باسمه تعالی

تکلیف سری سوم درس شبکه های کامپیوتری۲

سارا برادران (شماره دانشجویی: ۹۶۲۴۱۹۳)

Floodlight

This controller is based on the Beacon controller from Stanford University and works with physical and virtual OpenFlow switches. Some of the features of this controller are as follows: apache-licensed, Java-based (non-OSGI), modular, event-driven, asynchronous application framework, thread-based, and using a synchronized lock. Components of this controller are topology management, use for discovery of both OpenFlow and nonOpenFlow endpoints (LLDP), device management including MAC and IP tracking, path computation, infrastructure for web access which is used for management, counter store used for OpenFlow, BigDB (NoSQL, Cassandra-based database) and Quantum plug-in that causes interoperation with element agents supporting OpenFlow. These components are loadable services; and Floodlight Provider is the core module that handles inputs and outputs receives from and sends to switches. This module also translates OpenFlow messages into Floodlight events. There are REST APIs in controller for getting and setting the controller's state, event notification systems, and passing emitted events by Java Event Listeners. Floodlight also has sample applications including learning switch, hub application and static flow pusher, Firewall and load balancer.

OpenDaylight

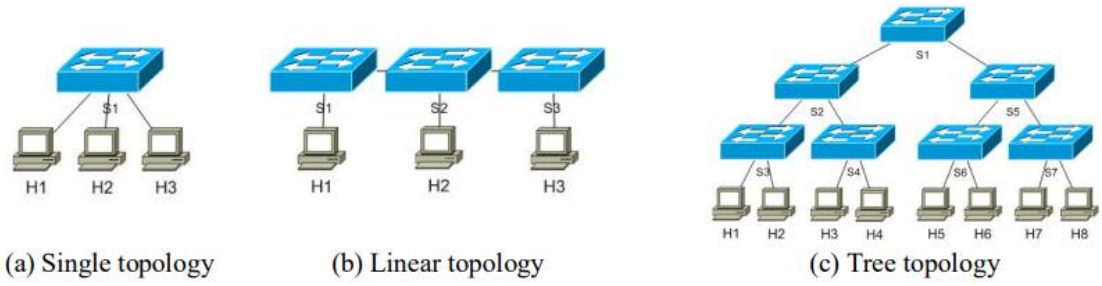
In 2013, the OpenDaylight Project consortium was formed as a Linux Foundation project. This controller creates a de facto northbound API standard so that different southbound protocols, like OpenFlow, I2RS, and NETCONF can be programmed. Some of the features of this controller are as follows: Java-based (OSGI), modular, pluggable and supporting of multiple southbound protocols. OpenDaylight supports the programming of a bidirectional REST and OSGI framework that supports applications running in the same address space as the controller. Internal and external requests for services will be mapped by a service abstraction layer to the appropriate southbound plugin and a basic service abstraction that higher-level services are built upon, will be provided; this feature depends on the capabilities of the plug-ins. Topology abstraction and discovery, PCE-P (and CSPF), OpenFlow, I2RS (as it evolves), and NETCONF are other built-in services within in OpenDaylight.

Table 1: A comparison of Floodlight and OpenDaylight [8, 9, 12]

Feature	Floodlight	OpenDaylight
Developer	Big Switch Networks	Linux Foundation
Supporters	Big Switch Networks	Cisco, HP, IBM , Juniper, VMWare, etc.
Written language	Java	Java
Supporting language	Java, Python and any language supports Rest API	Java
REST API	Yes	Yes
OpenStack networking (Quantum)	Yes	Yes
TLS supporting	Yes	Yes
Open-source	Yes	Yes
OF version	Full support for 1.0 and 1.3, experimental support for OpenFlow 1.1 and 1.2	1.0 , 1.3
User interface	web, Java	Web
Interfaces	southbound (OpenFlow), northbound (Java, REST)	southbound (OpenFlow and other SB protocols), northbound (Java RPC, REST)
Virtualization	Mininet, OpenVswitch	Mininet, OpenVswitch
Platform	Linux, Mac, Windows	Linux, Windows
Active community	Yes	Yes
Age	4 years	3 years
Documentation	Good (documentations exist in official website or other developers sites)	Medium
Mailing list activity	Very high	Medium
Handling mixed none-OpenFlow and OpenFlow networks	Yes	Yes
Installation	Very easy	Easy
Loop supporting	Topologies	Topologies and OF islands

Mininet

Mininet is a network emulator, which has been used by developers, teachers, and researchers to create virtual networks and SDN implementation. It was developed by the Mininet Team to run controllers, switches, and hosts in a virtual network on a single machine. The default topology in Mininet consists of an OpenFlow kernel switch connected to two hosts and an OpenFlow controller. Hosts on Mininet are able to run underlying Linux and file system commands. For example, “ipref” parses bandwidth between a client and server and “topo” makes topologies with the Python API for custom virtual networks. There are also three predefined common topologies in Mininet which are presented in Fig. 1: single, linear, and tree. A single topology has 1 controller, 1 switch, and a definable number of hosts. A linear topology has serial connections with definable number of switches and hosts. A tree topology consists of multiple topology levels with a definable number of levels (depth and fan-out). Remote controllers also can be used in Mininet. Therefore, a virtual network will be connecting to any remote controller in VM, local machine, or anywhere else. Some other advantages of using Mininet are system-level regression test supporting, complex topology testing and CLI which is topology-aware and OpenFlow-aware. Developed code for an OpenFlow controller, modified switch, or host on Mininet, can move to a real system with minimal deployment. So a network design in Mininet can usually move directly to hardware switches for line-rate packet forwarding.



(a) Single topology (b) Linear topology (c) Tree topology

Fig. 1: Three common topologies in Mininet.

installation

1.Use our Ubuntu package!

To install Mininet itself (i.e. 'mn' and the Python API) on Ubuntu 16.04+:

```
sudo apt-get install mininet
```

Note: this may install an older version of Mininet which may not support Python 3. If you would like the latest version of Mininet, consider installing from source as described in the next section.

2.Native installation from source

If you are running Ubuntu, Debian, or Fedora, you may be able to use our handy 'install.sh' script, which is in 'util/'. Please read the following sections first.

2.1. Obtaining the Mininet source code

If you're reading this, you've probably already done so, but the command to download the Mininet source code is:

```
git clone git://github.com/mininet/mininet.git
```

Note that the above git command will check out the latest and greatest Mininet (which we recommend!) If you want to run the last tagged/released version of Mininet, you can look at the release tags using

```
cd mininet
git tag
git checkout <release tag>
```

where <release tag> is the release you want to check out.

2.1.1 *CAUTION: USE AT YOUR OWN RISK!*

'install.sh' can be a bit intrusive and may possibly damage your OS and/or home directory, by creating/modifying several directories such as 'mininet', 'openflow', 'oftest', 'pox', etc.. We recommend trying it in a VM before trying it on a system you use from day to day.

Although we hope it won't do anything completely terrible, you may want to look at the script before you run it, and you should make sure your system and home directory are backed up just in case!

You can change the directory where the dependencies are installed using the -s <directory> flag.

```
util/install.sh -s <directory> ...
```

2.1.2 Running 'install.sh'

Installing a "minimal" version of Mininet with Open vSwitch should be reasonably non-perturbing since it should not create directories for other tools:

```
util/install.sh -nv
```

Note this will not install a controller, so you will have to either install your own controller, or use a switch such OVSBridge that does not require a controller:

```
sudo mn --switch ovsbr --test pingall
```

To install Mininet itself, the OpenFlow reference controller, and Open vSwitch, you may use:

```
util/install.sh -fnv
```

This should be reasonably quick, and the following command should work after the installation:

```
sudo mn --test pingall
```