Software Defined Networking – how BSS/OSS tools can help unleash innovation

Software Defined Networking (SDN) is currently a widely discussed topic in the telecommunications industry. There are high expectations regarding the technology, including reducing network maintenance costs and unleashing innovation, thus opening the way to new revenue sources and better network monetization. SDN is a concept where the main principle is to separate the control plane from the data plane, and to move the controller function out from today's routers, leading to the introduction of OpenFlow switches.

At first this definition does not sound very exciting, but placing the controller function centrally should enable much more "intelligent" network traffic control and, as a result, efficiently deliver new, innovative services for customers. The problem is that for now SDN is still only a concept, and currently the only tangible specification is OpenFlow, but it only defines the protocol between a controller and a switch.

To make the promises of SDN technology come true, there is a need for a platform, enabling a business application that will help opening up telecom networks. The specifications and APIs for this kind of a business application need to be defined to shape the network according to what is required and make it "smarter". In order for the latter to happen, a controller needs a comprehensive end-to-end view of the network and all connected services. However, the SDN concept does not define, how to provide such an end-to-end view.

One idea is to leverage the operators' existing assets like the BSS/OSS ecosystems and prove that SDN won't make BSS/OSS investments obsolete. This article describes an IT architecture, where BSS/OSS investments can not only be saved, but even act as a significant enabler for the SDN "revolution". This means that telecom operators will be able to provide significant added value to the SDN ecosystem.

What is SDN?

Since SDN is a concept, not a specification, it is in a way open for interpretation. However, the main principles seem to be well defined by the Open Networking Foundation, an industry organization that promotes and guides the adoption of SDN principles.

The main principle of SDN is the separation of control and data planes and centralizing the controller function, thus moving it away from individual network elements. The idea of separating the control plane from the data plane is by far nothing new, but in this context it means that the network can be "liberated" from the hardware limitations of routers.
As the behavior of the network is determined mostly by the control function, it also means that networks can be virtualized and can have resources allocated as required by the application layer. This is similar to virtual machines (VM) technology, widely used in data centers – a domain that in fact saw the first adoption of SDN. If virtual machines can adapt to application software needs, it would be great to have networks adapting as well.

The SDN architecture promoted by the Open Network Foundation in re-created form is depicted in Figure 1.

![Figure 1. SDN architecture (patterned on Open Flow Foundation materials).](image)

The architecture assumes using OpenFlow as a protocol between a controller and network elements. OpenFlow, unlike SDN, is not just a concept – it is a formal specification that facilitates building switches and controllers compliant with it. A whitepaper that has been published on OpenFlow [1] describes it as a way for researchers to run experimental protocols in the networks they use every day. At first glance it may not sound like something of great commercial value, but it’s not uncommon that something invented in universities as an internal tool for researchers ends up having huge commercial value. The history of internet (and hypertext in particular) proves this.

The fundamental element of OpenFlow is the separation of the data and control layers. The specification defines an OpenFlow switch, the primary role of which is to forward packets between ports according to flow tables – a concept that’s not new, as this is what switches usually do. What’s new is that the OpenFlow switch specification defines that the flow tables are managed by an external entity, a central controller for the entire network. This is a radical change compared to a previously observed evolution, which had led to more and more “intelligent” routers. The problem with those “intelligent” routers is that placing the controller function inside individual network elements makes the equipment more expensive. What’s even more important is that by being placed in individual network elements, the traditional controller does not get an end-to-end view of the network, inherently limiting the “smartness” of the controller's decisions.
With OpenFlow the situation is completely different. According to this concept, a centralized controller has an end-to-end view of the network and its “smartness” can evolve by just upgrading the software, without the need to upgrade the entire network hardware (i.e. switches). This is because there is a synergy with the Network Function Virtualization concept. It assumes that network functions are implemented in the software, not in the hardware. In other words, it means that software-based network functions do not require installing any extra hardware equipment. Controller software can be deployed on standard equipment (even PCs) and physical hardware boxes can be allocated to the software needs. This means that the SDN controller can have virtually unlimited “intelligence”, since it is not confined to dedicated individual IP Router hardware.

Explaining the SDN concept with OpenFlow may be difficult, but the best way to see its potential is to become familiar with use cases promoted by the Open Network Foundation, described in the next paragraph.

**SDN use cases – understanding the potential of the technology**

Since SDN is a new technology, its full potential is still being examined. Currently, a number of use cases have been presented by the Open Networking Foundation or by individual industry players. Presenting them all here would be impossible, so a few have been chosen to highlight its benefits.

**Ericsson and Smart Service Routers**

The use case scenario, demonstrated by Ericsson using their Smart Services Routers, assumes that home users are connected to the Internet by a communications service provider. When a user begins an internet session, the packets are forwarded to an authentication service. In the case of children accessing the Internet, the authentication is done according to a defined policy. Once the packets get authenticated, the central SDN controller defines the flow tables in a way that the whole traffic goes through a content filtering service. When it’s adults (parents) accessing the Internet, according to the agreed policy, the central SDN controller modifies the flow tables so that the traffic flows directly, by passing a content filter. Another scenario is e.g. accessing a trusted movie source, which would see the central controller reconfiguring the flow tables in a way that the traffic bypasses the content filtering and firewalls, thus reducing the burden on these network elements.

**Mobile wireless VoIP clients**

This use case is described in an OpenFlow whitepaper [1]. It presents the idea of a new call-handoff mechanism for Wi-Fi enabled phones. The scenario assumes that the SDN central controller is able to track, which Wi-Fi access point a phone is connected to. When a phone moves from one access point to another, the SDN controller can redefine the flow tables of the network switches and direct the traffic to the phone. This use case does not intend to present any alternative to cellular phones, but only demonstrates the huge potential of SDN technology. In this scenario service providers can benefit purely by implementing the software, without any hardware upgrades whatsoever.

**Providing Bandwidth on Demand, Bandwidth Exchange and Pay for Service Quality**

Another group of use cases is provided in another white paper by the Open Network Foundation [2]. It describes the concept of applying SDN technology to provide Bandwidth on Demand, Bandwidth Exchange and Pay for Service Quality. The main idea is to use the capability of a centrally located controller which, based on flow identification, can allocate network resources according to the agreed policy. The policy may depend on the extra fees that the customer is paying, e.g. for a fee they may receive higher bandwidth, better quality of service (QoS) etc.
All the above use cases have one thing in common – the potential of SDN originates from centrally implemented control functions, which can dynamically reshape the behavior of the network in a “smart” way. This “smartness” can be further improved with additional software and data which will enhance the view of the network that the controller manages.

**How to use the existing BSS/OSS infrastructure for introducing SDN**

As presented earlier, SDN technology seems to have huge potential that originates from centralizing the control function. The OpenFlow specification only describes how to manipulate the flow tables of the network switches, but it does not define what the controller should base its “smart” decision on.

On one hand it is an advantage of the specification, as it opens the field for innovation. On the other hand, there is a need for guidelines regarding the way in which SDN technology should be implemented, since the controller’s ”smartness” depends on its ability to obtain a holistic view of the network, services and defined policies.

---

**Figure 2. SDN Architecture – employing BSS/OSS system for an e2e view.**

In order to provide the SDN controller with a holistic view of the network, telecom operators can use the BSS/OSS tools that they already have. A Network Inventory Management system delivers a complete view of a multi-technology, multi-domain network. Using existing OSS systems can be easily justified, as it helps operators get more return on the investments they have already made and avoid the costs of implementing a dedicated inventory-like system for SDN technology.

A full view of products and services can be delivered by the Service and Product Inventory systems, which are part of a typical BSS/OSS infrastructure.
In addition, the controller may need information regarding the current status of the network from the alarm and quality perspectives. This could be provided by Fault Management, Service Monitoring, Performance Monitoring, and SQM systems, as well as by the much hyped Customer Experience Management solutions.

In addition, applying Big Data and the accompanying analytics can help to transform the information that operators already have at hand into valuable assets, enabling the SDN controller to make “smarter” decisions. What’s even more important, providing APIs for these assets is expected to facilitate the development of new network services just by some further programming of the network.

The relationship between SDN and BSS/OSS – the plug-in model

Traditional network architecture assumes that BSS/OSS controls the networks in a “push” model. It means, that e.g. during the process of new service fulfillment and delivery, the OSS ecosystem needs to orchestrate network elements and apply an appropriate configuration to the network. In this model the relationship between the OSS system and the network can be described as static. It assumes that once the network is configured, it works independently until a new orchestration order is applied.

The SDN architecture can transform this static relationship into a more dynamic “pull” model. This means that a controller (a network service depicted in Figure 1 and Figure 2) may access information available in the BSS/OSS infrastructure at runtime, by pulling the data needed to make “smart” decisions. This should also have a far-reaching effect on boosting innovation.

In the “push” model, OSS needs to know what data should be configured into the network elements. In the “pull” model, the OSS system does not need prior knowledge on all the details of controller algorithms. It is the controller software that is supposed to know, what it needs to implement in order to provide the best service. This means that in the “pull model” a controller needs to “plug” into the BSS/OSS system to gather the required data. From a network service development perspective it means opening up the API to the BSS/OSS ecosystem and to controller service developers.

Opening up the network – what does it really mean?

The main promise of SDN technology, which makes it so attractive, is that it should boost service innovation by “programming” the network, so that operators do not need to upgrade the existing infrastructure (“hardware”). But to achieve this, the network must be opened up with a set of APIs.
The real strength of SDN technology is in its potential to speed up innovation and open up the network. The "smartness" of the SDN controllers comes from the ability to access a complete end-to-end view of the network.

The problem is that SDN architecture does not define the API between the business applications and the controller layer (see Figure 1). It only defines the OpenFlow protocol between the controller and the infrastructure (switches). The question is, whether the traditionally strong separation between business applications and the network should be maintained. The right approach seems to be to allow business applications should have their own "private" networks, adapted to their own needs.

From an innovation perspective, what should be considered is whether opening up the API between the application and the controller layer is enough. Is it possible to open up the OpenFlow API for developers? It may sound shocking, but if we go back to the goal of OpenFlow (as defined in the Open Network Foundation's whitepaper [1]), it is a way for researchers to run experimental protocols in the networks they use every day. Could SDN then be a way for developers to experiment with applications and networks?

This definitely leads to many security issues related to developers manipulating the flow tables of the switches. But with appropriate control over the privileges and restrictions in the ability to effect the "private" virtual network, this might be a chance to really boost innovation. This approach assumes that there could be different developers employed for end-user applications and the network, but they would closely collaborate with each other and, as a result, improve the overall service quality for end customers.

In any case, opening up the network requires API exposure. Whether the developer's API would be restricted only to the one between the application and network layers or whether it should also include the ability for developers to develop their "own" network services, based on OpenFlow, remains an open question. Even if we assume that developers won't develop network services, to really get the innovation ball rolling would still require opening up the API to BSS/OSS systems, in order to enhance the view of the network that the controller has. This would also entail security issues, especially if it is to be available to third companies, such as operators' business partners. The current problem is that SDN does not define these APIs, just the OpenFlow ones.

Summary

The Software Defined Networking (SDN) technology is very promising and expected to help operators in reducing costs and boosting service innovation. The cost reduction factor derives naturally from centralizing the network control functions. Following the Network Function Virtualization concept, it ensures that the control function can be implemented on standard equipment (even PCs).

But the real strength of this technology is in its potential to speed up innovation and open up the network. The "smartness" of the SDN controllers comes from the ability to access a complete end-to-end view of the network. Instead of implementing a completely new infrastructure for an SDN controller, the end-to-end view can be delivered by the existing BSS/OSS systems.
In this way, combined with the potential of Big Data and analytics, the "smartness" of the controller could be virtually unlimited. But to achieve this, BSS/OSS APIs must be defined.

In addition, to implement the idea of defining "private" virtual networks for applications, the APIs must be available for developers. If opening up the OpenFlow API to developers is possible, it will probably raise more questions, but at the very least the application to control layer APIs must be defined and opened.

References

Comarch is a global supplier of IT products and services for the telecommunications industry that has been present on the market since 1993. Comarch’s areas of expertise include BSS/OSS, M2M and Cloud Service platforms, as well as a range of Managed Services. The company’s flexible solutions adhere to the best industry standards and have all been developed in-house. Having completed projects for over 50 telecom operators, Comarch has accumulated vast experience in the fields of designing, implementing, and integrating IT solutions. One of the company’s main strengths is the loyalty of its customers, who include leading brands such as: Vodafone, T-Mobile, Ö2, E-Plus, MTS and KPN.