MỤC LỤC

[CHƯƠNG I : Tổng quan về mạng SDN và Opendaylight 1](#_Toc90457421)

[1. Giới thiệu về mạng SDN 1](#_Toc90457422)

[2. Giới thiệu về Opendaylight 2](#_Toc90457423)

[3. Giới thiệu về mininet 3](#_Toc90457424)

[CHƯƠNG II : Xây dựng ứng dụng định tuyến trong mạng SDN sử dụng Mininet và Opendaylight 4](#_Toc90457425)

[1. Cài đặt Mininet 4](#_Toc90457426)

[2. Cài đặt Opendaylight 5](#_Toc90457427)

[3. Mô hình mạng SDN 7](#_Toc90457428)

[4. Ứng dụng định tuyến giữa các nút mạng trong SDN 11](#_Toc90457429)

[TÀI LIỆU THAM KHẢO 18](#_Toc90457430)

# CHƯƠNG I : Tổng quan về mạng SDN và Opendaylight

## Giới thiệu về mạng SDN

Kiến trúc của SDN gồm 3 lớp riêng biệt: lớp ứng dụng, lớp điều khiển, và lớp cơ sở hạ tầng (lớp chuyển tiếp). Trong đó:

Lớp ứng dụng: là các ứng dụng kinh doanh được triển khai trên mạng, được kết nối tới lớp điều khiển thông qua các API, cung cấp khả năng cho phép lớp ứng dụng lập trình lại (cấu hình lại) mạng (điều chỉnh các tham số trễ, băng thông, định tuyến, …) thông qua lớp điều khiển.

Lớp ứng dụng: là các ứng dụng kinh doanh được triển khai trên mạng, được kết nối tới lớp điều khiển thông qua các API, cung cấp khả năng cho phép lớp ứng dụng lập trình lại (cấu hình lại) mạng (điều chỉnh các tham số trễ, băng thông, định tuyến, …) thông qua lớp điều khiển.Và công việc ở trong bài tập này là lập trình chương trình đóng vai trò là Application thuộc lớp ứng dụng để định tuyến giữa các nút mạng

Lớp điều khiển: là nơi tập trung các bộ điều khiển thực hiện việc điều khiển cấu hình mạng theo các yêu cầu từ lớp ứng dụng và khả năng của mạng. Các bộ điều khiển này có thể là các phần mềm được lập trình. Ngoài ra để truyền thông điều khiển lớp cơ sở hạ tầng, lớp điều khiển sử dụng các cơ chế như OpenFlow, ONOS, 176 ForCES, PCEP, NETCONF, SNMP hoặc thông qua các cơ chế riêng biệt.Controller được sử dụng trong bài tập này là Opendaylight

Lớp cơ sở hạ tầng: là các thiết bị mạng thực tế (vật lý hay ảo hóa) thực hiện việc chuyển tiếp gói tin theo sự điều khiển của lớp điểu khiển. Một thiết bị mạng có thể hoạt động theo sự điều khiển của nhiều bộ điều khiển khác nhau, điều này giúp tăng cường khả năng ảo hóa của mạng.Việc xây dựng cơ sở hạ tầng là các switch được mô phỏng qua ứng dụng mininet

Diagram

Description automatically generated

## Giới thiệu về Opendaylight

OpenDaylight là phần mềm mã nguồn mở dành cho Software Defined Networking (SDN) sử dụng giao thức mở cung cấp khả năng kiểm soát tập trung, có khả năng lập trình được và theo dõi các thiết bị mạng. Giống như nhiều SDN Controllers khác, OpenDaylight hỗ trợ OpenFlow, cũng như cung cấp các giải pháp mạng khác sẵn sàng để cài đặt khi có yêu cầu.

OpenDaylight cung cấp giao diện cho phép kết nối các thiết bị mạng nhanh chóng và thông minh để tối ưu hiệu năng mạng. OpenDaylight Controller cung cấp northbound APIs, hay các giao diện hỗ trợ việc giao tiếp giữa các ứng dụng bên ngoài với nó. Các ứng dụng này có thể sử dụng controller để thu thập thông tin về mạng, chạy các thuật toán để kiểm soát, phân tích, sau đó sử dụng OpenDaylight Controller tạo các rules mới cho mạng.

OpenDaylight Controller viết bằng ngôn ngữ Java, có nghĩa là có thể sử dụng OpenDaylight Controller trên bất kì môi trường nào hỗ trợ Java. Tuy nhiên để đạt hiệu năng tốt nhất, OpenDaylight nên chạy trên môi trường Linux hỗ trợ JVM tối thiểu 1.7.

Diagram

Description automatically generated

## Giới thiệu về mininet

Mininet là một công cụ giả lập mạng, bao gồm tập hợp các hosts đầu cuối, các switches, routers và các liên kết trên một Linux kernel. Mininet sử dụng công nghệ ảo hóa (ở mức đơn giản) để tạo nên hệ thống mạng hoàn chỉnh, chạy chung trên cùng một kernel, hệ thống và user code.

Các host ảo, switch, liên kết và các controller trên mininet là các thực thể thực sự, được giả lập dưới dạng phần mềm thay vì phần cứng. Một host mininet có thể thực hiện ssh vào đó, chạy bất kì phần mềm nào đã cài trên hệ thống linux (môi trường mà mininet đang chạy). Các phần mềm này có thể gửi gói tin thông các ethernet interface của mininet với tốc độ liên kết và trễ đặt trước.

Mininet cho phép tạo topo mạng nhanh chóng, tùy chỉnh được topo mạng, chạy được các phần mềm thực sự như web servers, TCP monitoring, Wireshark; tùy chỉnh được việc chuyển tiếp gói tin. Mininet cũng dễ dàng sử dụng và không yêu cầu cấu hình đặc biệt gì về phần cứng để chạy: mininet có thể cài trên laptop, server, VM, cloud (linux).

# CHƯƠNG II : Xây dựng ứng dụng định tuyến trong mạng SDN sử dụng Mininet và Opendaylight

## Cài đặt Mininet

Nhóm sẽ sử dụng VMWare (hoặc VirtualBox) để tạo một máy ảo triển khai MininetVM trên đó

Trước tiên, ta thêm các card mạng Wifi của máy local host vào Virtual Network Editor dưới dạng Bridge để các máy ảo có thể kết nối đến mạng internet và Mininet kết nối tới controller là Opendaylight.

Graphical user interface, text, application

Description automatically generated

Chỉnh Network Adapter của 2 máy ảo thành Bridge

Tải file image của Mininet tại địa chỉ : [Releases · mininet/mininet · GitHub](https://github.com/mininet/mininet/releases). Và chọn phiên bản Mininet 2.3.0 dành cho Ubuntu Server 18.04

Sau đó, giải nén tệp zip của Mininet và nhập image bên trong thư mục vào VMWare

Tệp image trên đã được cấu hình sẵn Mininet , do đó chúng ta có thể sử dụng các câu lệnh triển khai các mạng ảo ngay mà không cần cài đặt các thư viện gì thêm

Trước tiên, chúng ta cần tạo một cấu trúc mạng ảo để thực hiện các công việc cấu hình. Ví dụ, ta có thể chạy câu lệnh sau trên giao diện dòng lệnh để tạo ra cấu trúc mạng hình cây 4 cấp

* $ sudo mn --topo tree,4 --mac --controller=remote,ip=[Opendaylight-IP] --switch ovs,protocols=OpenFlow13

Và kết quả hiển thị trên Opendaylight UI Topology:

Chart, radar chart

Description automatically generated

Khi đó, chương trình sẽ vào chế độ dòng lệnh của Mininet (mininet> ) để cho phép chúng ta cấu hình và thực hiện các thao tác giao tiếp với mạng

Một số câu lệnh điều khiển phổ biến trên Mininet như sau

* mininet> nodes : liệt kê thông tin các nút trong cấu trúc mạng
* mininet> links : liệt kê thông tin liên quan tới các liên kết trong cấu trúc mạng
* mininet> pingall : cho phép kiểm tra liên kết (ping) giữa tất cả các host trong mạng với nhau
* mininet> h1 ping h2 : cho phép kiểm tra liên kết giữa 2 host trong mạng với nhau
* mininet> dpctl dump-flows -O OpenFlow13 : xem toàn bộ bảng flow hiện có của tất cả các switch trong mạng
* mininet> sh ovs-ofctl dump-flows -O OpenFlow13 s1 : xem bảng flow của riêng switch s1
* mininet> dpctl del-flows -O OpenFlow13 : xóa toàn bộ bảng flow hiện có của tất cả các switch trong mạng
* mininet> sh ovs-ofctl del-flows -O OpenFlow13 s1 : xóa bảng flow của riêng switch s1
* mininet> sh ovs-ofctl add-flow -O OpenFlow13 s1 inport=1,actions=output:2 : tạo 1 flow cho phép các luồng liên kết đi tới cổng 1 của switch s1 sẽ được chuyển mạch đến cổng ra là cổng số 2
* mininet> exit : thoát khỏi chương trình Mininet

## Cài đặt Opendaylight

Trước tiên, ta cần cài đặt Java 8 (hoặc 11 tùy vào phiên bản) để tạo môi trường chạy cho Opendaylight

* $ sudo apt-get -y install openjdk-8-jre
* $ sudo update-alternatives --config java
* $ echo ‘export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre’ >> ~./bashrc
* $ source .bashrc

Ta vào trang chủ của Opendaylight để tải về bộ cài đặt của nó : <https://docs.opendaylight.org/en/latest/downloads.html>. Có rất nhiều phiên bản đã được phân phối, tuy nhiên, trong bài này, nhóm sử dụng phiên Carbon 0.6.4 để chạy ứng dụng

* $ wget <https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.6.4-Carbon/distribution-karaf-0.6.4-Carbon.tar.gz>

Sau đó, ta giải nén ra thư mục để sử dụng

* $ tar -xvzf distribution-karaf-0.6.4-Carbon

Khởi động Opendaylight bằng cách chạy tệp thực thi karaf trong thư mục bin và sẽ xuất hiện giao diện như sau

* $ cd distribution-karaf-0.6.4-Carbon
* $./bin/karaf

Text

Description automatically generated

Một số câu lệnh điều khiển phổ biến khi sử dụng Opendaylight

* feature:list -i : chỉ liệt kê các tính năng đã cài đặt
* feature:info <feature> : thông tin tính năng
* feature:install <feature> : cài đặt tính năng
* feature:uninstall <feature> : gỡ bỏ tính năng

Cài đặt các gói chức năng (feature) cần thiết cho việc sử dụng ứng dụng giao tiếp với Opendaylight bằng câu lệnh như sau:

* $ feature:install odl-restconf odl-l2switch-switch odl-dlux-core odl-dluxapps-nodes odl-dluxapps-topology odl-dluxapps-yangui odl-dluxapps-yangvisualizer odl-dluxapps-yangman

Khi đó, ta truy nhập vào địa chỉ của máy nhận Controller với cổng 8181 sẽ xuất hiện giao diện trình duyệt Opendaylight với các ứng dụng hỗ trợ thao tác cấu hình mạng. Nó sẽ yêu cầu đăng nhập trước khi sử dụng các dịch vụ bên trong. Tên đăng nhập và mật khẩu mặc định đều là admin

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

## Mô hình mạng SDN

Mô hình mạng SDN mô phỏng mà nhóm sẽ sử dụng trong quá trình thực hiện các thao tác cấu hình có cấu trúc gồm 10 switch và 10 host được liên kết với nhau. Hình bên dưới mô tả cấu trúc mạng của nhóm

Chart, diagram

Description automatically generated

Để tạo ra được cấu trúc mạng như này, ta sẽ định nghĩa một topo mạng với các nút và liên kết tương ứng trên một tệp được lập trình bằng Python và gửi cho Mininet để nó chạy mô phỏng.

* from mininet.topo import Topo
* *class* MyTopo(*Topo*):
* "Simple topology example."
* *def* \_\_init\_\_(*self*):
* "Create custom topo."
* Topo.\_\_init\_\_(self)
* # Add hosts and switches
* h1 = self.addHost('h1')
* h2 = self.addHost('h2')
* h3 = self.addHost('h3')
* h4 = self.addHost('h4')
* h5 = self.addHost('h5')
* h6 = self.addHost('h6')
* h7 = self.addHost('h7')
* h8 = self.addHost('h8')
* h9 = self.addHost('h9')
* h10 = self.addHost('h10')
* # Them cac switch
* s1 = self.addSwitch('s1')
* s2 = self.addSwitch('s2')
* s3 = self.addSwitch('s3')
* s4 = self.addSwitch('s4')
* s5 = self.addSwitch('s5')
* s6 = self.addSwitch('s6')
* s7 = self.addSwitch('s7')
* s8 = self.addSwitch('s8')
* s9 = self.addSwitch('s9')
* s10 = self.addSwitch('s10')
* # Add links
* self.addLink(h1, s1)
* self.addLink(h2, s1)
* self.addLink(h3, s2)
* self.addLink(h4, s3)
* self.addLink(h5, s4)
* self.addLink(h6, s6)
* self.addLink(h7, s7)
* self.addLink(h8, s9)
* self.addLink(h9, s10)
* self.addLink(h10, s10)
* self.addLink(s1, s2)
* self.addLink(s1, s3)
* self.addLink(s1, s5)
* self.addLink(s2, s4)
* self.addLink(s3, s6)
* self.addLink(s4, s5)
* self.addLink(s4, s7)
* self.addLink(s5, s6)
* self.addLink(s5, s8)
* self.addLink(s6, s9)
* self.addLink(s7, s8)
* self.addLink(s7, s10)
* self.addLink(s8, s9)
* self.addLink(s8, s10)
* self.addLink(s9, s10)
* self.addLink(s1, s7)
* self.addLink(s5, s9)
* self.addLink(s3, s5)

* topos = {'sdntopo': (*lambda*: MyTopo())}

Khởi động Mininet với controller là Opendaylight ở địa chỉ IP của máy chạy Opendaylight bằng câu lệnh bên dưới. Phần tham số dành cho –topo chính là tên của mô hình mạng được mô tả trong tệp topo.py ở trên. –mac với ý nghìa là tạo ra các host với địa chỉ MAC đơn giản bắt đầu từ 00:00:00:00:00:01

* $ sudo mn --custom topo.py --topo sdntopo –mac --controller=remote,ip=192.168.56.1 --switch ovsk,protocols=OpenFlow13

Sau đó, ta sẽ pingall để kiểm tra các liên kết và xem kết quả trên giao diện trình duyệt của Opendaylight thu được kết quả như sau

* mininet> pingall

Text

Description automatically generated

Chart, radar chart

Description automatically generated

## Ứng dụng định tuyến giữa các nút mạng trong SDN

* 1. Các chức năng của ứng dụng và API tương ứng của Opendaylight
* Lấy thông tin topo

Chức năng này được thực hiện bằng cách sử dụng API lấy thông tin topo ở dưới dưới dạng json do Controller trả về và xử lý gói tin đó để có được thông tin về cấu trúc mạng như các host (tên host, địa chỉ IP, địa chỉ MAC), các switch và liên kết giữa chúng

|  |  |
| --- | --- |
| URL | http://192.168.56.1:8181/restconf/operational/network-topology:network-topology/ |
| Method | GET |

* Kiểm tra bảng flow của một switch

|  |  |
| --- | --- |
| URL | http://192.168.56.1:8181/restconf/ operational /opendaylight-inventory:nodes/node/{node-id}/table/{table-id} |
| Method | GET |
| Param | * node-id : id của switch muốn lấy thông tin bảng flow. VD : openflow:1 * table-id : id của bảng flow bên trong switch. VD : 0 |

* Định tuyến :

Chương trình sẽ yêu cầu nhập vào tên host nguồn và tên host đích. Sau đó, dựa trên cấu trúc mạng lấy được ở trên, chương trình sẽ tìm đường đi giữa chúng thông qua các liên kết đã lấy được từ topo mạng. Việc định tuyến cũng có thể thực hiện thủ công bằng cách nhập tuyến đường chứa thứ tự các switch cần đi qua và chương trình sẽ kiểm tra xem tuyến đường có hợp lệ hay không

* Thêm flow mới vào switch

Ứng dụng sẽ lấy các thông tin như là id của switch, cổng vào, cổng ra để tạo gói tin json chứa các thông tin đó và PUT lên Opendaylight để chúng điều khiển các switch

|  |  |
| --- | --- |
| URL | http://192.168.56.1:8181/restconf/config/opendaylight-inventory:nodes/node/{node-id}/table/{table-id}/flow/{flow-id} |
| Method | PUT |
| Param | * node-id : id của switch muốn lấy thông tin bảng flow. VD : openflow:1 * table-id : id của bảng flow bên trong switch. VD : 0 * flow-id: id của flow mới cần thêm. VD : 10 |
| Body | {      "flow-node-inventory:flow": [          {              "id": "91",              "priority": 2,              "table\_id": 0,              "match": {                  "in-port": "3"              },              "cookie": 0,              "instructions": {                  "instruction": [                      {                          "order": 0,                          "apply-actions": {                              "action": [                                  {                                      "order": 0,                                      "output-action": {                                          "max-length": 65535,                                          "output-node-connector": "1"                                      }                                  }                              ]                          }                      }                  ]              },              "idle-timeout": 0          }      ]  } |

* Xóa flow của một switch

Chức năng này cho phép xóa toàn bộ bảng flow của switch để dễ dàng tự cài đặt và định tuyến bằng chương trình ứng dụng

|  |  |
| --- | --- |
| URL | http://192.168.56.1:8181/restconf/config/opendaylight-inventory:nodes/node/{node-id}/table/{table-id}/flow/{flow-id} |
| Method | DELETE |
| Param | * node-id : id của switch muốn lấy thông tin bảng flow. VD : openflow:1 * table-id : id của bảng flow bên trong switch. VD : 0 * flow-id: id của flow mới cần thêm. VD : 10 |

* 1. Kết quả đạt được
* Lấy thông tin các Host :

Text

Description automatically generated

* Lấy thông tin các Switch

Graphical user interface

Description automatically generated

* Định tuyến từ Host1 đến Host10 bằng Dijkstra. Trước khi định tuyến, bảng flow sẽ bị xóa nên không thể ping được từ Host1 đến Host10:

Text

Description automatically generated

* Định tuyến bằng ứng dụng của nhóm :

Text

Description automatically generated

* Sau khi thực hiện định tuyến thì ta có thể ping từ Host1 đến Host10 :

A screenshot of a computer

Description automatically generated with medium confidence

* Thêm flow vào cho switch 1 tại cổng vào 1 và cổng ra 2

Text

Description automatically generated

* Xóa bảng flow của tất cả các switch

Text

Description automatically generated

* Hỗ trợ người dùng trong việc sử dụng các câu lệnh

Text

Description automatically generated with medium confidence

# TÀI LIỆU THAM KHẢO

[1]. Mạng SDN, PGS.TS Trương Thị Diệu Linh, [Microsoft PowerPoint - 05-SDN network.pptx (hust.edu.vn)](https://users.soict.hust.edu.vn/linhtd/courses/AdvancedNet/05-SDN%20Network.pdf)

[2]. <https://docs.opendaylight.org/en/stable-carbon/user-guide/openflow-plugin-project-user-guide.html>

[3]. <https://docs.opendaylight.org/projects/openflowplugin/en/latest/users/flow-examples.html>

[3]. <https://docs.opendaylight.org/projects/openflowplugin/en/latest/users/operation.html>