International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 3 Issue 5, May 2016

ISSN: 2394-4404

# Openflow Extensions For Mininet 2.0

**Kehinde Adebusuyi**

Department of Electrical and Electronic Engineering,
Federal University Oye-Ekiti Nigeria

**Gerald Ijemaru**

Department of Electrical and Electronic Engineering,
Federal University Oye-Ekiti Nigeria

***Abstract:*** ***This paper presents an overview of OpenFlow 1.0 extension module. The new node developed for the emulation of OpenFlow based networks with Minimet 2.0, a well-known, widely-used core based emulation, which offers a high degree of experiment support. It focuses mainly on modelling the OpenFlow switch and controller with a particular attention on the aspects related to support Quality of Service, and the extensible match. Subsequently, we describe in detail the general architecture of the nodes and its components with particular emphasis on the switch nodes.***

***Keywords: Mininet, SDN, packets flows***



## I. MININET EMULATORS

We introduce an intelligent Platforms for Network Experimentation and Development at the control plane with the algorithms simulated in Mininet model. OpenFlow 1.3 extension module is based on the Mininet simulation framework, Mininet is an object-oriented modular discrete event network simulation framework that can be used to model Ethernet and IP networks, network protocols, and many others things. We will describe Mininet aspects that are more relevant to understanding OpenFlow 1.3 module. More information about Mininet framework can be found in [2].

Mininet is a lightweight container based emulator with modest hardware requirements, fast start-up, hundreds of nodes and command line tool, CLI, It has a simple Python API and it install using VM, Ubuntu package, or source

There are various emulators used in wired and wireless networks but Container-based emulators are great because Apps move seamlessly to and from hardware platforms.

Mininet node exist in various classifications and can be found in [3].

Mininet is not a framework or network simulator rather it is an emulator that means running unmodified code interactively on virtual hardware on a regular PC, providing convenience and realism at low cost – with some limitations like speed and detailed. This is in contrast to running on a hardware test bed which is fast, accurate, expensive and shared or a simulator which is cheap, detailed but perhaps slow and requiring code modifications. Mininet is similar to other Container-based emulators like CORE, virtual

Emulab, Trellis, Imunes, etc. provides infrastructure of components and tools called modules, which can be combined together to form the simulated network.

A major limitation of OpenFlow v1.2 is that it does not scale for multi-domains in terms of the size of forwarding tables, and large amount of traffic over large sized networks. One way to overcome such shortcomings could be the

Enhancement of an OpenFlow architecture wherein the controller is distributed amongst the OpenFlow switches and the minimization of the tables' size. In our major contribution would require the OpenFlow nodes to act as a distributed database storing its state in a local controller residing on themselves and communicating its state to other controllers residing on the other routers.

This paper is organized as follows. Section 2 brings a more detailed description of the problem and introduces the network model. In Section 3 the recent related work is presented. Section 4 proposes a new server placement rate assignment algorithm and further explains the computational model used in the heuristic approach and Section 5 describes the experimental scenarios. In Section 6 and 7 a mathematical model and a heuristic for the problem are presented, respectively. In Section 8 the computational results of the simulations are shown and Section 9 concludes the work.

## II. OPENFLOW PROTOCOL OVERVIEW

Regarding SDN, one of the most promising solutions introduced so far is OpenFlow [Pala09] which separates the data plane from the control plane. It consists of one or more OpenFlow switches with one or more controllers. The controller makes decisions to create an entry in the flow table or to delete an entry when a flow is no longer required. The controllers (NOX/Flow Visor, among others) are connected to the switches through a secure channel. A signalling protocol is also used in the OpenFlow network to communicate control decisions from the switches to the controller through a management overlay network. As such, the data path still resides on the switch but high-level routing decisions are removed to a separate controller.

OpenFlow recognizes a set of packets (called a flow) that satisfy particular conditions such as the destination, the bandwidth, or the type of application. Upon this, it defines a path for each type of applications in order to improve the network performance by providing shortest path routing capabilities over the network and reducing the latency, number of hops and energy needed for traffic forwarding of specific end to end communications.

OpenFlow version 1.0 recognises set of packets flows that matches a set of fields. It has recognises 18 header segment with 2 option fields. The set of instructions (1 required and null option field) with matching actions (2 required and 11 option field) in entry ports (6 required header segment and 2 option field).

The ingress port receives packets flows and occupies per table statistics. The Ethernet has incoming source packets and outgoing destination packets, Version 1.0 allows type and VLAN modifications with the processed queue as flow statistics based on flow entry and port. It also support internet version 4 packet headers statistics with incoming source packet and outgoing destination packets processed queued flows per ports. Version 1.0 also support queued statistics with TCP/UDP packet flows with incoming and outgoing destination packets ports

OpenFlow version 1.1 in contrast to 1.0 recognises Meta data and MPLS packet flows. It supports group statistics f packet flows and drops packets into buckets once the set limit is exceeded when the 23 headers segment and 2 optional header matches the received packets are the entry port. It does not require a set of instructions but matching actins f3 required header and 28 optional field. Once the matching actions are sent, the incoming source packet entry port and optional field at exit port. OpenFlow version 1.1 also supports MPLS label that is received on the edge routers after pop label that is tagged with a particular traffic class such as gold for voice, bronze for video and silver for data services. This class f service features ensures that round trip delay is reduced and latency as well.

The OpenFlow 1.1 switch keeps group statistics about flows that are processed based on traffic classes, If packet is compared with the packet against the flow table entries and it matches, a single flow entry, then a set of instructions are followed and counters are updated while if more than one matching entries are fund the entries with the highest priority is selected [2]. Since it supports traffic engineering, invariably it will support complex forwarding behaviours such as load balancing, fault tolerant and link aggregations []. The grp statistics consists of more grp entries. Each entry contains grp-id, grp type, action buckets and counters update.

OpenFlow 1.2 is an extensible match (0XM). It extension gives support to the 132 packet header length of the IPv6 OpenFlow switches. The flow label is identified via the source and destination addresses devoid of router solicitation ICMPv6 messages. The ICMPv6 field …. Unlike the version 1.1 does not allow grp entries. Hence n group statistics.

OpenFlow 1.3 is a defacto standard implemented for the southbound interface of the SDN architecture. It is more robust version that supports quality of services features. Quality of service features such as class f service, intrrserve and diffserve and basic rate limiting [2]. Similar t previous version 1.2, it supports flow match fields in extension headers. The entries are measured per each flow meter r per each received entry flow based on the links, bandwidth with the set of instructions matching.

OpenFlow version 1.4 is a proposed version with promising prospects t support varying optical port properties such as ….This should provide new features for the and more SDN support s well.

Left =20mm
Right=15mm

### A. THE EMULATOR MODEL

1. The Emulator model is based n SDN Apps placed top of controller plane for proper interaction with nodes and links. The nodes are connected by predefined connection links. The modules communicate with each other by sending and receiving messages through those modules' gates connection links. At the top of this hierarchy is the network, which has no gates to the outside world in this paper work are presented as follows:

- ✓ SDN Apps (CDNi) is an arbitrary connected graph.
- ✓ Controller plane All links in the network are have the same server clusters.
- ✓ All nodes

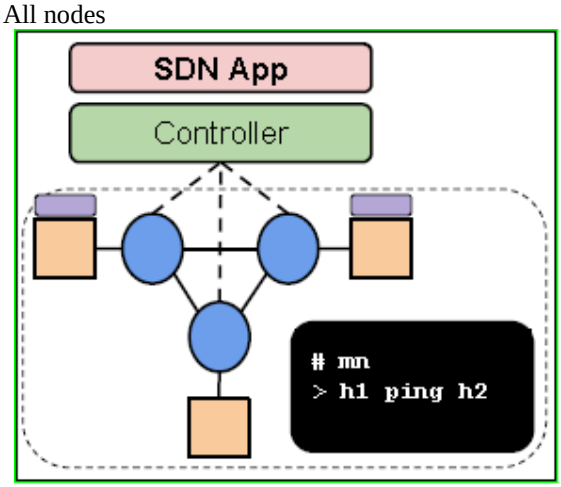


*Figure 1: A sample line graph using colors which contrast well both on screen and on a black-and-white hard copy*

Your goal is to adhere to this paper in appearance as closely as possible.

ISSN: 2394-4404

## III. SIMULATION MODEL

All paragraphs must be indented. All paragraphs must be justified alignment. With justified alignment, both sides of the paragraph are straight. The development of the OpenFlow 1.3 module for OMNeT++ simulation framework was carried out as an upgrade to the current module of OpenFlow 1.0 [1]. It is highly recommended to have a look at [1] before starting with this paper. The module provides a basic implementation of OpenFlow 1.3 devices, including OpenFlow 1.3 complaint switch nodes, Controllers, and secure channel. Due to the intrinsic complexity of the OpenFlow 1.3 standard, at the time of this writing the module does not support all the features specified by the OpenFlow 1.3 specification standard [3]. However, the proposed module has features that allow the simulation of at least several important aspects of OpenFlow 1.3 based networks; moreover, it provides a very good basis for further extensions as

### A. MININET CONTROLLER NODE

*LATEX… (BSD, PYTHON)*

The size of a lower-case "j" will give the point size by measuring the distance from the top of an ascender to the bottom of a descender.

### B. MININET. NOX

Every word in a title must be capitalized except for short minor words such as "a", "an", "and", "as", "at", "by", "for", "from", "if", "in", "into", "on", "or", "of", "the", "to", "with".

| OpenFlow Version | Match fields | Statistics | # Matches | | # Instructions | | # Actions | | # Ports | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Req | Opt | Req | Opt | Req | Opt | Req | Opt |
| v 1.0 | Ingress Port | Per table statistics | 18 | 2 | 1 | 0 | 2 | 11 | 6 | 2 |
| | Ethernet: src, dst, type, VLAN | Per flow statistics | | | | | | | | |
| | IPv4: src, dst, proto, ToS | Per port statistics | | | | | | | | |
| | TCP/UDP: src port, dst port | Per queue statistics | | | | | | | | |
| v 1.1 | Metadata, SCTP, VLAN tagging | Group statistics | 23 | 2 | 0 | 0 | 3 | 28 | 5 | 3 |
| | MPLS: label, traffic class | Action bucket statistics | | | | | | | | |
| v 1.2 | OpenFlow Extensible Match (OXM) | | 14 | 18 | 2 | 3 | 2 | 49 | 5 | 3 |
| | IPv6: src, dst, flow label, ICMPv6 | | | | | | | | | |
| v 1.3 | PBB, IPv6 Extension Headers | Per-flow meter | 14 | 26 | 2 | 4 | 2 | 56 | 5 | 3 |
| | | Per-flow meter band | | | | | | | | |
| v 1.4 | — | — | **14** | **27** | **2** | **4** | **2** | **57** | **5** | **3** |
| | | Optical port properties | | | | | | | | |

*Table 1: Font Specifications for A4 Papers*

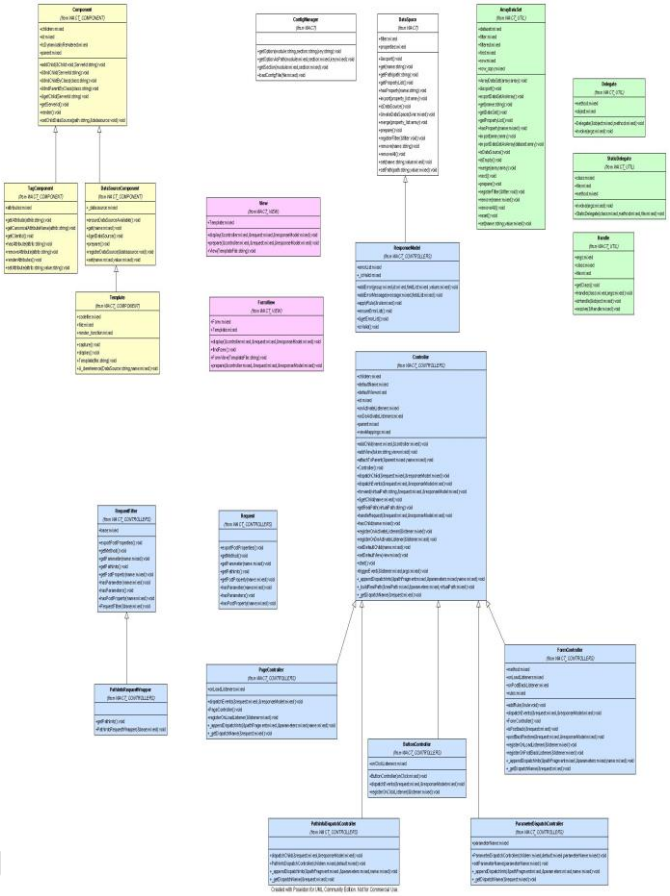

*Figure 2: Mininet Module Class UML diagram*

#### a. *CLASSES IN MININET*

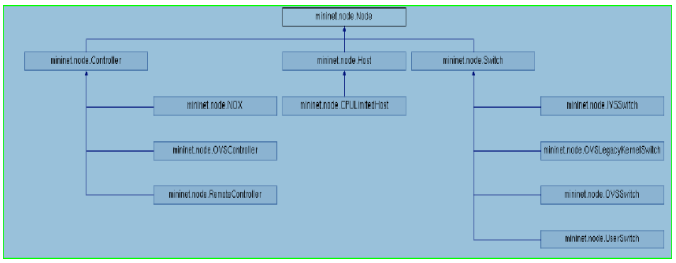


*Figure 3*

## IV. CONCLUSIONS AND FUTURE WORKS

A major limitation of OpenFlow v1.2 is that it does not scale for multi-domains in terms of the size of forwarding tables, and large amount of traffic over large sized networks. One way to overcome such shortcomings could be the enhancement of an OpenFlow architecture wherein the controller is distributed amongst the OpenFlow switches and the minimization of the tables' size. This would require the OpenFlow nodes to act as a distributed database storing its state in a local controller residing on themselves and communicating its state to other controllers residing on the other routers.

REFERENCES

[1] Mukhald A. Salih, John Cosmas, Yue Zhang "OpenFlow 1.3 Extension for OMNeT++" 2015.
[2] K. kaur, J. Singh and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform" International Conference on Communication, Computing & Systems (ICCCS–2014)
[3] Deepika M S1, K N Rama Mohan babu2, "An Approach to Effective Bandwidth Utilization using Software Define Networking" IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 2014, 5571-5574
[4] Aleˇs Friedl, Sven Ubik1, Alexandros Kapravelos "Realistic Passive Packet Loss Measurement for High-Speed Networks"
[5] Veena S, Chandan Pal, Ram P. Rustagi, K. N. B. Murthy "A Framework for Implementing Realistic Custom Network Topology in Mininet" International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064