A use-case based analysis of network management functions in the ONF SDN model

Alisa Devlic, Wolfgang John
Ericsson Research
Kista, Sweden
{alisa.devlic, wolfgang.john}@ericsson.com

Pontus Sköldström
Acreo AB
Kista, Sweden
pontus.skoldstrom@acreo.se

***Abstract*— The concept of software-defined networking (SDN) recently gained huge momentum in the industry, driven mainly by IT companies interested in datacenter applications. In this paper, however, we apply SDN to the carrier domain, which poses additional requirements in terms of network management functions. As a specific use-case we take a virtualized carrier network shared by multiple customers. We consider the current SDN model as defined by the Open Networking Foundation (ONF), including the OpenFlow and OF-config protocols. Through a step-by-step discussion of the procedures required to configure and manage the virtualized network, we analyze the applicability of the current SDN model as specified by the ONF. As a result, we identify shortcomings and propose necessary extensions to the ONF SDN model. The highlighted extensions include control network bootstrapping considerations, updates to the SDN and NOS model, and most importantly extensions of the OF-config management data model.**

I. Introduction

Traditional network elements have been designed as autonomous entities (Fig. 1, left) using a distributed control plane to communicate with the outside world. Various protocols allow autonomous decisions of what actions to take. Typically this involves a number of processes running within a closed operating system (OS) calling a proprietary API which in turn causes the OS to program specialized forwarding hardware, again using a proprietary API. Adding new functionality to a network element usually involves standardizing a new protocol that reinvents mechanisms such as distribution and signaling, and waiting for the vendors to implement the new protocol.

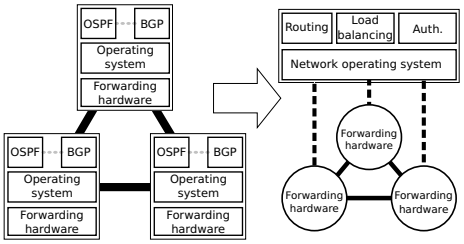

Fig. 1. From autonomous network elements to Software Defined Networking.

Software Defined Networking (SDN) proposes a new model by creating open APIs between the hardware and the operating system, and between operating system and *network applications*. In the SDN model (Fig. 1, right) a *Network Operating System (NOS)* is responsible for maintaining an up-to-date view[1] of the network and its current state. The NOS does not only maintain a view of the network, but is also responsible for handling changes to the view and transferring those changes both to the network hardware and to the network applications. Changes to the view come either from *network applications* running on top of the operating system or from the underlying network hardware, e.g. in the case of failures. The network applications are software modules that are able to access and modify the network view maintained by the NOS, which greatly simplifies the addition of new functionalities. It only takes to write a software module utilizing the API provided by the NOS, and the NOS is responsible for updating the network and distributing the new state.

SDN research and development is currently centered around an official standardization body, the Open Networking Foundation[2], which was established in 2011. The ONF has so far mainly focused on the specification of *OpenFlow* [1], an open protocol designed to expose the internals of a network element and provide an API to modify them. The protocol basically models a network element as *flow table(s)*. Flow tables contain rules that can be used to match incoming packets and associate them to a number of actions. If an incoming packet does not match an existing rule in the network element, the packet can be sent to the NOS where a network application can investigate the packet further and decide what to do, e.g., installing a new rule that takes care of all packets in this particular packet flow. The OpenFlow protocol is mainly intended for managing the flow table(s) by installing permanent or transient rules with a relatively high frequency.

A. Status of Network Management in SDN

The SDN framework enables centralized control of data path elements, independently of the network technology used to connect these devices originating from different vendors. The centralized control embeds all the intelligence and maintains the network-wide view of the data path elements and links that connect them. This centralized up-to-date view makes the NOS suitable to also perform network management (NM) functions.

Acknowledging the need for explicit NM functionalities in SDN, the ONF has recently proposed OF-config as additional

[1] A graph of the nodes and links in the network, with all their attributes.
[2] www.opennetworking.org

978-0-7695-4870-8/12 $26.00 © 2012 IEEE
DOI 10.1109/EWSDN.2012.11

configuration and management protocol besides OpenFlow. OF-config is based on NETCONF [2], a transactional protocol that uses remote procedure calls (RPCs) on top of a secure transport channel (such as SSL and SSH) to manage configurations on remote devices. It provides methods for installing, manipulating, and deleting configuration not only on a single device but also on multiple devices within a single transaction. While NETCONF itself is XML-based, the data model - that describes what can be configured and how different classes of configuration relate to one another - is written in YANG [3]. OF-Config adopts the NETCONF protocol, extending it with specific YANG models.

Figure 2 shows the current SDN architecture as defined by ONF [4]. In this model, an *OpenFlow capable switch*, which is a physical or virtual network element, is hosting one or more *OpenFlow logical switches*. The logical switches represent the actual OpenFlow network elements, which are controlled by one or more *OF Controllers* via the OpenFlow protocol. *Network Apps* on top of the OF Controller use the network via the OF Controller's northbound API (NB API)). Finally, an *OF Configuration Point* represents the service which communicates via the OF-Config protocol with an OpenFlow capable switch and partitions resources among OF logical switches (such as ports and queues). The relationship between OF Controller and OF Configuration Point is currently deliberately not defined by the ONF, however, both functional elements can be considered to be part of the NOS in general SDN terms.

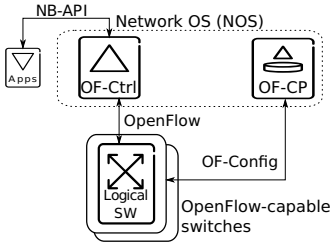

Fig. 2. The current ONF SDN model with an OF Configuration Point (OF-CP) separate from the OF Controller (OF-Ctrl).

B. Our contribution

SDN has gained momentum in the industry forum, being primarily driven by leading IT companies with a focus on data center applications. However, the focus of this paper is on SDN in modern carrier networks, as studied within the framework of the EU FP7 SPARC project[3]. In order to be considered a carrier grade network, an SDN solution needs to provide network and service management capabilities [5], [6].

In this paper, we analyze the ONF model by applying it on the use-case of a virtualized carrier network. We put the a focus on network configuration, but also consider fault and performance management functions[4]. As a result of this analysis, we derive an updated view of the ONF model and propose extensions to the ONF protocols.

[3]www.fp7-sparc.eu

[4]The FCAPS functions Security and Accounting are not considered here.

We describe the use case of a virtualized carrier-grade SDN in section II. We then discuss the procedures used to configure and monitor this type of network scenario in section III. In order to fulfill these required functional areas, we propose updates to the SDN model in terms of NOS and protocol extensions in section IV. Finally, we summarize and conclude the analysis in section V.

II. Use case

In this paper, we consider virtualization of carrier-grade networks in order to share physical infrastructure among multiple virtual network operators (VNOs) (see Fig. 3). Besides the expected OPEX and CAPEX savings for managing such a complex network, using SDN to control the entire infrastructure enables to go beyond traditional services such as E-Line or E-LAN by offering different abstraction levels to the customers. As example, some customers may want managed connectivity services (e.g., E-Lines), treating the underlying network as a black-box. Others may require more fine-grained exposure of network details in order to gain full control of the network, e.g. for advanced traffic engineering (TE) purposes.

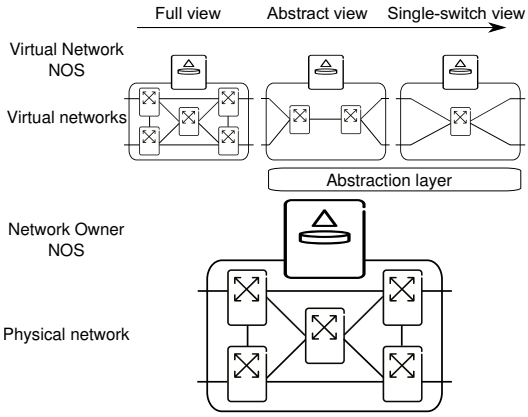

Fig. 3. The use case considered: A carrier-grade virtualized SDN able to provide clients with services ranging from point-to-point (E-Line like) services to fully virtualized SDN networks.

We desire an SDN design able to provide full network virtualization through which a customer can connect their own NOS to the logical switches. Additionally, we propose an abstraction layer, which provides an abstracted view of the virtual networks that are running on top of the physical infrastructure. An example of an abstracted view can be an abstract switch consisting of a number of virtual switches and ports. The advantages of having such a layer between customers and switches is that it provides any level of desired customer control, ranging from full dataplane exposure to a single-switch interface. Virtualization is, in this use case, done directly on top of the physical network [7]–[10]. Specifically, we consider datapath virtualization as described in [9], since it provides the highest level of flexibility by giving each VNO access to the full flow space. This virutalization approach applies encapsulation-based link separation combined with

flow table partitioning and strict resource isolation. Virtual connectivity is provided by tunnels, which can be monitored using OAM tools (e.g., BFD or performance monitoring tools) that are adapted to SDN/Openflow [11].

In the cases where network details are exposed to the customer (i.e., a VNO), the network is controlled via the VNO's own NOS. The advantages of this scenario could include more efficient network utilization through customer controlled TE and delegation of control to applications using the network (e.g., over-the-top, network-aware applications). Besides, the proposed approach is also reducing the amount of management tasks for the physical network owner. Advantages of this approach have already been identified in the data center/cloud environments [12], but we argue that this concept can be generalized to any type of networks, including carrier-grade access/aggregation and Fixed Mobile Convergence (FMC).

The desired topology, level of abstraction, switch capabilities, network bandwidth and switch resources of the virtual network are specified in an agreement between the VNO and the network owner. An SLA is also part of this agreement.

III. Configuration procedures

In this section, we discuss a step-by-step procedure list for configuration of virtual SDNs on top of one physical infrastructure, according to the use-case. For many of the steps, multiple alternative solutions might exist - in these cases we try to detail the solution which allows for the highest level of automation. Note that not all steps listed are currently supported by the ONF SDN model. The missing functions will be listed later in section IV.

The procedure list starts with bootstrapping the devices and configuration of the physical network. In order to fulfill SLAs, we configure performance and fault monitoring on the physical network. The procedure completes when the virtual network topologies are computed, configured, validated, and handed over to customers.

1) **Device configuration and network bootstrapping:**
   a) Newly connected switches request connection identifiers for connecting to the OF Configuration Point[5]. The connection identifiers required are at least the local address and the address for the OF configuration point. If non-default values are used, this may also include transport protocols and port numbers. For example, in an IP based control network **address (auto)configuration** can be done via DHCP [13] (see Fig.4). Additionally, if a certificate from an authentication service is required, the connection identifiers for this service also need to be provided.
   b) Credentials for a secure connection to the OF config point and the OF controller can be pre-configured on the device. For a more dynamical bootstrapping process, credentials can be requested from an **authentication and authorization** service (AA). In this case, initial trust is established with the AA service using an authentication mechanism (e.g. by adding a unique switch identifier to a database). Once trust is established, the required certificates (e.g. SSH keys and X.509 certificates) are transferred to the switch.
   c) The switch initiates an **OF-Config session** with the OF configuration point. They use the obtained certificates to mutually authenticate themselves.

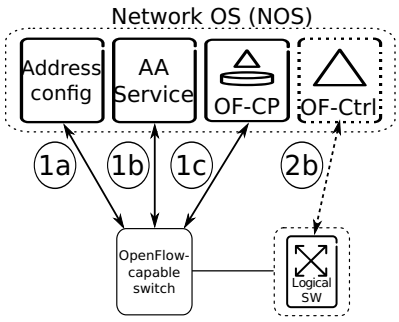

Fig. 4. Bootstrapping procedures: 1a) address configuration; 1b) authentication and authorization; 1c) OF-Config session to the OF control point (OF-CP); 2b) OF sessions to the OF controller (OF-Ctrl) are not established before a logical switch is instantiated. The entities depicted in the top of the figure can be separate nodes, but are here considered to be part of the NOS.

2) **Physical network configuration:**
   a) **Physical resources and capabilities** are discovered by the OF configuration point using the newly established OF-config session. Examples of physical resources are switch CPU, memory, and ports. Capabilities of the OF-capable switch are hardware and software capabilities, such as a list of supported OAM tools, OpenFlow actions and QoS functions.
   b) In order to connect the switch to an OF controller, the OF configuration point first needs to instantiate a logical switch with access to all physical ports, which we refer to as the **master logical switch**[6]. The master logical switch is assigned a minimum amount of resources (e.g., CPU, bandwidth) and a high priority in order to ensure control also during periods of high load. The OF configuration point then assigns connection identifiers and credentials for an OpenFlow session to a **master OF controller**, i.e., the controller of the infrastructure owner. The result is an established **OpenFlow session** with access to all physical ports.
   c) **Discovery of the physical topology** can be done in a controller-based (i) or distributed manner (ii).
   (i) When considering pure OpenFlow switches, the OF controller is required to take part in the physical topology discovery. The master OF controller that is connected to the master logical switch initiates physical topology discovery using controller based methods, e.g. the centralized LLDP mechanism used in the NOX discovery module. The discovered physical topology is shared with the master OF configuration point.

[5]We assume either an existing out-of-band or in-band control network.

[6]Please refer to step 4b for details on the instantiation.

(ii) Discovery of the physical topology can also be done in a distributed way, in which switches autonomously use methods such as LLDP or a Spanning Tree Protocol. The local adjacencies can then be retrieved by the OF-config point in order to create the global view of the topology. This approach assumes support for legacy protocols in the switches (i.e., hybrid switches).

3) **Physical fault and performance monitoring:** After discovering the physical topology, it is important to verify that the physical links and resources are correctly instantiated and to continuously monitor their performance. To achieve this, OAM tools for fault detection and performance monitoring are needed.
   a) **OAM** for fault detection and performance monitoring of physical links is configured on the OF-capable switches. Additionally, monitoring of other switch resources, such as CPU and memory, is configured. For each OAM tool the following parameters can be configured: parameters for scheduling of measurements and reporting, thresholds, and switch local actions.
   Scheduling of measurements and reporting includes the type of measurements (active, passive), the monitoring periods (i.e., how often the measurement is performed), and the reporting method (periodic, on demand, on change, or when a threshold is breached).
   Thresholds are configured on performance metrics such as bandwidth performance degradation or specific link state changes.
   Switch local actions are triggered upon threshold violations and can be corrective actions such as failover, flow table actions, or alarms that are sent to the NOS through the OpenFlow session.
   b) When an alarm or measurement data is received by the NOS it can react in several ways, for example through network restoration, physical resource reassignment, or potentially migration of the logical switch to another physical switch. Details about alarm and measurement handling at the NOS are out of scope of this paper.

4) **Virtual network setup:**
   a) Using the discovered physical topology and switch capabilites combined with pre-established customer agreements for virtual network topologies and SLAs, the virtual topologies can be computed. Based on these virtual topologies, the master OF configuration point can create **virtual links**. Virtual links are tunnels requiring configuration of tunnel endpoints (i.e., logical ports) and corresponding flow table configuration using OpenFlow. Bandwidth isolation is performed by applying QoS to the virtual links.
   b) After creating virtual links, the master OF configuration point is ready to **instantiate logical switches** for the virtual network. It does so by partitioning the physical resources and assigning the capabilities to each logical switch. This defines the view of resources and capabilities that are presented to the VNO through the OpenFlow protocol. Additionally, the view of resources and capabilities that is presented to the VNO through the OF-config protocol is specified here as well, defining for example which tunneling protocols can be used.
   c) Before handing over the newly created virtual network to a customer, topology and capabilities should be **validated**. In order to get the customer view, a special test-NOS is connected to the virtual SDN. With the help of the test-NOS, the network owner performs a capability discovery and a controller-based topology discovery. The obtained discovery results can be compared with the virtual network parameters to verify if they satisfy the customers requirements. Initial validation of the SLA is performed through OAM performance measurement tools.
   d) Optionally, the abstraction layer can be configured, e.g. defining the level of abstraction exposed (cf. Fig. 3). The abstraction system is external to the switches, running either as an separate entity, as a part of the Network Owner NOS (if it is provided as a service), or in the Service Provider NOS. How an abstraction system is configured is out of the scope for this paper.

5) **Virtual network operation:**
   Once the virtual network is created and validated by the network owner, the virtual network is handed over to the customer. The customer is given either a full view or an abstracted view as illustrated in Figure 3.
   The virtual network can have the pre-assigned customer-defined addresses and credentials that are configured by the network owner. Alternatively, the customer may choose to bootstrap its network in a similar fashion as described in step 1a for the physical network.
   From the customer's point of view the connected virtual network seems like any physical SDN network enabling the virtual service provider to perform control and management operations using its own NOS. Typical service provider operations include forwarding configuration, tunnel creation, OAM and QoS configuration, etc.

IV. Derived extensions

In order to perform the procedures listed above, several extensions to the ONF model and protocols are required. These include bootstrapping considerations, a revised SDN architecture, and extensions to the ONF protocols.

A. Control network bootstrapping

The current ONF OpenFlow specification does not describe how initial address assignment and control channel setup is performed. In this paper we do not propose a particular method, instead we point out that automatic bootstrapping requires mechanisms for control network setup, address assignment, authentication, and transfer of credentials. These mechanisms need to be supported by the OpenFlow switches and the NOS. Detailed discussions about bootstrapping procedures are planned for future work.

B. SDN architecture and NOS model

The current ONF controller model defines OF configuration points (OF-CP) and OF controllers (OF-Ctrl) as separate entities. As we can see from the procedures, there is a need for sharing data between these entities. In the list of procedures described in Section III in step 2a, physical capabilities are obtained by the OF-CP. However, the OF-Ctrl is responsible for obtaining and updating the physical topology through topology discovery (step 2c) and OAM data (step 3b). The combination of this data may be volatile and is required for instantiation and maintenance of the virtual networks. We argue that in order to quickly react upon this data, a system with tight coupling of the OF-CP and OF-Ctrl is needed. We indicate this in Fig. 5 with a shared data storage in the NOS.

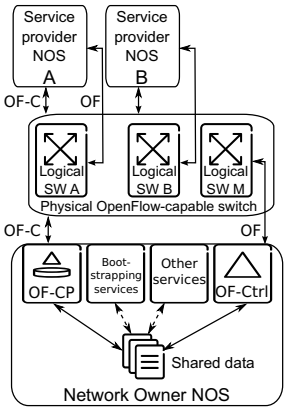

Fig. 5. Updated ONF model, including a network owner NOS, a master logical switch and limited per-customer OF-config interfaces.

Additionally, we see a need for being able to connect multiple OF-CPs with different views and capabilities to the OF-capable switches. In the ONF model, the network owner has full administrative capabilities, being able to instantiate logical switches, configure physical link level OAM, etc. However, a customer may also need an OF-config interface to configure its own tunnel endpoints, OAM tools, etc. Therefore, we propose to provide the customers with a reduced view of the switch and reduced configuration rights via OF-config interface. Through this reduced OF-config interface, the customer would see, for example, only one logical switch, which would seem to be connected directly to the physical ports.

C. Protocol requirements

Based on the configuration procedures we identified a number of missing functions in the current ONF protocols. These are primarily configuration related functions that naturally fit into the OF-Config protocol. Additionally, we suggest functions that deal with rapidly changing information to be better suited in the OpenFlow protocol.

In **OF-Config** we see four areas that need extensions:

(i) Physical resource discovery currently can only use the *OpenFlow Resource* structure in the OF-config model [4], which consists of details about physical ports, queues, flow tables, and certificates. The missing information here includes the supported OAM tools, CPU and memory resources, and upper limits of these resources. For OAM tools we need to discover the type of OAM tools supported and the maximum number of OAM tool instances that can be active. Additionally, if we are going to run multiple OpenFlow protocol instances on the switch (with different priorities) we want to assign hardware resources to them. Many upper limit values of physical resources are missing. While it is possible to configure OpenFlow resources, there is no information about the maximum number of the resources reported. This information is, however, crucial in order to instantiate logical switches and calculate the virtual network topologies.
We propose updates to the existing OpenFlow Resource structure and addition of two new structures in the model (see Figure 6):
OpenFlow Resource: the existing OpenFlow Resource structure (section 7.6 in [4]) extended with group tables and meters.
Physical Switch Resource: memory and CPU resources; list of available OAM tools; list of supported tunnel types.
Physical Switch Capabilities: Maximum number of logical switches on the OF capable switch; Upper limits of physical resources per logical switch.

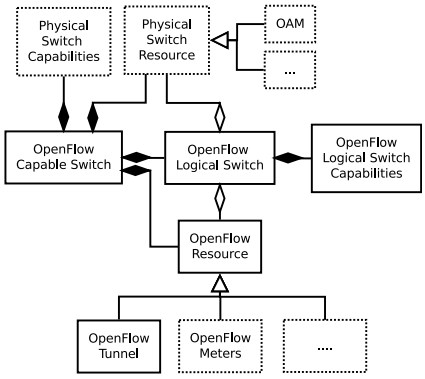

Fig. 6. Updated UML diagram of the OF-config model with new objects in dashed lines. Existing depicted objects are modified. Object with three dots indicate further sub-elements not depicted here, but listed in this subsection.

(ii) Logical link configuration is supported by OF-Config via the *OpenFlow Tunnel* structure, however, only for a small number of IP based tunneling protocols and with limited QoS capabilities. A logical link is represented as a logical port, which is actually a tunnel endpoint. The logical port can be set with a specific bit-rate but is missing other QoS configuration such as priority or physical port queue.
OpenFlow Tunnel: existing OpenFlow Tunnel structure (section 7.7 in [4]), extended with additional tunnel-types; QoS configuration parameters (priority, queue).

(iii) Logical switch instantiation is currently considered out of scope for OF-Config, however, using the existing (currently read-only) parameters in *OpenFlow Logical Switch*

Capabilities, logical switch instantiation should be possible. In addition to the existing parameters we propose to add more hardware resources (see (i)), masks for the supported OAM tools, and supported tunneling protocols. Additionally, upper limits need to be specified for logical switch resources shared between multiple logical switches.
OpenFlow Logical Switch: OpenFlow Resource structure (section 7.6 in [4]), extended with CPU and memory reservation; list of OAM tools; list of tunnel types; meters, group tables.
Logical Switch capabilities: Logical Switch Capabilities structure (section 7.4 in [4]), extended with upper limits for the number of flow and group table entries, queues, tunnels, OAM tools, and meters.

(iv) Device and link OAM configuration is currently not supported. The OAM functions we consider are monitoring of various parameters, such as CPU load and memory usage on the device, path connectivity verification, loss rate, delay, and bandwidth measurements on links. For OAM tools there are many tool-specific configuration parameters such as timing intervals and identifiers for BFD sessions, one or two-way measurements in case of delay measurements, etc.
However, OAM tools share a number of parameters as well, e.g., how often measurements are performed (on-demand or periodically), how often measurements are reported (on-demand or periodically), and thresholds specifying conditions which when reached should trigger sending of alarms or performing autonomous action(s) by the switch.
OAM:
OAM parameters: OAM instance identifier, OAM tool type, type of measurement;
Managed entity: type of resource, resource identifier;
Scheduling of measurements and reporting: monitoring period; reporting method;
Thresholds: threshold type; threshold value; list of actions;
Switch local actions: action type; action-specific parameters;
Optional parameters: OAM tool specific parameters.

Besides OF-Config, our procedures also require an extension of the **OpenFlow protocol**. While OAM tools are configured by OF-Config, the reporting of results and alarms are better suited for the OpenFlow protocol due to its real-time characteristics. OpenFlow is currently lacking a dedicated monitoring message, therefore we propose to add a new message type for both the asynchronous alarm messages and the periodic measurement results. The exact specification of these message structures will be highly dependent on the specific alarm and OAM tool.

V. Summary and Conclusions

The ONF currently specifies two protocols of an SDN environment, both interfacing network elements with the Network Operating System (NOS), which represents the centralized control and management plane. These protocols are OpenFlow, used to program flow-tables in flow and packet time scales; and the recent OF-config, with the main task of network wide configuration of the network elements in human time scales. In this paper, we apply the ONF's SDN model, resulting out of these two specifications, on the use-case of a carrier network which is virtualized and shared by multiple customers (VNOs). We outline the steps required to configure and manage this type of virtualized SDN, ranging from device and network bootstrapping to the setup of the virtual networks and final handover to the VNO. A per-step analysis of this procedure enabled us to identify shortcomings and propose necessary extensions to the ONF SDN model in terms of network management functionality. The highlighted extensions include control network bootstrapping considerations, updates to the SDN and NOS model, and most importantly extensions to the OF-config management data model. We believe that this type of use-case specific analysis is important in order to verify the applicability and usefulness of the ONF SDN model and its protocols in terms of network management functionality.

Acknowledgment

This work was funded by the European Commission under the FP7 ICT research project SPARC. The authors would like to thank all SPARC partners for discussions and comments.

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, pp. 69–74, 2008.
[2] R. Enns, "NETCONF Configuration Protocol," RFC 4741 (Proposed Standard), Internet Engineering Task Force, Dec. 2006, obsoleted by RFC 6241. [Online]. Available: http://www.ietf.org/rfc/rfc4741.txt
[3] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt
[4] Open Networking Foundation (ONF), "Openflow configuration and management protocol of-config 1.1," 2012. [Online]. Available: http://www.opennetworking.org/images/stories/downloads/of-config/of-config-1.1.pdf
[5] The SPARC consortium, "SPARC Delivarable D2.1: Initial definition of use cases and carrier requirements," 2011. [Online]. Available: http://www.fp7-sparc.eu/projects/deliverables
[6] Metro Ethernet Forum. (2012, April) Carrier ethernet defined. [Online]. Available: http://metroethernetforum.org/page_loader.php?p_id=140
[7] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.
[8] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, 2009.
[9] P. Skoldstrom and K. Yedavalli, "Network virtualization and resource allocation in openflow-based wide area networks," in *Proceedings of SDN'12: Workshop on Software Defined Networks*. IEEE ICC, 2012.
[10] B. Sonkoly, A. Gulys, J. Czentye, K. Kurucz, G. Vaszkun, A. Kern, D. Jocha, and A. Takacs, "Integrated openflow virtualization framework with flexible data, control and management functions," in *Proceedings of IEEE INFOCOM Demo*, 2012.
[11] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Skoldstrom, "Scalable fault management for openflow," in *Proceedings of SDN'12: Workshop on Software Defined Networks*. IEEE ICC, 2012.
[12] A. Fewell. (2012, April) Google showcases openflow network. [Online]. Available: http://www.networkworld.com/community/blog/google-pwns-networking-part-1
[13] OpenFlow.org. (2010) Openflow 1.0 reference implemenatation. [Online]. Available: http://www.openflow.org/wp/downloads