CS540 - Paper Review Report # XIII

Hailu Belay Kahsay - 20155624

Title:OpenFlow: Enabling Innovation in Campus Networks

Author: Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner

The authors claim that an OpenFlow protocol enables researchers can innovate in networking, allowing them to develop and test their innovations(experimental protocols) in realistic network settings. OpenFlow protocol is amenable to high-performance and low-cost implementations, capable of supporting a broad range of research, and is able to separate research traffic from production traffic, while do not require vendors to expose the internal workings of their switches.

It provides an open protocol to program the flow table in different switches and routers that allows researchers to run experiments on different switches and routers in a uniform way, without the need for vendors to expose the internal workings of their products, or researchers to write vendor-specific control software. It can also isolate the research traffic and the productive traffic in the same network. Researchers can control their own flows by choosing the routes their packets follow and the processing they receive. In this way, researchers can try new routing protocols, security models, addressing schemes, and even alternatives to IP.

In the paper it is stated that OpenFlow Switch consists of at least three parts: *Flow Table*, *Secure Channel* and *the OpenFlow Protocol*. The paper also talks about the two categories of OpenFlow switches: (1) Dedicated OpenFlow Switch, that do not support normal Layer 2 and Layer 3 processing, is a dumb datapath element that forwards packets between ports, as defined by a remote control process and (2) OpenFlow-enabled switches, general purpose commercial Ethernet switches and routers, to which the OpenFlow Protocol and interfaces have been added as a new feature by adding the *Flow Table*, *Secure Channel* and *OpenFlow Protocol*.

The paper also presents a list of examples, such as Network Management and Access Control, VLANs, Mobile wireless VOIP clients, Non-IP networks and processing packets rather than flows, that show how OpenFlow-enabled networks could be used to experiment with new network applications and architectures.

Discussion Points:

- Efficiency issue as the resources are shared by the research traffic and production traffic?
- It uses a controller. Single-point of Failure?
- Scalability of OpenFlow: How many flows can the flow table handle?
- In Ethane network users can be identified and bounded with hosts when they join the network. Is there a way to identify particular users in OpenFlow network?
- How secure is OpenFlow?

Title: **P4: Programming Protocol-Independent Packet Processors**

Author: Pat Bosshart , Dan Daly , Glen Gibb , Martin Izzard, Nick McKeown, Jennifer Rexford , Cole Schlesinger, Dan Talayco , Amin Vahdat , George Varghese, David Walker

The authors claimed that OpenFlow specification has become complicated, with many more header fields (from 12 fields to 41 fields within five years) and multiple stages of rule tables, and programming of new generation of switch chips is far from easy, as each chip has its own low-level interface.

The authors proposed P4, a high-level language for programming protocol-independent packet processors, that allows network programmers to declare and program with arbitrary packet header fields on OpenFlow-style switches. It raises the level of abstraction for programming the network, and can serve as a general interface between the controller and the switches. P4 can have the following impacts: It can make the switch more flexible for parsing packets and matching headers, it is independent with target devices, it allows programmers to define how to parse headers, which fields should be matched and what actions should be performed and it also allows programmers to define how the packet processing procedure should be

In designing P4, the authors have three goals: Reconfigurability, Protocol independence and Target independence. To support theses features, the packet parser and action table of hardware need to be configurable so that the switch's behavior can be changed in the field, and dedicated compilers for different types of switch are required to enable P4 programs to be written in target-independent way.

This paper presents an example for using P4, that take advantages of MPLS, PortLand protocols, using P4 creating mTag, then walk through the example as a introduction of P4 components: Headers, Parsers, Tables, Actions, Control of Programs.

Even though P4 will lead to future switches that provide greater flexibility, and unlock the potential of software defined networks, several aspects of a switch remain undefined (e.g., congestion-control primitives, queuing disciplines, traffic monitoring).

Discussion points:

- How does P4 relate to software-defined networking (SDN)?
- Scalability of P4?
- Compatibility issue?