ISBN 978-93-84422-37-0
2015 International Conference on Advances in Software, Control and Mechanical Engineering (ICSCME'2015)
Antalya (Turkey) Sept. 7-8, 2015 pp. 1-7

Design and Implementation of an Intra-domain routing module for an SDN controller for Traffic Engineering in SDN environment

Khirota Gorgees Yalda[1], Diyar Jamal Hamad[1] and Ibrahim Tanner Okumus[1]

[1]Kahramanmaras Sutcu Imam University

Abstract: *Software Defined Networking (SDN) provides an environment to test and use custom ideas in networking. One of the areas that needs this flexibility is routing in networking. In this study we design and implement a custom intra-domain routing approach in an SDN environment. In SDN routing can be implemented as part of a controller or as an application on top of a controller. In this study we implemented a module in Floodlight controller v1.1 with OpenFlow 1.3 support. This module interacts with another custom module that monitors active bandwidth use of inter-switch links inside a network. Using the information provided by monitoring module, routing module uses available capacity in inter-switch links to determine widest path between any given two points. We tested and evaluated the developed system to show its efficiency. Newly developed module can be used in traffic engineering with additional control options.*

Keywords: *SDN/OpenFlow1.3, Floodlight controller v1.1, switch OVSK and Mininet*

1. Introduction

Software Defined Networking (SDN) focuses on separation of the control plane of the network, which makes decisions about how packets should flow through the network, from the data plane of the network, which actually moves packets from place to place [1]. SDN requires some method for the control plane to communicate with the data plane. One such mechanism, OpenFlow [12].

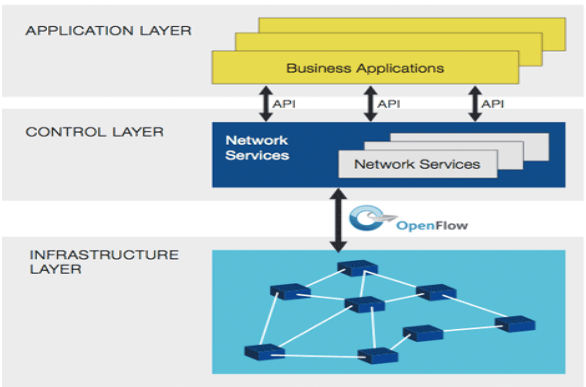

Fig.1:. SDN Architecture

Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine[3]. In traditional routing protocols, the decision process for route selection identifies a single best route to a specified set of destinations. In case there are several routes to the same destination that have the same degree of preference, the tiebreaking algorithm selects only one of these routes for inclusion in routing information base (RIB). In contrast, the OpenFlow-based Routing Service Application can intelligently have multiple routes to the same destination. Multipath routing also increases the probability of having at least one working path [4]. Researchers worked on various different routing approaches in literature, however before SDN, it was impossible to test these approaches on real networking devices and to use them in a real networking environment.

In this paper we are going to present design and implementation of an intra-domain routing module for an SDN controller that will be used for intra-domain traffic engineering in an SDN network. We will use mininet environment to simulate a network and test the performance of the routing module that we designed. We will use various traffic engineering scenarios to test the usability of the component.

2. Related work

A controller, in a computing base, is a hardware device or a software program that manages or directs the flow of data between two entities. In computing, controllers may be cards, microchips or separate hardware devices for the control of a peripheral device. In general, a controller can be idea of as something or someone that interfaces between two systems and manages communications between them[14].

An SDN network includes forwarding plane and control plane. The forwarding plane consists of OpenFlow switches (OF switches) that are responsible for directly forwarding packets in the network, and at the control plane there is a logically centralized controller which is responsible for managing OpenFlow switches in the network[7].

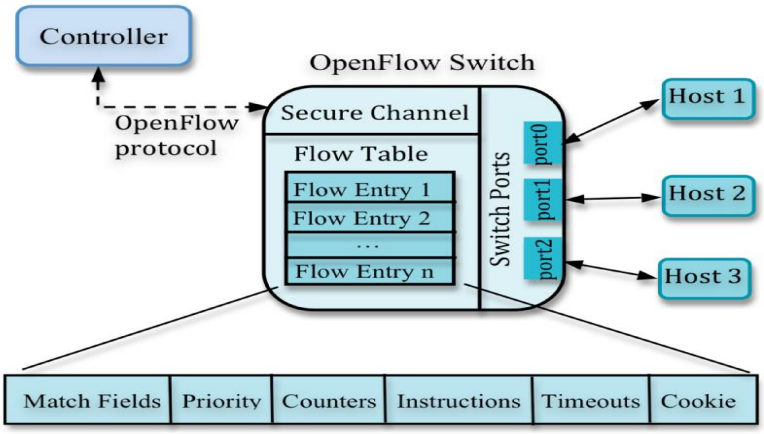

Fig 2: Architecture of an OpenFlow network[7].

Each OpenFlow switch contains one or more Flow Tables, which perform packet look-ups and forwarding, and a Secure Channel that connects the switch to the controller (fig. 2). This secure channel is the communication channel which the controller uses to operate all switches in the network via a standard OpenFlow protocol. Each table contains Flow Entries which are used as rules to process matching packets. The controller can add, update or delete flow entries in the flow table and this is the way it controls flow traffic in the network. Each flow entry basically consists of match fields which define the flow, instructions showing what to do with matching packets, and counters for statistical aim. When the OpenFlow switch receives a packet, it will

look-up its flow tables to find matching flow entry. If matching entry is found, the switch uses instructions defined in the entry to process the packet and update counters. If there is no matching entry, it sends the packet to the controller. The controller may add a new flow entry into the switch's flow table with instructions on how to process the packet, or simply drop the packet. This mechanism allows the controller to manage OpenFlow switches and control traffic in the OpenFlow network. In the newest version 1.3 of OpenFlow switch specification [10], each OpenFlow switch can establish connection with a single controller or with multiple controllers. With multiple controllers, the reliability is improved because the switches can continue to process packets in OpenFlow mode if they lose the connection with one controller. This idea allows multiple controllers work concurrently in an OpenFlow network but just the major one is responsible for directly managing all the switches in the network. Others controllers are used for recovery purpose to improve the reliability of the network. All the controllers are aimed to take care of the same network and there is still no mechanism that helps the controllers share their managing information, or cooperate all together to operate the network. With developing of the SDN many controllers released, one of them is Floodlight. Floodlight, a fork of Beacon that is managed by Big Switch Networks. While its beginning was based on Beacon it was built using Apache Ant which is a very popular software build tool that makes the development of Floodlight easier and more flexible[13].

3. Routing via OpenFlow

The Management Plane handles functions such as device management, firmware updates and external configuration via the CLI. The Data Plane refers to packet and frame forwarding through the device. The Control Plane includes functionalities such as routing protocols e.g. BGP & OSPF and switching protocols such as STP. The control plane will use the routing information to build the forwarding table used by data plane. The forwarding table is delivered to the data plane by the management plane. Thus when an Ethernet frame arrives on the switch interface, the data plane forwards it to output port. OpenFlow is a new method of control for flows in the network. To date, networking has always focused on managing frames and packets with routing protocols, but applications don't use single packets to deliver services. Rather, they exchange data between server and client, they create a stream of packets from a source to destination that is commonly known as flow. OpenFlow defines a standard for sending flow rules to network devices so that the Control Plane can add them to the forwarding table for the Data Plane. These flow rules contains fields for elements such as source & destination MAC, Source & destination IP, source and destination TCP, VLAN and QoS and more. The flow rules are then added to the existing forwarding table in the network device. The forwarding table is what all routers and switches use to decide how to send-out frame and packets to their exit ports [6]. There is a three extentions for routing via OpenFlow which explained as following:

3.1 FlowTable Extension:

Standard and open FlowTable structure composed with all forwarding elements, currently, is effective to control flows for experiment.

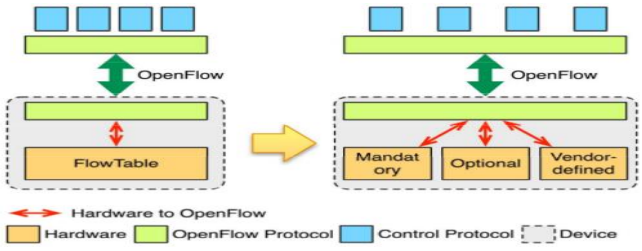

Fig. 3. FlowTable extension[2]

3.2 Control Mode Extension

Control mode of current OpenFlow is an intensive control mode that is powerful in control efficiency with a universal view, but is weak in scalability. In OpenRouter, distributed control protocols, such as OSPF, still keep running inside the device. The information generated by distributed control protocols inside the device, such as routing information, are shared with outside controller by encapsulated in OpenFlow protocol, as shown in Fig.4.

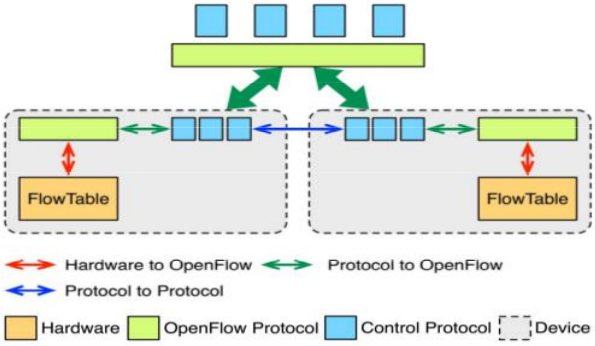

Fig. 4. Control mode extension with sharing data with inside/outside protocols by OpenFlow protocol encapsulation[2]

3.3 OpenFlow Protocol Extension

Network data in OpenFlow message are reorganized using TLV (Type, Length, and Value) format. It can not only support transferring of variable-length and arbitrary data, but also arrive the aim of flexible extension of FlowTable structure in future[2].

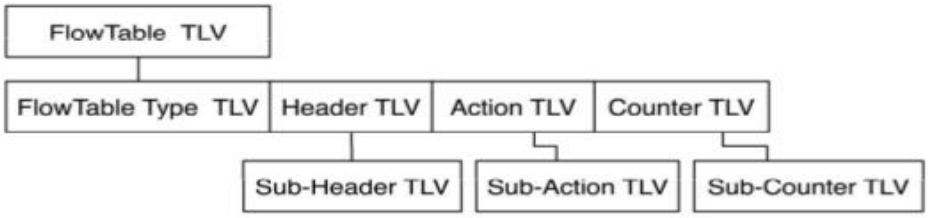

Fig. 5. OpenFlow protocol extension with TLV format and new-added FlowTable type field[2].

5. Test environment

In our results of changing the Shortest path to the Widest path according to the BandWidth we used Mininet to create SDN capable test topology. Mininet[8] is a network simulator for modeling software defined networks. With Python language, Mininet is simple to use and has a big flexibility. It can create components with its slight approach which uses an OS-level virtualization. The topology used for testing the Widest path is illustrated in Figure 6. This topology is enough to determine if it is work as expected.

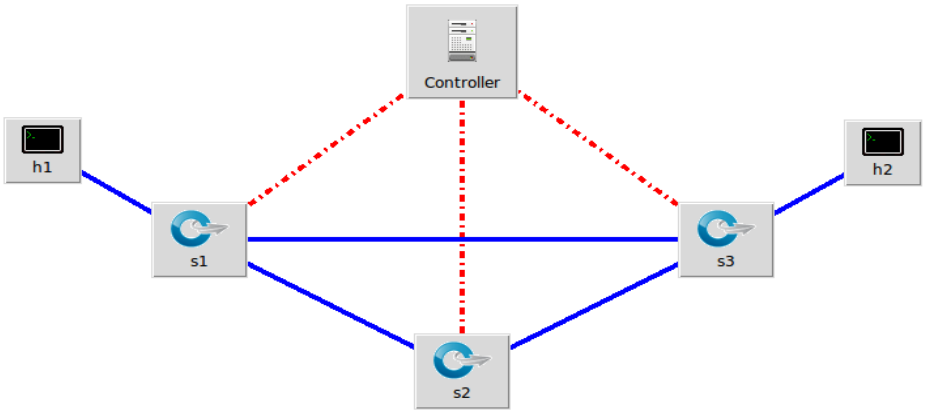

Fig 6: Topology

The OpenFlow network controller used in the test is Floodlight. Floodlight is an SDN Controller presented by Big Switch Network that works with the OpenFlow protocol to orchestrate traffic flows in a Software Defined Networking (SDN) environment [9]. One of OpenFlow switches is Open vSwitch (OVS). Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable enormous network automation through programmatic extension and supporting OpenFlow 1.3 protocols except Meters[11]. Our purpose is to change the Shortest path to the Widest path Routing. The Widest Path is a routing mechanism based on Dijkstra algorithm. Dijkstra is a greedy algorithm which makes at each step in the computation the best choice at that moment. The current version of Dijkstra algorithm in TopologyInstance.java module implemented in the Floodlight Controller is computed based on equal weights (i.e. equal costs on the links). For an accurate approach, the algorithm has been modified and it was considered that the weight on the links correspond to random loads on each aggregation switch. Therefore, each weight has been randomly assigned a positive value. Our work is based on finding the Widest path according to the Band Width. We changed the weight of the links and the module is going to find the mini max link that has the minimum Band Width. In order to check the functionality of Floodlight controller with the TopologyInstance Module, the (LLDP) exchanging between switches (as in Figure 5) the log messages displayed in the controller's console. one or multiple UDP flows are created between S1 towards S3 with data rate of 5 Mbps, through the testbed, Original Routing was according the Shortest path, from S1 to S3 directly, because it considered to be the Shortest path between them. each router builds a map of the network topology and the bandwidth available on each link. Then, each ingress router finds a route with sufficient bandwidth to an egress router for each traffic aggregate. after changing the Algorithm now we can see the path changed and routes from S1 to S2 then from S2 to S3.

When the Dijkstra was using Shortest path Algorithm, the packets were flowed in the Shortest path between two nodes without concern of BandWidth, but in the Widest path Algorithm the packets flows through the path that has the less traffic. To obtain the BandWidth we used another module which is added to Floodlight controller for calculating the BandWidth. The Bandwidth in this module is updating each 10 seconds.

We used the following pseudo-codes in Dijkstra Algorithm, The modified Floyd-Warshall's algorithm for the widest path problem is shown below. We have defined two 2D arrays, one for storing the bandwidth of each pair of vetices and another for generating the path corresponding to the bandwidth. The function FloydWarshall AllPairs WidestPathProblem is used for finding the bandwidth for all pairs and the function Trace Path is used to reconstruct the widest path for a given pair of vertices[15].

Pseudocode for Widest Path:

```
Initialize bandwidth as an array of size | V | | V | to be −∞
Initialize next as an array of size | V | | V | to be null
function FloydWarshall_AllPairs_WidestPathProblem (G(V, E))
for each vertex v in V do
      bandwidth[v][v] ← 0
      end for
      for each edge (u, v) in E do
      bandwidth[u][v] ← w(u, v)
      end for
      for k from 1 to | V | do
      for i from 1 to | V | do
      for j from 1 to | V | do
if min(bandwidth[i][k], bandwidth[k][j]) > bandwidth[i][j] then
      bandwidth[i][j] ← min(bandwidth[i][k], bandwidth[k][j])
      next[i][j] ← k
      end if
      end for
      end for
      end for
      end function
      function Trace Path(i, j)
      if bandwidth[i][j] == −∞ then
      return “There does not exists any path from i to j”
      end if
      k ← next[i][j]
      if k == null then
      return “ ”
      else
      return Trace Path(i, j) + k + Trace Path(k, j)
      end if
      end function
```

6. Conclusions

The goal of SDN is to make sure that all control-level decisions are taken at a central way, as compared to traditional networking, the control-level decisions are taken locally and intelligence is distributed in each switch. In this paper we mentioned about Routing in SDN controller generally, we worked on changing the

Shortest path to the Widest path Algorithm deployed with Floodlight controller in an OpenFlow network, we proposed that controller allows multiple path, it depends on link capacity and a solution for routing packets based on Widest path in Dijkstra, our design is a promising solution for Open-Flow scalability.

7. References

[1] available online: http://searchsdn.techtarget.com/definition/software-defined-networking- SDN

[2] Tao Feng, Jun Bi, and Hongyu Hu, OpenRouter: OpenFlow Extension and Implementation based on a Commercial Router, 19th IEEE International Conference on Network Protocols, 2011

http://dx.doi.org/10.1109/icnp.2011.6089045

[3] Vangie Beal, Routing

[4] GAUTAM KHETRAPAL, SAURABH KUMAR SHARMA, Demystifying Routing Services in Software-Defined Networking

[5] Ungureanu. Oana-Mihaela, Flexible and Programmable Evolved Packet Core: A New SDN-based Model, July 7, 2014

[6] Greg Ferro, OpenFlow and Software Defined Networking: Is it Routing or Switching ? 2011

[7] Xuan Thien Phan, Nam Thoai and Pierre Kuonen, A collaborative model for routing in multi-domains OpenFlow networks, IEEE conference 2013

[8] Mininet. Available: http://mininet.org/

[9] available online: https://www.sdxcentral.com/resources/sdn/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/

[10] "OpenFlow switch specification v1.3.0," Open Networking Foundation, 2012. Available: https://www.opennetworking.org/images/stories/downloads/specificat ion/openflow-spec-v1.3.0.pdf

[11] available online: http://openvswitch.org/

[12] available online: https://en.wikipedia.org/wiki/Software-defined_networking

[13] Sean Wilkins. A Guide To Software Defined Networking (SDN) Solutions, 2014

[14] available online: http://whatis.techtarget.com/definition/controller

[15] Venkat Sasank Donavalli, Algorithms for the Widest Path Problem, 2013