University of Würzburg
Institute of Computer Science
Research Report Series

A Compass Through SDN Networks

Thomas Zinner[1], Michael Jarschel[1], Tobias Hossfeld[1], Phuoc Tran-Gia[1] and Wolfgang Kellerer[2]

Report No. 488 December 2013

[1] University of Würzburg
Institute of Computer Science
Departement of Communcation Networks
Am Hubland, D-97074 Würzburg, Germany
{zinner,jarschel,hossfeld,trangia}@informatik.uni-wuerzburg.de

[2] Technische Universität München
Lehrstuhl für Kommunikationsnetze
Arcisstrasse 21, 80290 München, Germany
wolfgang.kellerer@tum.de

A Compass Through SDN Networks

Thomas Zinner, Michael Jarschel, Tobias Hossfeld, Phuoc Tran-Gia
University of Würzburg
Institute of Computer Science
Departement of Communcation Networks
Am Hubland, D-97074 Würzburg, Germany
{zinner, jarschel, hossfeld, trangia}@informatik. uni-wuerzburg.de

Wolfgang Kellerer
Technische Universität München
Lehrstuhl für Kommunikationsnetze
Arcisstrasse 21, 80290 München, Germany
wolfgang.kellerer@tum.de

Abstract

The term Software Defined Networking (SDN) is prevalent in todays discussion about future communication networks. However, no consistent definition regarding this technology has formed, yet. 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 research report is twofold. First, it gives a thoroug definition of SDN and its interfaces and a list of its key attributes. Second, it provides a mapping of interfaces and attributes to SDN use cases and highlights their relevance on a per-scenario basis. Thus, this report guides a potential adopter of SDN, whether SDN is in fact the right technology for his arbitrary use case.

1 Introduction

Today's networks face challenges that have not been expected during the time of their development 40 years ago. This is particularly illustrated by the rise of cloud systems and data centers. These systems rely on the high performance and flexibility of virtualized environments. If a service requires additional resources to support more customers they can easily be provisioned by dynamically adding new virtual instances running upon current server hardware. This flexibility, however, stands in contrast to static, properitary network components. Due to the heterogeneity of network equipment and vendors topology changes require much longer than provisioning a virtual machine.

Software-Defined Networking (SDN) is a technology that tries to overcome the ossification of the networks. It enables the decoupling of both data- and forwarding plane and offers a dynamic, logically centralized, cost-effective, and adaptable architecture. One of the main contributions of SDN is a standardized control protocol which allows the SDN controller to modify the forwarding tables of routers and switches and thus to change the data forwarding. The OpenFlow protocol is currently the most common realization of this protocol.

Parts of this manuscript have been submitted to IEEE Communications Magazine

SDN is prevalent in today's discussion about future communication networks. Like 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 paper represents an important step towards 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. Thus, the compass guides a potential adopter of SDN, whether SDN is in fact the right technology for an arbitrary use case.

2 Brief Overview of the OpenFlow Protocol

This section provides a brief overview of OpenFlow. More details on OpenFlow can be found in the white paper [1] as well as in the OpenFlow specification [2].

OpenFlow was designed as a new network paradigm, which enables researchers to test new ideas under realistic conditions on an existing network infrastructure. To be able to take action in the switching, OpenFlow separates the control plane from the data plane and connects them by an open interface, the OpenFlow protocol. The control plane is implemented in software in form of a controller on an external PC. For the communication between the switch and the controller, a secure channel is used. This allows researchers to be flexible with their work, while at the same time using high-performance hardware.

The OpenFlow switch itself holds a flow table which stores flow entries consisting of three components. First, a set of 12 fields with information found in a packet header that is used to match incoming packets. Second, a list of actions that dictates how to handle matched packets. Third, a collection of statistics for the particular flow, like number of bytes, number of packets, and the time passed since the last match.

When a packet arrives at the OpenFlow switch, its header information is extracted and then matched against the header portion of the flow table entries. If checking against entries in each of the switches' tables does not result in a match, the packet is forwarded to the controller, which determines how the packet should be handled. In the case of a match, the switch applies the appropriate actions to the packet and updates statistics for the flow table entry. This process is visualized in Figure 1.

Several papers have been published indicating possible uses for OpenFlow [3–6]. All these papers demonstrate that the concept of splitting the control plane from the data plane is useful in a variety of fields, like data center routing, energy saving, and network virtualization.

3 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

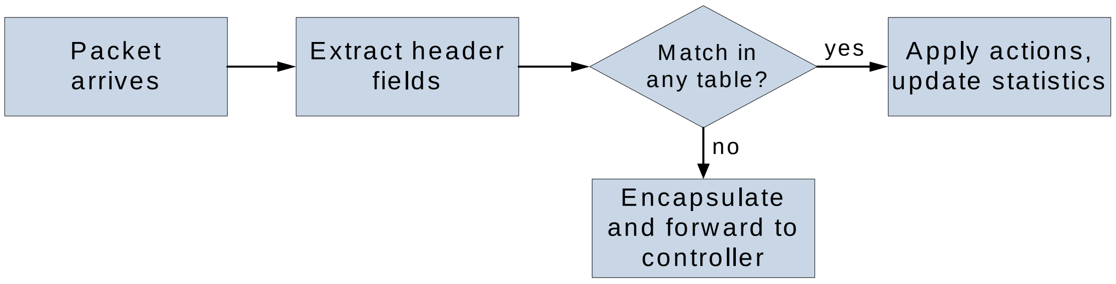
Packet arrives
Extract header fields
Match in any table?
yes
no
Apply actions, update statistics
Encapsulate and forward to controller

Figure 1: Handling of incoming packets in an OpenFlow switch.

provider space. SDN 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 considered as mandatory for classifying a technology as SDN.

3.1 Separation of Control- and Forwarding Plane

The physical separation of control- and forwarding- or data plane is the best known principle of SDN. 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 directly change the forwarding behavior of the network element. This enables several key benefits of SDN. 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. This has already introduced new and disruptive start-ups to the market that have sped up innovation in the network. Customers are also enabled to ”mix-and-match” products of different vendors and thus increase competition further.

3.2 Logically Centralized Control

The controller of an SDN network is a logically centralized entity, i.e. it can consist of multiple physical or virtual instances, but behaves like a single component. A network or virtual network has a single active controller which maintains a global state of that particular network or slice. In particular, this means that multiple active controllers per network domain/slice are not allowed. Multiple controllers are not required as a controller can be implemented as a distributed system and backups can be defined in case of a controller failure. Depending on the implementation, such a controller can operate networks ranging from a virtual network on a single network device to a global network spanning a multitude of nodes. The global knowledge such a central controller possesses about the network enables it to adapt its network policy much faster than a system of traditional routers could.

3.3 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. 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 (cf. Southbound API). This is also true for the remaining interfaces, which are discussed in greater detail in Section 4.1.

3.4 Programmability

The most important principle of SDN from our point of view is the programmability of the network. This is enabled through the external software controller and the 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. This represents the fundamental paradigm shift in networking SDN has initiated.

4 Key Interfaces and Features of Software Defined Networking

In this section, we define the key interfaces of an SDN system and highlight the features of Software Defined Networking - key components of the SDN compass.

4.1 Definition and Significance of SDN Interfaces

The interfaces of Software Defined Networking as we see them are illustrated in Figure 1 for a generic, exemplary network. This network consists 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). The mapping of SDN interfaces to this example network is discussed below.

4.1.1 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. Its realization is a standardized instruction set for the networking hardware, comparable to OpenGL or DirectX in the graphics field. Implementation examples are the IETF ForCES Protocol [7] and most notably the OpenFlow protocol [8].

4.1.2 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 what we call the ”application control plane”. What kind of information is exchanged, in which

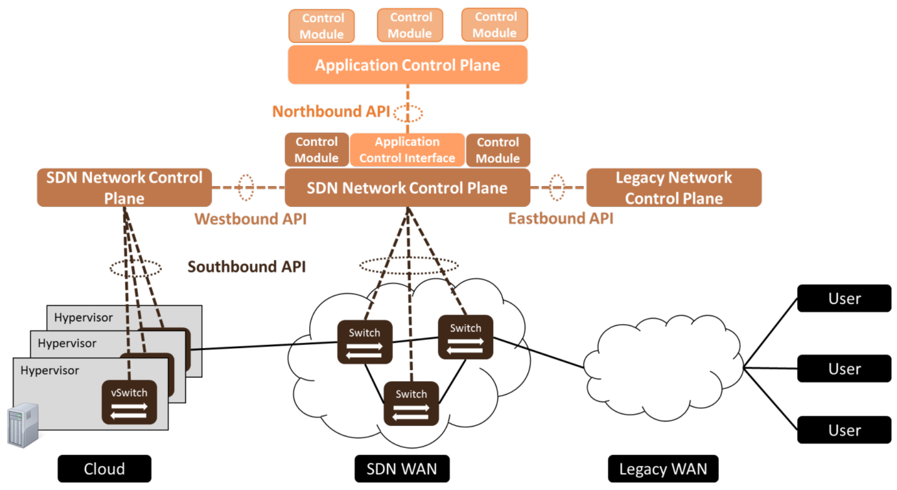

Figure 2: Example - Interfaces of a Software Defined Network

form, and how often, depends on the kind of application and network in question along with the network policy. Standardization of this interface only makes sense for common scenarios, however, all implementations should be 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 as part of a single application instance to a central entity for the entire network responsible for all applications.

4.1.3 Westbound-API

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

4.1.4 Eastbound-API

Communication with the control planes of non-SDN domains, e.g. a Multi-Protocol Label Switching (MPLS) control plane, uses the Eastbound-API. 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. 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.

4.2 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 operation. Matching SDN's unique features and their importance to one's particular use case can help a potential adopter of SDN to determine, whether SDN is in fact the right technology for that specific purpose.

4.2.1 Programmability

Programmability is not only a principle but also the key feature of SDN and drives most SDN use cases. It is enabled by an open interface between data and control plane, which allows the control plane to be realized in software external from the data plane devices. This opens the control plane to broad modifications and 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. traffic/power pricing) the routing in a network is adapted automatically to lower the cost [9].

4.2.2 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 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 [10].

4.2.3 Dynamic

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, the traffic is efficiently rerouted with minimal delay [11].

4.2.4 Granularity

Networking spans different protocol layers and also levels of data flow aggregates. SDN also features the ability to control 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: A network administrator wants all traffic to a specific TCP port, e.g. 80 (HTTP), sent via a dedicated link. SDN can reroute the traffic according to the corresponding field in the packet header [11]..

4.2.5 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 [12]

5 Use Cases for Software Defined Networking

This section introduces several use cases which we consider representative of SDN. This list is not meant to be exhaustive, but serves as a method for classifying SDN in terms of its features and interfaces (cf. Section 6).

5.1 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 SDN controller as well as 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. One approach to create such an interface is Meridian introduced by Banikazemi et al. [13].

5.2 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, the 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 [14], a use case for a data center load balancer is described and a solution based on OpenFlow is proposed.

5.3 Routing

The API between data plane forwarding and a centralized control plane in SDN provides ample opportunities for 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, e.g. 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, i.e. IPv6. The opportunities for using SDN/OpenFlow for a centralized routing control platform are described in [9].

5.4 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 by Yu et al. [15] 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 of introducing 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.

5.5 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 greater the required configuration effort becomes. Hence, a once set 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. 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 by Kim et al. [16].

Recently, Google [17] presented its a network management solution based on SDN used in its own private WAN, B4. This WAN connects Google's data centers and utilizes Open-Flow enabled switches. Due to three characteristics presented in the following, the SDN technology can be applied resulting in a high usage of the network links of up to 100% in single cases and 70% on average.

First, Google does not only control the network entities in the WAN, but also applications, servers and the local area networks. Second, the bandwidth-intensive applications, i.e., data copies between the data centers, are based on TCP and can easily adapt the transmision rate to the available bandwidth. Third, the small number of data centers allows a central control of the WAN with a single SDN controller.

This results in a higher utilization of the data links, 70% instead of 40%, and thus a significant reduction of expenditures.

5.6 Application-Awareness

Using network resources efficiently and optimizing traffic flows towards 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, which 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 [18]. With the Northbound-API of the SDN controller, the application itself can inform the network about its properties and state. This way, the network controller can direct traffic flows to complement rather than disrupt each other. Furthermore, a once 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 like user-centrality or energy-efficiency is possible. An example aiming at improving the QoE of applications in an enterprise like network is discussed in [11].

6 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 aim at creating an understanding of the SDN definition and validate the presented features and their mapping to

Interface / Use Case	Southbound Interface	Northbound Interface	Eastbound Interface	Westbound Interface
Cloud Orchestration	✓	✓	✗	(✗)
Load Balancing	✓	✓	✗	(✓)
Routing	✓	✗	✓	✓
Monitoring and Measurement	✓	✓	✓	✓
Network Management	✗	✓	✓	✗
Application-Awareness	✗	✓	✗	✗

Table 1: Mapping of Use Cases to SDN Interfaces. Reliance on an interface is checked, whereas non reliance is marked with an X. Tendencies, i.e. there are some exemptions from the rule, are marked in brackets.

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. Tendencies, i.e. there are some exemptions from the rule, are marked in brackets. 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. This shows that the choice of use cases appears to be a good mix covering all interfaces and many combinations. The most used interface appears to be the ”Northbound-API” interface with all but one use cases relying on it. However, a clear statement about the importance of an interface is not possible as it depends on the specific implementation of a use case, how an interface is used, and therefore how crucial it is.

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, and
- 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 using 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 in Section 4.2, 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. 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.

7 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 towards an image of being a universal solution in networking. As a basis for such a methodology, we (1) defined four main interfaces in an SDN-enabled network control architecture, namely the southbound, eastbound, westbound, and northbound interfaces, and (2) we identified the five main features provided by SDN technologies, namely 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, measurement, network management) hints at a methodology for assessing the importance of SDN as an enabler for certain use cases. Our discussion shows that the use cases depend on (a) different application areas (Data center, Enterprise, WAN). Referring to Table 2 we cannot determine a specific area of application where SDN excels. Furthermore, (b) different interfaces are needed for each use case, i.e., 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, i.e., 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, i.e., 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 which cannot be realized in today’s networks. These use cases can benefit from or are even enabled by SDN features like the presented application awareness use-case. Despite the operational use-cases discussed above,

Feature / Use Case	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) 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.

SDN drives innovation also in other areas. Particularly in research testbeds [19], for prototyping [20] 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.

8 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 paper is a major step towards 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. The main contribution of this paper is therefore 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.

Acknowledgments

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008.

[2] "OpenFlow Switch Specification, Version 1.0.0," December 2009.

[3] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastic Tree: Saving Energy in Data Center Networks," in *7th USENIX Symposium on Networked System Design and Implementation (NSDI)*, (San Jose, CA, USA), pp. 249–264, April 2010.

[4] M. Casado, D. Erickson, I. A. Ganichev, R. Griffith, B. Heller, N. McKeown, D. Moon, T. Koponen, S. Shenker, and K. Zarifis, "Ripcord: A Modular Platform for Data Center Networking," Tech. Rep. UCB/EECS-2010-93, EECS Department, University of California, Berkeley, June 2010.

[5] S. Das, G. Parulkar, P. Singh, D. Getachew, L. Ong, and N. McKeown, "Packet and Circuit Network Convergence with OpenFlow," in *Optical Fiber Conference (OFC/NFOEC'10)*, (San Diego, CA, USA), March 2010.

[6] R. Braga, E. S. Mota, and A. Passito, “Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow,” in *35th Annual IEEE Conference on Local Computer Networks*, (Denver, CO, USA), pp. 416–423, October 2010.

[7] A. Doria, R. Haas, and J. Salim, “Forces protocol specification.” `http://www.ietf.org/internet-drafts/draft-ietf-forces-protocol-08.txt`, 2006.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 60–74, 2008.

[9] C. Rothenberg, M. Salvador, C. Corrła, S. Lucena, and R. Raszuk, “Revisting routing control platforms with the eyes and muscles of software-defined networking,” in *Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks*, 2012.

[10] G. Hampel, M. Steiner, and B. Tian, “Applying software-defined networking to the telecom domain,” in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013.

[11] M. Jarschel, F. Wamser, T. Hhn, T. Zinner, and P. Tran-Gia, “Sdn-based application-aware networking on the example of youtube video streaming,” in *2nd European Workshop on Software Defined Networks (EWSDN 2013)*, 2013.

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, “Onix: a distributed control platform for large-scale production networks,” in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI’10, (Berkeley, CA, USA), pp. 1–6, USENIX Association, 2010.

[13] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, “Meridian: an SDN platform for cloud network services,” *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 120–127, 2013.

[14] R. Wang, D. Butnariu, and J. Rexford, “OpenFlow-based server load balancing gone wild,” in *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, pp. 12–12, USENIX Association, 2011.

[15] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, “Flowsense: Monitoring network utilization with zero measurement cost,” in *Passive and Active Measurement*, pp. 31–41, Springer, 2013.

[16] H. Kim and N. Feamster, “Improving network management with software defined networking,” *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, 2013.

[17] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, “B4: Experience with a globally-deployed software defined wan,” in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 3–14, ACM, 2013.

[18] T. Hossfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 28–36, 2012.

[19] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?," in *OSDI*, vol. 10, pp. 1–14, 2010.

[20] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.