Are we ready for SDN? - Implementation Challenges for Software-Defined Networks

Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan
CSIT, Queen's University Belfast

Barbara Fraser, David Lake - Cisco Systems

Jim Finnegan, Niel Viljoen - Netronome

Marc Miller, Navneet Rao - Tabula

ABSTRACT

Cloud services are exploding and organizations are converging their data centres in order to take advantage of the predictability, continuity, and quality of service delivered by virtualization technologies. In parallel, energy-efficient and high-security networking is of increasing importance. Network operators, service and product providers require a new network solution to efficiently tackle the increasing demands of this changing network landscape. Software-Defined Networking has emerged as an efficient network technology capable of supporting the dynamic nature of future network functions and intelligent applications while lowering operating costs through simplified hardware, software, and management. In this article, the question of how to achieve a successful carrier grade network with Software-Defined Networking is raised. Specific focus is placed on the challenges of network performance, scalability, security and interoperability with the proposal of potential solution directions.

1 INTRODUCTION - What is Software-Defined Networking?

Network configuration and installation requires highly-skilled personnel adept at configuration of many network elements. Where interactions between network nodes (e.g. switches, routers, etc.) are complex, a more systems-based approach encompassing elements of simulation is required. With the current programming interfaces on much of today's networking equipment, this is difficult to achieve.

In addition, operational costs involved in provisioning and managing large, multi-vendor networks covering multiple technologies have been increasing over recent years, whilst the predominant trend in revenue for operations has been decreasing. Coupled with increasing scarcity

of human resources and increasing costs of real-estate, this "perfect storm" for service providers is leading to renewed interest in solutions that can unify network management and provisioning across multiple domains. A new network model is required to support this.

The term Software-Defined Networking (SDN) has been coined in recent years. However, the concept behind SDN has been evolving since 1996 driven by the desire to provide user-controlled management of forwarding in network nodes. Implementations by research and industry groups include Ipsilon (proposed General Switch Management protocol, 1996), The Tempest (a framework for safe, resource-assured, programmable networks, 1998) and IETF Forwarding and Control Element Separation, 2000, and Path Computation Element, 2004. Most recently, Ethane (2007) and OpenFlow (2008) have brought the implementation of SDN closer to reality. Ethane is a security management architecture combining simple flow-based switches with a central controller managing admittance and routing of flows. OpenFlow enables entries in the Flow Table to be defined by a server external to the switch. SDN is not, however, limited to any one of these implementations, but is a general term for the platform.

For clarity, SDN is described in this article with the Open Networking Foundation (ONF) [1] definition: *"In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications."*

SDN focuses on four key features:

- Separation of the control plane from the data plane,
- A centralized controller and view of the network,
- Open interfaces between the devices in the control plane and the data plane, and
- Programmability of the network by external applications.

Our vision of the future SDN architecture is described in Figure 1. This architecture encompasses the complete network platform.

The bottom tier of Figure 1 involves the physical network equipment including Ethernet switches and routers. This forms the data plane.

The central tier consists of the controllers that facilitate setting up and tearing down flows and paths in the network. The controllers use information about capacity and demand obtained from the networking equipment through which the traffic flows. The central tier links with the bottom tier via an Application Programming Interface (API) referred to as the southbound API. Connections between controllers operate with east and westbound APIs. The controller-application interface is referred to as the northbound API.

Functional applications such as energy-efficient networking, security monitoring and access control for operation and management of the network are represented at the top of Figure 1 highlighting the user-control/management separation from the data-plane. An application in this article refers to a service provided by the network operator. A detailed insight into every

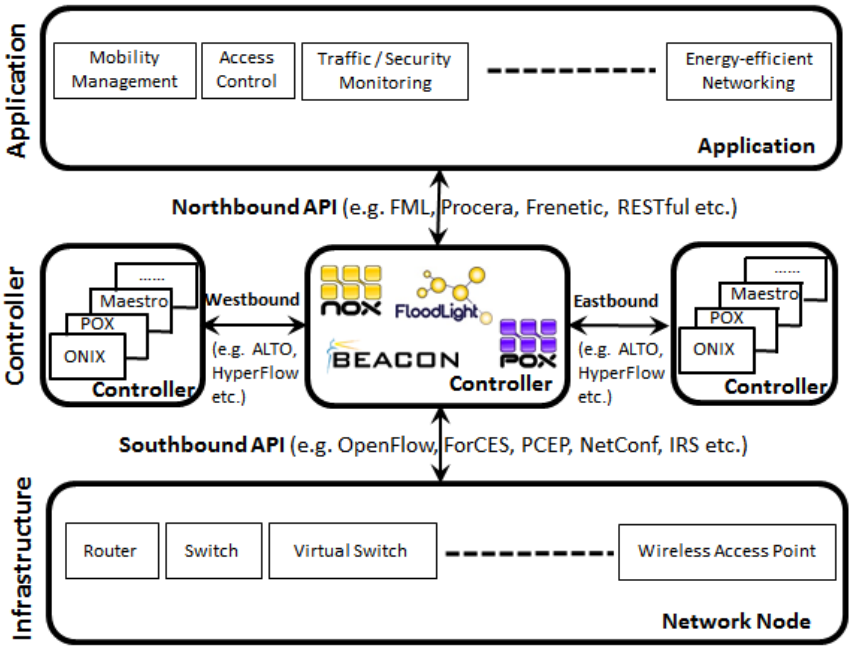

Figure 1: SDN Functional Architecture illustrating the infrastructure, control and application elements of which the network is comprised.

element of the architecture in Figure 1 is beyond the scope of this article. Instead, the transition from the traditional network to state-of-the-art in SDN today is presented.

A key challenge of SDN relates to separation of the control and data plane and maintaining carrier grade service within this framework. The architecture requirements to meet operational expectation in carrier grade networks are scalability, reliability, Quality-of-Service (QoS) and service management [2]. Four specific questions arising from the control-data plane separation challenge are discussed in Section 3. A series of solutions to these identified issues are studied and the article concludes with the outline of our vision for the future of SDN.

2 BACKGROUND -Why SDN?

The fundamental purpose of the communication network is to transfer information from one point to another. Within the network the data travels across multiple nodes and efficient and effective data transfer (forwarding) is supported by the control provided by network applications/services.

Networking - The Old Way:

In traditional networks, as shown in Figure 2, the control and data planes are combined in a network node.

The control plane is responsible for configuration of the node and programming the paths that will be used for data flows. Once these paths have been determined they are pushed down

to the data plane. Data forwarding at the hardware level is based on this control information.

In this traditional approach, once the flow management (forwarding policy) has been defined, the only way to make an adjustment to the policy is via changes to the configuration of the devices. This has proven restrictive for network operators who are keen to scale their networks in response to changing traffic demands, increasing use of mobile devices and the impact of "Big Data".

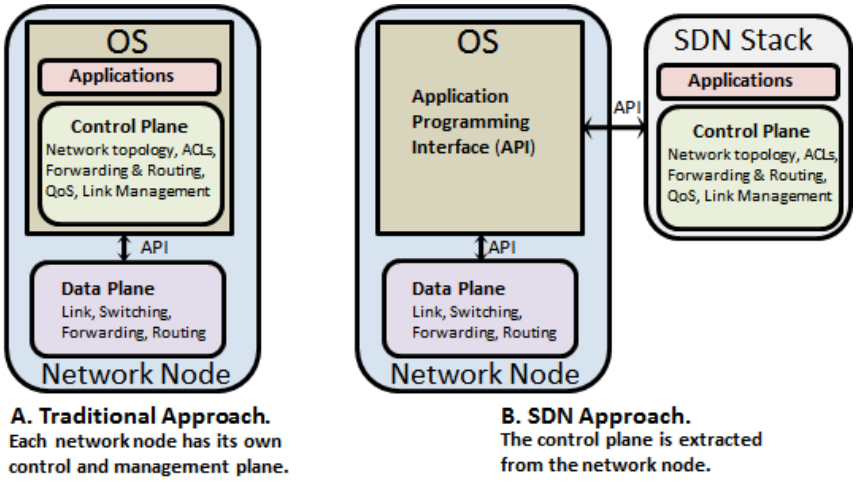

Figure 2: Traditional Network View compared with SDN Network View

Networking - The SDN Way:

From these service-focussed requirements, SDN has emerged. Control is moved out of the individual network nodes and into the separate, centralized controller. SDN switches are controlled by a Network Operating System (NOS) that collects information using the API shown in Figure 2B and manipulates their forwarding plane, providing an abstract model of the network topology to the SDN controller hosting the applications.

The controller can therefore exploit complete knowledge of the network to optimize flow management and support service-user requirements of scalability and flexibility. For example, bandwidth can be dynamically allocated into the dataplane from the application.

In Figure 3, once the first packet of a new flow arrives at the switch from the sender (Step 1) the switch checks for a flow rule for this packet in the SDN cache (Step 2). If a matching entry is found, the instructions associated with the specific flow entry are executed e.g. update counter, packet/match fields, action set, metadata. Packets are then forwarded to the receiver (Step 5).

If no match is found in the flow table, (Step 3) the packet may be forwarded to the controller over a secure channel. Using the southbound API (e.g. OpenFlow, ForCES, PCEP etc.), the controller can add, update, and delete flow entries, both reactively (in response to packets) and proactively. The controller executes the routing algorithm and (Step 4) adds a new forwarding

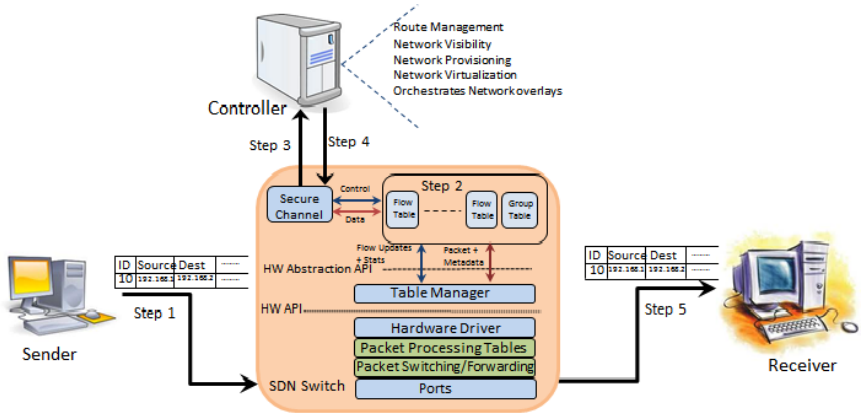

Figure 3: The Operation of SDN (Controller-Node)

entry to the flow table in the switch and to each of the relevant switches along the flow path. The switch then forwards the packet to the appropriate port to send the packet to the receiver (Step 5).

Where does SDN take us?

SDN implementation opens up means for new innovation and new applications. Dynamic topology control i.e. adjusting switch usage depending on load and traffic mapping becomes possible with the global network view. This introduces scope for network-wide access control, power management and home networking, for which the network view is not beneficial but absolutely necessary.

Furthermore, the network programmability possible in SDN allows seamless communication at all levels, from hardware to software and ultimately to end users (network operators). Programmability makes applications aware of the network and the network aware of applications. This enables greatly improved use of resources and opens up the potential for new applications with the associated potential for revenue-generation e.g. flow-metering in which cost plans can be defined based on a level of service provision.

3 KEY CHALLENGES

SDN holds great promise in terms of simplifying network deployment and operation along with lowering the total cost of managing enterprise and carrier networks by providing programmable network services. However, a number of challenges remain to be addressed. This section focuses on four specific questions arising from the challenges of SDN.

Performance vs. Flexibility: How can the programmable switch be achieved?

One fundamental challenge of SDN is how to handle high touch, high security, high per-

formance packet processing flows in an efficient manner. There are two elements to consider; performance and programmability/flexibility.

In this section, performance refers specifically to the processing speed of the network node considering both throughput and latency. Programmability means the capability to change and/or accept a new set of instructions in order to alter functional behaviour. Flexibility is the ability to adapt systems to support new unforeseen features (e.g. applications, protocols, security measures).

There are a number of initiatives [3, 4] underway to allow programmability of existing network technologies in a manner conformant with the goals of SDN. Beyond these, the SDN programmability and performance problem remains a challenge to achieve node bandwidth beyond 100 Gbps.

Figure 4 outlines the main technologies used for network processing in terms of their relationship (trade-off) between programmability/flexibility and performance.

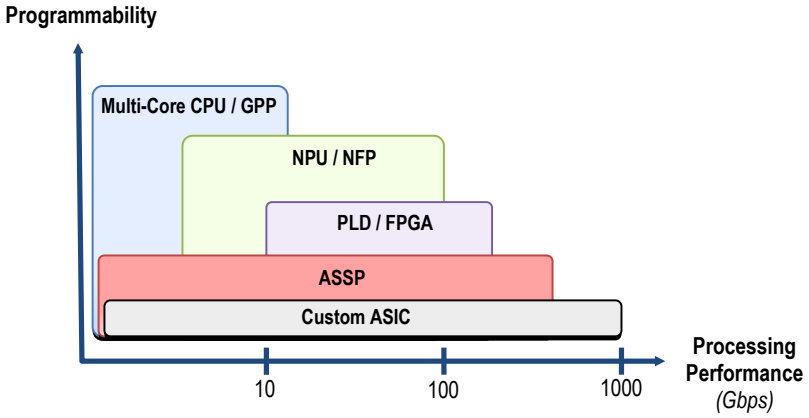

Figure 4: Network Processing - Performance vs. Programmability

General Purpose Processors (CPU/GPP) provide the highest flexibility. High-level programming languages and design tools enable the highest design abstraction and the rapid development of complex packet processing functions. The limitation of CPU implementation, however, is its performance and power dissipation, constrained by the general purpose architecture. Nevertheless, multi-core processors such as those of the Intel Xeon family [5] can achieve several tens of Gigabits of throughput by load balancing traffic onto multiple cores.

Network Flow Processors (NPU/NFP) are optimized processor architectures for network processing. Instructions and interconnects are tailored for processing packetized data. Dedicated hardware accelerators and various interface technologies are used for acceleration while reducing power dissipation. However, the flexibility of implementation is reduced as more detailed knowledge of the device is required in order to define the packet/flow processing function

and to take full advantage of the device's parallel processing capabilities. State-of-the-art NPUs such as the Netronome NFP6xxx [6] offer 216 micro-cores promising flow processing performance of over 200 Gbps line-rate per device and well over 100 Mpps.

Programmable Logic Devices (PLD) or Field Programmable Gate Arrays (FPGAs) have evolved into a technology for telecommunication and network processing. In comparison to microprocessors, PLDs are configured using hardware design tools. This technology is ideal for implementing highly parallel and pipelined data paths that are tailored for individual network processing functions. PLD technologies such as the Tabula ABAX2 [7] can achieve custom data-path processing of over 200 Gbps per device e.g. 200 Mpps switching.

Application Specific Standard Products (ASSP) are the cornerstone of high-performance networks. They are designed and optimized for widely used functions or products aiming for high-volume. The drawback of ASSPs is their limited flexibility. Core ASSP domains are physical and data-link layer products, switching and wireless products. In recent years, SDN-specific ASSPs have been introduced by Intel, Broadcom and Marvell targeting primarily high-performance Ethernet switching with virtualization and OpenFlow support for over 500 Gbps switching.

Application Specific Integrated Circuits (ASIC) are proprietary devices custom-built by system vendors (e.g. Cisco, Huawei, Juniper etc.) when standard products are unavailable and programmable solutions are unable to meet performance constraints. As an application-specific solution, ASICs offer the lowest flexibility while providing the highest performance, power and cost benefits. SDN products are expected to be comprised of proprietary ASICs to implement the SDN data plane.

Taking into account the programmability/performance trade-off of data processing technologies, it is evident that only a hybrid approach will provide an effective technology solution for SDN. Main SDN node functions can be decomposed into clusters of sub-functions such that feature-specific technologies (within or across nodes) are used to satisfy the best performance versus programmability trade-off in terms of power dissipation, cost and scalability.

For example, building a platform based on custom-built devices (e.g. PLDs/ASSPs) combined with NPUs/NFPs and a CPU/GPP presents a hybrid programmable architecture. Such a platform can support fast forwarding on established flows in the network along with programmability and controlled processing for encapsulated traffic and new flows.

One goal of SDN is to develop networks built on general purpose hardware. The combination of technologies as described in the hybrid architecture supports this goal. With a programmable interface built on standard hardware, a multi-vendor equipped network becomes a possibility.

Scalability: How to enable the Controller to provide a global network view?

Assuming that the performance requirements can be achieved within the hybrid programmable archtitecture, a further issue that has seen some discussion but limited solution is scalability in SDN.

The issue can loosely be split into controller scalability and network node scalability. The focus here is on controller scalability in which three specific challenges are identified. The first is the latency introduced by exchanging network information between multiple nodes and a single controller. The second is how SDN controllers communicate with other controllers using the east and westbound APIs. The third challenge is the size and operation of the controller back-end database.

Considering the first issue, a distributed or peer-to-peer controller infrastructure would share the communication burden of the controller. However, this approach does not eliminate the second challenge of controller-to-controller interactions for which an overall network view is required.

Traditional packet networks lend themselves to scalable solutions because they do not require extensive state to be held between system units. Each network node is autonomous, requiring only limited knowledge of its neighbours. Routing protocols have been designed to control traffic with this in mind. In order to create resilient networks, alternative paths and secondary equipment are required. It may then be necessary to hold some state between systems to ensure that should a failure occur, there is little or no interruption in service. Typical systems that require this functionality include network elements such as Load Balancers and Firewalls.

Within a pure SDN environment, a single controller or group of controllers would provide control plane services for a wider number of data-forwarding nodes, thus allowing a system-wide view of network resources.

Other approaches that match the goals of SDN with existing routing protocols involve addition of an orchestration layer exposing an API that application elements may use to request desired performance from the transport layer.

An extension to the Application Layer Traffic Optimization (ALTO) data model has been proposed by various organizations in which the ALTO server hosts aggregated information to which each controller has a link. The goal of ALTO is to guide applications in their selection of one of several hosts capable of providing the desired resource. A vertical architecture with bi-directional information flow between each SDN controller and the ALTO server is proposed in [8] to support the global network view. In terms of improving application performance, ALTO with SDN would be a powerful tool.

A specific solution to controller scalability is HyperFlow [9]. HyperFlow is a controller application that sits on the NOX controller and works with an event propagation system. The

HyperFlow application selectively publishes events that change the state of the system and other controllers replay all the published events to reconstruct the state. By this means all the controllers share the same consistent network-wide view.

Indeed, this concept of providing the network view by distributing the state over multiple controllers is highlighted in [10] in which a series of solutions to controller scalability are described. Notably, in [10], the authors conclude that the flexibility of SDN provides an opportunity in terms of network manageability and functional scalability.

On the way to achieving full scalability for SDN, an evolutionary approach to network programmability will be necessary. For example, with the hybrid architecture a volume of queries can be resolved in the node CPU, which would otherwise be transferred to the controller for processing. This can potentially reduce the database size at the controller and simultaneously reduce communication between the controller and its nodes.

Security: How can the Software-Defined Network be protected from malicious attack?

There has been limited industry and research community discussion to date on the security issues associated with SDN. A greater focus on security is therefore required if SDN is going to be acceptable in broader deployment. Indeed a security working group has been set up within ONF with this in mind. A number of issues are highlighted here that underscore the need for further study and development of security solutions.

Potential security vulnerabilities exist across the SDN platform. At the controller-application level, questions have been raised around authentication and authorization mechanisms to enable multiple organizations to access network resources while providing the appropriate protection of these resources [11]. Not all applications require the same network privileges and a security model must be put in place to isolate applications and support network protection.

One potential solution is role-based authorization. FortNox [12] is proposed to resolve the situation when a controller receives conflicting flow rules from two different applications. Role-based authorization alone, however, does not present a solution for the complexity of SDN requiring isolation of applications or resources.

The controllers are a particularly attractive target for attack in the SDN architecture open to unauthorized access and exploitation. Furthermore, in the absence of a robust, secure controller platform, it is possible for an attacker to masquerade as a controller and carry out malicious activities. In the past, such attacks have targeted DNS servers e.g. Kaminsky DNS attack [13]. Considerably greater damage could be done by such an attack on an SDN controller.

A security technology such as Transport Layer Security (TLS) with mutual authentication between the controllers and their switches can mitigate these threats. Current specifications of OpenFlow [1] describe the use of TLS. However, the security feature is optional and the

standard of TLS is not specified. A full security specification for the controller-switch interface must be defined to secure the connection and protect data transmitted across it.

With a single controller controlling a set of network nodes, implementation of authentication with TLS may provide the necessary security. However, with multiple controllers communicating with a single node or multiple control processes communicating with a single, centralized controller, authorization and access control becomes more complex. The potential for unauthorized access increases and could lead to manipulation of the node configuration and/or traffic through the node for malicious intent.

One potential malicious attack is the Denial of Service (DoS) attack. Within the operation of SDN, as illustrated in Figure 3, there are two options for the handling of a new flow when no flow match exists in the flow table. Either the complete packet or a portion of the packet header is transmitted to the controller to resolve the query. With a large volume of network traffic, sending the complete packet to the controller would absorb high bandwidth.

However, if only header information is transmitted to the controller, the packet itself must be stored in node memory until the flow table entry is returned. In this case, it would be easy for an attacker to execute a DoS attack on the node by setting up a number of new and unknown flows. As the memory element of the node can be a bottleneck due to high cost, an attacker could potentially overload the switch memory.

Furthermore, with the introduction in SDN of open interfaces and known protocols to simplify network programming by any application provider, the door is thrown open to attackers. With full knowledge of how to control the network, with access to the controller, the operation of the network can quickly and easily be subverted to the benefit of the attacker. Even at a lower level, individual network nodes, hosts or users could be targeted undermining the desired network performance. Such issues must receive due consideration in the SDN platform design.

On the plus side, the SDN architecture supports a highly reactive security monitoring, analysis and response system. From the security perspective SDN can support:

- **Network Forensics:** facilitate quick and straightforward, adaptive threat identification and management through a cycle of harvesting intelligence from the network, analyzing it, updating policy and then reprogramming to optimize from network experience.

- **Security Policy Alteration:** allow you to define a security policy and have it pushed out to all the infrastructure elements, reducing the frequency of mis-configuration and conflicting policies across the infrastructure.

- **Security Service Insertion:** facilitate security service insertion where applications like firewalls and Intrusion Detection Systems (IDS) can be applied to specified traffic according to the organization's policies.

However, the security of SDN will only be as good as the defined security policy. Implementation of existing authentication and authorization mechanisms can resolve some aspects of the security challenge. Meanwhile, threat detection and protection techniques will continue to evolve. The key, though, is for individual organizations to effectively and comprehensively define their security policies in order to exploit the full extent of available network protection.

Interoperability: How can SDN solutions be integrated into existing networks?

To answer this question requires consideration of interoperability andstandardization to support the transition from the traditional network model to SDN.

It would be straightforward to deploy a completely new infrastructure based on SDN technology. For this, all elements and devices in the network would be SDN-enabled. However, there exists a vast, installed-base of networks supporting vital systems and businesses today. To simply "swap-out" these networks for new infrastructure is not going to be possible and is only well suited for closed environments such as data centres and campus networks.

The transition to SDN therefore requires simultaneous support of SDN and legacy equipment. The IETF Path Computation Element (PCE) [14] could help in gradual or partial migration to SDN. With PCE, the path computation component of the network is moved from the networking node to a centralized role while traditional network nodes not using PCE continue to use their existing path computation function. A specific protocol (PCEP) enables communication between the network elements. However, PCE does not provide complete SDN. The centralized SDN controller supports complete path computation for the flow across multiple network nodes.

Further development is required to achieve a hybrid SDN infrastructure in which traditional, SDN-enabled and hybrid network nodes can operate in harmony. Such interoperability requires the support of an appropriate protocol which both introduces the requirements for SDN communication interfaces and provides backward compatibility with existing IP routing and MPLS control plane technologies. Such a solution would reduce the cost, risk and disruption for enterprise and carrier networks transitioning to SDN.

Introducing a new protocol requires consideration of standardization and where this standardization will be of most benefit. ETSI Network Function Virtualisation (NFV) Industry Specification Group [4] intends to standardize components within the core network that may be virtualized to provide efficient scalability and placement of those services. IETF's Forwarding and Control Element Separation (ForCES) WG has been working on standardizing interfaces, mechanisms and protocols with the goal of separating the control plane from the forwarding plane of IP routers. ONF is standardizing OpenFlow as a communication protocol within the network and is driving the standards of related protocols, such as the OpenFlow management

and configuration protocol. Many programming languages such as Frenetic, Procera etc. are also being proposed to resolve the northbound API link.

The work of the IETF, ETSI, ONF and other industry working groups must be coordinated in order to take advantage of existing standards in networking while proposing and developing the most effective standards to support migration from the traditional network model to SDN.

4 CONCLUSION

SDN has emerged as a means to improve programmability within the network to support the dynamic nature of future network functions. As bandwidth demand escalates, the provision of additional capabilities and processing power with supportfor multiple 100GE channelswill be seamless through an SDN-based update and/or upgrade. SDN promises flexibility, centralized control and open interfaces between nodes enabling an efficient, adaptive network.

In order to achieve this goal, a number of outstanding challenges must be resolved. In this article we have presented a discussion of a number of challenges in the area of performance, scalability, security and interoperability. Existing research and industry solutions could resolve some of these problems and a number of working groups are also discussing potential solutions. In addition to these, the hybrid programmable architecture could be a means to counter performance and scalability issues introduced by SDN. The objective of the model is to optimize flow processing in the network.

The original data networks were formed out of a combination of computing devices with data and network nodes to transfer this data between the source and destination. The ability to provide "X"-as-a-service (XaaS) through virtualization technology has increased the volume of data in the network. This has set a baseline for a new communication method by pushing computation into the network devices increasing machine-to-machine communications.

The future of networks will be shaped around this progression. The goal is to provide effective communications and services where network, data and computation are fused into a service architecture. In the future, for a specific process, data will request the computing, storage and connection that it requires before launching the application. The location of the network elements might be distributed physically and virtually but this will be entirely opaque to the end user. All the user will observe is the quality of delivery of the requested service.

SDN will contribute to this vision of future communications. However, significant issues must be addressed in order to meet expectations. Indeed consideration of the potential for application-driven networks might lead us to wonder whether SDN as currently envisioned is even sufficient. Nevertheless, it is certain that SDN is here to stay as an evolutionary step for paving the way for a highly optimized ubiquitous service architecture.

References

[1] "Software-Defined Networking: The New Norm for Networks," Open Networking Foundation, White Paper. [Online]. Available: https://www.opennetworking.org

[2] "ITU.T Recommendation Y.1731 OAM Functions and Mechanisms for Ethernet based Networks." [Online]. Available: http://www.itu.int/itudoc/itu-t

[3] "Interface to the Routing System," IRTF Working Group. [Online]. Available: https://datatracker.ietf.org/wg/irs/charter/

[4] "Network Function Virtualisation," ETSI Industry Specification Group. [Online]. Available: http://portal.etsi.org/portal/server.pt/community/NFV/367

[5] R. Ozdag, "Intel Ethernet Switch FM6000 Series - Software Defined Networking." [Online]. Available: http://www.intel.co.uk/content/dam/www/public/us/en/documents/white-papers/ethernet-switch-fm6000-sdn-paper.pdf

[6] "Netronome NFP6XXX Flow Processor." [Online]. Available: http://netronome.com/pages/flow-processors/

[7] "Tabula." [Online]. Available: www.tabula.com

[8] "Use Cases for ALTO with Software Defined Networks," Internet Engineering Task Force ALTO Working Group. [Online]. Available: http://tools.ietf.org/pdf/draft-xie-alto-sdn-use-cases-01.pdf

[9] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, 2010.

[10] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, February 2013.

[11] ""Security Requirements in the Software Defined Networking Model"," IETF Network Working Group Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-hartman-sdnsec-requirements/

[12] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A Security Enforcement Kernel for OpenFlow Networks," in *Proceedings of the 1st Workshop on Hot topics in Software Defined Networks*. ACM, 2012, pp. 121–126.

[13] "Kaminsky DNS Attack." [Online]. Available: http://dankaminsky.com

[14] "Path Computation Element," IETF Working Group. [Online]. Available: http://datatracker.ietf.org/wg/pce/charter/