# Proposal for a NETCONF Interface for Virtual Networks in Open vSwitch Environments

Roberio Gomes Patricio*, Bruno Lopes Alcantara Batista*, Joaquim Celestino Junior* and Ahmed Patel†
* State University of Ceará
Fortaleza, Brazil
Emails: {roberio, bruno, celestino}@larces.uece.br
† University Kebangsaan Malaysia
Selangor Darul Ehsan, Malaysia
Email: whinchat2010@gmail.com

***Abstract*—The Internet is predominantly viewed as widely successful for existing users and service providers. But it suffers from ossification in the underlying infrastructure to exploit and scale to network virtualization for content providers and third-party hosting of cloud services through overlay networking by creating virtual ecosystems that enable and leverage new business opportunities. This paper proposes a Network Configuration Protocol (NETCONF) interface, modeled in YANG language, a data modeling language for networks. It provides standardized, simple and easy to use interfaces that facilitate the process of automating the creation of virtual networks using virtual switches, tested with Open vSwitch (OVS).**

***Keywords*—NETCONF, YANG, VLAN, distributed system.**

## I. INTRODUCTION

The increase of Internet data traffic and demands for faster and efficient network to accommodate social networking, big data and a stream of new virtualized cloud computing applications together with many different types of business opportunities is accelerating the tide of the next generation of the ubiquitous Internet. It scours the necessity of new technologies and new protocols that can support the creation of new kinds of networks to facilitate not only Internet evolution but also end user applications supported by core overlay network infrastructures. This has created a void that has become known as the ossification of the Internet.

New architectures and topologies have been proposed to resolve the Internet ossification problem [1], among them, Virtual LAN (VLAN) [2], Virtual Private Network (VPN) [3], and Overlays network [4] together with a series of underlying management protocols to support them. All of these enable the building of virtual network, that use the underlying native network infrastructure, which already exists to transform it into a new type of overlay network with new topologies and new management protocols. This allows the building of new computing based habitats, a kind of micro ecosystems, that are completely isolated from the nitty gritty of the underlying network infrastructures and their idiosyncrasies. It enables and leverages new business opportunities and strongly supports Cloud-Based Virtual Networks (CVN) [3]. CVNs are on their way that will transform the way ICT works, the way we humans and machines works.

To realize the benefits of virtualization, we need an architecture for network virtualization that encompasses the key players and providers such as players insides providers (PIPs) and service providers (SPs), virtual network providers (VNPs) for assembling virtual resources from one or multiple PIPs into a virtual network, and virtual network operators (VNOs) and virtual network providers (VNPs) for assembling virtual resources from one or multiple PIPs into desired virtual network. On the technical side, we need standardized interfaces between the players to automate the setup of virtual networks, ie, a common control plane. Moreover, we need ways in which each player can check if it is being provided with the service it is paying for (eg in terms of quality of service (QoS) and quality of experience (QoE).

These related initiatives contribute towards creating and using of the principles of Software Defined Networking (SDN), which represents a consolidation of the previously cited virtualization networking models that today are the object of study both in academia and within the major telecommunication, networking and ICT companies. They are promising to tide the next big application level networking wave.

When discussing about SDN and the initiatives of industrial applicability and related research, it is difficult to find anyone who does not cite Open vSwitch (OVS), a totally software based virtual switch that is able to provide a wide range of network services, such as among them VLAN and VPN. With OVS it is possible to create new virtual networks using a set of commands and a Command Line Interface (CLI) [5].

Within the last five years, end system virtualization, eg via Xen or VMware, has revamped server business. Router vendors such as Cisco and Juniper offer router virtualization, and existing techniques such as MPLS (Multiprotocol Label Switching) [6], GMPLS (Generalized MPLS) [7] and VPNs (Virtual Private Networks) [8] offer some coarse grained link virtualization. Overlays such as peer-to-peer (P2P) networks over the Internet (e.g., BitTorrent) can also be seen as a virtual network, but they suffer from a lack of sufficient isolation. VPNs (e.g., realized via MPLS), can also be seen as virtual networks. Open vSwitch is a production quality, multilayer virtual switch that attempts automation but is stuck with traditional forms of network interfaces. However, they suffer from a lack of node programmability using the latest software engineering tools.

Virtual networks can be tailored to meet a specific set of service provider and customer requirements that satisfy

 ISBN: 978-1-61208-360-5 

specific user groups under either public or private configuration management. While OVS as a tool goes some way to manage the configuration of theses environments, it lacks mechanisms of more versatile automation because, in particular, of the cumbersome and restrictive CLI that is out of phase with today's advanced software development and deployment technologies.

This paper proposes a Network Configuration Protocol (NETCONF) [9] interface, modeled in YANG language [10], a data modeling language for networks. It provides standardized, simple and easy to use interfaces that facilitate the process of automating the creation of virtual networks using virtual switches, tested with Open vSwitch (OVS).

The remainder of this paper is organized as follows: section II presents an overview of network configuration management using NETCONF and YANG; section III describes the related works done by other authors in this area; section IV describes the requirements and the modeling process used in this paper; section V presents a set of tests and results to evaluate and validate the proposed data model and section VI concludes the work with some conclusions and providing new ideas about future works.

## II. THEORICAL REFERENCE

### A. NETCONF

The configuration management of a wide number of network elements and devices is still a major problem nowadays because of their complexities and vendor specific proprietary style of interactions. The mechanisms to retrieve and modify the configuration data are largely something specific of each device provider, and the configuration interfaces are difficulty to maintain and quite costly to achieve a high level of efficiency and reliability through automation especially when dealing with issues of maintenance and version control [11].

According to Case el al., [12] the NETCONF exceeds the deficiency of Simple Network Management Protocol (SNMP) and emerges as a promising approach to standardizing the mechanism of network management based in eXtensible Markup Language (XML). NETCONF provides a better configuration interface for network devices due to the effective use of technologies like XML and others. The philosophy behind NETCONF is the necessity of an interoperable programmable interface between the different network equipment vendors to manipulate the devices' configuration state of the entire network into a systematic whole [11].

### B. YANG

The NETCONF protocol describes a communication model between network devices that need to be configured and managed. However, the specification of this protocol does not describe how the manipulated information in the data layer must be represented. This issue is taken up and addressed by the YANG data modeling language, that emerged from the working group called Netmod Standard Working Group (Netmod WG) [10] [11].

The fact of using XML messages, many other options of data representation emerged to work with NETCONF protocol, like XML and RelaxNG [2]. Despite the great power of expression of these languages and their wide adoption by the community, the Netmod WG chose to define their own data modelling language, aiming to have total control of it to achieve total independence from proprietary vendors. This is to avoid having to cater for specific formats and meanings that require data mapping transformation to achieve interoperability.

Thus, the YANG language as a data modeling language permits describing network elements. It covers not only information about the data configuration parameters and options, but allows handling data that describes the current state of the device and providing important and relevant data pertaining to network management. This goes way beyond just configuration management. It also allows tunneling the data and information to other aspects of network management such as accounting, security, performance and fault, in ISO parlance for network management, FCAPS [13].

Conceptually, according to McCloghrie et al. [14], YANG can be compared with Structure of Managed Information (SMI) [14], the language used by SNMP protocol [12] to define and construct network.

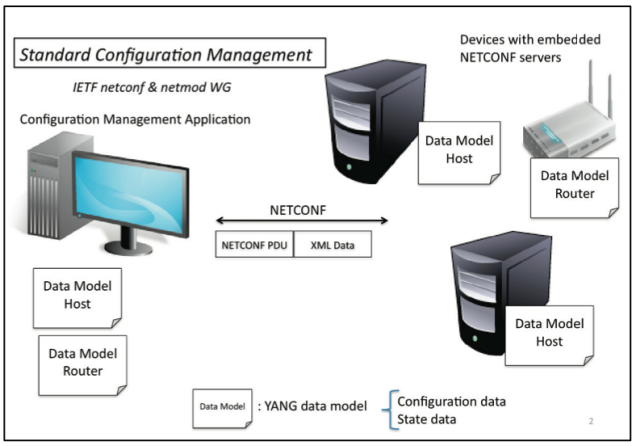


Fig. 1: The YANG and NETCONF integration

Management Information Base (MIB) that can be easily manipulated by the YANG data modeling language where such data are distributed and accessible only through NETCONF protocol. Figure 1 shows how to use the YANG language, its applicability and iteration with the NETCONF protocol.

## III. RELATED WORKS

The Open Networking Foundation [15] discusses about the OpenvSwitch implementation and compares its performance with the Linux Bridge, which is the de facto reference implementation in the open source world with this purpose.

The drawback is that they do not offer sufficient in depth scenarios and examples of how to accommodate NETCONF or other essential forms of network management. One positive aspect of OpenvSwitch is, it offers centralized management control in a distributed environment, creation of VPNs and virtualized mobility between IP subnets.

Pfaff and Davie [5] proposed an OpenvSwitch's database management protocol based in JSON-RPC calls. They presented and discussed about insufficient details in the database

 ISBN: 978-1-61208-360-5 

schema and the Application Programming Interface (API) of calls to configuration management of instances of OVS, with their names, parameters and return types.

However, with the adoption of API based in JSON-RPC for integrating the activities of distributed systems and managing the configuration tasks of multiples instances in the absence of more robust and secure mechanisms for allowing transactional configurations, it actually ends up limiting the scope and horizons of virtualization applicability.

Furthermore, the granularity of exported services via JSON-RPC exposes the database in a way that violates the encapsulation of data and increases the demand of coupling of any clients who want to consume such services. It directly affects the evolution of the model by preventing inclusions of new requirements in the information model or updating the MIB.

In addition, none of the current APIs are able to present OVS with sufficient granularity in defining and accessing of their services from the point of view of the basic setup operations necessary for their proper functioning.

## IV. The YANG Modeling Process

By using the OVS in real scenarios, we realized that we required an agile way of interaction with it. This was necessary to allow automated programmability of the configuration of OVS and its utilization in order to create virtual networks using VLANs as comprehensively as possible.

Initiating from a well known managed network environment concept, using the NETCONF protocol and data modeling language YANG, resulted in an innovative protocol that would offer a new communications interface in addition to the existing CLI and JSON-RPC. This also facilitated the OpenFlow specification [15], which states that the NETCONF protocol must be used for the most basic function of management and configuration in OpenFlow switches.

The following subsection shows the most important aspects of the proposed information model together with its major requirements. It also provides additional scenarios as visions of this model for a better understanding of its programmability and potential operational behavior. Beyond this, the data structures necessary to store the configuration data, the operations used via CLI for creating VLAN are also present in this model.

### A. Multiple switch instances

We envisage to have inside of the one OVS process several other instances of virtual switches. This opens the possibilities to provisioning a specific instance for each client, starting from the same instance. In other words, a given OVS process shall actuate as several virtualized switches simultaneously, where each switch can be responsible for a distinct VLAN operation. To make this possible, for this to be functional ,we propose the following additions to our policy:

*1) Strategy:* The information model must anticipate/contemplate an object of a bridge type, which is contained within a collection called bridges as shown in Figure 2.

*2) Vision:* Figure 2 also shows the structure of the information model supporting multiple instances.

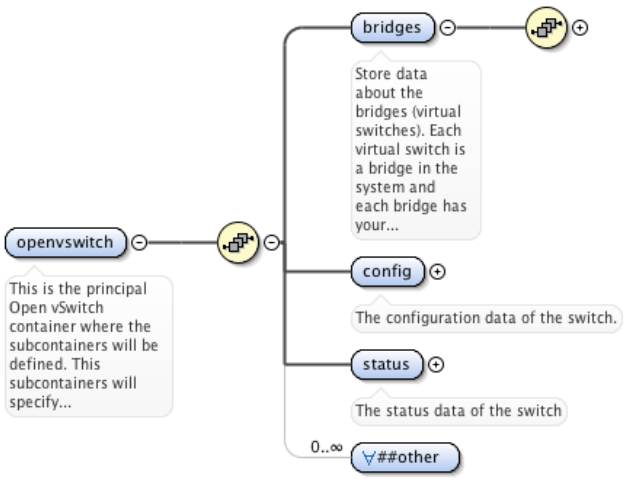


Fig. 2: The YANG and NETCONF integration

*3) Normative Considerations:* Following a document-based approach, the objects of the bridge type are grouped into a superior entity called bridges. This approach prevents such objects from being scattered inside the model. These may unnecessarily and considerably increase the NETCONF calls when the manager's function wants to obtain an overview of the whole, part, or only to access the instances of the OVS at once.

In this multiple switch instances case, using any other approach other than YANG would require unnecessary data normalization. Also using YANG over more traditional modeling approaches of MIBs in ASN1, which generally works with data structures that tie closely to standard Data Base Management Systems (DBMS), gives greater flexibility and independence of data types and DBMSs.

Tables would be costly in terms of operational efficiency and programmability. The YANG model is used as a container element to represent the bridges' objects and a list of elements for the bridges' objects. This approach is far more efficient than the traditional table approaches.

### B. Multiply Ports in the same Switch Instance

Any virtual switch to add and configure more ports will be possible, with its respective network interfaces and VLAN tags when applicable. In this proposed policy the following are essential.

*1) Strategy:* The switch ports are to be mapped onto a port object, which are contained in a collection called ports.

*2) Vision:* Figure 3 shows the hierarchical root structure of the information model of how multiples ports can be supported in the same switch instance.

*3) Normative Considerations:* Group the port objects into a superior entity called ports. So it is possible to obtain information of all ports of a switch instance with a minimum of NETCONF calls.

Furthermore, the process of batch configuration of ports on a switch occurs in a much more rapid fashion, once all

 ISBN: 978-1-61208-360-5 

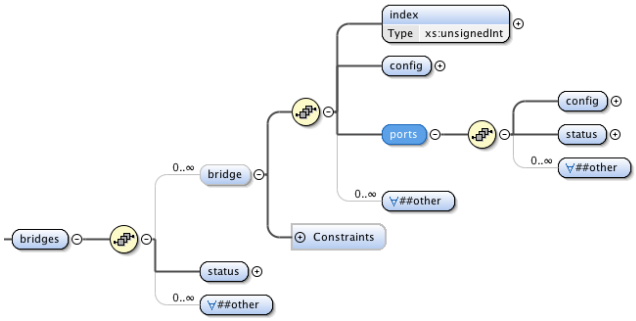


Fig. 3: The vision of multiples ports model

ports are under one entity. In terms of YANG, the container element represents the set of ports object and the list element represents the individual port object.

### C. Configuration Data Must Not Be Exposed to Any User

The configuration data of a given entity, whether it is an OVS instance or a VLAN, is to be inaccessible in the model. The NETCONF protocol proposes a clear separation between all the configuration data and the state data.

Furthermore, it will be possible to access and retrieve statistical data of an object as well as configuration. In this proposed policy:

*1) Strategy:* The configuration data and the state data of a given element must be separated in distinct branches, opening the possibility to restrict the data access through defined rules on the NETCONF server.

*2) Vision:* Figure 4 shows a data model element containing two objects: config and status. The element called config contains the related data about the configuration of the device, while the element status stores the state data of the parent object, which can be used for statistical purpose.

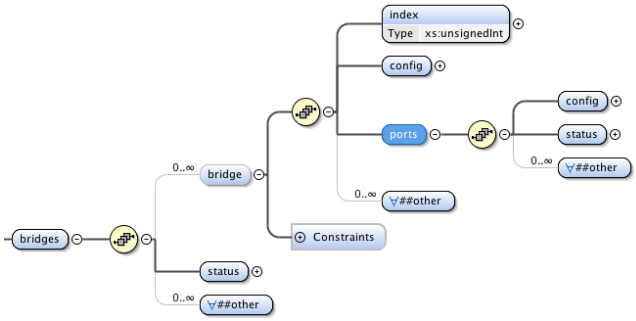


Fig. 4: The vision of config and status container model

*3) Normative Considerations:* The data separation is done through of two YANG containers. The definition of these containers is based on the kind of information that should be labeled with the YANG instructions, *config true* or *config false*.

Elements that have the YANG instruction *config true* must be stored in a config container and this data have read and write permissions. On the other hand, elements that have the YANG instruction *config false* must be stored in state container and this data have read-only permission. It is easy to associate access profiles to these elements in the Network Configuration Protocol (NETCONF) server and thus obtain more control over the access of these elements.

### D. VLAN support

To create VLANs and associate them to the ports of a given switch instance, they must be properly identified by a tag or an ID, which should be provided during the creation of a virtual network. The policy for this case is:

*1) Strategy:* Each port of a given switch must be associated to an object called interface and can still be linked to a VLAN. This construction puts under one port all needed data for total VLAN support.

*2) Vision:* Figure 5 below shows the configuration of (1) above where the VLAN elements are connected in a given port of a given switch instance.

With this type of structure it is easy to configure a VLAN to a given port, since the data needed for this are easily accessible from the same branch in the tree of objects.

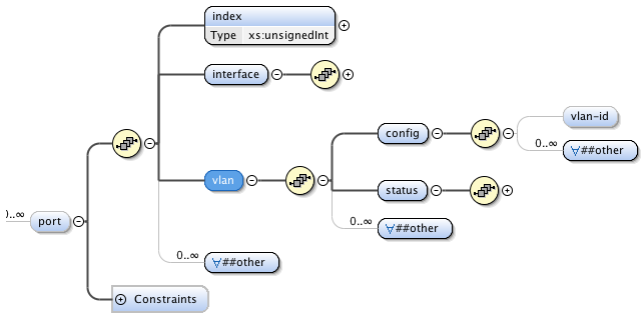


Fig. 5: The vision of VLAN support model

*3) Normative Considerations:* Here, two YANG containers which are nested and associated to a port container are used. The VLAN element is marked as optional, and may well be having the switch ports that are not being used in the construction of VLANs.

### E. Well Defined API

A well defined API with a set of RPC operations could be used to manipulate the information model without the NETCONF client having the need to interact directly with the information model.

The granularity of these operations should not be much different to other user interfaces, such as, CLI to which the user is already accustomed. It also allows for making it easy to learn and take advantage of the previously obtained experiences working with other interfaces.

*1) Strategy:* Since the main operations are already supported by CLI, they become easily exportable to the NETCONF RPC call function. This can be executed in the context of an application of the network management system (NMS) in the standard programmable manner.

 ISBN: 978-1-61208-360-5 

TABLE I: Current Supported Actions

| Operation | Description | Params |
|---|---|---|
| add-bridge | This RPC call adds a bridge (virtual switch) | bridge name |
| del-bridge | This RPC call deletes a virtual swicth | bridge name |
| add-port | This RPC call adds a port in the specified switch | bridge name, interface name |
| del-port | This RPC call deletes a port in the specified switch | bridge name, interface name |
| add-vlan-port | This RPC call adds a port in the specified switch with a vlan tag | bridge name, interface name, vlan tag |

Table I shows the currently supported operations by this model with their respective parameters.

## V. RESULTS

A YANG model, as described above, for the OVS is created to attend to the requirements already mentioned previously. We use the Yuma Server [16]. In our implementation of the NETCONF protocol, the Yuma Server is used to load the data model created by the YANG model by using the yangcli tool, that allows a NETCONF client to populate the data model in the associated MIB.

In the first test, we create two instances of OVS within the data model using the yangcli and as we can see in Listing 1. The model fulfills the requirements of multiple bridges. For reasons of simplicity, only the switches index are filled in the model, but in a real scenario all the other associated fields must be filled as well.

Listing 1: Multiple bridges on the NETCONF data model

```
rpc-reply{
  data{
    openvswitch{
      bridges{
        bridge 1 {
          index 1
        }
        bridge 2 {
          index 2
        }
      }
      status{
      }
    }
  }
}
```

The YANG model allows us to create several ports for a given switch instance, and according to the requirements, to have multiple ports in the same switch instance. Listing 2 shows a switch instance with two ports.

Listing 2: Multiple ports in the switch on the NETCONF data model

```
rpc-reply{
  data{
    openvswitch{
      bridges{
        bridge 1{
          index 1
          ports{
            port 1{
              index 1
              interface{
                config{
                  name eth0
                }
                status{
                  status up
                }
              }
            }
            port 2{
              index 2
              interface{
                config{
                  name eth1
                }
                status{
                  status up
                }
              }
            }
          }
        }
      }
    }
  }
}
```

In Listing 2, it can be observed that the configuration data and status data are separated into their respective containers, since the configuration data must be used exclusively by that process without exposure to any process.

In port 1 of switch 1 we have a container interface that has two sub containers, config and status, where the configuration data is within the config container (in this case the leaf name) and in the status container is the status data (in this case the leaf status).

Other requirement discussed in the previous section regarding VLAN support, follow the data modelling which allows determining the port of a given switch belonging to a VLAN. Listing 3 shows how a port is associated with a VLAN within the data model.

Listing 3: VLAN configuration in the switch port on the NETCONF data model

```
rpc-reply{
  data{
    openvswitch{
      bridges{
        bridge 1{
          index 1{
            ports{
              port 1{
                index 1
                interface{
                  config{
                    name eth0
                  }
                  status{
                    status up
```

```
                    }
                  }
                }
                vlan{
                  config{
                    valn-id 10
                  }
                  status{
                  }
                }
              }
            }
          }
        }
      }
    }
  }
```

The implemented model also has defined operations specified as requirements that make the appropriate API easily accessible in remote configuration using the NETCONF protocol. Besides implementing the data model in YANG, the API is written in C programming language that invokes the OVS operations.

Listing 4 shows an example of a virtual switch creation within the data model that is simple!.

Listing 4: Creating a switch and a port using API on the NETCONF data model

```
yangcli root@localhost> add-bridge bridge-name=sw01

RPC OK Reply 25 for session2:

yangcli root@localhost>
```

## VI. Conclusion and Future works

This work proposed a NETCONF interface for the configuration management of virtual switches that should be used to create virtual networks based on VLAN.

The proposed information model and the currently supported operations were modeled in the YANG data modelling language, utilizing OVS as a reference of virtual switches. The interface currently supported by OVS may not be the most appropriate when one has to automate the process of configuring this type of switch, but it is sufficient to allow further work to be performed to overcome this deficiency.

In this paper, we listed the basic requirements that a virtual switch must meet to support virtual networks for VLAN. Each one of those requirements was attended to in the information model based on YANG and the solution was presented in a simple visual way using a XML Schema Definition (XSD) models. In addition, other policy considerations devolving into strategies used in the modeling were also presented.

The proposed information model was not intended to exhaust all the possibilities to take all the necessary requirements to a virtual switch, but, however to illustrate the information requirements and data modeling together with the set of the operations to realize in building virtualized networks based in VLAN with virtual switches. It is a principle that has huge potentials in creating the next generation of virtual networks supporting a variety of virtual content based applications in virtualized cloud computing. Simply put, it is putting up the ante for the next generation of the Internet that offers huge technical and business possibilities.

Further improvements can be implemented in this model, certainly the support of other approaches used to build virtual networks (such as OpenFlow for instance) are already being studied and can be easily incorporated in the proposed modeling.

Thus, using a NMS it will be possible to complete the configuration management of programmable virtual switches through a robust interface standards based that is consolidated every day as a great alternative to traditional managed network interfaces.

## References


[1] G. N. Rouskas, “Tutorial on network virtiualization,” Presented at OFC/NFOEC, pp. 1393–1398, March 2012.

[2] N. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” Comput. Netw., vol. 54, no. 5, pp. 862–876, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.10.017

[3] T. Choi, K. Nodir, T.-H. Lee, D. Kim, and J. Lee, “Autonomic management framework for cloud-based virtual networks.” in APNOMS. IEEE, 2011, pp. 1–7. [Online]. Available: http://dblp.uni-trier.de/db/conf/apnoms/apnoms2011.html#ChoiNLKL11

[4] O. vSwitch, “Open vSwitch: a open virtual switch,” [accessed April 2014], 2013. [Online]. Available: http://openvswitch.org/

[5] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, “Extending networking into the virtualization layer,” in HotNets’09, 2009, pp. –1–1.

[6] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol Label Switching Architecture,” RFC 3031 (Proposed Standard), Internet Engineering Task Force, Jan. 2001, updated by RFCs 6178, 6790. [Online]. Available: http://www.ietf.org/rfc/rfc3031.txt

[7] E. Mannie, “Generalized Multi-Protocol Label Switching (GMPLS) Architecture,” RFC 3945 (Proposed Standard), Internet Engineering Task Force, Oct. 2004, updated by RFC 6002. [Online]. Available: http://www.ietf.org/rfc/rfc3945.txt

[8] L. Andersson and T. Madsen, “Provider Provisioned Virtual Private Network (VPN) Terminology,” RFC 4026 (Informational), Internet Engineering Task Force, Mar. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4026.txt

[9] R. Enns, “NETCONF Configuration Protocol,” RFC 4741 (Proposed Standard), Internet Engineering Task Force, December 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4741.txt

[10] Ietf, “RFC 6020: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt

[11] H. Xu and D. Xiao, “Data modeling for netconf-based network management: Xml schema or yang,” pp. 561–564, 2008.

[12] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, “Simple network management protocol (snmp),” United States, 1990.

[13] H. Zimmermann, “Osi reference model–the iso model of architecture for open systems interconnection,” pp. 425–432, 1980.

[14] K. McCloghrie, D. Perkins, and J. Schoenwaelder, “Structure of Management Information Version 2 (SMIv2),” RFC 2578 (Standard), Internet Engineering Task Force, April 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2578.txt

[15] O. N. Foundation, “OpenFlow ,” [accessed April 2014], 2013. [Online]. Available: https://www.opennetworking.org/index.php?option=com_content&view=category&layout=blog&id=57&Itemid=175&lang=en

[16] YumaWorks, “NETCONF ,” [accessed April 2014], Jan. 2014. [Online]. Available: https://www.yumaworks.com/netconfd-pro/netconf/


 ISBN: 978-1-61208-360-5 