Open Source Platforms, Applications and Tools for SDN and 5G research

TBD

Technical Report C-2014-X
University of Helsinki
Department of Computer Science

Helsinki, August 11, 2014

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author

TBD

Työn nimi — Arbetets titel — Title

Open Source Platforms, Applications and Tools for SDN and 5G research

Oppiaine — Läroämne — Subject

Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Technical Report C-2014-X	August 11, 2014	9

Tiivistelmä — Referat — Abstract

Software-Defined Networking (SDN), a novel solution to network configuration and management, has shown great potential to simplify the existing complex and inflexible network infrastructure. For the design strength, SDN is gaining various investment from both industry and academia in terms of experimental implementation and deployment. In this report, we present. Our focus is on . We discuss the potential research directions and share our perspectives in this domain.

Avainsanat — Nyckelord — Keywords

SDN, Open Source, 5G Mobile Networks

Säilytyspaikka — Förvaringsställe — Where deposited

University of Helsinki / Kumpulan Kampuskirjasto

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

2 Open Source SDN Components

Even though the concept of software defined networking dates back to year 2004 [25] the first widely recognised applications followed only several years later. The most notable one is the widespread OpenFlow protocol [13] which made its official debut in 2009 and has since become the most prevalent protocol for SDN. The available applications on the market reflect the pervasiveness of OpenFlow as practically all of them are built to support the protocol. Keeping SDN's basic concepts and architectural solutions in mind it doesn't come as a surprise that the network controllers and switch implementations are the most common SDN related applications. In this section, we present a comprehensive survey of existing open source components for SDN.

2.1 Controllers

Most OpenFlow controllers are organized in the same fashion as shown in figure 1. Most of the current controllers offer a so-called Northbound interface (usually a REST API) to communicate with applications and to offer them services. The southbound interface is for communicating with the actual physical and virtual switches in the network.

Hailed as the first OpenFlow controller, NOX was developed side-by-side with OpenFlow but peculiarly released a year before OpenFlow's official release [10]. NOX is written in C++ and is meant for developing further customised SDN controllers, but it is commonly regarded as complex and cumbersome to use for anything but really performance critical applications. The developer Murphy McCauley himself recommends using POX [15], the Python version of NOX, for most tasks [22]. There is also a newer version of NOX that according to developers has more streamlined architecture, cleaner codebase and is more efficient. At the time of writing this the latest modification to source code was an over 7 months old bugfix, so NOX's actual viability in current SDN development is questionable.

Beacon is another notable OF controller [1]. It was released in 2010 and it's written in Java. Despite Java's reputation as an inefficient programming language Beacon has fared well performance-wise in comparison with other controllers [23]. Java was chosen to address C's and C++'s portability problems and to reduce developer's burden with significantly shorter compilation times and better error logging. Other controllers with similar focus on easier development are for example Python-based Ryu [17] and Ruby-based Trema [19].

One of the features that makes Beacon perhaps more notable than other aforementioned controllers is the fact that it serves as basis for one of the most

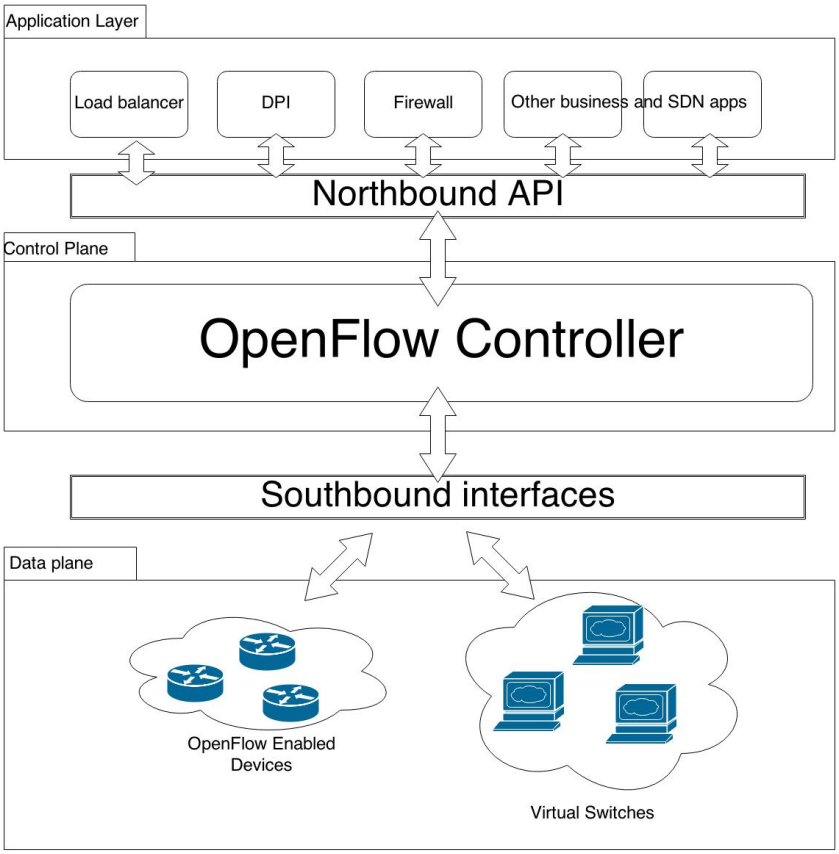

Figure 1: General SDN controller architecture

popular OpenFlow controllers: Floodlight [2]. Floodlight is developed by Big Switch Networks, a startup founded by networking veterans. Unlike Beacon, Floodlight doesn't rely on OSGi framework and offers more features such as REST API and support for non-OpenFlow domains. The documentation for Floodlight is also generally good and surpasses the Beacon documentation with ease. Big Switch has in the past lobbied for Floodlight to be included in Open Daylight Project's codebase [21] but the project opted for Cisco's controller implementation.

Open Daylight Project [11], or ODP in shorthand, is commonly considered the leading SDN project with it's substantial company backing and large developer community. The project members involved include practically every company relevant to the field. These are for instance Cisco, Microsoft, IBM, Hewlett-Packard, Juniper, Red Hat and VMware and many more. Feature highlights of Open Daylight include a robust REST API and southbound communication which allows for using other non-Openflow protocols: OVSDB, NETCONF, LISP, BGP, PCEP and SNMP. Naturally ODP contains components to take advantage of the protocols as well as other components

not found in any other SDN projects e.g. DDOS detection and counter-measures The basic architecture is rather similar to other controllers, but different core components, applications built on top of them and protocol plugins that offer different services and use different interface complicate the organization a bit. Figure 2 shows the architecture and some of the components in the bundle. For prototyping applications that only make use of OpenFlow the sheer number of different components in ODP may be intimidating and distracting, but generally ODP community offers decent documentation and guides though quality varies from one component to another. In comparison with Floodlight ODP is likely to be more viable in real business environment because of the features it offers and fast evolving nature but when developing quick prototypes or SDN applications that only make use of OpenFlow, Floodlight may be a better choice due to it being simpler, smaller and at the moment better documented.

Other lesser known controllers and controller frameworks are for example a very bare bones Libfluid [6], Java-based Maestro [7], On.Lab's Flowvisor [4] which is meant to be used with other On.Lab's SDN components, FLOWer [3] which is written with Erlang, the event-driven Resonance[16], Node.js -based NodeFlow [9], lesser-known KulCloud Open MUL and SNAC citeMUL, SNAC, flowforwarding.org's Warp [20], network virtualization bundle Open VNet [14], IRIS which uses Floodlight and Beacon components to offer horizontal scalability [5] and Juniper's OpenContrail focusing on SDN and Big Data [12]. Generally these pieces of software are either not in active development or not as well received or otherwise as notable as the controllers mentioned earlier.

2.2 Switches

OpenFlow relies fundamentally on switches: there has to be at least one in the network to connect to a controller, otherwise using OpenFlow is not possible. OpenFlow being near synonymous to SDN and network functions virtualisation or NFV being a hot topic for conversation along with SDN there's a demand and supply for standardised, vendor-agnostic virtual and physical switches. With SDN and network virtualization it is possible to eliminate the need for specialised switches, routers, middleboxes etc. and utilise the network elements on ordinary X86 servers.

VMWare's Open VSwitch is undoubtedly the most well known of virtual switches. It has been ported to multiple virtualization platforms such as VirtualBox and KVM, is included in few Linux kernels and bundled with many SDN projects and is also integrated to various virtual management systems like OpenStack. Open VSwitch offers many features like several QoS and security components and it supports few different protocols.

Other software switches include Indigo Virtual Switch meant to be used with Floodlight and Indigo framework, CPqD's OFSoftwsitch, LINC-switch,

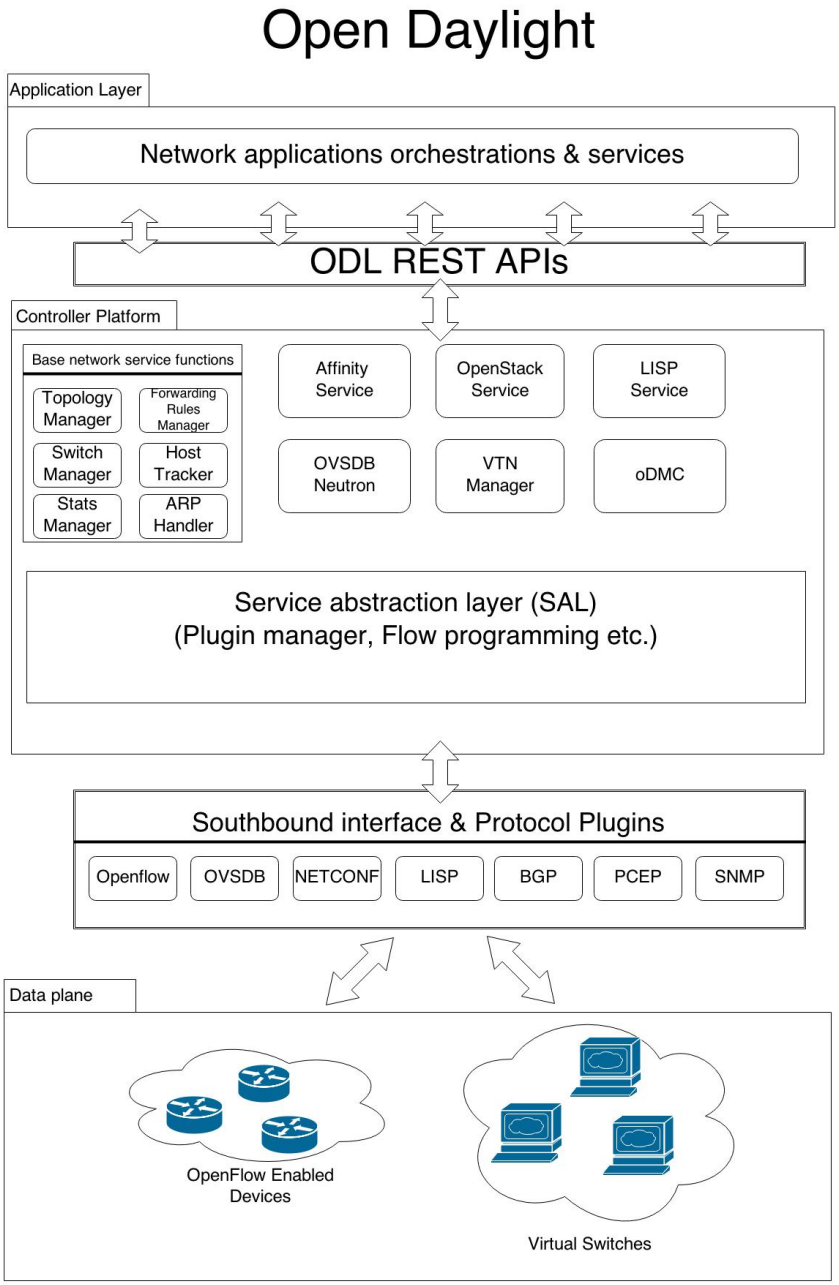

Figure 2: Open Daylight architecture

Snabb.co's virtualized ethernet switch called Snabb Switch and protocol oblivious POFSwitch. The previously mentioned open VNet also includes a virtual switch.

Other switch related projects are Centec's Lantern which includes a modified Open VSwitch but is mainly design for implementing custom

hardware based SDN switches with accompanying software bundle. Other partly similar software is Pantou. It can be used to turn normal commercial routers to OpenFlow-enabled switches.

2.3 Testing and Simulating

Like in any other field in computer science, fast and professional development and research is supported by tools for testing and quality assurance. SDN technology is not an exception. Tools fall roughly in two categories: Some are used for simulating networks and network events while others focus on performance benchmarking and monitoring.

The most well known tool for simulating networks is Mininet. Mininet could be considered essential for conducting research on SDN and testing software in a realistic network environment. Mininet supports custom network topologies which can be easily created with its Python API and its switches support OpenFlow. Connecting to real networks should be quite straightforward though a single Linux OS can support over four thousand switches and hosts simultaneously [8].

A little bit more specific tool is STS which stands for SDN Troubleshooting System. It can be used to simulate and troubleshoot specific devices on the network. A bit similar program is ns-3 which is used for simulating network events. For performance testing and benchmarking there are OFLOPS and PerfSonar and for testing SDN software itself one can utilize NICE for controller applications, OFTest for compliance testing and TestOn for testing automatization. It's also worth to mention HyperGlance by Real Status. It is a network visualization tool which supports Open Daylight. It could most likely to be used for debugging as in addition to network topology it displays traffic too. Unlike other pieces of software mentioned HyperGlance isn't an open source project or free software, but at the time of writing it was in open beta phase and acquiring the software required only registration to the site.

2.4 Other SDN projects

Apart from controllers, switches, testing and simulation tools there are many notable projects that don't straightforwardly fall into these categories.

There hasn't been as much talk about security with SDN as there have been on for example network virtualization, but the centralized network control introduces fundamental vulnerabilities that need to be addressed. Roy Chua of SDNCentral.com points out, that the SDN controller should always be closely controlled and also protected from an outside attack like DDOS because without it SDN network's functionality is crippled [18] . Other aspects to be aware of are protecting communication throughout the network and maintaining and monitoring network performance and function. Open Daylight for example includes a component for protection from Denial

of Service attacks and OpenFlowSec.org offers multiple security solutions for SDN ranging from malware protection to security policy enforcement.

Other projects aim to enhance the performance of the network. InCntre's Flowscale is a network load balancer, but the development has stopped after version 0.6. This may be because of many controller applications including load balancer of their own. Both the BIRD by CZ.NIC Labs and CPqD's RouteFlow offer IP routing.

Some components are made to enhance SDN development. These include domain specific languages like Frenetic and Pyretic developed in Princeton University and Haskell based Nettle from Yale university. In same vein FlowForwarding.org offers protocol libraries for NetConf and OpenFlow written in Erlang, Midokura has one for OpenFlow written in Python and Ericsson provides a library made with Node.js. They are named enetconf, of_protocol, OpenFaucet and Oflib-Node respectively.

Other various project include Big Switch's Indigo, a hardware abstraction layer and configuration layer meant for communication between floodlight and the physical layer, MirageOS, used for constructing unikernels for network applications, wakame-vdc for data center virtualization and for wireless communication there is nwEPC - EPC SAE Gateway by Thomas Batia and Open Source IMS Core from Fraunhofer Fokus and OpenEPC of which latter is not an open source project but otherwise worth mentioning in this context.

EU has various research projects under moniker Seventh Framework Programme (FP7) which has produced both research and open source software. Some individual projects in FP7 like FELIX and OFELIA and FIRE research test-beds for use and frameworks for using them (OFELIA). Other projects focus more straightforwardly to OpenFlow. ALIEN project for instance offers an OpenFlow enabling hardware abstraction layer for non-OpenFlow enabled devices and integration with OFELIA whereas OFERTIE aims to improve delivery of real-time interactive applications on the networks. SPARC project aims to enhance standard SDN architecture by splitting the control and forwarding functions by utilizing MPLS.

3 SDN-based service chaining for mobile networks

SDN (software-defined networking) has revolutionized designing and managing networks over the past few years. Routers and switches operate with complex software, which is closed source and dedicated to the private network companies. Moreover, each of these devices come with their own configuration interfaces that is different between vendors, which makes work of network administrators exhausting to configure each of these individual network devices. [24]

SDN changes the way networks are managed and designed by defining two characteristics. First, SDN detaches control plane from the data plane.

Second, SDN unifies the control plane, thus a single software program (controller) can control numerous data plane elements. SDN controller has direct access over the data plane through a well-defined API such as OpenFlow. An OpenFlow switch has packet-handling table, which different rules can be defined in the table. Therefore, based on different rules that match with a subset of traffic, different actions (such as dropping, forwarding and header modification) can be taken. Thus, based on different rules installed on OpenFlow by the controller, an OpenFlow switch can act as a router, firewall, network address translator, switch or something in between. [24]

As discussed earlier, network instruments are closed and proprietary, which is a barrier to openness. Technical innovation can break down the barrier caused by industry and government in order to breed openness. The future perspective of wireless mobile networks is that any mobile device can connect to any network and move from one network to another seamlessly and freely. Thus, handheld devices can connect to any network regardless of what radio technology it uses and who owns the network. Figure 3 illustrates an overview of OpenFlow Wireless. [27]

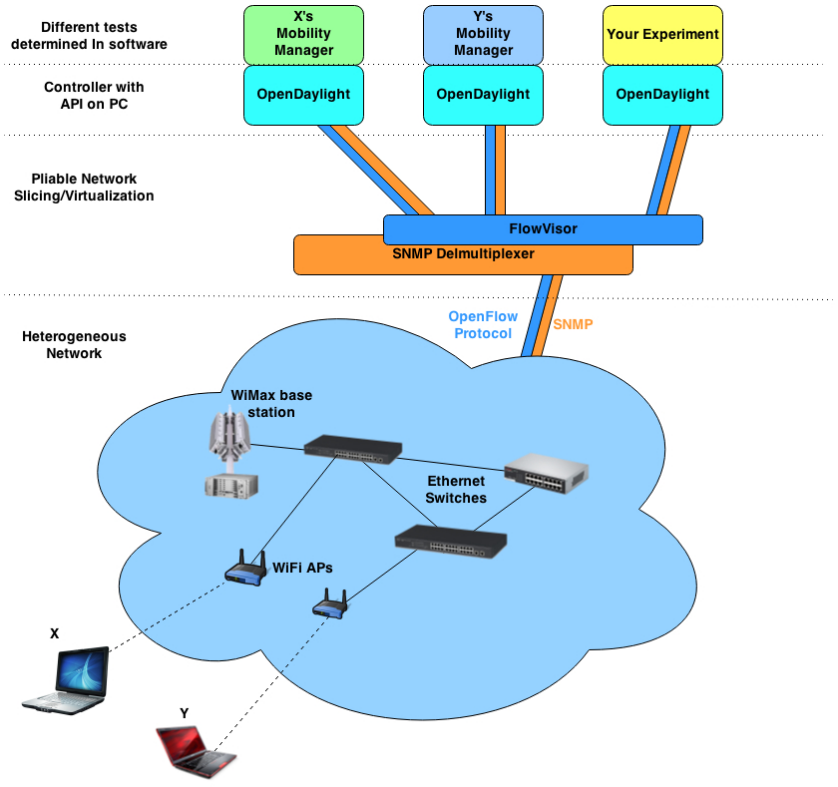

Figure 3: OpenRoads Architecture [26]

References

[1] *Beacon controller*. https://openflow.stanford.edu/display/Beacon/Home, visited on 07-08-2014.

[2] *Floodlight project*. http://www.projectfloodlight.org/floodlight/, visited on 07-08-2014.

[3] *Flower*. https://github.com/travelping/flower, visited on 07-08-2014.

[4] *Flowvisor*. https://openflow.stanford.edu/display/DOCS/Flowvisor, visited on 07-08-2014.

[5] *Iris*. http://openiris.etri.re.kr/, visited on 07-08-2014.

[6] *Libfluid*. http://opennetworkingfoundation.github.io/libfluid/, visited on 07-08-2014.

[7] *Maestro-platform*. https://code.google.com/p/maestro-platform/, visited on 07-08-2014.

[8] *Mininet overview*. http://mininet.org/overview/, visited on 05-08-2014.

[9] *Nodeflow*. http://garyberger.net/?p=537, visited on 07-08-2014.

[10] *Nox controller*. http://www.noxrepo.org/nox/about-nox/, visited on 07-08-2014.

[11] *Open daylight project*. http://www.opendaylight.org/, visited on 07-08-2014.

[12] *Opencontrail*. https://github.com/Juniper/contrail-build, visited on 07-08-2014.

[13] *The openflow switch specification*. http://OpenFlowSwitch.org, http://OpenFlowSwitch.org.

[14] *Openvnet*. http://openvnet.com/, visited on 07-08-2014.

[15] *Pox controller*. http://www.noxrepo.org/pox/about-pox/, visited on 07-08-2014.

[16] *Resonance*. http://resonance.noise.gatech.edu/, visited on 07-08-2014.

[17] *Ryu controller*. http://osrg.github.io/ryu/, visited on 07-08-2014.

[18] *Security challenges in SDN.* http://www.sdncentral.com/security-challenges-sdn-software-defined-networks/, visited on 06-08-2014.

[19] *Trema controller.* http://trema.github.io/trema/, visited on 07-08-2014.

[20] *Warp.* http://flowforwarding.github.io/warp/, visited on 07-08-2014.

[21] Banks, Ethan: *Big Switch leaves OpenDaylight, toutes white-box future.* http://www.networkcomputing.com/networking/big-switch-leaves-opendaylight-touts-white-box-future/a/d-id/1234231?

[22] Chua, Roy: *NOX, POX and controllers galore – Murphy McCauley interview*, 2012. http://www.sdncentral.com/sdn-nfv-open-source/nox-pox-controllers-murphy-mccauley/2012/09/.

[23] Erickson, David: *The Beacon Openflow controller.* In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 13–18, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2178-5. http://doi.acm.org/10.1145/2491185.2491189.

[24] Feamster, Nick, Rexford, Jennifer, and Zegura, Ellen: *The road to sdn.* Queue, 11(12), December 2013. http://doi.acm.org/10.1145/2559899.2560327.

[25] Robert, B.I.I. and Carman, D.Z.: *System for regulating access to and distributing content in a network*, February 2012. http://www.google.com/patents/US8122128, US Patent 8,122,128.

[26] Yap, Kok Kiong, Kobayashi, Masayoshi, Sherwood, Rob, Huang, Te Yuan, Chan, Michael, Handigol, Nikhil, and McKeown, Nick: *Openroads: Empowering research in mobile networks.* SIGCOMM Comput. Commun. Rev., 40(1), January 2010, ISSN 0146-4833. http://doi.acm.org/10.1145/1672308.1672331.

[27] Yap, Kok Kiong, Sherwood, Rob, Kobayashi, Masayoshi, Huang, Te Yuan, Chan, Michael, Handigol, Nikhil, McKeown, Nick, and Parulkar, Guru: *Blueprint for introducing innovation into wireless mobile networks.* In *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '10, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0199-2. http://doi.acm.org/10.1145/1851399.1851404.