each of the use cases shown in the previous section to them. We validate the presented features and their mapping to the use-cases. Thus, the compass guides a potential adopter of SDN, whether SDN in fact is the right technology for an arbitrary use case.

In the first step of our analysis, we map the use cases to the SDN interfaces as shown in Table 1. Reliance on an interface is checked, whereas non-reliance is marked with an X. As can be seen, not all use cases depend on all interfaces. In fact, the only use case leveraging all interfaces of SDN is the "Monitoring and Measurement" use case. Overall, the use cases are quite heterogeneous in terms of SDN interface dependency. The most used interface appears to be the "Northbound-API" interface. However, no conclusion is drawn about the importance of an interface here. This depends on the specific implementation of a use case, on how an interface is used, and therefore how crucial it is. To apply the SDN compass it is important enough to understand which interfaces to use for a considered use case to be implemented using SDN.

In the second step of our use case analysis, we classify the use cases according to:

- The importance of each of the above identified features as a use case enabler: HIGH (***) = enables service, MEDIUM (**) = improves service significantly, LOW (*) = nice to have.
- The area of application in which these features are most important for each use case: Data Center = one data center, WAN = ISP Core network, Enterprise = Enterprise network without enterprise data center.

As a result of the use case classification applying these simple metrics, we obtain Table 2. Here we observe horizontal clusters as well as local clusters of importance across the different use cases and areas of application. This analysis does not only validate the five SDN features as described previously, but also allows us to identify the importance of each feature for certain classes of SDN use cases.

Let us have a closer look at the table. There are entire rows filled with a background color where all features are classified as highly important enablers for a use case. These horizontal lines are limited to one application area only. Cloud orchestration, for example, is a use case that is focused on data center environments. This is also confirmed by the table where the horizontal line marking each SDN feature with high importance runs in the data center application area. A similar observation can be made for monitoring. Here enterprise networks benefit most from all SDN features due to the high application mix that has to be monitored in enterprise networks.

Use case \ Feature	Area of application	Elasticity	Programmability	Protocol independence	Dynamic	Granularity
Cloud orchestration	Data center	* * *	* * *	* * *	* * *	* * *
	WAN	*	*	**	*	***
	Enterprise	*	*	**	*	***
Load balancing	Data center	*	*	*	***	**
	WAN	*	*	**	**	**
	Enterprise	*	*	**	**	**
Routing/forwarding	Data center	*	***	*	***	**
	WAN	**	****	***	**	**
	Enterprise	*	***	**	***	**
Monitoring and measurement	Data center	**	**	***	**	***
	WAN	*	**	***	*	***
	Enterprise	***	***	***	***	***
Network management	Data center	**	***	***	**	**
	WAN	*	***	***	*	**
	Enterprise	*	***	***	*	**
Application-awareness	Data center	*	***	***	***	***
	WAN	*	***	***	***	***
	Enterprise	*	***	***	***	***

Table 2. Mapping of Use Cases to SDN Features. The importance of each of the above identified features as a use case enabler: HIGH (***) = enables service, MEDIUM (**) = improves service significantly, LOW (*) = nice to have. The area of application in which these features are most important for each use case: Data Center = one data center, WAN = ISP Core network, Enterprise = Enterprise network without enterprise data center. Local clusters of importance (dark grey) point to a particular importance of one SDN feature for a use case across all three areas of application. Complete rows marked with a background color indicate that all features are classified as highly important enablers for the use case, but limited to one application area only.

Local clusters of importance (dark) point to a particular importance of one SDN feature for a use case across all three areas of application. For example, the time-dynamics to be realized for SDN-based routing for path protection are based on the dynamic feature of SDN as the key enabler. Monitoring is mostly concerned with data gathering across protocols and different levels of granularity. This is confirmed by the table with the SDN features protocol independence and granularity expressing high importance markings. Network management is based on the features programmability and protocol independence as here the configuration aspect is a key feature enabled by SDN.

The general use case application awareness shows four importance clusters, namely: programmability, protocol independence, dynamic, and granularity.

Key Derivations

The above definitions and use case analysis aim to create an understanding of the applicability of the SDN principles. There is a lack of clear definitions and a lack of methodology for assessing the suitability of SDN concepts for certain use cases. Current discussions direct SDN toward an image of being a universal solution in networking.

As a basis for such a methodology, we defined four main interfaces in an SDN-enabled network control architecture — the southbound, eastbound, westbound, and northbound interfaces — and we identified the five main features provided by SDN technologies: programmability, protocol independence, dynamic, granularity, and elasticity.

Our analysis of selected use cases based on related publications mainly taken from recent workshops, conferences, and journal articles (cloud orchestration, load balancing, routing,

This approach can be adapted to help classify other use cases and gauge the potential benefits of using SDN in their context. Their main features can be identified and weighted, and the implementation focus of the required network applications can be planned accordingly.

measurement, network management) provides a methodology for assessing the importance of SDN as an enabler for certain use cases.

Our discussion shows that the use cases depend on different application areas (Data center, Enterprise, WAN). Referring to Table 2 we cannot determine a specific area of application where SDN excels. Furthermore, different interfaces are needed for each use case, that is, not all interfaces have to be implemented. Accordingly, development guidelines for specific controllers can be derived based on the analysis of the specific use-case in relation to the features and interfaces used. Different use cases are based on different SDN features, that is, the implementation of the features depends on the specific use-case.

Similarly, a corresponding analysis of a new use case can reveal whether the use case can benefit from SDN technology, that is, is there at least one feature with "high." Furthermore, the benefit of SDN for a certain use case increases with the number of important features identified. Additionally, we can observe that there are more advanced use cases that cannot be realized in today's networks. These use cases can benefit from or are even enabled by SDN features such as the presented application awareness use-case.

Despite the operational use-cases discussed above, SDN also drives innovation in other areas. Particularly in research testbeds [18], for prototyping [19], and for service rollout (Beta Slice), the capabilities of SDN enable innovation within networks. Accordingly, the areas of application are not limited to the discussed use-cases, but are expected to expand beyond the scope of today's networks. However, this discussion is not the intended topic of this article.

Conclusion

Due to its innovation potential, SDN is seen as one key technology to enable and operate next-generation networks. However, different definitions and meanings of the term SDN currently exist, leading to a fragmented view.

This article is a major step toward a better understanding of SDN, its necessary interfaces, as well as its key attributes. Based on an inductive approach, we derived a mapping of interfaces and attributes to SDN use cases. In a second step, a mapping of use cases to SDN features, highlighting the importance of the specific features to the use-cases and areas of application, is performed. This approach can be adapted to help classify other use cases and gauge the potential benefits of using SDN in their context. Their main features can be identified and weighted, and the implementation focus of the required network applications can be planned accordingly.

Therefore, the main contribution of this article is to supply SDN adopters with a compass and a map to reach the desired answer to the question of using SDN in a specific scenario.

References

[1] T. Nadeau and K. Gray, "SDN: Software Defined Networks," O'Reilly Media, Sept. 2013, ISBN: 978-1449342302.

[2] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, 2012.

[3] S. Sezer *et al.*, "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks," *IEEE Commun. Mag.*, vol. 51, no. 7, July 2013, pp. 36–43.

[4] A. Doria, R. Haas, and J. H. Salim, "ForCES Protocol Specification," http://www.ietf.org/internet-drafts/draft-ietf-forces-protocol-08.txt, 2006.

[5] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 2, April 2008, pp. 69–74.

[6] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *Proc. ACM SIGCOMM 2013 Conference*, Aug. 2013, pp. 3–14.

[7] A. Devlic, W. John, and P. Sköldström, "Carrier-Grade Network Management Extensions to the SDN Framework," *Proc. 8th Swedish National Computer Networking Wksp. SNCNW 2012*, June 2012.

[8] G. Hampel, M. Steiner, and B. Tian, "Applying Software-Defined Networking to the Telecom Domain," *Proc. 16th IEEE Global Internet Symp.* in conjunction with *IEEE INFOCOM*, Apr. 2013, pp. 133–8.

[9] M. Jarschel *et al.*, "SDN-based Application-Aware Networking on the Example of YouTube Video Streaming," *Proc. 2nd European Wksp. Software Defined Networks* (EWSDN 2013), Oct. 2013, pp. 87–92.

[10] T. Koponen *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks," *Proc. 9th USENIX Conf. Operating Systems Design and Implementation* (OSDI 10), Oct. 2010, pp. 351–64.

[11] M. Jarschel and R. Pries, "An OpenFlow-Based Energy-Efficient Data Center Approach," P*roc. ACM SIGCOMM 2012 Conf.*, Aug. 2012, pp. 87–8.

[12] C. Monsanto *et al.*, "Composing Software Defined Networks," *Proc. 13th USENIX Symp. Networked Systems Design and Implementation* (NSDI 13), Apr. 2013, pp. 1–13.

[13] C. E. Rothenberg *et al.*, "Revisiting Routing Control Platforms with the Eyes and Muscles of Software-Defined Networking," *Proc. First Workshop on Hot Topics in Software Defined Networks* (HotSDN 12), Aug. 2012, pp. 13–18.

[14] C. Yu *et al.*, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," *Proc. Passive and Active Measurement Conf.* (PAM 2013), Jan. 2013, pp. 31–41.

[15] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," *IEEE Commun. Mag.*, vol. 51, no. 2, Feb. 2013, pp. 114–19.

[16] T. Hoßfeld *et al.*, "Challenges of QoE Management for Cloud Applications," *IEEE Commun. Mag.*, vol. 50, no. 4, Apr. 2012, pp. 28–36.

[17] G. Wang *et al.*, "Programming Your Network at Run-Time for Big Data Applications," *Proc. First Workshop on Hot Topics in Software Defined Networks* (HotSDN 12), August 2012, pp. 103–08.

[18] R. Sherwood *et al.*, "Can the Production Network be the Testbed," *Proc. 9th USENIX Conf. Operating Systems Design and Implementation* (OSDI 10), Oct. 2010, pp. 365–78.

[19] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," *Proc. 9th ACM SIGCOMM Wksp. Hot Topics in Networks* (Hotnets-IX), article no. 19, Oct. 2010.

Additional Reading

[1] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-based Server Load Balancing Gone Wild," *Proc. 11th USENIX Conf. Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (HotICE 2011), Mar.s 2011.

Biographies

Michael Jarschel is a Ph.D. candidate working in the "Next Generation Networks" research group at the Chair of Communication Networks in Würzburg. He received his diploma in computer science from the University of Würzburg in 2009. His main research interests are Software Defined Networking, QoE, and Cloud Networks, with a focus on performance evaluation.

Thomas Zinner is heading the NGN research group "Next Generation Networks" at the Chair of Communication Net-

works in Würzburg. He finished his Ph.D. thesis on "Performance Modeling of QoE-Aware Multipath Video Transmission in the Future Internet" in 2012. His main research interests cover the performance assessment of novel networking technologies, in particular software-defined networking and network function virtualization, as well as network-application interaction mechanisms.

TOBIAS HOSSFELD is heading the FIA research group "Future Internet Applications & Overlays" at the Chair of Communication Networks in Würzburg. He finished his Ph.D. in 2009 and his professorial thesis "Modeling and Analysis of Internet Applications and Services" in 2013. He has published more than 100 research papers in major conferences and journals, receiving four best paper awards, three awards for his Ph.D. thesis, and the Fred W. Ellersick Prize 2013 (IEEE Communications Society).

PHUOC TRAN-GIA is a full professor at the Institute of Computer Science and head of the Chair of Communication Networks at the University of Würzburg, Germany. His current research areas include architecture and performance analysis of communication systems, and planning and optimization of communication networks. He has published more than 100 research papers in major conferences and journals, and recently received the the Fred W. Ellersick Prize 2013 (IEEE Communications Society).

WOLFGANG KELLERER is a full professor at the Technische Universität München (TUM), heading the Institute for Communication Networks in the Faculty of Electrical Engineering and Information Technology. Until 2012 he has been director and head of wireless technology and mobile network research at NTT DOCOMO's European research laboratories for more than ten years. His research resulted in more than 100 publications in the area of mobile networking and service platforms.