SNMP MIB for OpenFlow-Capable Switch

A Thesis

Presented to

The Faculty of Engineering Technology

University of Houston

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

In

Engineering Technology

By

Dina Dalibalta

December 3, 2015

Acknowledgement

I am grateful and thankful to God, who always enriches me with endless blessings. Due to His blessings, I was able to successfully complete my thesis, had the pleasure to work with a knowledgeable and inspiring professor, Dr. Deniz Gurkan, and been lucky to have the support of my husband and the two little angels that we have together.

Abstract

Network Management Systems (NMSs) rely heavily on Simple Network Management Protocol (SNMP) for exchanging information between managed devices. In this thesis, we thought of introducing SNMP to Software Defined Network (SDN); the new promising paradigm in the world of networking.

While MIBs (Management Information Bases) are central to SNMP functionality, we focused on investigating and designing effective MIBs for SDN managed entities such as switches and routers. In the designing stages, we emphasized our analysis and investigation on the requirements and specifications of the standard configuration and management protocol (OF-CONFIG) of all SDN operational contexts and its companion OpenFlow protocol. The outcome of our design is a data model of OF-CONFIG MIB mapped to SNMP. This data model is considered as a specific framework of SDN switch MIBs. Therefore, this thesis briefly explains a road map of MIB development in an SDN environment. As a result, the information in this thesis can be used as a guide for further design extensions, testing and implementation of SDN effective MIBs.

Table of Contents

Table of Contents

Table of Figures

Chapter 1: Introduction

There is no doubt we are witnessing a new milestone in the history of networking with the rise of SDN paradigm. We can think of SDN as a revolution in the nature of networking where all the network building blocks are being defined as software entities. This new approach to building networks will provide numerous benefits to network operators, including reductions in both capital and operating costs. Moreover, the implementation of SDN seems to satisfy the Bring Your Own Device (BYOD) trend and its requirements of flexible and secure networks. Therefore, the continuous development of such a technology is valuable as long as it is manageable and controlled. With all the promising types of benefits that potentially arise from the application of SDN; this new environment is still lacking any aspect of management. One of the keys to dealing with this situation is to make use of SNMP MIBs, a legacy mean of data monitoring and information managing over the internet.

Toward this goal, this thesis presents the first design of an SNMP MIB model dedicated to the SDN-based switches. SNMP MIBs have been and are still serving network engineers to better manage and control their networks' various devices. In nowadays data networks, the functionality controlling the network is split into three planes: the data plane, the control plane and the management plane. The data plane needs to implement, in addition to next-hop forwarding, functions such as tunneling, accessing control, address translation, and queuing [3]. To implement these functions, multiple entities such as the switches/routers are governed. Therefore, tucked inside our designed MIB are states and instances to help tackle the multiple routing processes running on the same router/switch and modify them for management and control.

1.1 Background

1.1.1 SNMP

SNMP (Simple Network Management Protocol) is a protocol that manages devices on an IP network. These devices can vary from servers to routers to switches to printers and many more, and all of these devices support SNMP [52]. SNMP is highly used in most of the network management systems. SNMP can be used to monitor built-in and externally-attached network devices. Internet Engineering Task Force (IETF) defines SNMP as a subset of the Internet Protocol suite. SNMP consists of an application layer protocol, data objects, and database schema that address the data objects. SNMP carries the system configurations in the form of variables across the managed network. SNMP uses a management database schema, called the MIB (Management Information Base), to store these variables. SNMP enables network administrators with the capability to actively monitor, manage and configure the network devices via remote access of the MIBs.

A typical SNMP scenario consists of one or more administrative units called the SNMP manager that is responsible to monitor and manage the network devices. On each network device, resides a software component called the SNMP agent that communicates with the manager using designated SNMP messages (see figure 1). SNMP agents make use of the management data stored in the MIBs in their communication with the SNMP manager. In some cases, more than one SNMP agents are gathered to report to one master agent and the master agent in its turn, reports directly to the manager. SNMP provides the agents with an extra feature where they can send notifications instantly to the manager without the need of sending request messages and

waiting for response. These notifications are asynchronous notifications of the agent's states and they are called traps. For that purpose, a section of the SNMP manager is dedicated to receive this type of messages and is called the trap receiver.

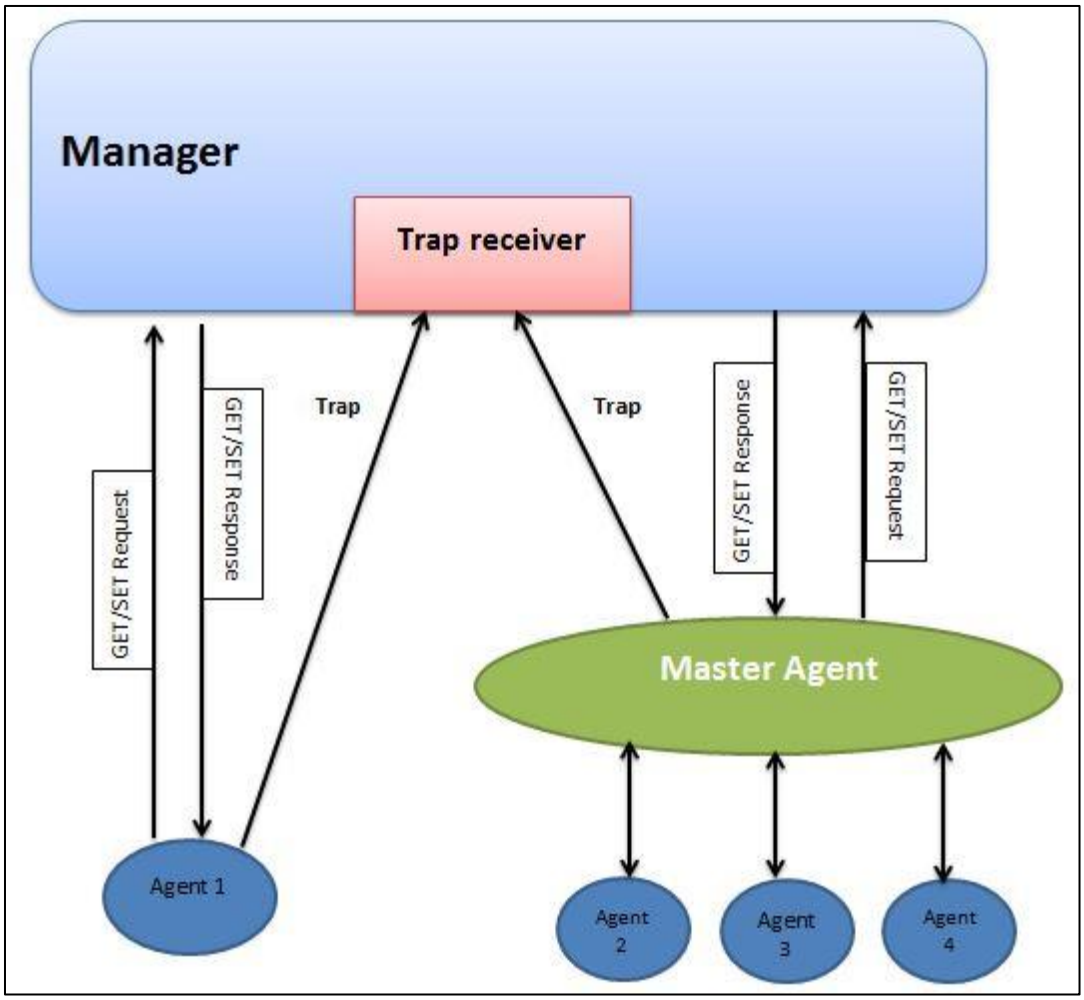

Figure 1 SNMP Communication Principle

1.1.1.1 SNMP Architecture

SNMP architecture is shaped by a manager-agent communication model. A typical SNMP-based management system consists of a manager, a managed element and a set of SNMP protocol messages. The managed element contains the SNMP agent with the managed objects. In their communication, the SNMP manager and agent refer to the MIB to exchange defined data. As mentioned earlier, the manager and agent communicate via specific messages provided by the SNMP protocol. Through this type of communication, the agent provides an interface for the manager to reach the managed objects. On other hand, provides an interface for the human network administrator to reach the management system (see figure 2).

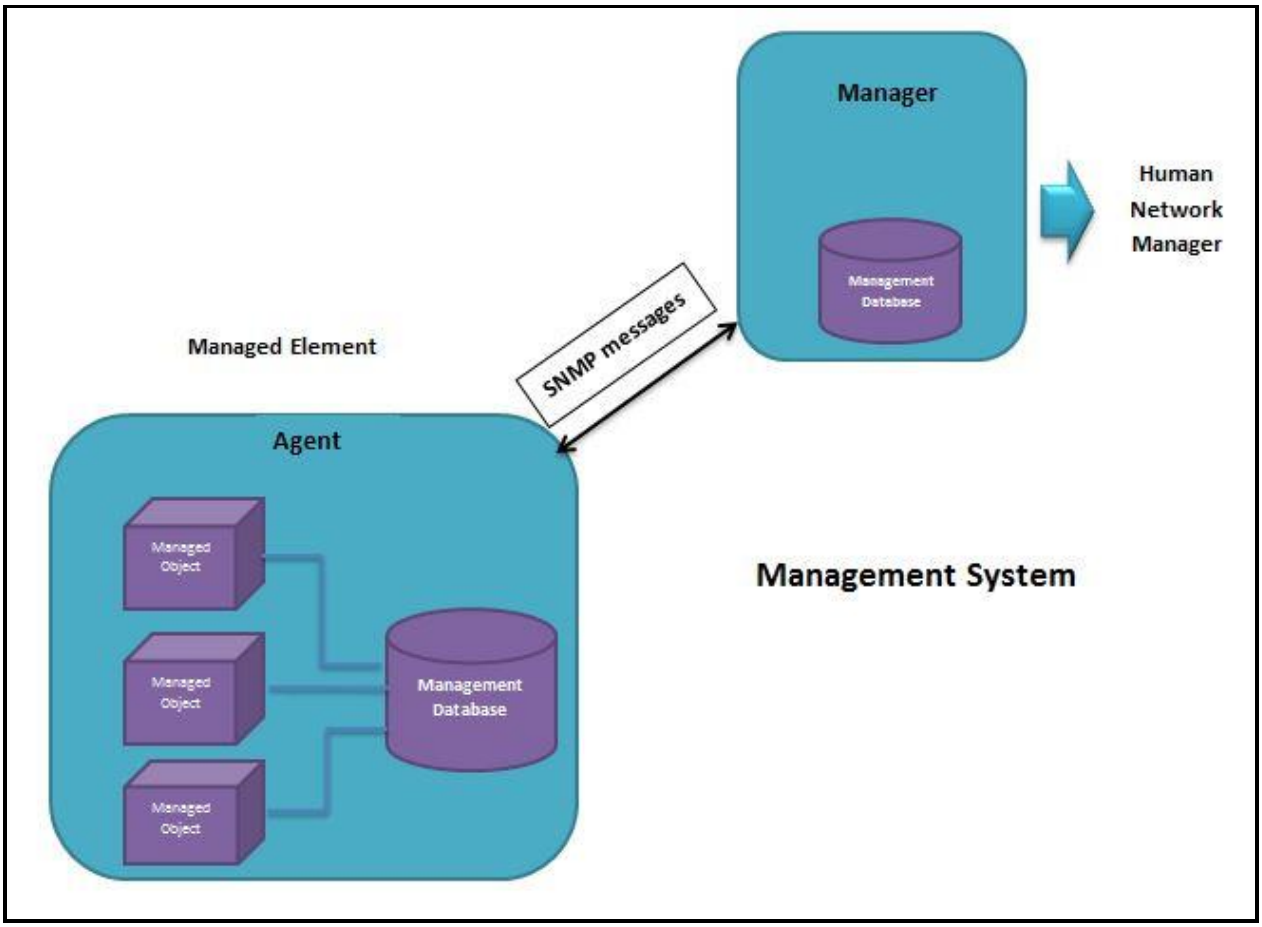

Figure 2 SNMP Management System

SNMP stands for "Simple Network Management Protocol"[64]. The protocol is "simple" (but not necessarily its implementation) as it contains only a few important operations. (See Table 1)

Table 1 SNMP Message Types

SNMP Message	Description
GET	Retrieve data from a network node
GETNEXT	Retrieve the next element from a network node
SET	Send configuration or control commands to a network node
TRAP	A network node can send a notification to the management station
INFORM	An acknowledged trap (network nodes can try and send it again if no acknowledgement is received)

1.1.2 SNMP-managed network

An SNMP network [66] generally consists of SNMP entities, each consisting of one or more SNMP agent and one or more SNMP managers (although an entity may comprise both an agent and manager) that communicate using SNMP messages. An SNMP manager (or NMS (network management station)) is responsible for managing one or more SNMP agents within the domain of the SNMP manager. An SNMP agent is included in each node (or host) of the network (e.g., computer, server, etc) that is managed by an SNMP manager. Each agent collects data about the respective managed node and provides the appropriate information to the designated SNMP manager. The agent sends the appropriate information (configuration, parameters...etc.) as responses to its manager's requests. Each SNMP agent maintains a local MIB where it stores all

the data collected about the managed node and its surroundings. In the case of maintaining a collection of managed objects within one managed element, the SNMP agent can store all the collected data in the form of MIB modules.

1.2 MIB

1.2.1 What is MIB?

MIB (Management Information Base) [30] is a database used for managing the entities in the communication network. A Management Information Base [29] is a map of the hierarchical order of all of the managed objects or MIB variables. Each system in a network (workstation, server, router, bridge, and so on) maintains a MIB that reflects the status of the managed resources on that system. A MIB consists of a collection of MIB variables that are managed by the SNMP manager. Each variable within a MIB is referenced with a unique OID. An OID refers to the location of a managed object within the MIB namespace. OIDs are arranged in a tree-like structure that begins with a root and expands downwards into branches. Each point in a MIB tree is known as a node.

1.2.2 How it works? MIB & SNMP

The MIB is a text file written in ASCII code describing SNMP managed objects [12]. The MIB organizes the defined data of the managed objects in a nested list. Every element in the MIB list is identified by a unique number. The main function of the MIB is to translate the identification numbers into human-readable text whenever asked by an SNMP manager. In other words, the SNMP manger uses the MIB as a dictionary or a codebook to identify the desired requested data.

In this way, a MIB allows the SNMP manager to retrieve and modify data that is maintained by an agent.

The identification number given to each defined object in the MIB list is referred to as OID (Object Identifier) (OID is defined later in section 3.1). The MIB, as well, provides a text-label for each OID. Each message sent by an SNMP device contains an OID or more of certain object in its string of numbers and/or text-label formats. The SNMP manager uses the MIB to process the messages. The SNMP manager compiles the MIB through compiling software called the compiler. The compiler converts the MIB from ASCII format into binary format to be used by the managing device. Without the MIB, the message is just a meaningless string of numbers. The SNMP manager imports the MIB through a software function called compiling. Compiling is the process of converting the MIB from its raw ASCII format into a binary format that the SNMP manager can use. Thus, it is very important for each component on the managed network to be defined in the MIB, otherwise, from an SNMP device perspective the component doesn't exist.

1.3 SDN

SDN (Software Defined Networking) is the future of networks where all the network's building blocks are defined as software abstractions [56]. The main concept in SDN is decoupling the control and data planes. The inventors and vendors of these systems claim that this simplifies networking.

In network architecture there are three planes:

Data plane: carries the network user traffic

Control plane: carries signaling traffic

Management plane: carries administrative traffic and is considered a subset of the control plane

SDN decouples the data and control planes, removes the control plane from network hardware and implements it in software instead [48], which enables programmatic access and as a result, makes network administration much more flexible. A network administrator can have a landscape overview of the network traffic without individually accessing each component on the network. The administrator can change any network switch's rules when necessary.--prioritizing, de-prioritizing or even blocking specific types of packets with a very granular level of control.

1.3.1 OpenFlow

SDN requires some method for the control plane to communicate with the data plane. One such mechanism is OpenFlow. OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. OpenFlow enables controllers to determine the path of network packets through the network of switches. OpenFlow allows switches from different suppliers — often each with their own proprietary interfaces and scripting languages — to be managed remotely using a single, open protocol. Its inventors consider OpenFlow an enabler of Software defined networking (SDN) [43].

Over time, numerous paths to SDN implementation will emerge, but one of the most explored strategies so far decouples the control plane of a physical network and places management in a centralized controller. That controller uses OpenFlow as a southbound protocol to direct specific flows between nodes on the network, allowing for granular network programmability.

While OpenFlow efficiently manages flows and determines how packets are forwarded between individual source and destination pairs, it does not provide the configuration and management functions necessary to allocate ports or assign IP addresses. That's where OpenFlow configuration protocols come in.

In a traditional network, vendors use proprietary configuration and management methods. Some depend on SNMP to configure products and monitor devices. Others use command lines to configure each device on the network.

But one of the potential benefits of SDN is to move away from programming individual switches on the network. SDN controllers instead give engineers a holistic view of every component on the network and then offer the ability to set policy and manage traffic across the complicated matrix of devices.

In that scenario, OpenFlow defines packet flow operation, but it doesn't specify switch configuration databases or a management protocol. OpenFlow configuration protocols, however, establish the relationship between controllers and switches, enabling a standard configuration and management method for switches. Using the same management configuration, network managers can select switches from any vendor, choose the best device for each network location and set the parameters for communication between controllers and switches [22].

1.3.2 OF-CONFIG

The OpenFlow Management and Configuration Protocol (OF-Config) is a special set of rules that defines a mechanism for OpenFlow controllers to access and modify configuration data on an OpenFlow switch. OF-Config provides network engineers with the configuration and management functions that are needed to allocate ports or assign IP addresses. OF-Config also

helps in giving network engineers an overall view of every area of the network and the ability to set policies and manage traffic across devices [35].

1.3.2.1 OF-Config basics

OpenFlow Management and Configuration Protocol version (OF-Config) was designed to apply to all OpenFlow implementations and on both physical and virtual switches [22].

The OF-Config protocol addresses the following components of controller-switch management:

1. OpenFlow configuration point: The OF-Config point issues OF-Config commands.
2. OpenFlow capable switch: A physical or virtual switching device contains a number of ports and queues.
3. OpenFlow logical switch: A logical switch within the OpenFlow capable switch allocates a subset of the ports and queues that make up an OpenFlow capable switch.

The OF-Config Point can be located in the same server or workstation as an OpenFlow controller or within traditional network management products. Either way, configuration points can manage multiple OpenFlow-capable virtual or physical switches. A configuration point may manage multiple OpenFlow capable switches, and a capable switch may be managed by more than one configuration point.

The configuration point also communicates with OpenFlow logical switches that live within the OpenFlow capable switch. Specifically the control point supplies each logical Switch with the IP addresses and port numbers of the OpenFlow controllers that will control individual packet flows through the switch. It also specifies whether TCP or TLS will be used to communicate between the switch and controller, and it configures certificates for communications between switches and controllers. Each OpenFlow logical switch operates independently of the other logical switches within the same OpenFlow capable switch.

A configuration point can discover the resources allocated to a logical switch, configure tunnels, set port parameters, turn ports off and on, and retrieve switch status. It receives error codes from a switch if a configuration operation fails and it can roll back the operation in the event of a partial failure.

1.3.3 OpenFlow Switch

An OpenFlow switch is a software program or hardware device that forwards packets in a software-defined networking (SDN) environment. OpenFlow switches are either based on the OpenFlow protocol, or compatible with it [44].

In a conventional switch, packet forwarding (the data plane) and high-level routing (the control plane) occur on the same device. In software-defined networking, the data plane is decoupled from the control plane. The data plane is still implemented in the switch itself, but the control plane is implemented in software and a separate SDN controller that makes high-level routing decisions. The switch and controller communicate by means of the OpenFlow protocol.

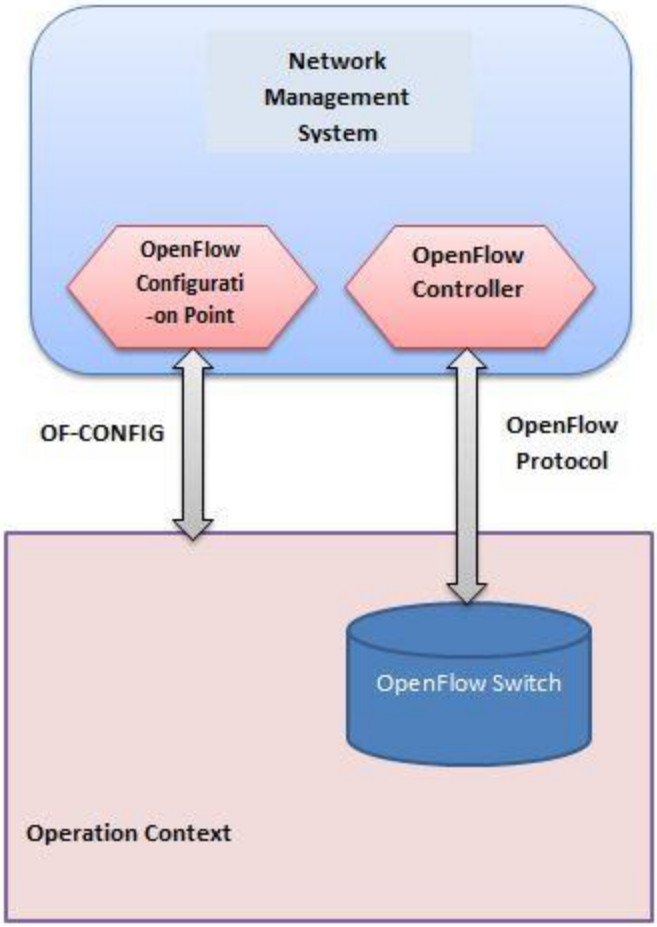

Figure 3 Controller and switch communications

1.3.3.1 Switch Components

An OpenFlow Logical Switch can contain a number of flow tables and group tables that are responsible for packets forwarding and lookups, as well as, OpenFlow channels for external communications with the controller. Through these channels, the controller can manage the switch using OpenFlow protocol. The controller is capable of modifying the data contained in the flow tables inside the switch. Inside each flow table there is a set of flow entries. Flow entries can be counters, matching fields or instructions for packet matching applications.

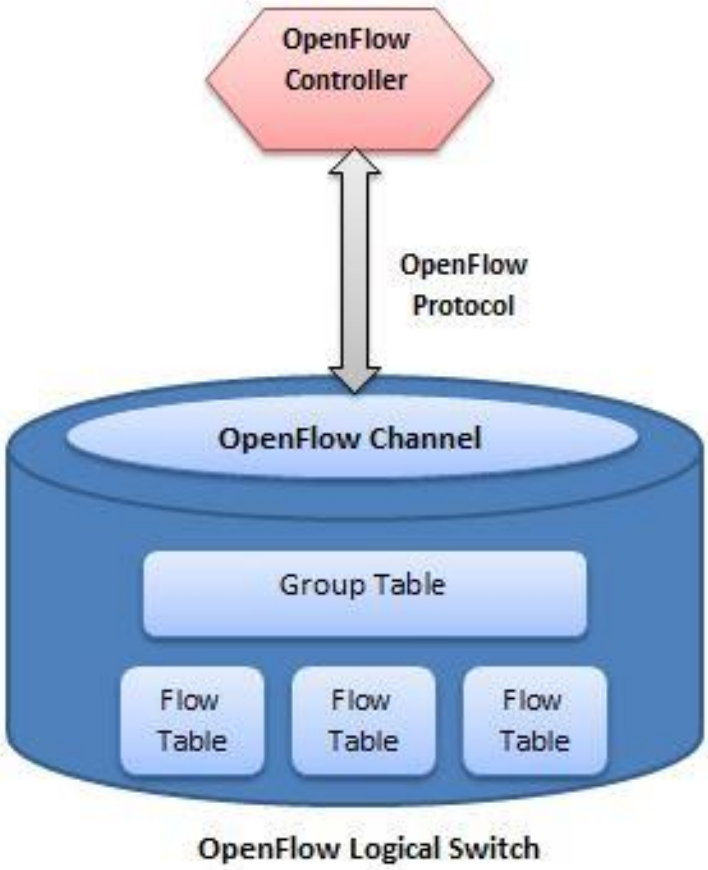

Figure 4 OpenFlow Logical Switch components

As a result, a virtual switch such as the OpenFlow switch can implement all of the sophisticated functionality in a straightforward manner which makes it far more flexible and integrate more tightly with host software than do physical switches [45].

Chapter 2 Literature Review

2.1 Motivation

As networks grow complex, managing them gets more complex. With the rise of the SDN paradigm and its concept of adapting open-source software entities and a variety of vendors-specific devices with their proprietary protocols, the mission to orchestrate such a complicated system can be nearly impossible. On the other hand, SDN made it more challenging for network researchers and designers to compete and produce an ultimate managing tool that would best serve in such an overgrowing networking environment. Under the influence of this competition, we decided to introduce the SNMP MIBs for OF-Capable switches as a promising tool capable of monitoring and controlling SDN-based entities.

2.2 Why SNMP

2.2.1 SIMPLE

In our perception of the networks' growing complexity we decided to resolve such a problem by using a "simple" tweak, SNMP-based management approach that breaks down any complex system into small detailed and manageable entities. The SNMP approach is generic and can be used to manage a variety of different systems [47]. Almost any real-time system consisting of a collection of independent communicating elements can use SNMP.

As stated by Alia Atlas [2], the second most popular interface for the interrogation of a device's state, statistics, and configuration is the Simple Network Management Protocol and a set of relevant standards-based and proprietary Management Information Base (MIB) modules. SNMP

has a strong history of being used by network managers to gather statistics and state information about devices, including their routing systems.

In his article [3 SDN technology for network management better be more than middleware], Patrick Hubbard stated that the ubiquity of the SNMP -- and its admittedly unsexy utility -- stem from it's being baked into pretty much any network device --*not* from it's being in a box attached to the device. And all vendors do it the same way, which eliminates any possibility that potential customers can't integrate it into their management systems. When it comes to SDN, the management vendors that are the most agnostic have the most to gain. They will be able to integrate SDN frameworks from different vendors with existing non-SDN management technologies into a single pane of glass.

In his post [4 Monitoring a Software Defined Network, Part 4 SDN offers an opportunity to redesign network monitoring][http://www.nojitter.com/post/240166288/monitoring-a-software-defined-network-part-4], Terry Slattery wrote; " If we're not careful, we'll create this new SDN thing and it will be great...until something breaks. Because there is a separate monitoring system that's not well integrated, we'll have organizations that are running their networks blind simply because they are unaccustomed to effectively using a network management system". He also wrote that SDN can make a large collection of switches look like one big switch/router. That's part of the advantage of SDN—it hides implementation details. But when one of the components is experiencing health problems, we need to know about it. He then stated that SNMP could be used to monitor device health data. And that there are some things that SNMP has done well. Being able to walk the MIB of a device is very handy, even without the original MIB definition file. He was sometimes able to determine useful information about a device by walking its MIB using SNMP.

2.2.2 Comprehensive MIBs

MIBs could be thought of as a folder containing several titled documents each describing the features of different network elements. MIBs are comprehensive databases in a way that they can be addressed to any network element. MIBs are also flexible so that anyone can create/write his own MIBs, register and implement them through MIB tree extensions on any hardware or software device.

2.2.3 Proprietary Extensions

Any vendor can design and create a MIB for any chosen product of its own. There are standard MIBs defined by the IETF that can be used as well by any vendor. Vendors can add new objects under the Standard MIBs through sub-tree extensions. These extensions are designated for proprietary use. The main tree branch for such extensions is usually named "enterprise". All extensions added under the enterprise has to be first registered by the IANA (Internet Assigned Numbers Authority). The IANA assigns a unique MIB registration number to each vendor applicant [46].

2.3 Bringing SNMP to SDN

2.3.1 Related Work

An early example of SNMP in the context of SDN is discussed in Peregrine.[68] Designers of the ITRI container computer devised an innovative data center network architecture called *Peregrine*, which employs only commodity Ethernet switches as dump packet forwarding engines, but removes most of the control plane functionalities in the traditional Ethernet architecture, such as spanning tree, source learning, flooding and broadcast-based ARP query, and centralizes the address look-up, routing and fast fail-over intelligence on dedicated servers.

While *Peregrine* depends on SNMP or the CLI interface to program Ethernet switches, OpenFlow proposes an open API for programming the switches' control plane for every OpenFlow-enabled switch from an Open-Flow controller. To support lights-out management, the ITRI container computer incorporates a comprehensive SNMP-based environmental monitoring and control subsystem to protect itself, including a fire-and-smoke detection system backed up by a clean-agent gas-fire suppression subsystem, a physical security alarm subsystem, and an early earthquake detection system that proactively shuts itself down in the event of an earthquake.

Traditional NMS (Network Management System) adopts SNMP as management protocol and has achieved great success with many advantages. SDN has many advantages as well, but it still lacks mature management tools. Therefore, a group of researchers at the Beijing University of Posts and Telecommunications decided to merge NMS with SDN and they came up with SDNMP [68]. They presented the design of SDNMP, which is an approach for managing SDN using traditional NMS. . By adding data acquisition function, data processing and storage function and view function in the controllers, SDNMP provides a unified SNMP interface to traditional NMSs. To verify their approach, they built and implemented a prototype in their own testbed. By deploying virtual networks and services, results showed that SDNMP works well in practice. Experiment results also showed that SDNMP enables the management of physical/virtual networks, flow table, and services using SNMP. This approach introduces a way to link NMS with SDN and it is considered the closest to ours, but it still far from directly embedding SNMP MIBs in SDN as our approach does.

In order to let traditional systems manage new network architecture, there are some studies that use SNMP to manage the new architecture. M. Madan and M.Mathur showed in their paper [28] International Journal on Cloud Computing: Services and Architecture (IJCCSA), 2014 that CNMM improves SNMP to manage the cloud network. A. Bianco with others defined a [6] management architecture and a manager-agent communication model to coordinate the information residing on the single elements of the multi-stage router to present a unified view to the external network management station issuing SNMP requests. The IRTF rfc7426, [54] it was mentioned that MIBs can be used to describe for the forwarding planes. The ITRI container computer also incorporates a SNMP-based monitoring subsystem in the context of SDN. But it is not presenting an NMS based on SNMP to manage SDN. Meanwhile, there have been some studies on SDN management. For example,[27] was introduced as a high-level policy language that is designed to solve the problem in the low level configuration where it effectively serves as the glue between high-level event-driven network policies and low-level network configuration. On the other hand, the main idea of Resonance [39] is that it allows network devices to operate on the granularity of the flows. This function is enabled by the emerging OpenFlow standard and has origins in the designs of earlier protocols and programmable switch architectures.

One of the recent implementations in the SDN management plane was conducted in a GENI experimental [40] that introduced a new network management tool (NETMAN) that allows the user to gain and manipulate the complete set of parameters of OpenFlow switches. NETMAN which has a core of HTML/JavaScript code was experimented on a LINC switch that uses the NETCONF protocol for configurations. XML-coded OpenFlow parameters were parsed and

displayed via shell script done with HTML/JavaScript/XML code. We saw that a lot of coding in different languages was involved to display some parameters of the OpenFlow switch and the authors mentioned that “sometimes people don’t have clarity that behind simplicity lays tons of work and hundreds lines of code in order to just have one automatically configured function or command”. We absolutely appreciate the hard work behind the NETMAN but the question is why we would create a complicated tool to help manage an already complicated system. Another approach was done by the same NETMAN experimenters consisted of mapping between OF-CONFIG (*OpenFlow Management and Configuration Protocol*) and OVSDB (Open vSwitch Database Management Protocol) protocols. Knowing that OVSDB and OFCONFIG have absolutely different data models, OFCONFIG was implemented over the OVSDB protocol to enable OpenFlow parameters to be programmed for an operational context. The results showed that almost all the possible functions of OVSDB can be implemented in the OFCONFIG proving that the two protocols are not significantly different even though they took different approaches in managing network resources. Upon this discovery, we can see that it is not obligatory nor it does make any advancement to have a proprietary or vendor- specific management tool or protocol to better manage SDN network devices. Moreover, SNMP is widely supported on almost all managed network devices, and the standards for MIBs are comprehensive, covering almost all data network devices

Da Paz Ferraz Santos, P.R. with Pereira Esteves, R. and Zambenedetti Granville, L. investigated the management interfaces based on SNMPv2c, SNMPv3, NETCONF, and RWS for managing physical routers supporting virtualization [11] . They compared the interfaces by evaluating the performance of each one in terms of response time, CPU time, memory consumption, and

network usage for three basic VR management operations: creation, retrieval, and removal. The comparison showed that the SNMPv2c presented the best results for most evaluated metrics. The second best one was the NETCONF interface, which achieved results quite close to the SNMPv2c interface. Results also showed that the SNMPv2c interface is the most suitable one for small NVEs (network virtualization environments) without strict security requirements and NETCONF is the best choice to compose a management interface to be deployed in more realistic scenarios, where security and scalability are major concerns. On the other hand, Garry baker stated in his blog [5] that SNMP has been around since the dawn of IP networking and still a lot of networks do not utilize it to its full capacity. Based on some quotes from the NETCONF working group, replacing SNMP is not even part of the charter for NETCONF. "This is *NETCONF* – configuration specific, not out to replace SNMP notifications, traps, etc." – Randy Bush and "The work in the SNMP community (if there is any) doesn't have much impact on the NETCONF WG, since NETCONF is not intended to be a replacement for SNMP." – Andy Bierman.

In the IRTF (Internet Research Task Force) rfc 7426 it was shown that [50] SNMP MIBs can be used to describe DAL (Device and resource Abstraction Layer) for the forwarding and operational planes. Similar to YANG, SNMP MIBs are able to describe DAL for the forwarding plane. SNMP, similar to NETCONF, is suited for the MPSI (Management-Plane Southbound Interface).

A white paper by Ashton, Metzler & Associates [60] discussed what IT organizations should look for as a key feature is that it should be possible to monitor the SDN controller using standard protocols and techniques. It should, for example, be possible to monitor the health of the controller and the virtual networks that the controller supports using SNMP. The IT

organization should determine if the SDN controller provides support for a wide range of standard MIBs and also provides private MIBs to enable the IT organization to monitor the virtual network elements.

2.3.1.1 OLD BUT GOLD

The legacy SNMP is still influencing recent researches and studies of SDN. OpenDaylight is an Open Source Software project under the Linux Foundation with the goal of furthering the adoption and innovation of Software Defined Networking (SDN) through the creation of a common industry supported platform. The OPEN DAYLIGHT TSC (Technical Steering Committee) recently proposed the SNMP4SDN plugin project [49] The SNMP project formally joins the OpenDaylight Lithium Simultaneous Release. The SNMP Project addresses the need for a southbound plugin that allows applications and controller services to interact with devices using SNMP.

SNMP4SDN scope:

- Enable Ethernet switch on the SDN paradigm by means of SNMP & CLI as the south bound protocol
- New mechanisms for topology discovery for Ethernet switch in SDN
- Realize flow configuration of Ethernet switches over the forwarding table, ACL, and VLAN table

For the SDN controller to support building an SDN using Ethernet switches, it needs to be able to configure flows on the Ethernet switches. In addition, in initial, it has to discover which switches are under its management and then can configure flows on them. Also, the connectivity topology among switches is necessary information for the controller and applications. In this plugin, flow configuration on Ethernet switch would be done via SNMP or CLI, switch

discovery would be achieved via SNMP trap sent from the switch, and topology discovery would be resolved by reading LLDP (The Link Layer Discovery Protocol) data on the switches.
The following figure depicts the described SNMP4SDN as a southbound plugin in OpenDaylight architecture:

In his article [41], Tom Nolle wrote that one could argue that the framework of OpenFlow today is a byproduct of some limited experiments, rather than a mechanism designed to support generalized network service creation. Many have pointed out that OpenFlow cannot accomplish basic device setup, which would mean that more traditional management standards, such as the SNMP protocol, has to be used to achieve this goal.

A couple of students from Lanzhou University, China [51] presented an OpenFlow-based dynamic load balancing server cluster architecture to solve the problems that traditional dedicated load balancer faced. In their experiment, they wrote an SNMP module and LoadBalancer module that are executed by the Floddlight Controller [16]. The experiment showed that their load balancing architecture consisting of an OpenFlow switch based NetFPGA with a Floodlight Controller has more flexibility and low-cost. This can replace traditional dedicated hardware load balancer.
Taeyoung Song [57] with others developed techniques for conventional network devices that are monitored by SNMP to deal with the increasing amount of management traffic on networks.

The paper on Adaptive flow monitoring in SDN architecture [4] presented a new method to lessen the burden of SDN switch by making the SDN controller poll the SDN switch adaptively,

instead of making the SDN switch wait for the event all the time. This method resembles the way of SNMP function working more efficiently when it goes to trap method instead of polling method.

A new invention by Senthil Gurusamy CS and Rajesh Kidambi [10] provides a single SNMP view of OpenFlow flow tables that are dynamically defined by user or controller. They invented an apparatus to create a single SNMP table for multiple Openflow tables, comprising: first index with flow table as in SNMP that has device/context; second index with OpenFlow table Ids; and third index with flow count, wherein such table acts as Index to view multiple flow tables by iterating each row in flow table in SNMP, to provide for a single view display, thus providing dynamic tables view using SNMP single table. While displaying single view, OpenFlow table acts as index and is used to view multiple flow tables by iterating each row in the flow table in SNMP. As the index is to OpenFlow table, this invention addresses dynamic tables view using SNMP single tables. This is useful for OpenFlow enabled switches and remote monitoring of the OpenFlow switches from an SNMP manager on all flows installed in the switch.

In a StackExchange blog, Fred Thomsen [62] stated that SNMP is still heavily used and relied on for the following reasons:

- The use of UDP as the underlying protocol makes SNMP very efficient. Since most monitoring/management is done within your own data center you don't need to be as concerned with packets getting lost over the public internet and TCPs acknowledgement and flow control are overkill. SNMPv2 addresses some of SNMP original inefficiencies, for example, adding support for BULK GET.

- SNMP is universal across networked devices. Almost all networking equipment implements a SNMP agent. Having MIBs ensures there is a global space where information can be added by different vendors in a controlled fashion and thus makes looking up information on what OIDs to query easier and mostly vendor agnostic.
- Finally, there hasn't been a good candidate to drop in as a replacement. SNMP may not be great, but it's good and good is good enough. Several network devices now have APIs to get the same and additional information, but as I stated in my second point, the ways of querying these APIs obviously vary across devices and no endpoints are standardized across devices.

Trying to keep backward compatibility with existing network management approaches, one initiative at IRTF SDNRG [26] proposes a management plane at the same level of the control plane. It classifies solutions in two categories: control logic (with control plane southbound interfaces) and management logic (with management plane southbound interfaces). In other words, the management plane can be seen as a control platform that accommodates traditional network management services and protocols, such as SNMP, BGP, PCEP, and NETCONF.

The ForCES (Forwarding and Control Element Seperation) helps in standardizing of information being exchanged between the forwarding plane and the control plane in a ForCES NE (ForCES Network Element). It defines a framework for protocols responsible for the whole information exchange mechanism. ForCES defines Logical Function Blocks (LFBs) using XML encoding just as SNMP uses ASN.1 language to define the MIBs. Also, same as in SNMP-based networks, every entity belonging to the Forwarding Element (FE) being configured by the Control Element (CE), must be defined in the LFB. The LFB's entities are identified by component ids, each consisting of a 32-bit numerical string. The identified entities are then stored in a scheme to be

addressed by protocol constructs. When used to address specific data, the component IDs are organized in a hierarchal scheme that is called a path. A path is defined by the model [17] as a dotted numerical string of 32-bit component IDs (e.g, 1.0.9.4).

Chapter 3: The Model

3.1 OID

The managed objects are stored in the MIB as variables and are organized in a hierarchal structure called the MIB tree, where the variables for the leaves of the tree. Each branch and leaf of the MIB tree is given a unique address. This address consists of a string of numbers and is called an OID (Object Identifier). The OID helps in telling the exact location of any object on the MIB tree. The MIB tree structure always begins with a root that expands into multiple branches. The root is a node with no address because it's only a starting point. The OID numbering starts from the first node after the root and expand downwards (see figure 5). Figure 5 shows the standard first level of the MIB tree as defined by the IETF. The standard in SNMP MIBs is to start from node (iso) with OID (1), org node with OID (3), dod node with OID (6) then the internet node with OID (1). We have several branches under the internet subtree; one of them is the "private" branch where any vendor can put his own MIB. Another one is the "experimental" branch where MIBs are still undergoing development by the IETF. Usually defined objects in proprietary MIBs of certain organization are located under the "private" branch, "enterprises" subtree. MIBs can be registered under the enterprises subtree through the IANA (Internet Assigned Numbers Authority).The IANA assigns an enterprise number dedicated to the applied party where the rest of the MIB OID numbering is determined locally within the organization.

3.2.1 What is the function of an OID?

All SNMP messages sent by an SNMP device carry a series of OIDs identifying the desired data objects [12]. OIDs are a dotted sequence of numbers. These numbers are non-negative real integers separated by dots; each integer defining a new node and each dot defining a new branch. Moreover, the OID sequence can have the exact name of the desired object instead of the numerical value. However, OID sequences can also carry both numerical and textual values separated by dots as well. OIDs can have the following syntax formats [46]:

```
 {<name>.,<name>.<name>….}
Or
{<number>.<number>.<number>…..}
Or
{…<number>.<number>.<name>…..}
Where
<name> refers to an object name and
<number> refers to the object value
For example:
{internet}     can be defined as {iso(1)org(3)dod(6)1}
{private}      can be defined as {internet 4}
{enterprises}  can be defined as 1.2.6.1.4
                 or
               iso.org.dod.internet.private.1
                 or
               1.3.6.1.4.enterprises
                 or
               Simply enterprises can be defined as private 1
```

To define an OID, the choice of any of these syntax depends on the location of the object and where the OID is used. There are no restrictions on the names used with OID as long as they are unique for each object defined by the tree.

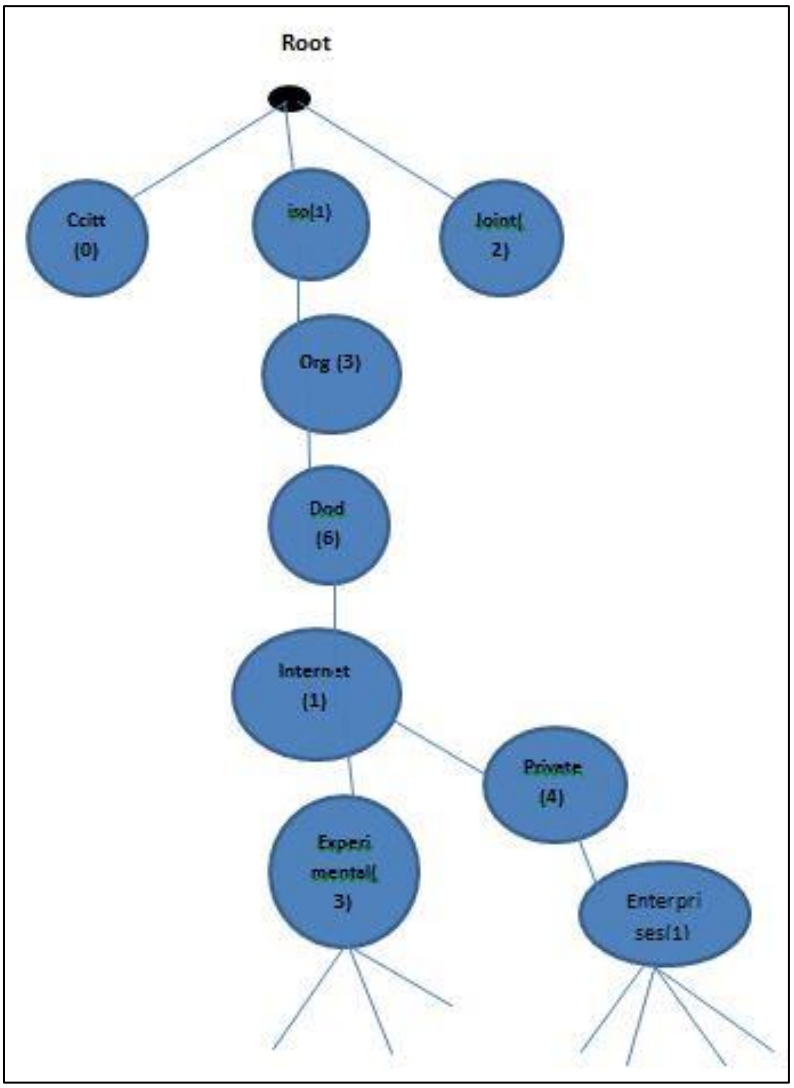

Figure 5 OID Tree

3.3 ASN.1

The MIBs are written in ASN.1 language. ASN.1 stands for Abstract Syntax Notation 1. The ISO (International Organization for Standardization) maintains ASN.1 as a standard notation [12]. ASN.1 is human-readable and extensible so that it can be used in any system where communication between different devices is required. It can describe any type of data transmitted over a network, regardless of the communication protocols and implementation language used. However, ASN.1 is not a programming language; it is only used to describe the structural aspects of the data. One of the important characteristics of ASN.1 is that any term defined within this notation can be capable of defining another separate term. This criterion is essential in hierarchal structures as used in MIBs where constructed types of objects are defined. ASN.1 uses basic data types such as:

INTEGER, BIT STRING, SEQUENCE…..

Below is an example of how ASN.1 can define a component as a sequence of sub-components:

```
ofFlowTable  DEINITIONS ::= BEGIN
ofFlowTable OBJECT-TYPE
        SYNTAX SEQUENCE OF ofFlowTableEntry

ofFlowTableEntry OBJECT-TYPE
        SYNTAX ofFlowTableEntry

       INDEX     {ofFlowTableIndex}
        ::= {ofFlowTableEntry 1}

        ::= {of 1.1.7}
OFFlowTableEntry ::=SEQUENCE {
```

```
            ofFlowTableIndex     INTEGER
}
ofNextFlowTables OBJECT-TYPE
        SYNTAX INTEGER
        ::= { ofFlowTableEntry 2}
ofFlowTableMatch OBJECT-TYPE
        SYNTAX INTEGER

        ::= { ofFlowTableEntry 3}
END
```

3.4 SMI
(The Structure of Management Information)

SMI is considered a subset of ASN.1 used by the MIBs as a framework to define managed objects [46]. The SMI outlines the layout of the MIB tree structure. SMI specifies the names, data type allowed and syntax of manageable objects within the MIBs. SMI can be thought of as a schema for a database system. When designing MIBs for any selected device, it is essential to begin with building a model of abstractions describing the main components of the device. These components are later to be defined as managed objects by the MIB tree. The SMI helps in defining the managed objects model and the different types of actions that are used to access these objects .

3.5 Modules

Modules are the main building blocks of the MIBs as defined by the ASN.1 syntax. Modules' mechanism is used to organize managed objects. Modules define objects by unique names and group them according to their functions and features. Therefore, Modules apply some restrictions on object-naming and value assigning mechanisms. These restrictions prevent MIBs from having duplicate names/values. For example, it is not allowed to define objects with different OID values, but carrying the same name, thus, all object names must be unique within a module namespace. However, the MIB objects must have a common prefix associated with their names within the whole defined module [46].

3.5.1 MIB Modules

The MIB term is sometimes used to describe management data contained in one or more modules. The modules in their turn are found in one or more documents [46]. Some may use the whole collection of management documents as one entity and name it "the MIB".

Proprietary MIBs are used for existing MIBs extensions and specifications of new objects that are not covered in IETF-standard MIBs defined areas.

It is mentioned earlier that names must be unique across the whole MIB modules; this is strictly the case for MIBs that are developed to be IETF standards. In the case of experimental and proprietary MIBs, name restrictions are not strictly enforced. MIB module names are formed by letters and digits, all upper case, separated by hyphens. There is no limitation for the number of letters and digits used, but there some restrictions on the use of hyphens. It is not allowed to use consecutive hyphens nor is it allowed to start or end a module name by a hyphen. Below is the format MIB modules' syntax:

```
<mib>=<module>…

<module>=

     <modName> DEFINITIONS ::= BEGIN

     ["IMPORTS" <importlist>;…]

   END
```

<importlist> =[<impotitem> , <importitem>]…

FROM <importmodName>

Where

<modName> is the name of the MIB module;

<importitem> is an item defined in another MIB module;

<importmodName> is the name of another previously defined MIB module;

The IMPORTS clause (defined later in section 3.5.1.2) allows MIB designers to use items previously defined in other modules and import them in the newly defined modules. Therefore, any item cannot be imported unless it is previously defined in Textual conventions. The imports also help in using the OBJECT-TYPE macro and choosing its definitions and syntax as defined in the Concise MIB RFC [33] and in the SMI RFC [34].

3.5.1.1 SNMP MIB Specifications

SNMP MIB specifications consist of three sections [47] (see figure 6) the first contains prose descriptions, the next consists of one or more MIB modules, and the last section contains references to other documents. Usually MIB specifications developed by IETF working groups are published as RFCs. The prose descriptions in a MIB specification are needed to map between the managed device or software and the definitions of the management information contained in the MIB module. The references section completes a MIB specification by providing a list of all the sources used or which can provide additional information to understand the item being managed.

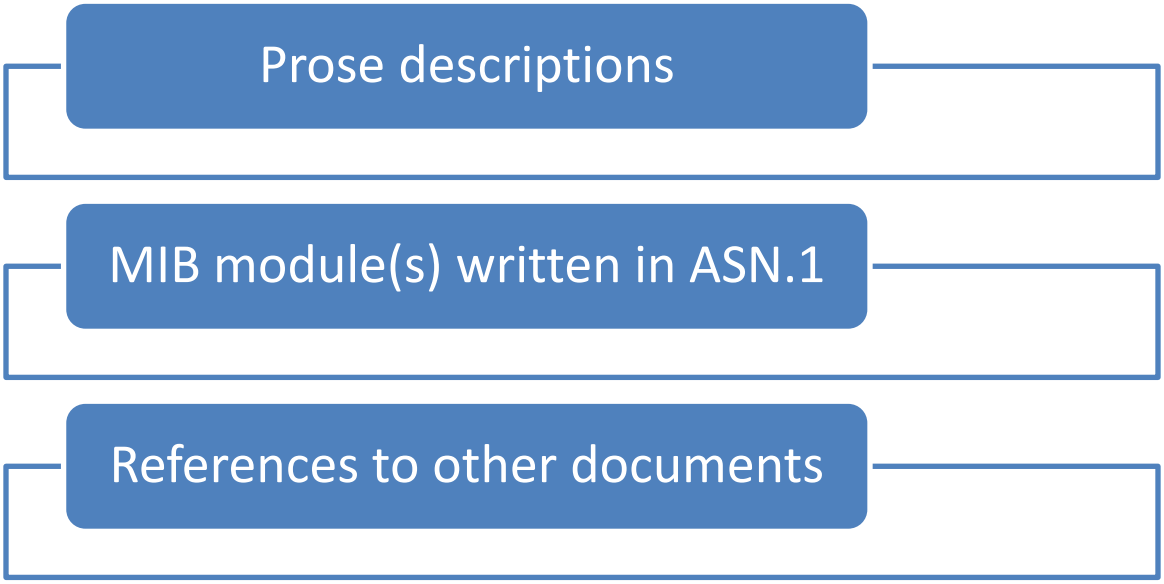

Figure 6 Contents of a MIB specification

3.5.1.2 SNMP MIB Modules

SNMP MIB modules are made to be human-readable and program-readable at the same time [47]. The main programs used by SNMP MIB are called MIB compilers. A MIB compiler is software used to perform parsing.

Items within a MIB module are defined by a language that consists of constructs from ASN.1 and constructs created for SNMP by means of ASN.1 macros [47]. ASN.1 macros are a mechanism to extend what can be specified in ASN.1 modules. To a user of the ASN.1 language, the macros appear to augment the language with additional constructs). The purpose of the macros is to define SNMP management information, SNMP events, describe SNMP MIB modules and specify implementation characteristics.

The macro that we basically used in designing our MIBs is the OBJECT-TYPE.

Example:

```
--the name of the MIB module is OF-CAPABLE-SWITCH-MIB
 OF-CAPABLE-SWITCH-MIB  DEFINITIONS : :=  BEGIN
-- The imports section specifies items defined in
-- other modules and referenced in this module.
-- The imports section must precede the definitions.
  IMPORTS
        OBJECT-TYPE
            FROM RFC1155-SMI
        ofResource
            FROM OF-CAPABLE-SWITCH-ROOT-MIB;

-- MIB module for enterprise OFCapableSwitch
-- for management of Resource.
ofResource     OBJECT IDENTIFIER
         -- resource OF objects
         : : = {of 1.1}
--a definition of management information
```

```
ofResource OBJECT-TYPE
        SYNTAX OBJECT IDENTIFIER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Assigns ports and queues associated with ofCapableSwitch and OFlogicalSwitch."
        ::= {of 1.1}

ofPort OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS current
        DESCRIPTION
            "Represent a physical port or a logical port."
        ::= {of 1.1.1}
    --the rest of the module would go here
END
```

3.5.1.3 MIB Module Layout and Elements

MIB modules should have outer wrapper that names it and differs it from other MIB modules. The IMPORTS section comes inside the wrapper as a linkage section to specify items previously defined in other modules and imported in the current module. After the IMPORTS section, modules are required to define the identity of the MIB module, especially in the case of SNMPv2 MIB modules where MODULE-IDENTITY construct should be used. The remaining of the modules is where the definitions come. (See figure 7).

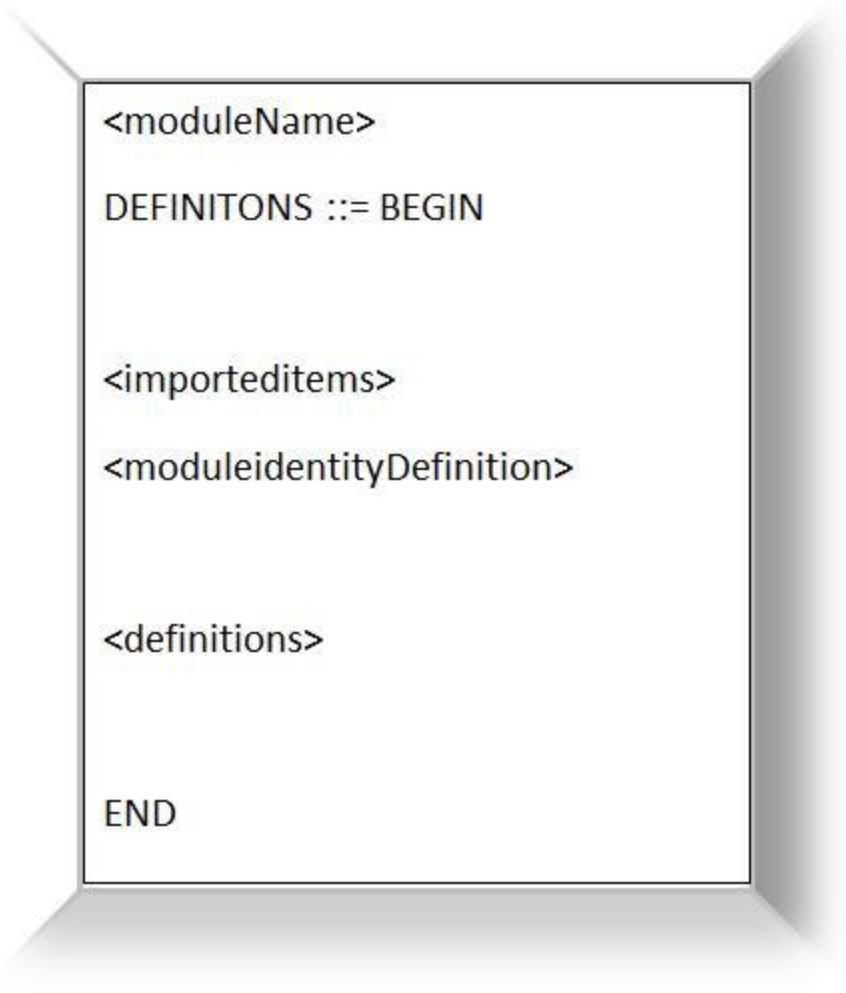

Figure 7 Layout of SNMP MIB Module

3.5.1.4 SNMP MIB Modules lexical Rules

SNMP MIB modules are contained in an ASCII file. The ASCII file contains printable characters and line terminators. A terminator is an end of file (EOF). Printable characters usually form a stream of tokens. Tokens consist of identifiers, punctuation, keywords, and white space. White space can have space characters, tab characters, comments, and line terminators. If not used to separate adjacent identifiers, literals and keywords, the white space is ignored.

3.5.1.5 Constructs in MIB Modules

The IMPORTS statement as defined before specifies items that are used in the current MIB module from other MIB modules. The definitions can be either ASN.1 "type assignments" for defining textual conventions and sequences, or ASN.1 "value assignments" for defining all other items in a MIB module. The constructs in MIB modules are formed of both the IMPORTS statement and the definitions.

There are different types of constructs that can be used to define items in SNMP MIB modules such as constructs defined in Table 2.

Table 2 Constructs used to Define Items in SNMP MIB Module

Construct	Description
MODULE-IDENTITY definition	Specifies information about an SNMP MIB module and registers the module with an OID value
OBJECT-TYPE definition	Defines management information and registers it with an OID value
OBJECT-IDENTITY definition	Defines a unique item and registers it with an OID value
OBJECT-GROUP definition	Groups management information and registers that grouping with an OID value
NOTIFICATION-TYPE definition	Defines an event and registers it with an OID value
TRAP-TYPE definition	Defines an event and identifies it with two numbers and an OID value
SEQUENCE definition	Specifies the columns in an SNMP table

3.5.1.5.1 OBJECT-TYPE Construct

In a MIB module, most of the definitions are for management information and use the OBJECT-TYPE construct to define several objects such as table, row and columnar. In our design of the OF-CAPABLE-SWITCH-MIB we used the OBJECT-TYPE construct. This construct is basically used to define an *object type.* An object type could be either a mechanism to organize related object types or a class of management information.

There are some clauses of the OBJECT-TYPE construct that can be used for definitions of table, row, and scalar objects. (See Table 3).

Table 3 Allowed clauses of the OBJECT-TYPE construct

Clause	Example
SYNTAX	SYNTAX Counter
UNITS	UNITS "inches"
ACCESS	ACCESS read-only
MAX-ACCESS	MAX-ACCESS read-only
STATUS	STATUS mandatory
DESCRIPTION	DESCRIPTION "physical port representation"
REFERENCE	REFERENCE "xxx co."
INDEX	INDEX {ifindex}
AUGMENTS	AUGMENTS {xxPiplineEntry}
DEFVAL	DEFVAL {2} *for columnar

We used the following clauses for the OF-CAPABLE-SWITCH-MIB:

3.5.1.5.1.1 SYNTAX Clause

The SYNTAX clause must present to define the structure of the data describing the respective object defined abstraction. We have to note that MIB objects with Object Syntax are exclusively supported by SNMP as defined in SMI [33]. However, even if SNMP operations are exclusively applied on scalar objects, MIB developers are able to construct virtual table indexed structures for adjunctive collection of objects within the same MIB. The SYNTAX clause that we used in our MIB design specified as INTEGER, OCTET STRING, TABLE ENTRY or OBJECT IDENTIFIER.

3.5.1.5.1.2 ACCESS Clause

The ACCESS clause [33] specifies the required support level for a certain object type. The ACCESS clause [47], which must be present, specifies the allowed access to the leaf object according to SMIv1. There was confusion within the SNMP community on the use of the clause, and even the allowed values. In SMIv1, the values are specified as “read-only”, “read-write”, and “write-only”. (See Table 4)

Table 4 Values for the ACCESS clause in SMIv1

Value	Description
not-accessible	Not allowed for scalar or columnar
read-only	The object type may be an operand in only retrieval and event report operations.

read-write	The object type may be an operand in modification, retrieval and event report operations.
write-only	The object type may be an operand in modification, retrieval and event report operations. However, the result is undefined. Note: this value should not be used, since it is obsolete.

3.5.1.5.1.3 MAX-ACCESS Clause

There was also confusion over whether the access specified for a leaf object was the minimal needed for conformance, or the maximum allowed [47]. The rules for SMIv2 clarify the intent of this clause by changing the name to MAX-ACCESS and specifying that the value specified is the one that makes "protocol sense" and not a value for conformance.

In SMIv2, the access values for the leaf object are "accessible-for-notify", "read-only", "read-write", and "read-create". (See Table 5)

Table 5 Values for the MAX-ACCESS clause in SMIv2

Value	Description
Not-accessible	The object type is a column in a table that is used as an index.
Accessible-for-notify	The object type is a special operand for event report operations.
read-only	The object type may be an operand in only retrieval and event report operations.

read-write	The object type may be an operand in modification, retrieval and event report operations.
read-create	The object type may be an operand in modification, retrieval and event report operations. And the object type may be an operand in modification request that creates a new instance of the object type.

3.5.1.5.1.4 STATUS Clause

The STATUS clause, which must be present, defines the implementation support required for that object type [33]. The value of the STATUS clause for a columnar object must be consistent with the value of the STATUS clause of the containing table [47]. That is, if the status of the containing table is "obsolete", then all the columnar objects must have a status of obsolete. If the status for the containing table is "deprecated", then the columnar objects of the table may have a status of "deprecated" or "obsolete". (See Table 6).

Table 6 Consistent STATUS values for Columnar Objects

Value for containing Table/Row	**Allowed values for Index Columnar Objects**	**Allowed values for Non-Index Columnar Objects**
mandatory in SMIv1	mandatory in SMIv1	mandatory, deprecated, or obsolete in SMIv1
current in SMIv2	current in SMIv2	current, deprecated, or obsolete in SMIv2
deprecated	deprecated	deprecated or obsolete

obsolete	obsolete	obsolete

3.5.1.5.1.5 DESCRIPTION Clause

The DESCRIPTION clause doesn't have to be present. This clause is comprehensive in which it is used to briefly describe a certain identified object. Using the DESCRIPTION clause, a developer can write all necessary definitions associated with the managed object in a textual format. It is critical to though for the value of this clause to be enclosed in double quotation marks as required by the ASN.1 syntax [33].

Chapter 4: The Design

4.1 MIB Development Road Map

4.1.1 Problem statement

We have set a task aiming to create an information framework for OpenFlow-Capable Switches. An OpenFlow switch main function is to forward packets throughout the network. Moreover, the OpenFlow switch must support three types of OpenFlow ports: physical ports, logical ports and reserved ports. Besides the ports, the OpenFlow switch contains logical entities that take care of packets lookup such as flow tables. For its communication with the controller, the OpenFlow switch contains OpenFlow channels as well. With this sophisticated structure and features, our goal is to make it possible to create OpenFlow switch management software that exploits its key features.

4.1.2 Framework Requirements

We provided a list of what we want to do and what we are trying to accomplish with this framework:

- Determine what components are present in the system: controller, logical switch etc.
- Examine the attributes of each component.
- Control each component: enabling/ disabling ports.
- Build Forwarding tables and decide forwarding actions.
- Monitor multiple OpenFlow data paths.
- Configure resources: queues and ports.

- Configure certificates for secure communication between the OpenFlow Logical Switches and OpenFlow Controllers.

4.1.3 Analysis

This was the first and most important phase in the MIB where we outlined the manageable objects we were trying to model in the MIB. (See Table 7). In our case, we made use of the OF-Config 1.2 specifications defined by UML (Unified Modeling Language) diagrams and XML (Extensible Markup Language) encoding to guide us in the analysis process.

Table 7 Manageable Objects to be modeled

MANAGABLE OBJECTS	DESCRIPTION	PRIMARY PURPOSE
OpenFlow Capable Switch	Is a physical or virtual switching device	Contains & manages OpenFlow Resources
OpenFlow Configuration Point	Is a Logical service entity	Configures one or more OpenFlow Capable Switches Via OF-CONFIG protocol
OpenFlow Logical Switch	Is an abstraction of OpenFlow switch containing a set of OpenFlow resources (e.g data path, control channel ...)	Communicates with the controller via OpenFlow protocol
OpenFlow Controller	Is a software	Controls OpenFLow Logical switches via OpenFlow protocol
OpenFlow Resource	Is a set of resources (ports, queues, flow tables…)	provide matching, forwarding, and packet modification
OpenFlow Queue	Is a queuing resource of an OpenFlow Logical Switch	Schedule packets according to their priority on an output port to provide Quality-ofService (QoS)
OpenFlow Port	Is a forwarding interface of an OpenFlow Logical Switch	Where packets enter and exit the OpenFlow pipeline (set of linked flow tables)
NDM (Negotiable Datapath Model)	Is an abstract switch model	Describes specific switch forwarding behaviors controllable via the OpenFlow-Switch protocol

4.1.4 Object Analysis

The first step after outlining the manageable objects was to define the main components (See Figure 8) of these objects and breaking them down into the attributes, statistics, and states. To accomplish that, we filled out a simple sheet for each selected component and assigned four columns representing the various aspects of that component: cardinality, attributes, statistics, and state. And with respect to these categories we characterized each subcomponent identified within the main selected component. (See Table 8).

A.OF-Capable Switch

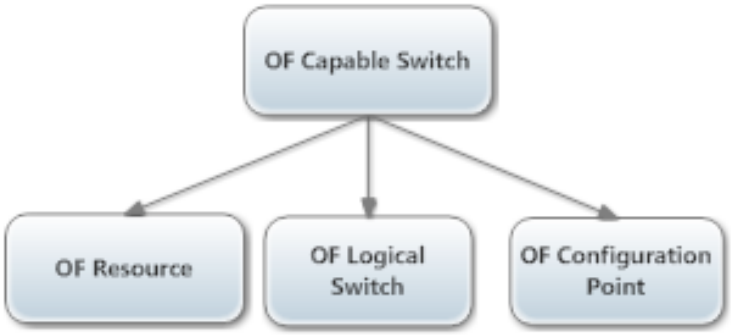

Figure 8 OpenFlow Capable Switch main components

Table 8 OF-Capable Switch Object analysis Worksheet

component	cardinality	attributes	statistics	state
OpenFlow Capable Switch	1			
Resource	1 to number of switches	Ports/queues/tables	Flow entries/ packets flow rate	

Logical switch	1 to number of switches	Flow tables/ OpenFlow channels	Flow entries/ packets flow rate	
Configuration point	1 to number of switches			

After completing the worksheet of the first component we followed some rules to guide us through the translation process to MIB syntax:

- Sub-components with cardinality greater than 1 should be part of a table.
- Attributes [47] can take on two forms: OCTET STRINGS can be used to represent items such as human readable text descriptions or binary data, and integer types are used to represent measurable quantities such as packets flow rate.
- Statistics representing high or the low packet rate or flow entries are integer type, INTEGER.
- Statistics representing increasing values are counter type, COUNTER
- States representing discrete stages of operation use an enumerated INTEGER. Each enumerated value is one of the states mentioned.
- States that have fluctuating values are gauge type, GAUGE.

A.1 Translating a Model into a MIB

Using the above worksheet, we were able to construct the first piece of the MIB declaration for the OpenFlow switch component. First, we could recognize that we had only 1 OpenFlow switch whose components all had cardinality between 1 and the number of given switches. Hence, there's no cardinality greater than 1 so we directly assigned the switch and its components

OBJECT IDENTIFIER SYNTAX. We then added the attributes, the statistical objects and the state objects respectively. So we could define the objects as follows:

ofCapableSwitch OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"OFConfig identification and version"

::= {of 1}

ofResource OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Assigns ports and queues associated with ofCapableSwitch and OFlogicalSwitch."

::= {of 1.1}

ofLogicalSwitch OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Identification of OFConfig, datapath and provides means for checking connection-behavior and controller-certificate."

::= {of 1.2}

ofConfigurationPoint OBJECT-TYPE

SYNTAX DISPLAY STRING

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"A textual description of the entity. This value should include OFConfigID and protocol."

::= {of 1.3}

You can note that we gave our MIB modules the prefix "of" as for representing OpenFlow.

We proceeded with the translation following the same rules identified first to cover all components of the OpenFlow switch.

B. OF-Resource

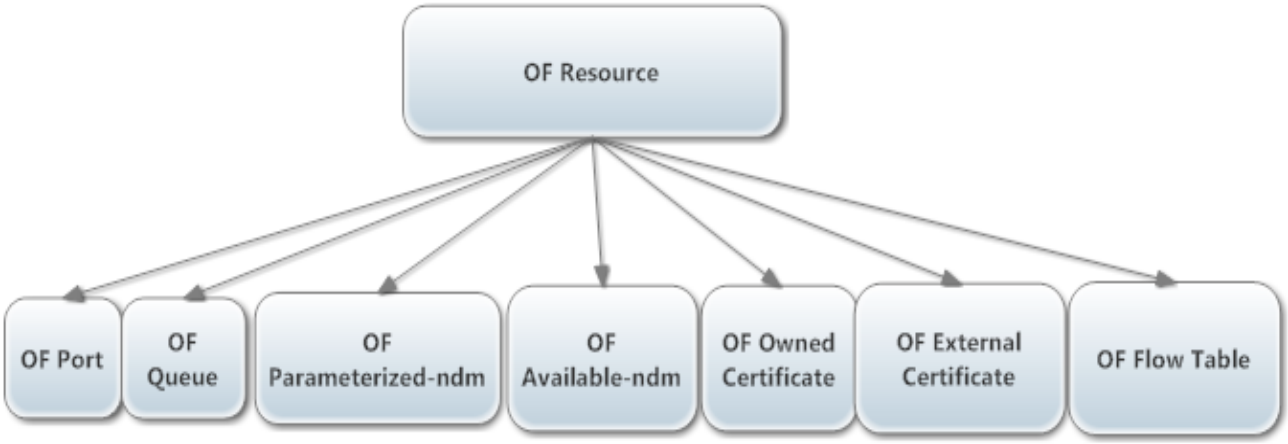

Figure 9 OpenFlow Resource main components

Table 9 OF-Resource Object Analysis Worksheet

component	cardinality	attributes	statistics	state
Resource	1			
Port	1 to number of resource		Rate of packets	Up/down
Queue	1 to number of resource	Flow tables/ OpenFlow channels	Flow entries/ packets flow rate	
Flow table	1 to number of resource	Queue id	Duration in Nano seconds	
Owned certificate	1			
External certificate	1 to number of resource			
Parameterized-ndm	1			
Available-ndm	1			

B.1 MIB translation of OF-Resource:

ofResource OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Assigns ports and queues associated with ofCapableSwitch and OFlogicalSwitch."

::= {of 1.1}

ofPort OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS current

DESCRIPTION

"Represent a physical port or a logical port."

::= {of 1.1.1}

ofQueue OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS current

DESCRIPTION

"Represents a queue as described in the OF protocol specification."

::= {of 1.1.2}

ofParameterized-ndm OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"A textual description of specific switch forwarding behaviors."

::= {of 1.1.3}

ofAvailable-ndm OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Includes name, type and version of the entity"

::= {of 1.1.4}

ofOwnedCertificate OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised."

::= {of 1.1.5}

ofExternalCertificate OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"It can be used by an OFLogicalSwitch for authenticating itself to a controller when a TLS connection is established."

::= {of 1.1.6}

ofFlowTable OBJECT-TYPE

SYNTAX SEQUENCE ofFlowTableEntry

ACCESS read-write

STATUS current

DESCRIPTION

"A logical context which represents a flow table."

::= {of 1.1.7}

C.OF-Logical Switch

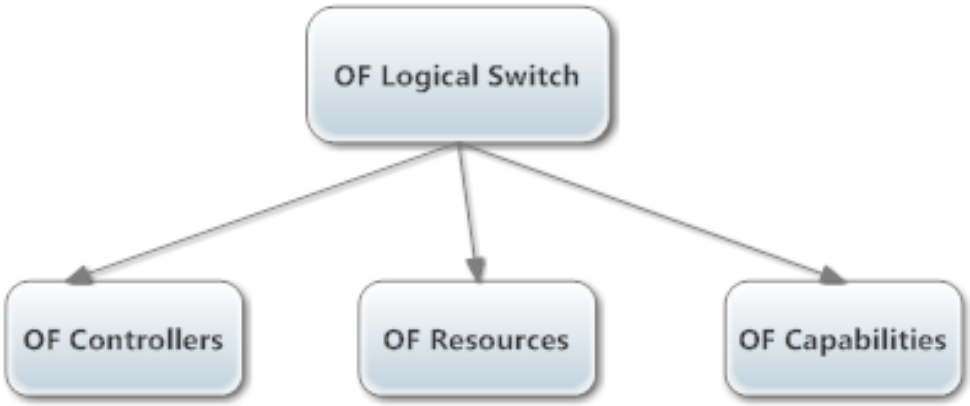

Figure 10 OF-Logical switch main components

Table 10 OF-Logical switch Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Logical switch	1			

controllers	1 to number of switch		Flow entries	
resources	1 to number of switch	Ports/queues/flow tables	Flow entries/ packets flow rate	
capabilities	1 to number of switch	Ports/queues/flow tables	Tables-sizes/ Flow entries/ packets flow rate	

C.1 MIB translation of OF-Logical switch:

ofLogicalSwitch OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"A set of resources from an ofCapableSwitch which can be associated with a specific ofController."

::= {of 1.2}

ofController OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"Contains attributes that indicate the role of the controller and parameters of the OF connection to the controller."

::= {of 1.2.1}

ofResources OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"Assign ports and queues associated with ofCapableSwitch and ofLogicalSwitch."

::= {of 1.1}

ofCapabilities OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"Identify tables-lengths, ports-statistics, queues and flow statistics."

::= {of 1.2.3}

D.OF-Controller

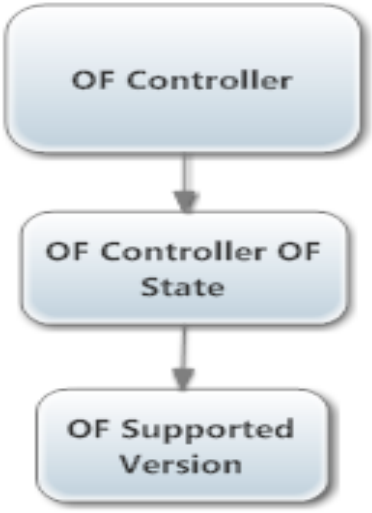

Figure 11 OF-Controller main components

Table 11 OF-Controller Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Controller	1			
Controller OpenFlowState	1 to number of controller		Connection state	Up/down
Supported version	1 to number of controller	Version index	Flow entries/ packets flow rate	updated

D.1 MIB translation of OF-Controller:

ofController OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"ofConfig ID, Role (master, slave, equal), IP-address, port number and protocol (TCP, TLS)."

::= {of 1.2.1}

ofControllerOFState OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the connection-state (up, down) and current version."

::= {of 1.2.1.1}

ofSupportedVersion OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the version(1.3, 1.2, 1.1, 1.0)."

::= {of 1.2.1.1.1}

E.OF-Capabilities

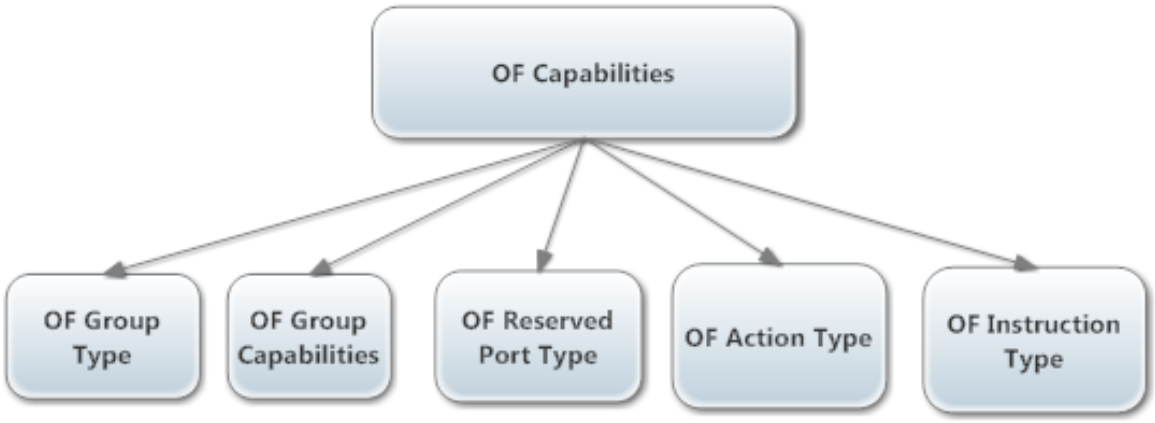

Figure 12 OF-Capabilities main components

Table 12 OF-Capabilities Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Capabilities	1			
Group type	1 to number of switch		Connection state	Up/down
Group capabilities	1 to number of group type	Version index	Flow entries/ packets flow rate	updated
Reserved port type	1 to number of switch		Packet flow rate	
Action type	1 to number of capabilities	Queues/IP-fragments	Packet flow rate	
Instruction type	1 to number of capabilities			

E.1 MIB translation of OF-Capabilities:

ofCapabilities OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS current

DESCRIPTION

"Values of max-buffered-packets, max-tables, max-ports, flow-statistics and block-looping ports."

::= {of 1.2.3}

ofReservedPortType OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates port-statistic."

::= {of 1.2.3.1}

ofGroupType OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates group-statistics."

::= {of 1.2.3.2}

ofGroupCapabilities OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates table-statistics."

::= {of 1.2.3.3}

ofActionType OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates queue statistics and IP-fragments."

::= {of 1.2.3.4}

ofInstructionType OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates flow-statistics."

::= {of 1.2.3.5}

F.OF-Flow Table

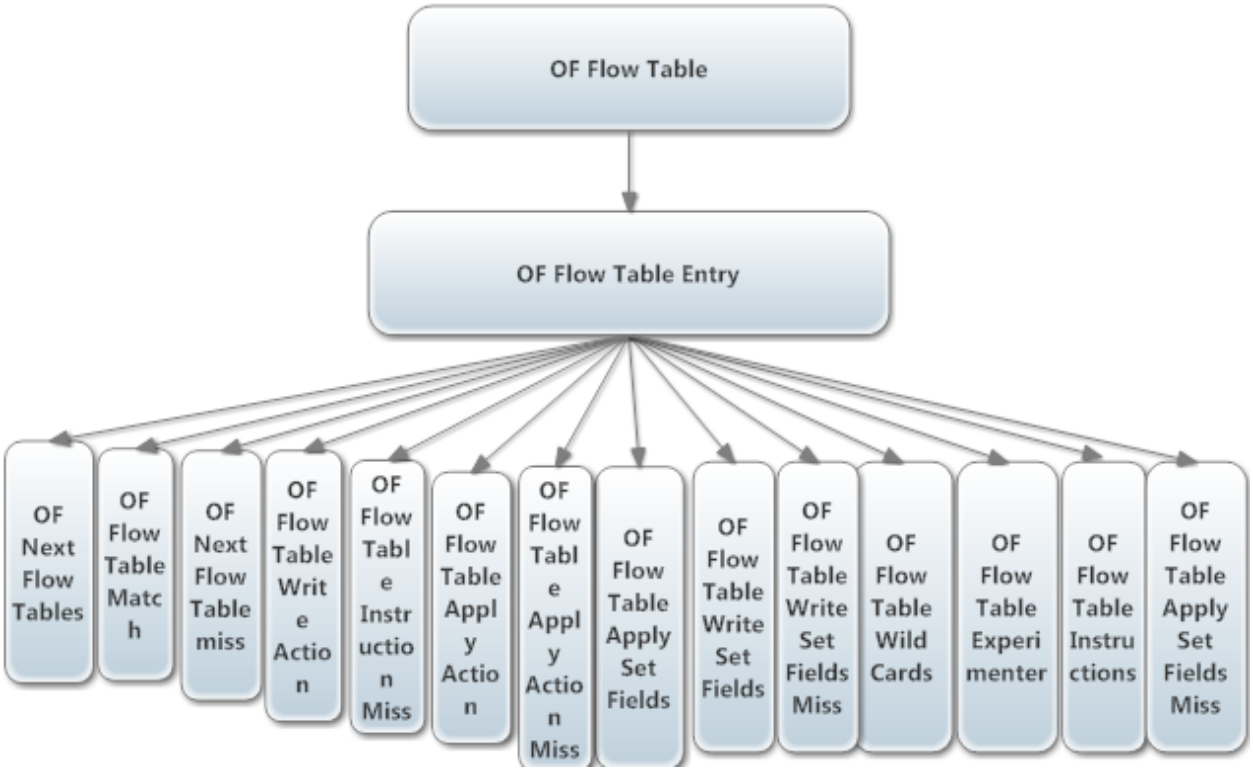

Figure 13 OF-Flow Table main components

Table 13 OF-Flow Table Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Flow table				
Flow table entry	1 to number of flow table entries	Match fields/counters	Matching packets	
Next flow table	1 to number of flow table entries	Match fields/counters	Matching packets	
Flow table match	1 to number of flow table entries	Match fields/counters	Matching packets	
Next flow table miss	1 to number of flow table entries	Match fields/counters	Matching packets	
write action	1 to number of flow table entries	Match fields/counters		
Apply action	1 to number of flow table entries	Match fields/counters		
Apply action miss	1 to number of flow table entries	Match fields/counters		
instructions	1 to number of flow table entries	Match fields/counters		
instruction miss	1 to number of flow table entries	Match fields/counters		
Apply set fields	1 to number of flow table entries	Match fields/counters	Field length	
Apply set field miss	1 to number of flow table entries	Match fields/counters	Field length	
Write set fields	1 to number of flow table entries	Match fields/counters	Field length	
Write set fields miss	1 to number of flow table entries	Match fields/counters	Field length	
Table wild cards	1 to number of flow table entries	Match fields/counters		
Table experimenter	1 to number of flow table entries	Match fields/counters		

The above worksheet showed that the cardinality of all components of the OpenFlow Flow Table is between 1 and the number of flow table entries. Since the cardinality of all these components

varied between 1 and the number of flow table entries we had to use a table indexed by flowtable number.

F.1 MIB translation of OF-FlowTable:

```
ofFlowTable OBJECT-TYPE
        SYNTAX SEQUENCE OF ofFlowTableEntry
        ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "A logical context which represents a flow table."
        ::= {of 1.1.7}
ofFlowTableEntry OBJECT-TYPE
        SYNTAX ofFlowTableEntry
        ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "A logical context which represents a flow table."
       INDEX    {ofFlowTableIndex}
        ::= {ofFlowTableEntry 1}
```

Since the table was indexed by the FlowTable number, it was the first column in our table. So, we first added the columnar objects to the SECUENCE construct for the table, and then we created the OBJECT-TYPE definitions for them.

```
OFFlowTableEntry ::=SEQUENCE {
            ofFlowTableIndex     INTEGER
}
```

Looking back at the worksheet, we saw that each flowtable entry contained one of each of the components within the flowTable. By adding the attributes of these items, we were able to define the objects as follows:

ofNextFlowTables OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates a counter for table id number given a maximum number of entries."

::= { ofFlowTableEntry 2}

ofFlowTableMatch OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the metadata-match."

::= { ofFlowTableEntry 3}

ofNetFlowTableMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates a counter for missed entries, it increments whenever an entry is missed."

::= { ofFlowTableEntry 4}

ofFlowTableWriteAction OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the metadata-write."

::= { ofFlowTableEntry 5}

ofFlowTableInstructionMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the missed instructions."

::= { ofFlowTableEntry 6}

ofFlowTableApplyAction OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates type of action."

::= { ofFlowTableEntry 7}

ofFlowTableApplyActionMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Type of action missed."

::= { ofFlowTableEntry 8}

ofFlowTableApplySetFields OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates Field length."

::= { ofFlowTableEntry 9}

ofFlowTableWriteSetFieldsMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates metadata-write field length missed."

::= { ofFlowTableEntry 10}

ofFlowTableWriteSetFields OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Metadata-write field length."

::= { ofFlowTableEntry 11}

ofFlowTableApplySetFieldsMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Field length missed."

::= { ofFlowTableEntry 12}

ofFlowTableWildCards OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates wildcard number."

::= { ofFlowTableEntry 13}

ofFlowTableExperimenter OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates id number."

::= { ofFlowTableEntry 14}

ofFlowTableInstructions OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates instructions."
::= { ofFlowTableEntry 15}

G.OF-Owned Certificate

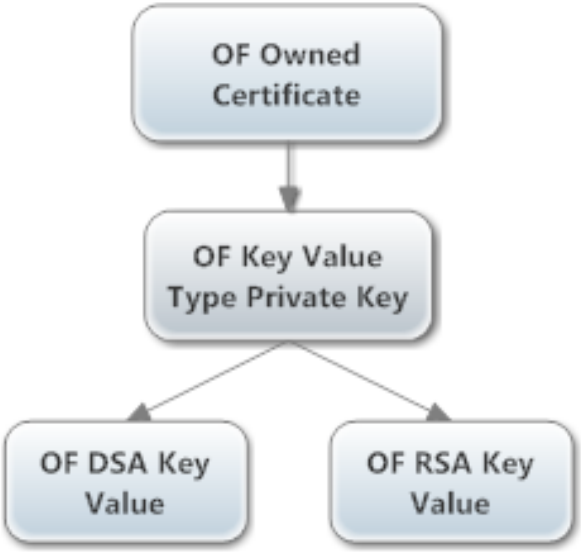

Figure 14 OF-Owned Certificate main components

Table 14 OF-Owned Certificate Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Owned certificate	1			
Key Value Type Private-Key	1 to number of Owned certificate	Key type Value		
DSA Key Value	1 to number of Owned certificate	Pingen counter speed	mbps	

RSA Key Value	1 to number of Owned certificate	Modulus Value		

G.1 MIB translation of OF-OwnedCertificate:

ofOwnedCertificate OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised."

::= {of 1.1.5}

ofKeyValueTypePrivate-Key OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of key type."

::= {of 1.1.5.1}

ofDSAKeyValue OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of PgenCounter, speed, P, Q, J, G and Y."

::= {of 1.1.5.1.1}

ofRSAKeyValue OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of Modulus and exponent."

::= {of 1.1.5.1.2}

H.OF-Queue

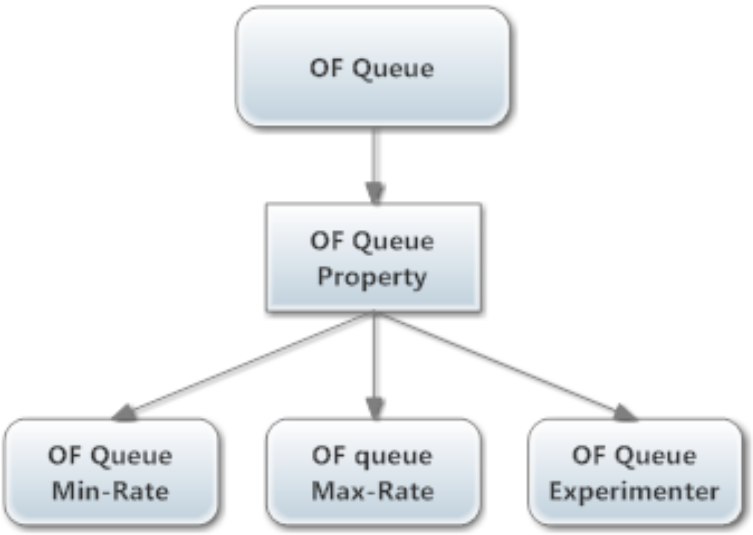

Figure 15 OF-Queue main components

Table 15 OF-Queue Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Queue	1			
property	1 to number of Queue	ID/Port number		
Min-Rate	1 to number of Queue	Min-Rate Value	percentage	
Max-Rate	1 to number of Queue	Max -Rate Value	percentage	

Experimenter	1 to number of Queue	ID		

H.1 MIB translation of OF-Queue:

ofQueue OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Represents a queue as described in the OF protocol specification."

::= {of 1.1.2}

ofQueueProperty OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"ID and port number."

::= {of 1.1.2.1}

ofQueueMin-Rate OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Value of queue min-rate (percentage 0.0 to 100.0 to 1/10 of a percent)."

::= {of 1.1.2.1.1}

ofQueueMax-Rate OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Value of queue max-rate (percentage 0.0 to 100.0 to 1/10 of a percent)."

::= {of 1.1.2.1.2}

ofQueueExperimenter OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Experimenter id and data."

::= {of 1.1.2.1.3}

I.OF-Port

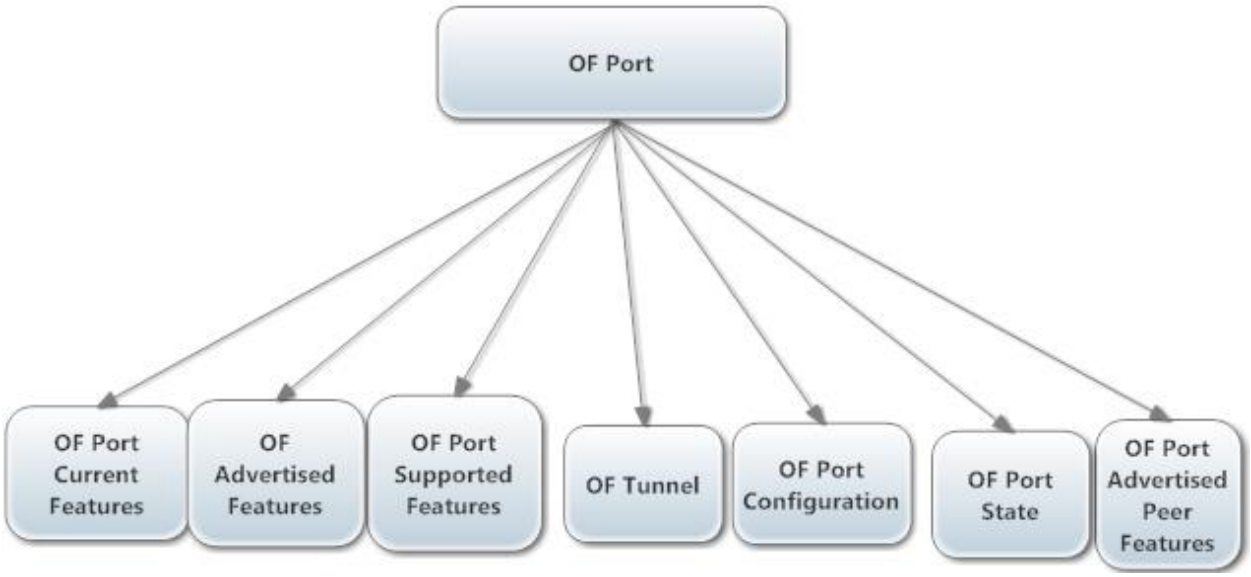

Figure 16 OF-Port main components

Table 16 OF-Port Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Port	1			
Current Features	1 to number of port s	Match fields/counters	Matching packets	

Advertised Features	1 to number of port s		Matching packets	
Supported Features	1 to number of port s			
Tunnel	1 to number of port s	IP addresses		
Configuration	1 to number of port s			
Port State	1 to number of port s			Blocked/ live Up/ down
Advertised Peer Features	1 to number of port s			

I.1 MIB translation of OF-Port:

ofPort OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Represent a physical port or a logical port."

::= {of 1.1.1}

ofPortCurrentFeatures OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"DESCRIPTION value of port current features state."

::= {of 1.1.1.1}

ofPortAdvertisedFeatures OBJECT-TYPE

SYNTAX SEQUENCE ofPortFeaturesEntry

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised features state."

::= {of 1.1.1.2}

ofPortSupportedFeatures OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port supported features state."

::= {of 1.1.1.3}

ofTunnel OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Addresses of remote and local endpoints."

::= {of 1.1.1.4}

ofPortConfiguration OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Value of admin-state up or down."

::= {of 1.1.1.5}

ofPortState OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Value of oper-state up, , blocked or live."

::= {of 1.1.1.6}

ofPortAdvertisedPeerFeatures OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised peer features status."

::= {of 1.1.1.7}

J.OF-Port Advertised Features

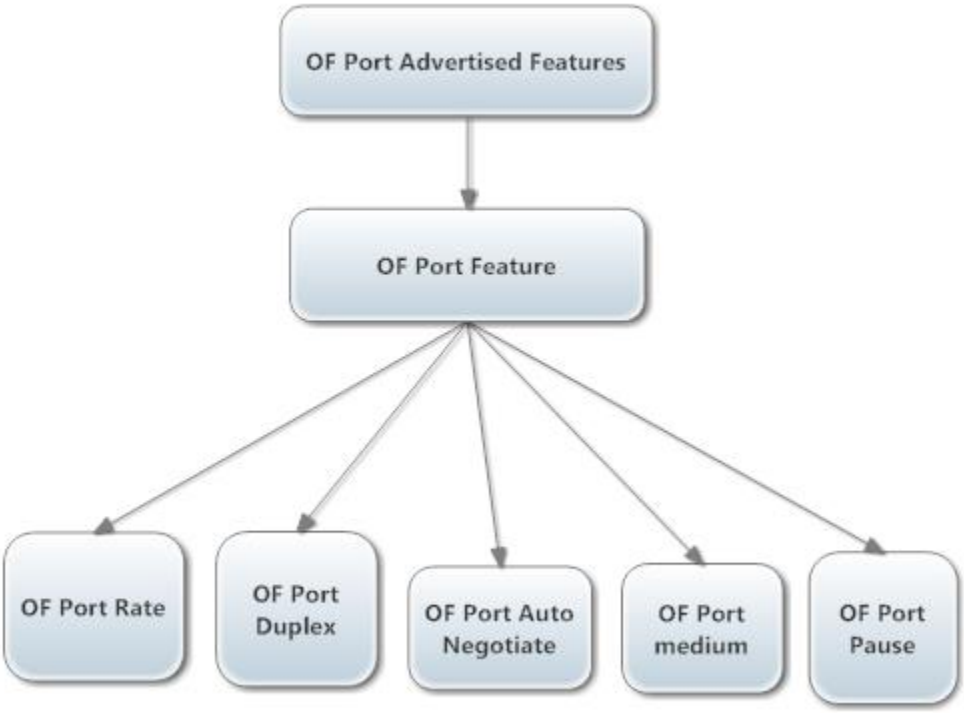

Figure 17 OF-Port Advertised Features main components

Table 17 OF-Port Advertised Features Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Advertised Features	1			

Port features	1 to number of port features			
Rate	1 to number of port features		Bits per second	
Duplex	1 to number of port features		Matching packets	Half/full
Auto Negotiate	1 to number of port features		Matching packets	Enabled/disabled
Medium	1 to number of port features	Copper/fiber	Index value	
Pause	1 to number of port features			Symmetric/asymmetric

J.1 MIB translation of OF-PortAdertisedFeatures:

ofPortAdvertisedFeatures OBJECT-TYPE

SYNTAX SEQUENCE ofPortFeaturesEntry

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised features state."

::= { ofPortFeaturesEntry 1}

OfPortFeature OBJECT-TYPE

SYNTAX ofPortFeatureEntry

ACCESS read-write

STATUS current

DESCRIPTION

"Listing OF port features as described in OF protocol."

::= { ofPortFeaturesEntry 2}

ofPortRate OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"A value in bits."

::= { ofPortFeaturesEntry 3}

ofPortDuplex OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"A value which indicates if the port is half or full."

::= { ofPortFeaturesEntry 4}

ofPortAutoNegotiate OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"A value which indicates if the port auto-negotiate is enabled or disabled."

::= { ofPortFeaturesEntry 5}

ofPortMedium OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"A value which indicates if the media is copper or fiber."

::= { ofPortFeaturesEntry 6}

ofPortPause OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"A value which indicates if the port status is symmetric or asymmetric."

::= { ofPortFeaturesEntry 7}

K.OF-Tunnel

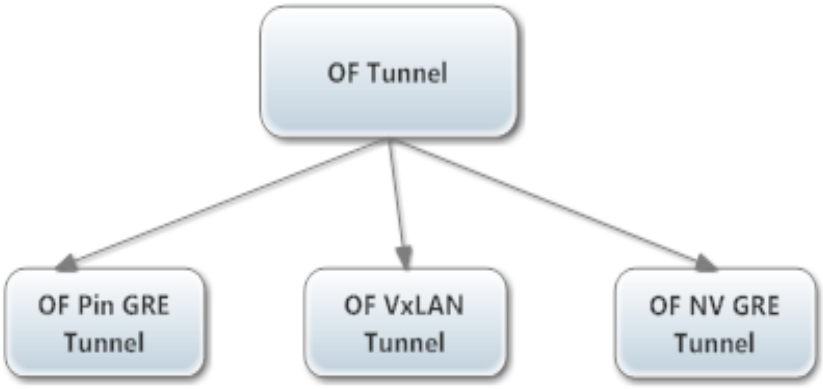

Figure 18 OF-Tunnel main components

Table 18 OF-Tunnel Object Analysis Worksheet

Component	Cardinality	Attributes	Statistics	State
Tunnel	1			
IP in GRE	1 to number of tunnels	sequence		
VxLAN Tunnel	1 to number of tunnels	IP addresses		
NVGRE Tunnel	1 to number of tunnels	IP addresses		

K.1 MIB translation of OF-Tunnel:

ofTunnel OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Addresses of remote and local endpoints."

::= {of 1.1.1.4}

ofIPinGRE OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"A textual description of the entity including checksum-present, key-present and sequence number present."

::= {of 1.1.1.4.1}

ofVxLANTunnel OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"A textual description of the entity including UDP-checksum status, source, destination, ports status, and IP address."

::= {of 1.1.1.4.2}

ofNVGRETunnel OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"A textual description of the entity including TNI multicast group and net IP address."

::= {of 1.1.1.4.3}

Chapter 5: The Tree

Usually the position where the defined manageable object should be located in the OID tree, is determined before writing the MIB module. For private MIBs as in our case, a new branch should be added under an “enterprise” branch in the “internet private” subtree. Never the less, it is often required to create branches under the “experimental” subtree for testing and experimenting purpose as well as a branch for each released MIB module. Once a MIB is published, its items cannot be changed. Therefore, a newly designed MIB should be placed under an “experimental” branch, tested then moved to a “standard” branch whenever the MIB document is published. OIDs are used extensively via SNMP with no specific constraints. The OIDs can have unlimited length with unlimited size of component values as assigned by the ASN.1[46].

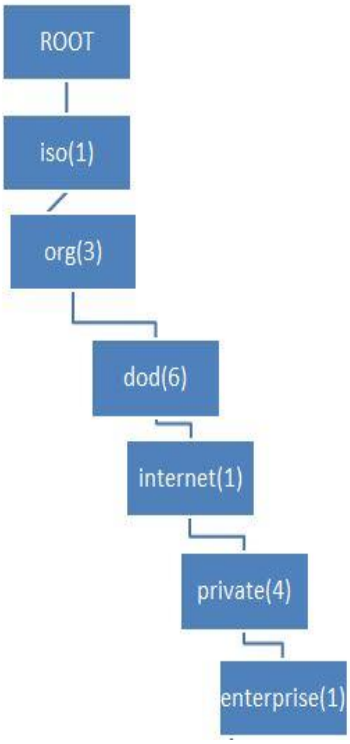

Figure 19 Standard MIB OID tree

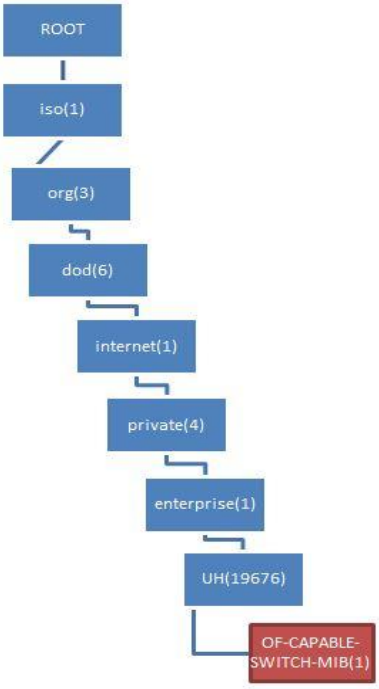

Figure 20 Standard MIB OID tree with OF-CAPABLE Switch MIB added

As for the structure of the MIB tree, all MIBs are established in a standard topology; iso.org.internet.private.enterprises (See Figure 19). Starting from the root of a tree that has no identification, the variables of our OID infrastructure layout start from the branch of OID 1.3.6.1.4.1.19676, where "19676" is assigned to the University of Houston (See Figure 20). The structure of the upper branches of the universal OID tree is fixed, so we only had control over the OID area for enterprise and this is a common case for any framework designer. Figure 21 shows the OID infrastructure layout where the top level of our MIB's namespace resides. We numbered the sub-trees with the registrations 1 through 3 for the three main components of the OF-Capable Switch: ofResource(1), ofLogicalSwitch and ofConfigurationPoint.

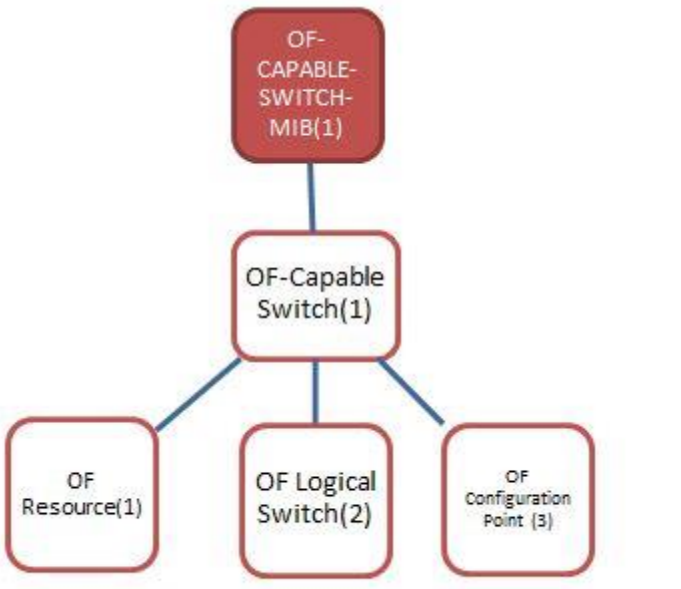

Figure 21 Top level OF-Capable Switch Namespcace Organization

The full OF-Capable Switch MIB in ASN.1 syntax is provided in the APPENDIX.

Chapter 6 Conclusions and future work

6.1 Challenges

As in any design project, a designer might face a lot of challenges. To overcome those challenges a designer should turn these challenges into targeted goals that need to be fulfilled. In our design of the OpenFlow Capable Switch MIB, the major challenge that we faced was the translation of the switch objects into the MIB syntax. It was tricky to carefully outline the switch components that could be managed. Especially that we were dealing with all software entities, it was hard to define attributes and link them to those components. If we want to describe it, it was like making a virtual thing touchable by giving it an ID and character.

It was also quite challenging for us to come up with a MIB model that might be able to cover all types of SDN-based entities. Therefore, we based our design on OpenFlow and OFCONFIG protocols, the basic pillars in the SDN environment. While OpenFlow efficiently manages flows and determines how packets are forwarded between individual source and destination pairs, it does not provide the configuration and management functions necessary to allocate ports or assign IP addresses. That's where OpenFlow configuration protocols come in. As a result, we can say that OpenFlow and OFCONFIG complement each other and it was of high importance for us to consider both protocols in our design. In general, the idea of merging the legacy SNMP into the new SDN

environment and have it compete with other newly designed and up-to-the date monitoring tools and protocols, was a challenge by itself.

6.2 Future work

An approach to fulfill the goal behind our designed MIB is to implement this MIB in a working SDN-based switch and put it under testing. We can suggest using NET-SNMP tool to monitor an OpenFlow-Capable switch as a LINC switch after adding our designed MIB. Our MIB could be added to the switch by injecting it into the standard imported MIBs via NET-SNMP. Most of the standard MIBs specify branches, like the enterprise subtree for extensions or new objects to be added.

6.3 Summary

In this thesis, we discussed how to go about analyzing systems and services with SNMP in mind, and how to translate that analysis into a MIB. Based on the OF-CONFIG data model defined in the OF-CONFIG 1.2 specifications through UML diagrams and XML encoding, we created the OF-CONFIG MIBs Tree and mapped it to SNMP.

The SNMP architectural model with the MIB tree structure helped in classifying the main elements of an operational context which is capable of supporting an OpenFlow Switch using OF-CONFIG. The MIB tree provided a description of the real capabilities of the OpenFlow Capable Switch starting from OF-Configuration Point, OF-Resources and OF-Logical Switch to the ports and queues‘ Features and properties. This MIB tree is to be later tested using an SNMP monitoring tool.

6.4 Conclusion

This thesis looks into how management in SDN environment can be approached. It makes use of proprietary-based SNMP MIBs and maps them to the new SDN-based switches. The purpose of this thesis is to build a unified information framework that can be implemented over a wide range of OpenFlow-Capable switches. According to Martin Taylor (2014) “a past that has been dominated by special- purpose network elements based on proprietary hardware will give way to a future in which network functions are implemented almost entirely in software running on shared pools of standard hardware resources, just like cloud-based IT workloads” (p.2) [59].

References

[1] Aron, M., Sanders, D. Druschel, P. and W. Zwaenepoel, "Scalable content-aware request distribution in cluster-based network servers," in Proceedings of the 2000 USENIX Annual

[2] Atlas, A., Nadeau, T., and Ward, D. Interface to the Routing System Problem Statement. IETF draft 2015.

[3] A. Greenberg *et al.*, "A Clean Slate 4D Approach to Network Control and Management," *ACM Comp. Commun. Rev.*, vol. 35, no. 5, 2005, pp. 41–54.

[4] Baik, S., Lim, Y., Kim, J., and Lee, Y. "Adaptive flow monitoring in SDN architecture". kt Infra Laboratory Daejeon, Korea, 2015.

[5] Baker, G. "Using NETCONF + YANG To Configure Network Devices And Why It Does Not Replace SNMP". Packetpushers.net (blog). November 2011. http://packetpushers.net/using-netconf-yang-to-configure-network-devices-and-why-it-does-not-replace-snmp.

[6] Bianco, A., Birke, R., Debele,F., and Giraudo, L. "snmp management in a distributed software router architecture,", IEEE Intl. Conference on communications (icc'11), pp. 1–5, jun. 2011.

[7] Bemstein, L., and Yuhas, C.M. *Basic concepts for managimng telecommunication networks:Copper to sand to glass to air (Network and systems management).,* 1999th ed. Springer, 1999.

[8] Birke, F.,Debele, G., and Giraudo, L., "SNMP Management in a Distributed Software Router Architecture," 2011 IEEE International Conference Communications (ICC), pp. 1-5, June, 2011.

[9] Chiueh, T., Tu, C., Wang, Y., Wang, P., Li, K., and Huang, Y. "Peregrine: An All-Layer-2 Container Computer Network". IEEE Fifth International Conference on Cloud Computing, 2012.

[10] CS, G and Kidambi, R. "Unites States Patent Application: 20150236918-Method and system for creating single snmp table for multiple openflow tables, August 20, 2015.

[11] Da Paz Ferraz Santos, P. R., Esteves, R. P., and Granville, L. Z. "Evaluating SNMP, NETCONF, and RESTful web services for router virtualization management. IFIP/IEEE International Symposium on Integrated Network Management, 2015.

[12] DenHartog, M., "Demystifying the MIB". DPS Telecom, February 5, 2008.

[13] Doria, A. *et al*., "Forwarding and Control Element Separation (ForCES) Protocol Specification" .IETF RFC 5810, March 2010.

[14] Feamster, N., Rexford, J., and Zegura, E." The Road to SDN."Queue, 11(12):20:20–20:40, December 2013.

[15] Feng, T., Bi, J., Xiao, P., and Zheng, X. "Hybrid SDN architecture to integrate with legacy control and management plane: An experiences-based study". Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium, pp 754-757.

[16] "Floodlight". Project. http://www.projectfloodlight.org/floodlight/.

[17] "Forwarding and Control Element Separation (ForCES) Forwarding Element Mode". IETF RFC5812, March 2010.

[18] Hamid, A , Kawahara, Y. and Asami, T. "Web cache design and implementation for efficient SNMP monitoring towards internet-scale network management," IEICE Transactions on Communications, vol. 94, no. 10, pp. 2817–2827, 2011.

[19] Hillbrecht, R., and De bona , R. "A SNMP-based virtual machines management interface", IEEE/ACM fifth international conference on utility and cloud computing, 2012.
[20] Hubbard, P., "SDN technology for network management better be more than middleware", April 2013. [Online]. Available: http://searchsdn.techtarget.com/news/2240181264/SDN-technology-for-network-management-better-be-more-than-middleware.
[21] " Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)".International Organization for Standardization International Standard 8824, December 1987.
[22] Jacobs, D., "OpenFlow-configuration-protocols-Understanding-OF-Config-and-OVSDB", April 2013. [Online]. Available: http://searchsdn.techtarget.com/tip/OpenFlow-configuration-protocols-Understanding-OF-Config-and-OVSDB.
[23] John, A., Vanderveen, K., and Sugla, B. " An XML-based Framework for Dynamic SNMP MIB Extension". Active Technologies for Network and Service Management: 10th IFIP/IEEE International Workshop on Distributed Systems , Zurich, Switzerland, October, 1999.
[24] John, W. *et al*., "Scalable Software Defined Monitoring for Service Provider DevOps". Software Defined Networks (EWSDN), 2015 Fourth European Workshop, pp 61-66.
[25] John, W. *et a*l., " Research directions in network service chaining In Future Networks and Services", 2013 IEEE SDN for, pages 1–7, Nov 2013.
[26] Kreutz, D. *et al*., "Software-Defined Networking: A Comprehensive Survey ", 2014.
[27] Kim, H., and Feamster, N. " Improving network management with software defined networking". Communications Magazine, IEEE, volume: 51 Issue: 2, pp 114-119, 2013.
[28] Madan, M.,and Mathur, M. "Cloud Network Management Model A Novel Approach to Manage Cloud Traffic". International Journal on Cloud Computing: Services and Architecture (IJCCSA) ,Vol. 4, No. 5, October 2014.
[29] "Management Information Base", Microsoft Developer Network Library. Last modified August 28, 2008. https://msdn.microsoft.com/en-us/library/aa909982.aspx.
[30] "Management information base", Wikipedia. Last modified October 28, 2015. https://en.wikipedia.org/wiki/Management_information_base.
[31] Marschke, D., Doyle, J., and Moyer, P. SDN: anatomy of openflow. Lulu.com, 2015.
[32] Matthew Roughan, "A Case Study of the Accuracy of SNMP Measurements", Australia, Journal of Electrical and Computer Engineering Volume 2010, Article ID 812979, 7 pages.
[33] McCloghrie, K., and Marshall, T.R., "Concise MIB Definitions". IETF RFC 1212, March 1991.
[34] McCloghrie, K., and Marshall, T.R., "Structure and Identification of Management Information for TCP/IP-based Internets". IETF RFC 1155 (previously RFC 1065), May 1990.
[35] McNickle, M., "OF-Config (OpenFlow Configuration and Management Protocol) definition", August 2013. [Online]. Available:http://searchsdn.techtarget.com/definition/OF-Config-OpenFlow-Configuration-and-Management-Protocol.
[36] McCloghrie, K., and Marshall, T.R., "Towards Concise MIB Definitions", IETF RFC 1212, March 1991.

[37] McKeown, N., Anderson, T. , Balakrishnan, H. , Parulkar, G. M. , Peterson, L. L., Rexford, J. , Shenker, S. and Turner, J. S. "Openflow: enabling innovation in campus networks," Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.
[38] McCloghrie, K., *et al.,* "Structure of Management Information Version 2 (SMIv2)", IETF RFC 2578, April 1999.
[39] Nayak, A., Reimers, A., Feamster, N., and Clark, R. " Resonance: Dynamic Access Control for Enterprise Networks" School of Computer Science, Georgia Tech, 2010.
[40] Network Management considerations for OpenFlow-based SDN.
[41] Nolle, T. "SDN's missing links: Five barriers blocking SDN adoption by providers". Article, Techtarget.com. May 2013.
[42] "Object Identifier". Wikipedia. Last modified September 23, 2015.https://en.wikipedia.org/wiki/Object_identifier.
[43] "OpenFlow", Wikipedia. Last modified November 29, 2015. https://en.wikipedia.org/wiki/OpenFlow
[44] "OpenFlow Switch Specification", (2014), Version 1.3.4 (Protocol version 0x04) (White paper). Open Networking Foundation.
[45] Pfaff , B., Pettit, J., Koponen, T., Amidon, K., Casado, M., Shenker, S. Extending Networking into the Virtualization Layer. Hotnets, 2009.
[46] Perkins, D., " Understanding SNMP MIBs", September 1993.
[47] Perkins, D. and McGinnis, E. Understanding SNMP MIBs. New Jersey: Prentice Hall PTR, 1997.
[48] "Plane (in networking)", Networking and communications glossary. Last modified January 2013. http://whatis.techtarget.com/definition/plane-in-networking.
[49] "Project Proposals: SNMP Plugin". Opendaylight.org. Last modified January 13, 2015. [Online]. Available: https://wiki.opendaylight.org/view/Project_Proposals:SNMP_Plugin.
[50] "SDN layers and architecture terminology". IRTF (Internet Research Task Force) RFC7426. http://tools.ietf.org/html/rfc7426
[51] Shang, Z., Chen, W., Ma, Q., and WU, B. "Design and implementation of server cluster dynamic load balancing based on OpenFlow". Lanzhou University Communication Network Center Lanzhou,China, 2013.
[52] "Simple Network Management Protocol", Wikipedia. Last modified November 10, 2015. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol.
[53] Slattery, T., "Monitoring a Software Defined Network, Part 4", February 2014. [Online]. Available: http://www.nojitter.com/post/240166288/monitoring-a-software-defined-network-part.
[54] "Software-Defined Networking (SDN): Layers and Architecture Terminology". IETF RFC7426. Online at http://tools.ietf.org/html/rfc7426.
[55] "Software-defined networking: The new norm for networks," White Papers. OFN, 2012, pp. 3–6. [Online]. Available: http://goo.gl/ElEJv.
[56] "Software-defined networking", Wikipedia. Last modified November 13, 2015. https://en.wikipedia.org/wiki/Software-defined_networking.
[57] Song, T., Kawahara, Y., and Asami, T. "Cache management algorithm of load balancer for large-scale SNMP monitoring system". Globecom 2013 Workshop- The 5th IEEE International Workshop on Management of Emerging Networks and Services.

[58] Swarna, J., Senthil, C., and Dr.K.S.ravichandran ,"Cloud monitoring based on snmp". Journal of Theoretical and Applied Information Technology 30th June 2012. Vol. 40 No.2.
[59] Taylor, M., " A guide to NFV and SDN ". White paper, Metaswitch Networks, 2014 Technical Conference (USENIX-00). Berkeley, CA: USENIX Ass., Jun. 18–23 2000, pp. 323–336.
[60] Ten Things to Look for in an SDN Controller". Ashton, Metzler and Associates.
[61] "The Open vSwitch Database Management Protocol". IETF RFC7047, 2013.
[62] Thomsen, F. "Is SNMP still used widely as of 2015?". StackExchange/serverfault blog. August 2015.http://serverfault.com/questions/709005/is-snmp-still-used-widely-as-of-2015.
[63] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. "On controller performance in software-defined networks," Hot-ICE'12 Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services,2012.
[64] "TUT: SNMP", Net-SNMP Wiki. Last modified November 4, 2010. http://net-snmp.sourceforge.net/wiki/index.php/TUT:SNMP
[65] Van Adrichem, N. L. M., Doerr, C., and Kuipers, F.A., "OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks". Network Architectures and Services, Delft University of Technology, The Netherlands.
[66] Yost, W. H., "United States Patent: 7290142 B1- System and method for initializing a simple network management protocol (SNMP) agent", October 30, 2007.
[67] Z. Cai, "Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane", Ph.D. thesis, 2011.
[68] Zhang, Y., Gong, X.,Hu, Y., Wang , W., and Que, X., " SDNMP: Enabling SDN Management Using Traditional NMS". Workshop on Advances in Software Defined and Context Aware Cognitive Networks, IEEE ICC, 2015.

APPENDIX

<OfCapableSwitch>

```
OF-CAPABLE-SWITCH-MIB  DEFINITIONS : :=  BEGIN
IMPORTS
        OBJECT-TYPE
            FROM RFC1155-SMI

ofCapableSwitch OBJECT-TYPE
        SYNTAX OBJECT IDENTIFIER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "OFConfig identification and version"
        ::= {of 1}

ofResource OBJECT-TYPE
        SYNTAX OBJECT IDENTIFIER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Assigns ports and queues associated with ofCapableSwitch and OFlogicalSwitch."
        ::= {of 1.1}

ofLogicalSwitch OBJECT-TYPE
        SYNTAX OBJECT IDENTIFIER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Identification of OFConfig, datapath and provides means for checking connection-behavior and
controller-certificate."
        ::= {of 1.2}

ofConfigurationPoint OBJECT-TYPE
        SYNTAX DISPLAY STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```

"A textual description of the entity. This value should include OFConfigID and protocol."
::= {of 1.3}

ofResource OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Assigns ports and queues associated with ofCapableSwitch and OFlogicalSwitch."
::= {of 1.1}
ofPort OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS current
DESCRIPTION
"Represent a physical port or a logical port."
::= {of 1.1.1}
ofQueue OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS current
DESCRIPTION
"Represents a queue as described in the OF protocol specification."
::= {of 1.1.2}
ofParameterized-ndm OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"A textual description of specific switch forwarding behaviors."
::= {of 1.1.3}
ofAvailable-ndm OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"Includes name, type and version of the entity"
::= {of 1.1.4}
ofOwnedCertificate OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"Value of port advertised."
::= {of 1.1.5}
ofExternalCertificate OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current

DESCRIPTION
"It can be used by an OFLogicalSwitch for authenticating itself to a controller when a TLS connection is established."
::= {of 1.1.6}

ofFlowTable OBJECT-TYPE
SYNTAX SEQUENCE ofFlowTableEntry
ACCESS read-write
STATUS current
DESCRIPTION
"A logical context which represents a flow table."
::= {of 1.1.7}

ofLogicalSwitch OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS current
DESCRIPTION
"A set of resources from an ofCapableSwitch which can be associated with a specific ofController."
::= {of 1.2}

ofController OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS current
DESCRIPTION
"Contains attributes that indicate the role of the controller and parameters of the OF connection to the controller."
::= {of 1.2.1}

ofResources OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS current
DESCRIPTION
"Assign ports and queues associated with ofCapableSwitch and ofLogicalSwitch."
::= {of 1.1}

ofCapabilities OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS current
DESCRIPTION
"Identify tables-lengths, ports-statistics, queues and flow statistics."
::= {of 1.2.3}

ofController OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write

STATUS current
DESCRIPTION
"ofConfig ID, Role (master, slave, equal), IP-address, port number and protocol (TCP, TLS)."
::= {of 1.2.1}

ofControllerOFState OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates the connection-state (up, down) and current version."
::= {of 1.2.1.1}

ofSupportedVersion OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates the version(1.3, 1.2, 1.1, 1.0)."
::= {of 1.2.1.1.1}

ofCapabilities OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS current
DESCRIPTION
"Values of max-buffered-packets, max-tables, max-ports, flow-statistics and block-looping ports."
::= {of 1.2.3}

ofReservedPortType OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates port-statistic."
::= {of 1.2.3.1}

ofGroupType OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates group-statistics."
::= {of 1.2.3.2}

ofGroupCapabilities OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates table-statistics."
::= {of 1.2.3.3}

ofActionType OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates queue statistics and IP-fragments."
::= {of 1.2.3.4}
ofInstructionType OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Indicates flow-statistics."
::= {of 1.2.3.5}

ofFlowTable OBJECT-TYPE
SYNTAX SEQUENCE ofFlowTableEntry
ACCESS not-accessible
STATUS current
DESCRIPTION
"A logical context which represents a flow table."
::= {of 1.1.7}
ofFlowTableEntry OBJECT-TYPE

SYNTAX ofFlowTableEntry

ACCESS not-accessible

STATUS current

DESCRIPTION

"A logical context which represents a flow table."

INDEX {ofFlowTableIndex}

::= {ofFlowTableEntry 1}

ofNextFlowTables OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates a counter for table id number given a maximum number of entries."

::= { ofFlowTableEntry 2}

ofFlowTableMatch OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the metadata-match."

::= { ofFlowTableEntry 3}

ofNetFlowTableMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates a counter for missed entries, it increments whenever an entry is missed."

::= { ofFlowTableEntry 4}

ofFlowTableWriteAction OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the metadata-write."

::= { ofFlowTableEntry 5}

ofFlowTableInstructionMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates the missed instructions."

::= { ofFlowTableEntry 6}

ofFlowTableApplyAction OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates type of action."

::= { ofFlowTableEntry 7}

ofFlowTableApplyActionMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Type of action missed."

::= { ofFlowTableEntry 8}

ofFlowTableApplySetFields OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates Field length."

::= { ofFlowTableEntry 9}

ofFlowTableWriteSetFieldsMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates metadata-write field length missed."

::= { ofFlowTableEntry 10}

ofFlowTableWriteSetFields OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Metadata-write field length."

::= { ofFlowTableEntry 11}

ofFlowTableApplySetFieldsMiss OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Field length missed."

::= { ofFlowTableEntry 12}

ofFlowTableWildCards OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates wildcard number."

::= { ofFlowTableEntry 13}

ofFlowTableExperimenter OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates id number."

::= { ofFlowTableEntry 14}

ofFlowTableInstructions OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Indicates instructions."

::= { ofFlowTableEntry 15}

ofOwnedCertificate OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised."

::= {of 1.1.5}

ofKeyValueTypePrivate-Key OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"Value of key type."

::= {of 1.1.5.1}

ofDSAKeyValue OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"Value of PgenCounter, speed, P, Q, J, G and Y."
::= {of 1.1.5.1.1}
ofRSAKeyValue OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"Value of Modulus and exponent."
::= {of 1.1.5.1.2}

ofQueue OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Represents a queue as described in the OF protocol specification."
::= {of 1.1.2}
ofQueueProperty OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"ID and port number."
::= {of 1.1.2.1}
ofQueueMin-Rate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Value of queue min-rate (percentage 0.0 to 100.0 to 1/10 of a percent)."
::= {of 1.1.2.1.1}
ofQueueMax-Rate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Value of queue max-rate (percentage 0.0 to 100.0 to 1/10 of a percent)."
::= {of 1.1.2.1.2}
ofQueueExperimenter OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write

STATUS current
DESCRIPTION
"Experimenter id and data."
::= {of 1.1.2.1.3}

ofPort OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"Represent a physical port or a logical port."
::= {of 1.1.1}
ofPortCurrentFeatures OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"DESCRIPTION value of port current features state."
::= {of 1.1.1.1}
ofPortAdvertisedFeatures OBJECT-TYPE
SYNTAX SEQUENCE ofPortFeaturesEntry
ACCESS read-write
STATUS current
DESCRIPTION
"Value of port advertised features state."
::= {of 1.1.1.2}
OfPortFeature OBJECT-TYPE
SYNTAX ofPortFeatureEntry
ACCESS read-write
STATUS current
DESCRIPTION
"Listing OF port features as described in OF protocol."
::= {of 1.1.1.2.1}
ofPortRate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"A value in bits."
::= {of 1.1.1.2.1.1}
ofPortDuplex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"A value which indicates if the port is half or full."
::= {of 1.1.1.2.1.2}

ofPortAutoNegotiate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"A value which indicates if the port auto-negotiate is enabled or disabled."
::= {of 1.1.1.2.1.3}
ofPortMedium OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"A value which indicates if the media is copper or fiber."
::= {of 1.1.1.2.1.4}
ofPortPause OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"A value which indicates if the port status is symmetric or asymmetric."
::= {of 1.1.1.2.1.5}
ofPortSupportedFeatures OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS current
DESCRIPTION
"Value of port supported features state."
::= {of 1.1.1.3}
ofTunnel OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"Addresses of remote and local endpoints."
::= {of 1.1.1.4}
ofIPinGRE OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION
"A textual description of the entity including checksum-present, key-present and sequence number present."
::= {of 1.1.1.4.1}
ofVxLANTunnel OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS current
DESCRIPTION

"A textual description of the entity including UDP-checksum status, source, destination, ports status, and IP address."

::= {of 1.1.1.4.2}

ofNVGRETunnel OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS current

DESCRIPTION

"A textual description of the entity including TNI multicast group and net IP address."

::= {of 1.1.1.4.3}

ofPortConfiguration OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Value of admin-state up or down."

::= {of 1.1.1.5}

ofPortState OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Value of oper-state up, down, blocked or live."

::= {of 1.1.1.6}

ofPortAdvertisedPeerFeatures OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS current

DESCRIPTION

"Value of port advertised peer features status."F

::= {of 1.1.1.7}