

SDN 운용

목차

I 프로젝트 소개

1. 프로젝트명
2. 개요
3. 목적
4. 구축방법
5. 측정

II 프로젝트 구축1

1. Pox-dhcp 설정하기
2. Floodlight controller 설정 및 연동하기
3. Mininet 설정 및 구동하기
4. Wireshark로 결과 측정하기

III 프로젝트 구축2

1. Floodlight controller 설정하기
2. Mininet 설정 및 구동하기
3. Wireshark로 결과 측정하기

1. 프로젝트소개

프로젝트명

- Pox-dhcp와 Floodlight를 혼용한 SDN 구성
- Miniedit과 Floodlight를 혼용한 SDN 구성

개요

- SDN 컨트롤러인 POX, Floodlight Controller를 통해 Mininet 상에 SDN 기반의 네트워크로 구성하고, L3 스위칭/DHCP/Loadbalancing/Firewall/ACL 시험이 가능하도록 한다.
- SDN 기반의 네트워크 망을 통해서 시험을 시행하며 데이터를 수집하여 Openflow 프로토콜 분석을 시행하여 분석보고서를 작성하도록 한다.

목적

- pox-dhcp 동작 확인 및 측정
- floodlight 를 사용한 ACL/Firewall 구현
- Mininet과 floodlight를 연동하여 동작 확인 및 결과 측정

구축방법(실습1)

- 1단계 : pox Controller에서 pox-DHCP 설정 및 실행 (1번 서버)
- 2단계 : floodlight Controller 실행 (2번 서버)
- 3단계 : Miniedit에서 Pox_DHCP_Floodlight.mn 실행
- 4단계 : 각 서버의 IP와 Port 설정 및 DHCP로 IP 할당
- 5단계 : Floodlight Web에서 측정 및 확인
- 6단계 : Wireshark로 측정되는 결과 확인

(두번째 실습 또한 마찬가지로 pox없이 miniedit과 floodlight을 연동해서 진행함)

구축방법(실습2)

- 1단계 : floodlight Controller 실행 (1번 서버)
- 2단계 : Miniedit에서 miniedit.mn 실행 (1,2,3번 서버)
- 3단계 : Controller 서버와 Remote서버의 IP와 Port 설정 및 DHCP로 IP 할당
- 4단계 : Floodlight에서 ACL과 firewall 시험
- 5단계 : Wireshark로 측정되는 결과 확인

측정

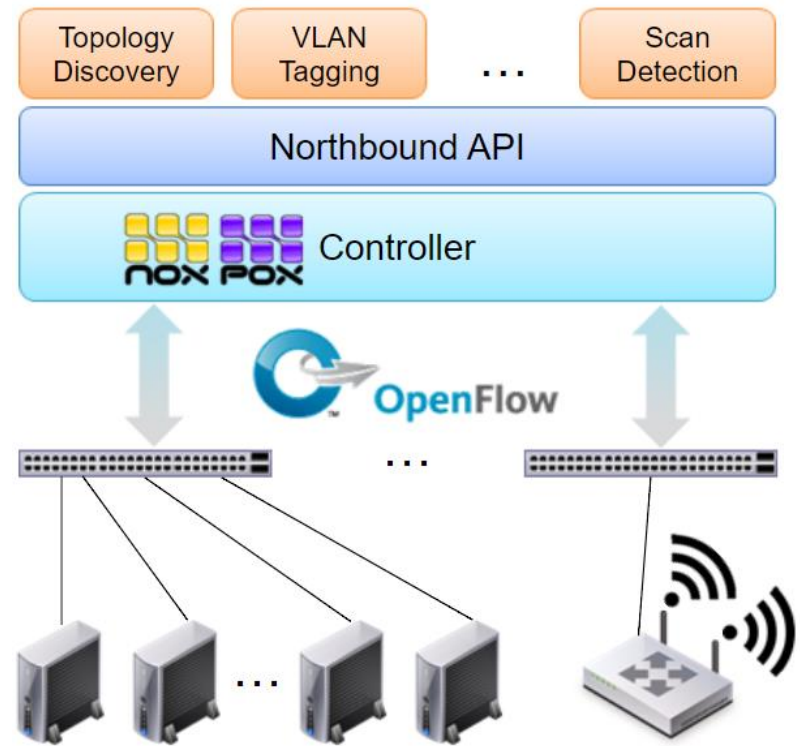
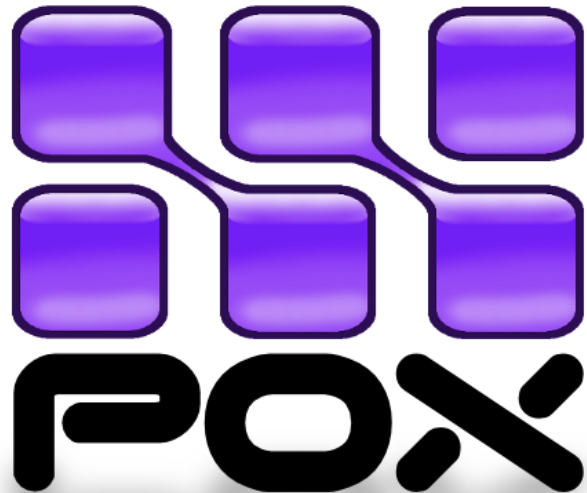
- sdn 스위치와 sdn 컨트롤러 간에 openflow 프로토콜을 수집 및 분석한 결과를 작성
- sdn 스위치와 sdn 컨트롤러 간에 Secure Channel 형성과정 측정 및 분석한 결과를 작성
- sdn 컨트롤러의 dhcp 동작 상태 측정 및 분석한 결과 작성
- sdn 컨트롤러의 flow-entry를 측정하고 분석한 결과를 작성
 - openflow 교재를 참조하여 자세히 기록

1. POX-DHCP 설정

POX Controller로 POX-DHCP 구성

POX Controller 란?

- Python 언어로 구성된 POX Controller는 학교나 연구 기관을 대상으로 하며, 리눅스, MAC, OS, 윈도우에서 설치가 가능한 네트워크 제어 플랫폼이다. 네트워크 관리 및 제어 애플리케이션을 구성할 수 있는 프로그램 인터페이스를 제공하며, 전체 네트워크에 대한 중앙 집중화된 프로그램 모델을 가능하게 한다. OpenFlow v1.3까지 지원하고 있으며 사용방법이 쉬운 것이 장점이다.
- <https://github.com/noxrepo/pox> 에서 다운로드 가능하다.

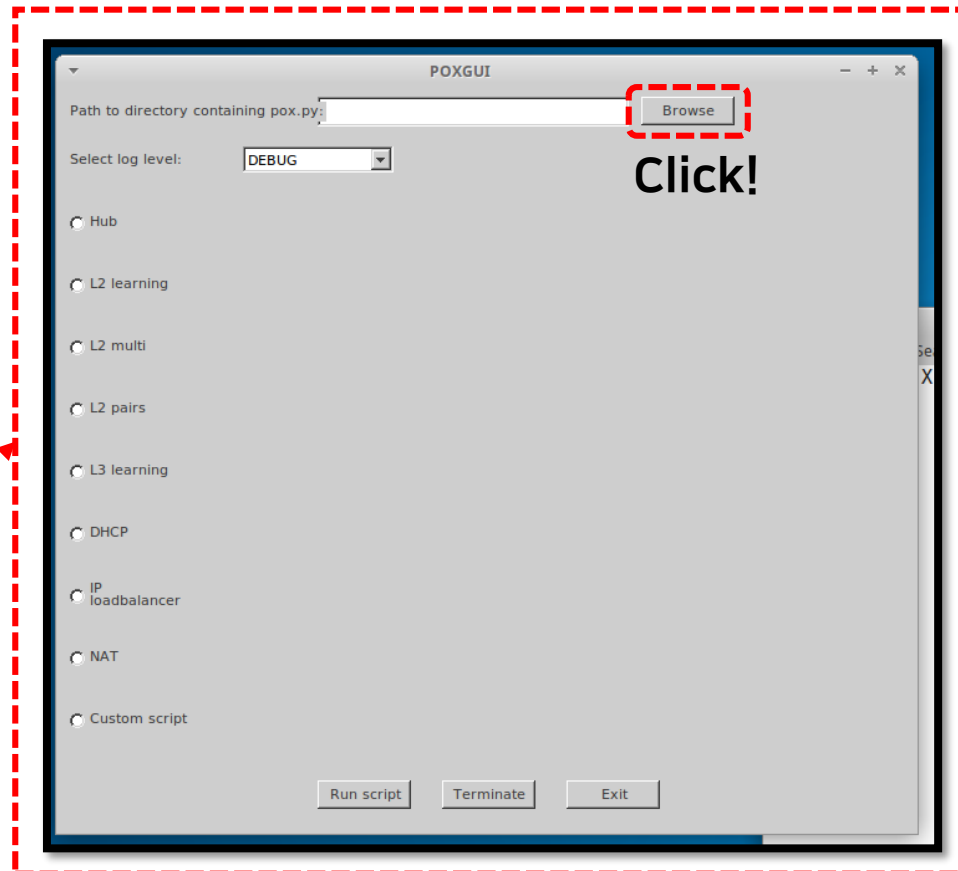
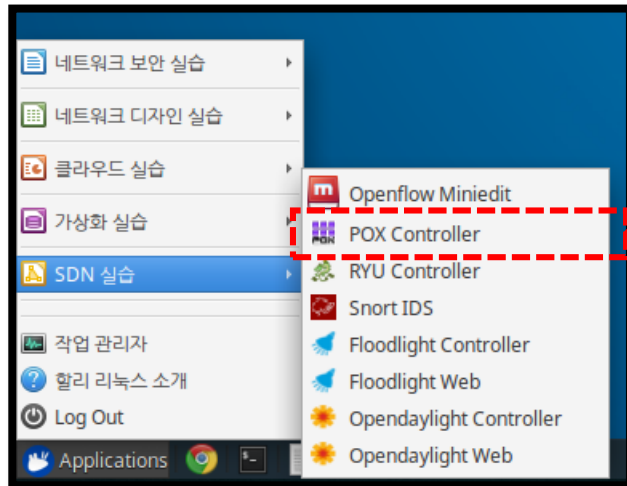


POX-DHCP 설정

1) Pox Controller 열어서 POX-DHCP 설정하기

- 미리 구성한 서버1의 hali-linux 환경에 설치된 POX Controller 열기

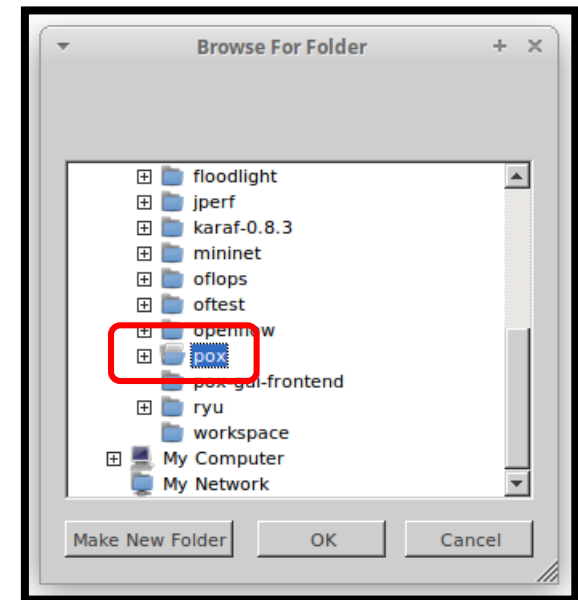
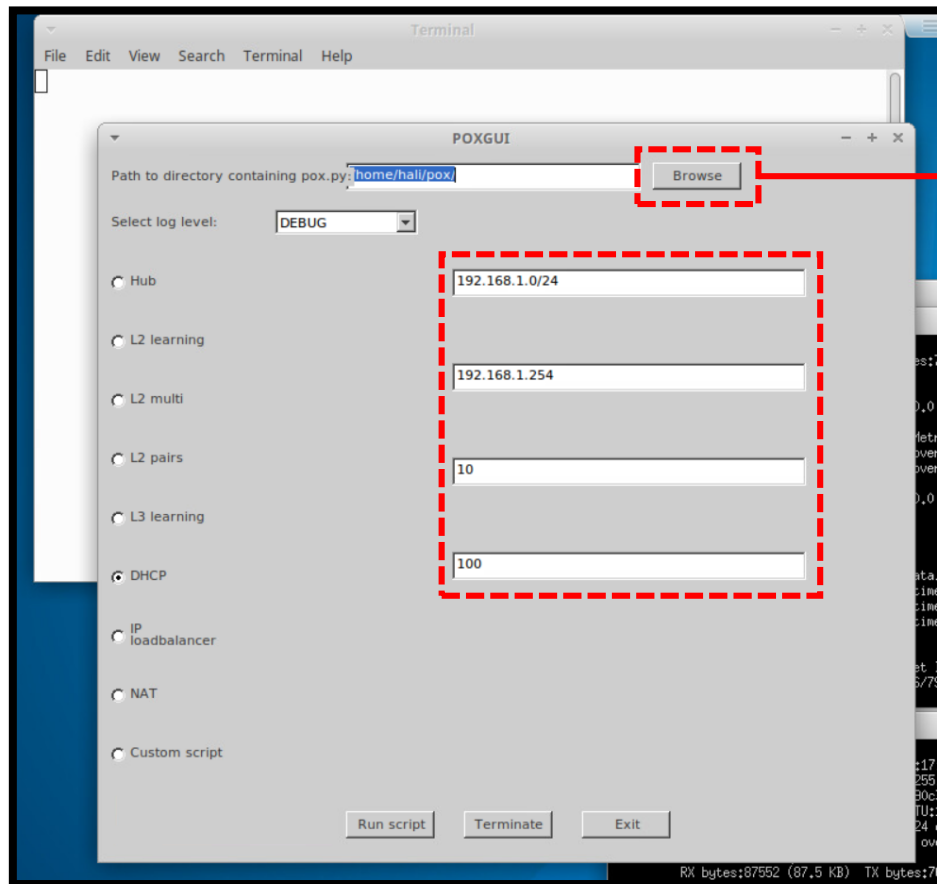
➤ 파일 위치 : Applications > SDN 실습 > POX Controller



POX-DHCP 설정

1) Pox Controller 열어서 POX-DHCP 설정하기

- POX Controller에서 경로 /home/hali/pox로 설정 후에 DHCP를 선택해서 입력하기.
 - 위에서부터 나눌 Subnet 범위/DHCP Server로 쓸 IP, 길이의 시작부터 끝을 가리키고 있음.



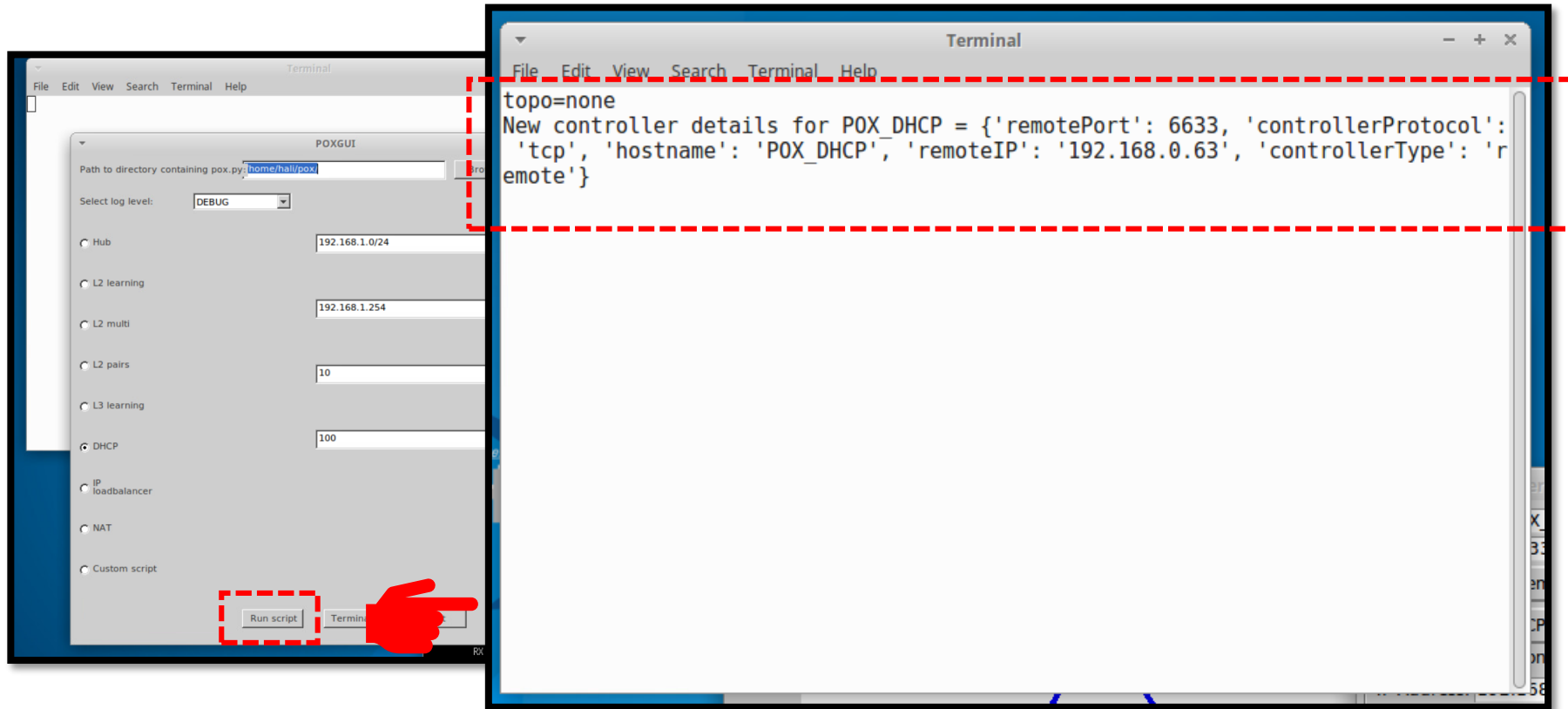
➤ Home/hali/pox

POX-DHCP 설정

2) Pox Controller에서 POX-DHCP 실행하기

- POX Controller에서 run script 눌러 실행하기

- 입력 후에 Run scrip를 누르면 pox가 구동된다.
- 새로운 POX_DHCP 라는 Controller를 인식했음을 알 수 있다.

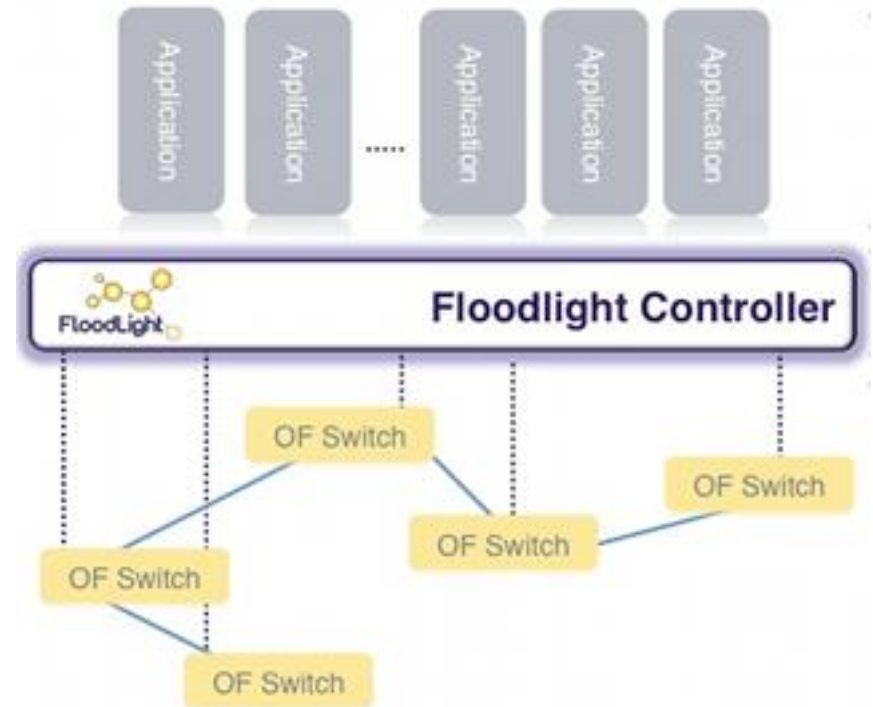


2. Floodlight 연동

Floodlight Controller로 연동시키기

Floodlight 란?

- Floodlight 컨트롤러는 자바 기반 오픈소스 컨트롤러이며 openflow v1.5까지 지원한다. 오픈 커뮤니티로 FAQ 및 지식 교류도 활발히 운영되고 있다. 스탠포드 대학교 출신들이 세운 빅스위치란 회사에서 개발한 SW이다. 본 프로젝트에서는 Floodlight 컨트롤러를 Miniedit과 연동하여 사용할 것이다.
- <https://www.projectfloodlight.org/> 에서 다운가능하다.

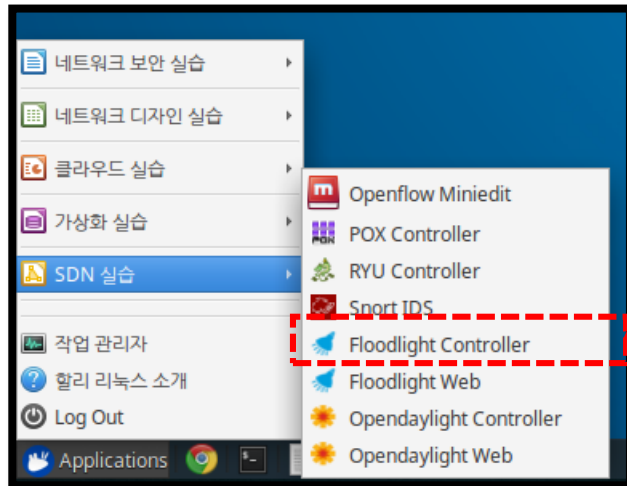


Floodlight Controller 설정

1) Floodlight Controller를 구동한다.

- 미리 구성한 서버2의 ESXi 환경에 설치된 Floodlight Controller를 연다.

- 파일 위치 : Applications > SDN 실습 > Floodlight Controller
- 열면 하단 그림과 같이 터미널 하나가 생기면서 Floodlight Controller가 켜졌음을 보여준다.

A screenshot of a terminal window titled 'Terminal'. It displays the startup logs of the Floodlight Controller. The logs show the controller starting on port 8080, receiving a new switch connection from 192.168.0.114:39704, negotiating the OpenFlow version (OF_13), and binding the switch to the controller. There is an error message about 'OFGGroupModFailedError' but it seems to be a non-fatal warning. The logs also show the topology manager recomputing the topology and the debug server starting on port 6655.

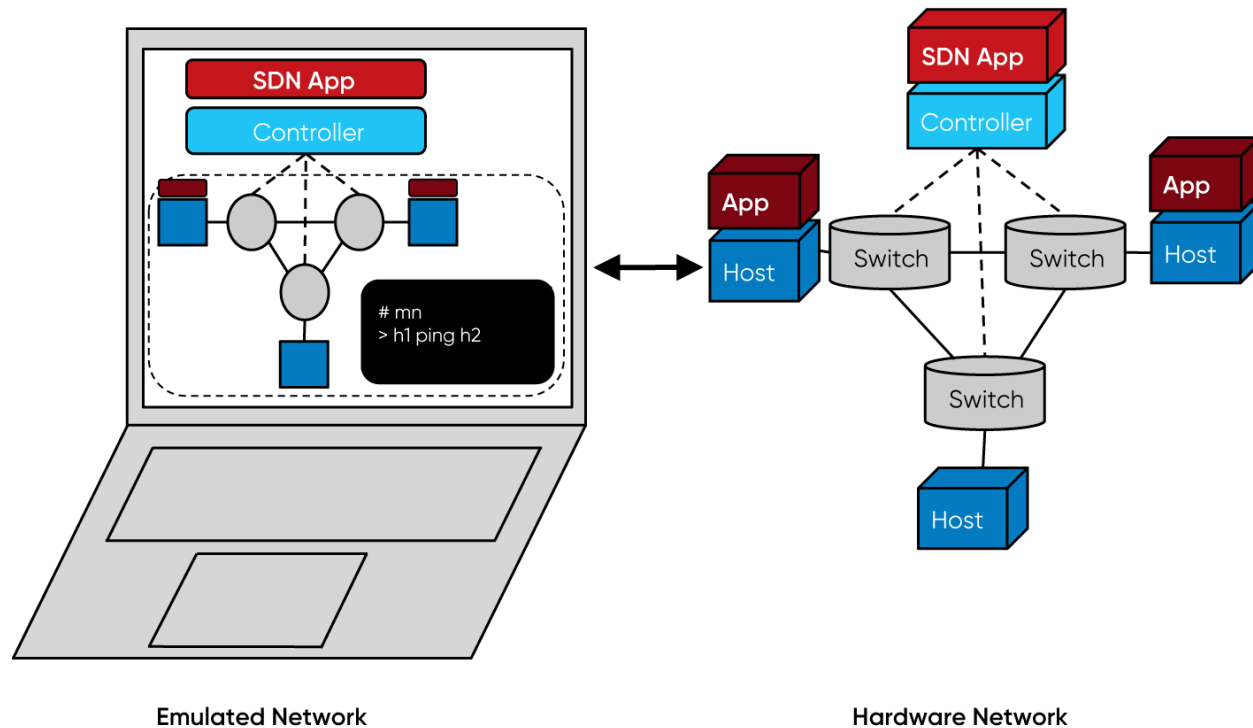
```
server on port 8080
2019-01-28 10:48:31.650 INFO [org.restlet] Starting net.floodlightcontroller.re
stserver.RestApiServer$RestApplication application
2019-01-28 10:48:31.694 INFO [n.f.c.i.OFChannelHandler] New switch connection f
rom /192.168.0.114:39704
2019-01-28 10:48:31.707 INFO [n.f.c.i.OFChannelHandler] Negotiated down to swit
ch OpenFlow version of OF_13 for /192.168.0.114:39704 using lesser hello header
algorithm.
2019-01-28 10:48:31.741 INFO [n.f.c.i.OFSwitchHandshakeHandler] Switch OFSwitch
DPID[00:00:00:00:00:00:10] bound to class class net.floodlightcontroller.cor
e.internal.OFSwitch, description SwitchDescription [manufacturerDescription=Nici
ra, Inc., hardwareDescription=Open vSwitch, softwareDescription=2.5.4, serialNum
ber=None, datapathDescription=s1]
2019-01-28 10:48:31.919 INFO [n.f.c.i.OFSwitchHandshakeHandler] Clearing flow t
ables of 00:00:00:00:00:00:10 on upcoming transition to MASTER.
2019-01-28 10:48:31.985 ERROR [n.f.c.i.OFSwitchHandshakeHandler] OFGroupModFaile
dErrorMsgVer13(xid=12, code=INVALID GROUP, data=OFGGroupDeleteVer13(xid=12, group
Type=INDIRECT, group=all, buckets=[])) from switch OFSwitch DPID[00:00:00:00:
00:00:10] in state net.floodlightcontroller.core.internal.OFSwitchHandshakeHan
dler$MasterState@32644e23
2019-01-28 10:48:32.78 INFO [n.f.t.TopologyManager] Recomputing topology due to
: link-discovery-updates
2019-01-28 10:48:32.997 INFO [n.f.j.JythonServer] Starting DebugServer on :6655
```

3. Miniedit 구동

Miniedit 구동시키기

Miniedit (Mininet Edit GUI Tool) 란?

- Miniedit 는 mininet에서 제공하는 에디터이다. CLI 방식은 불편하기에 Miniedit을 이용하여 간단하고 쉽게 토폴로지를 구성하고 각종 Controller를 연결하여 SDN 실습을 진행할 수 있다. 본 프로젝트에서는 이 Miniedit을 통해 Floodlight Controller와 연동, 혹은 Pox controlle와 연동하여 그 결과를 측정할 것이다.



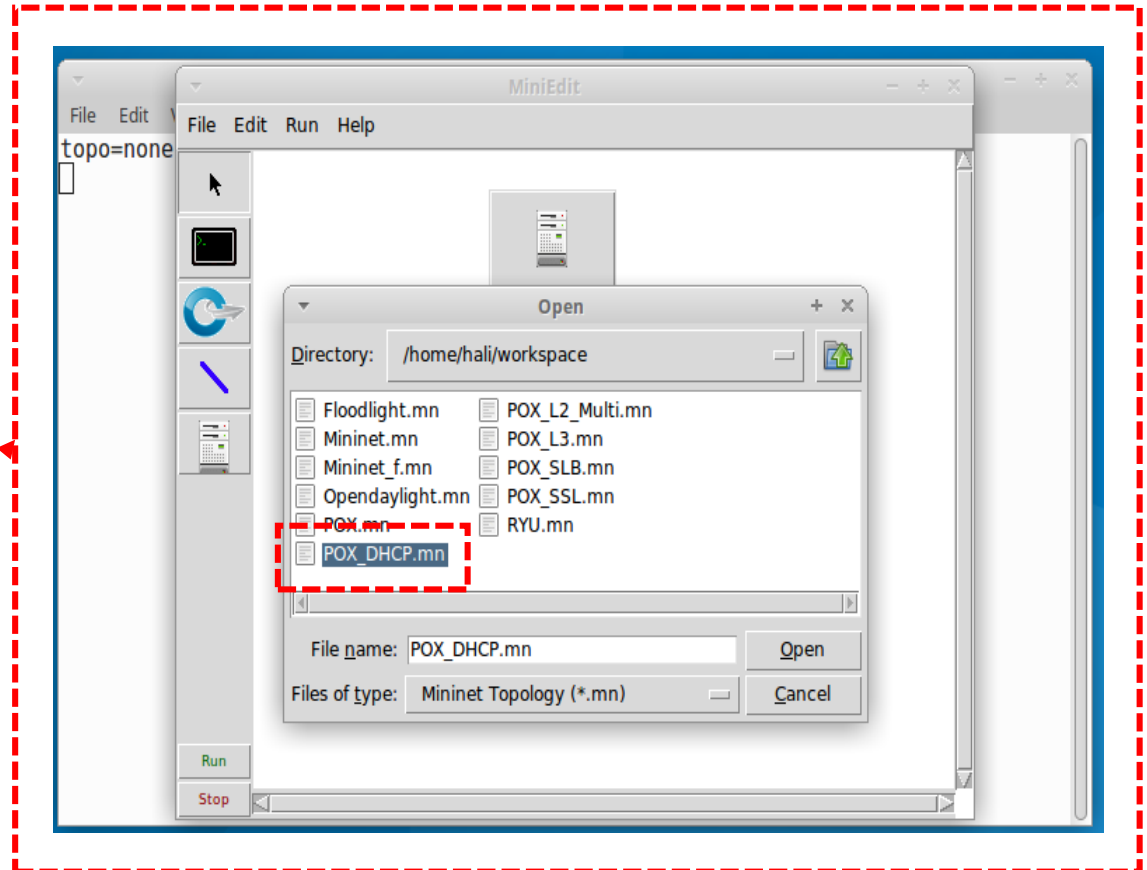
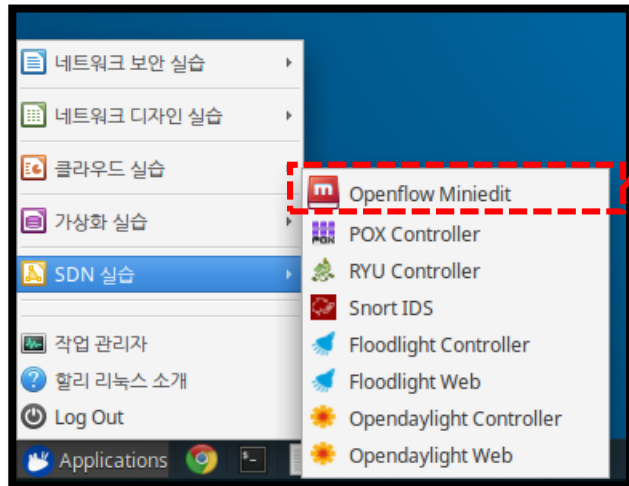
Miniedit 구동 및 Controller 연동하기

1) Pox Controller 와 Miniedit 연동하기

- 미리 구성한 서버1의 hali-linux에 설치된 Openflow Miniedit을 연다. (단, POX Controller가 먼저 켜져야 함)

➤ 파일 위치 : Applications > SDN 실습 > Openflow Miniedit

➤ File > Open > POX_DHCP.mn을 불러온다. (기본 틀이 되는 예시를 가져와서 쓸 것)

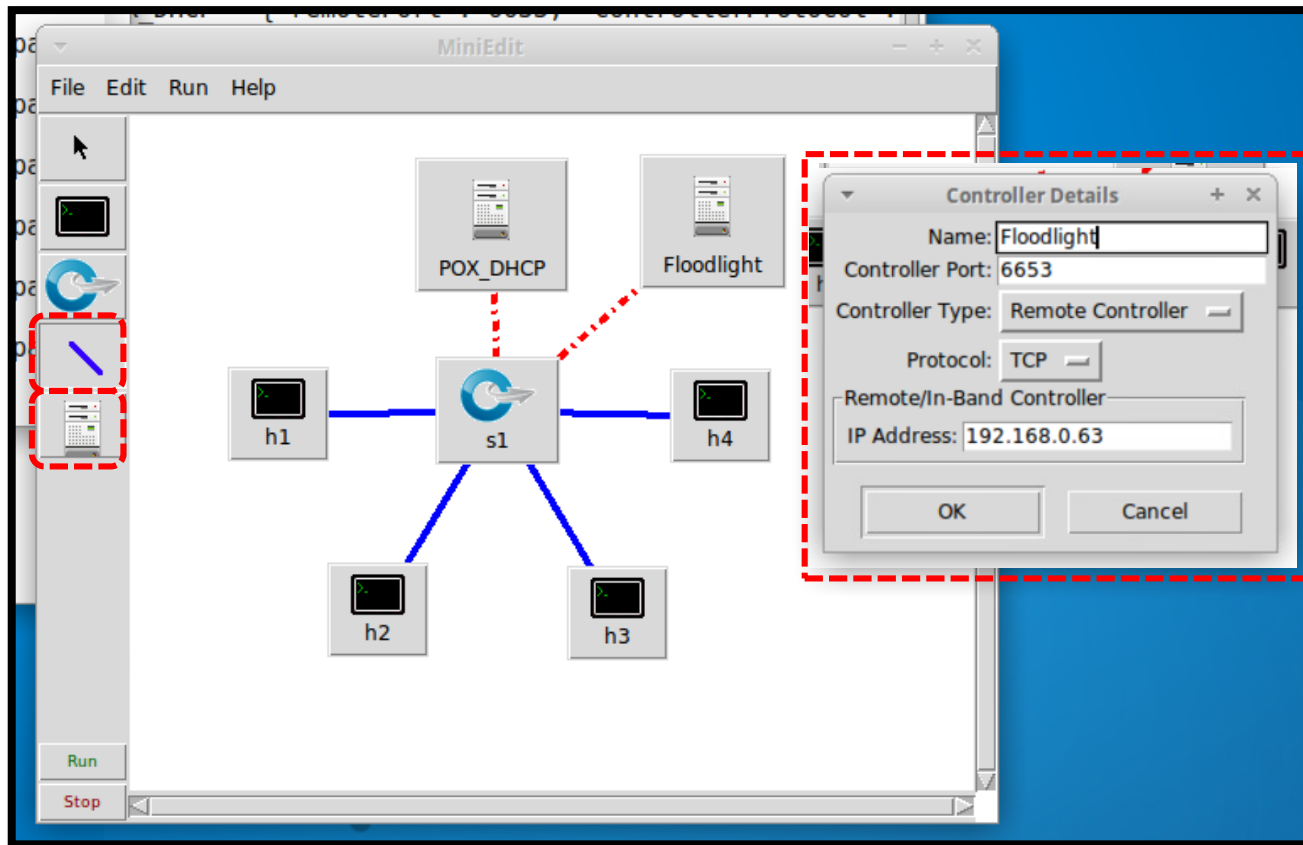


Miniedit 구동 및 Controller 연동하기

1) Pox Controller 와 Miniedit 연동하기

- POX_DHCP.mn 기본틀에서 Floodlight Controller 하나를 더 추가해준다.

- Controller Name : Floodlight / Controller Port : 6653 / IP : 192.168.0.63 / Controller Type: Remote Controller
- Remote Controller : 서버2가 되므로 서버2의 ip를 써준다.

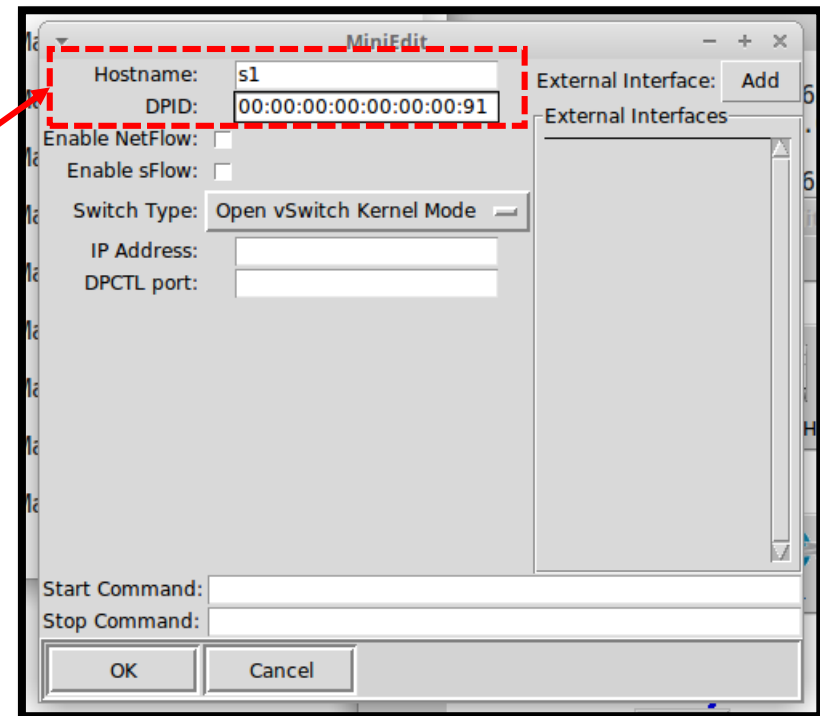
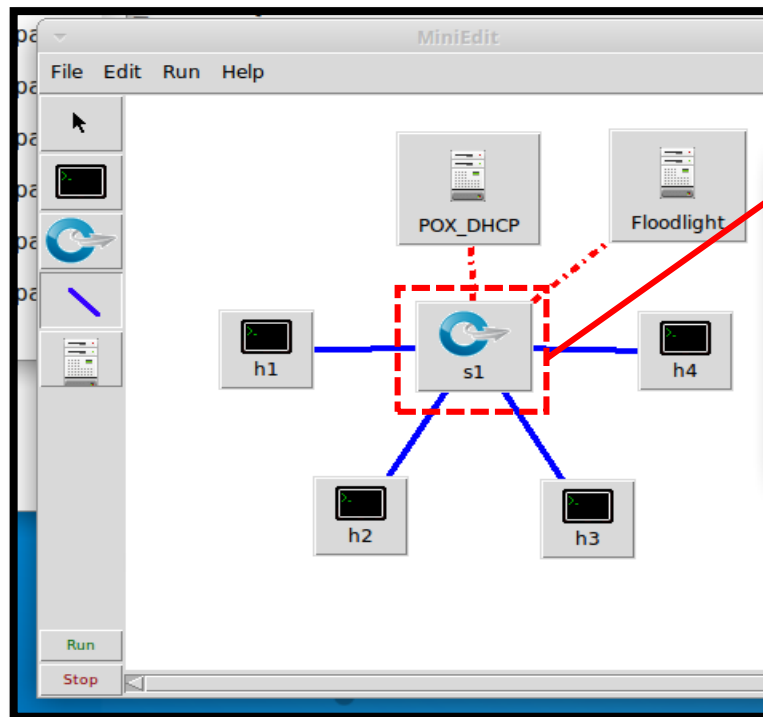


Miniedit 구동 및 Controller 연동하기

1) Pox Controller 와 Miniedit 연동하기

- POX_DHCP.mn s1 (스위치)설정을 서버2의 DPID와 겹치지 않도록 임시로 지정한다. (91로 지정함)

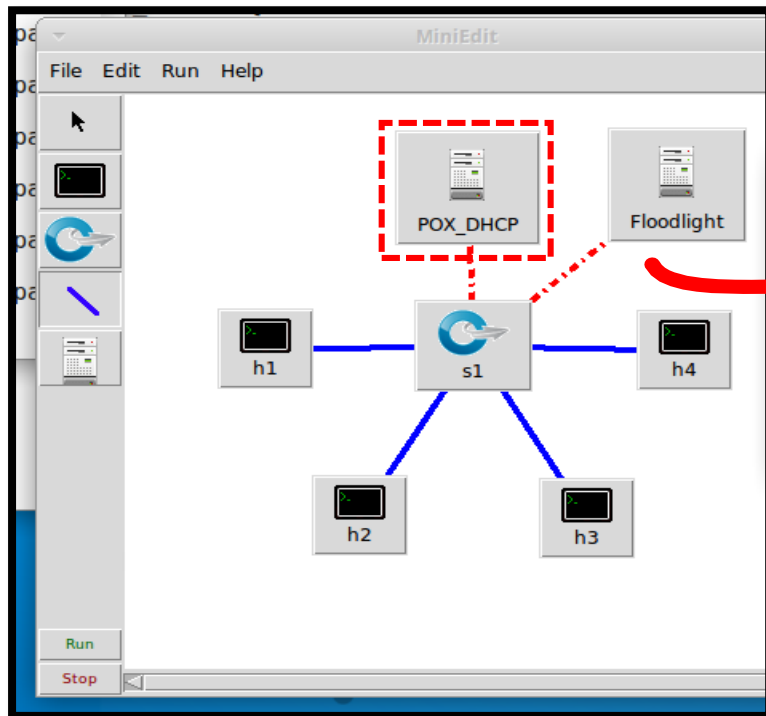
➤ 서버1의 s1 DPID -> 00:00:00:00:00:00:91



Miniedit 구동 및 Controller 연동하기

1) Pox Controller 와 Miniedit 연동하기

- POX_DHCP.mn 기본틀에서 POX_DHCP Controller 의 설정을 바꿔주고서 POX_DHCP_floodlight.mn으로 저장한다.
 - Controller Name : POX_DHCP / Controller Port : 6633 / IP : 127.0.0.1 / Controller Type: Remote Controller
 - Remote Controller : 자기자신이 되므로 자신의 로컬IP를 써준다.



The 'Controller Details' dialog box is shown with the following configuration:

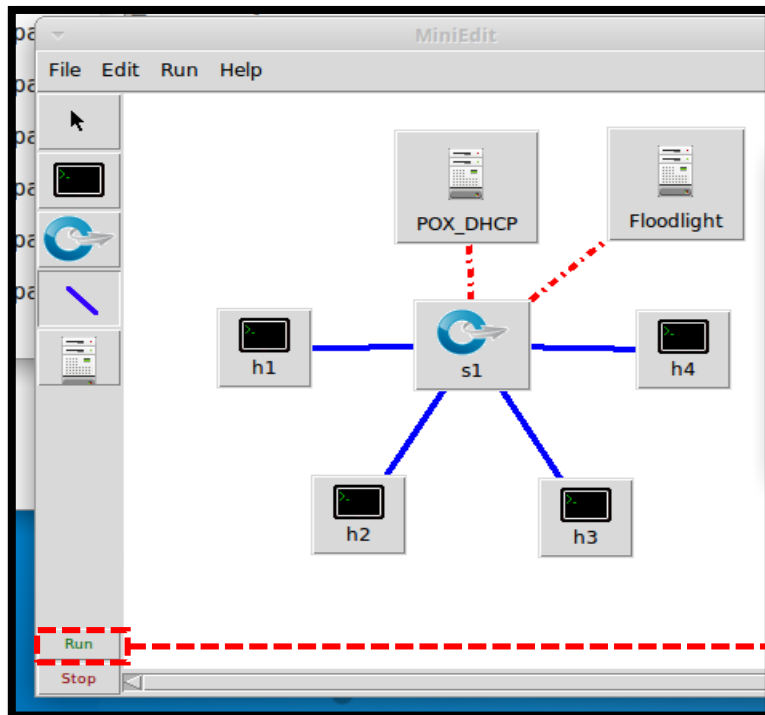
- Name: POX_DHCP
- Controller Port: 6633
- Controller Type: Remote Controller
- Protocol: TCP
- Remote/In-Band Controller: (checked)
- IP Address: 127.0.0.1

Buttons: OK, Cancel

Miniedit 구동 및 Controller 연동하기

1) Pox Controller 와 Miniedit 구동하기

- 왼쪽 하단에 있는 Run 버튼을 누르면 Mininet이 실행되어 Terminal에 실행되고 있음을 알리는 확인창이 뜬다.
 - 해당 터미널에서 어떤 Controller가 어떤 IP, PORT로 연결되었는지 다 볼 수 있으니 확인 차 보는 것도 좋다.
 - 올바르게 mininet이 구동되었다면 CLI 명령어를 칠 수 있는 란이 나온다.



```
File Edit View Search Terminal Help
'tcp', 'hostname': 'POX_DHCP', 'remoteIP': '192.168.0.63', 'controllerType': 'r
emote'}
New controller details for POX_DHCP = {'remotePort': 6633, 'controllerProtocol':
'tcp', 'hostname': 'POX_DHCP', 'remoteIP': '192.168.0.63', 'controllerType': 'r
emote'}
Getting Hosts and Switches.
Getting controller selection:remote
Getting Links.
*** Configuring hosts
h1 h3 h2 h4
**** Starting 1 controllers
POX_DHCP
**** Starting 1 switches
s1
No NetFlow targets specified.
No sFlow targets specified.

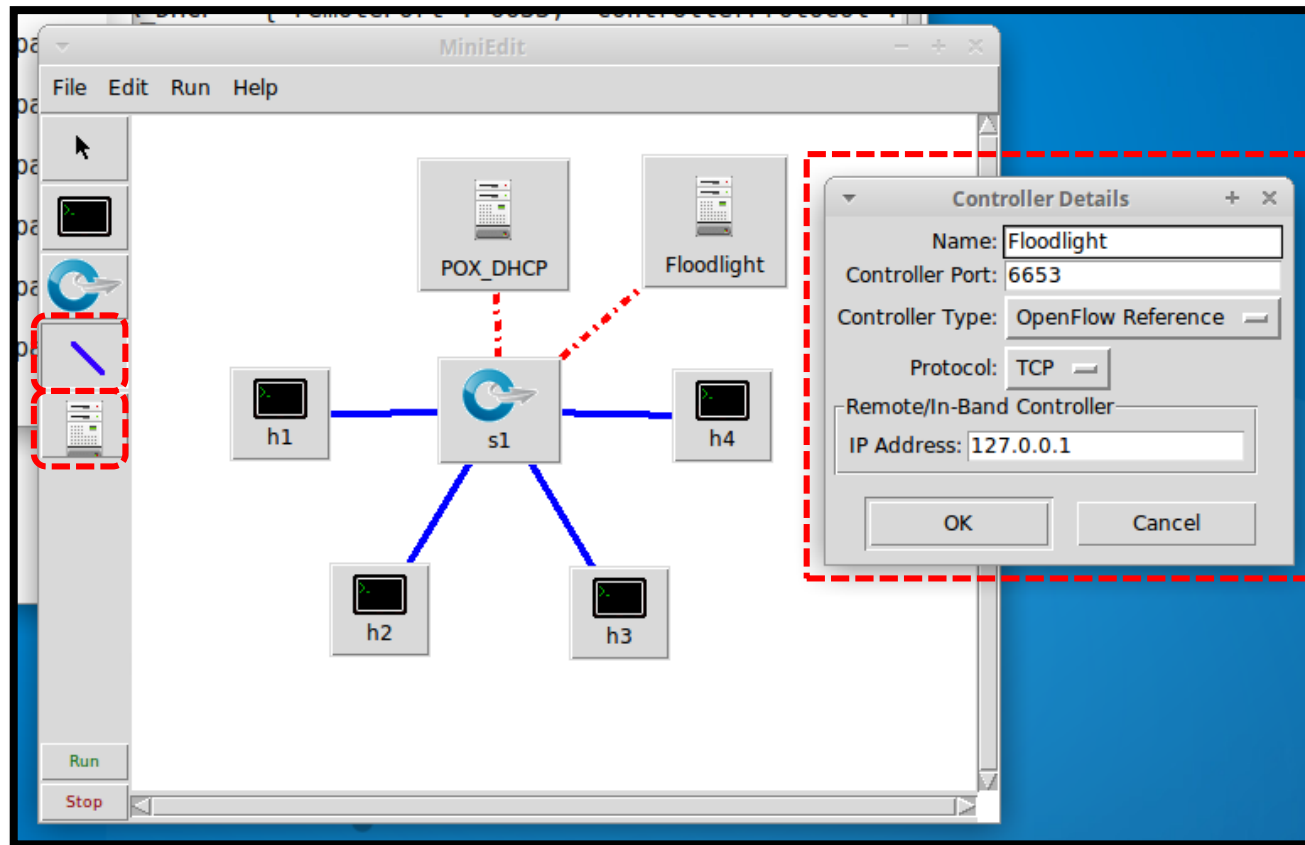
NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exi
ting will prevent MiniEdit from quitting and will prevent you from starting the
network again during this session.

*** Starting CLI:
mininet>
```

Miniedit 구동 및 Controller 연동하기

2) Floodlight Controller 와 Miniedit 연동하기

- 1)과정이 비슷하나 로컬이 다른 부분을 수정해서 설정해준다. (단, 서버2는 Floodlight Controller가 먼저 켜져야 함)
 - Controller Name : Floodlight / Controller Port : 6653 / IP : 127.0.0.1 / Controller Type: Remote Controller
 - Remote Controller : 자기 자신이므로 로컬 IP를 써준다.

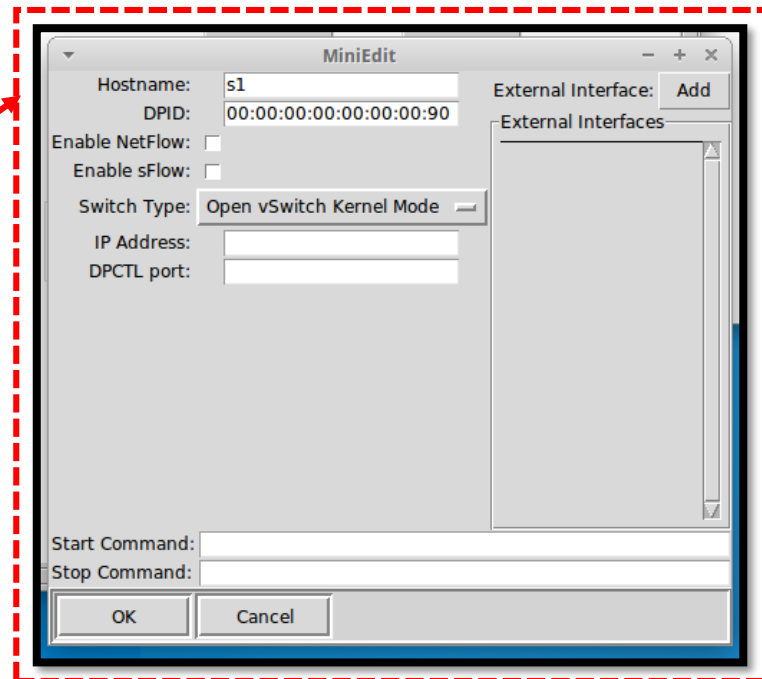
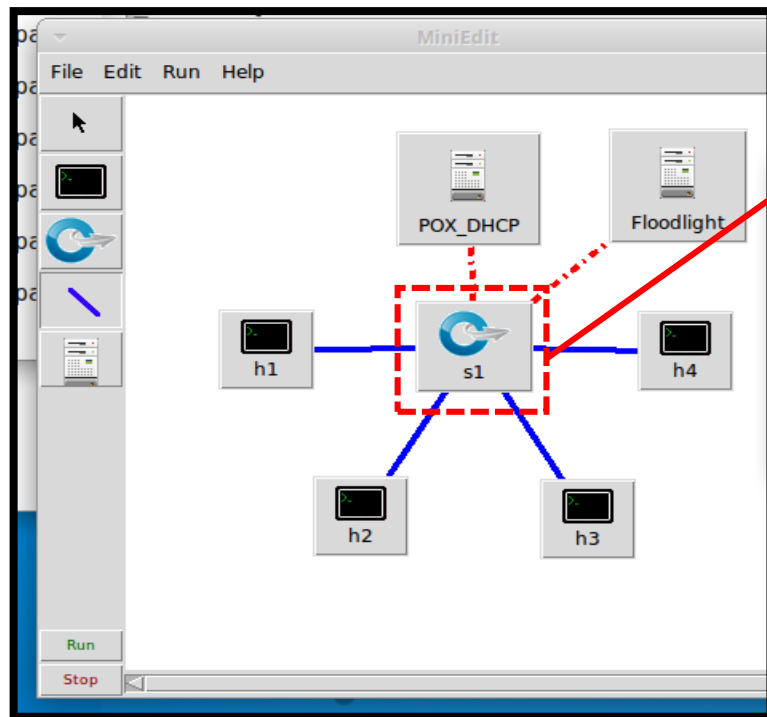


Miniedit 구동 및 Controller 연동하기

2) Floodlight Controller 와 Miniedit 연동하기

- POX_DHCP.mn s1 (스위치) 설정을 서버1의 DPID와 겹치지 않도록 임시로 지정한다. (90로 지정함)

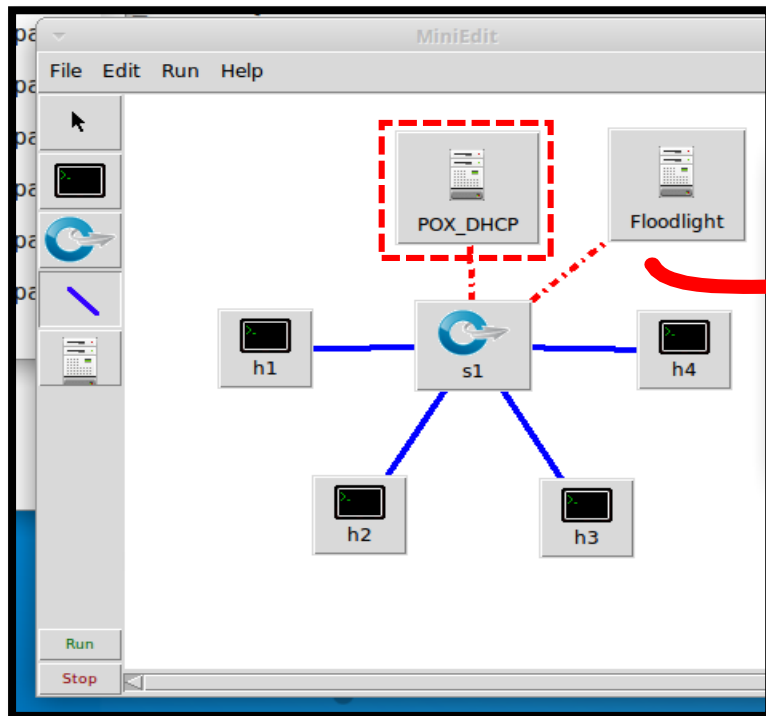
➤ 서버2의 s1 DPID -> 00:00:00:00:00:00:90 (서버1은 91이었음)



Miniedit 구동 및 Controller 연동하기

2) Floodlight Controller 와 Miniedit 연동하기

- POX_DHCP.mn 기본틀에서 POX_DHCP Controller 의 설정을 바꿔주고서 POX_DHCP_floodlight.mn으로 저장한다.
 - Controller Name : POX_DHCP / Controller Port : 6633 / IP : 192.168.0.185 / Controller Type: Remote Controller
 - Remote Controller : 서버1의 IP를 적어준다.



The 'Controller Details' dialog box is shown with the following configuration:

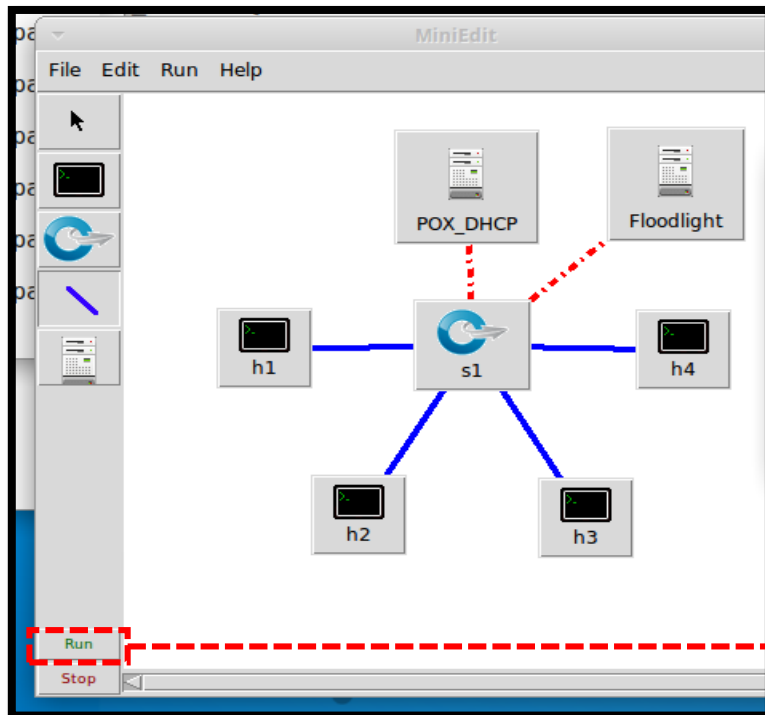
- Name: POX_DHCP
- Controller Port: 6633
- Controller Type: Remote Controller
- Protocol: TCP
- Remote/In-Band Controller: IP Address: 192.168.0.185

Buttons: OK, Cancel

Miniedit 구동 및 Controller 연동하기

2) Floodlight Controller 와 Miniedit 연동하기

- 왼쪽 하단에 있는 Run 버튼을 누르면 Mininet이 실행되어 Terminal에 실행되고 있음을 알리는 확인창이 뜬다.
 - 해당 터미널에서 어떤 Controller가 어떤 IP, PORT로 연결되었는지 다 볼 수 있으니 확인 차 보는 것도 좋다.
 - 올바르게 mininet이 구동되었다면 CLI 명령어를 칠 수 있는 란이 나온다.



```
Terminal
File Edit View Search Terminal Help
topo=None
New controller details for POX_DHCP = {'remotePort': 6633, 'controllerProtocol':
'tcp', 'hostname': 'POX_DHCP', 'remoteIP': '192.168.0.185', 'controllerType': '
remote'}
New controller details for floodlight = {'remotePort': 6653, 'controllerProtocol
': 'tcp', 'hostname': 'floodlight', 'remoteIP': '127.0.0.1', 'controllerType': '
remote'}
Getting Hosts and Switches.
Getting controller selection:remote
Getting controller selection:remote
Getting Links.
*** Configuring hosts
h2 h1 h4 h3
**** Starting 2 controllers
POX_DHCP floodlight
**** Starting 1 switches
s1
No NetFlow targets specified.
No sFlow targets specified.

NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exi
ting will prevent MiniEdit from quitting and will prevent you from starting the
network again during this session.

*** Starting CLI:
```

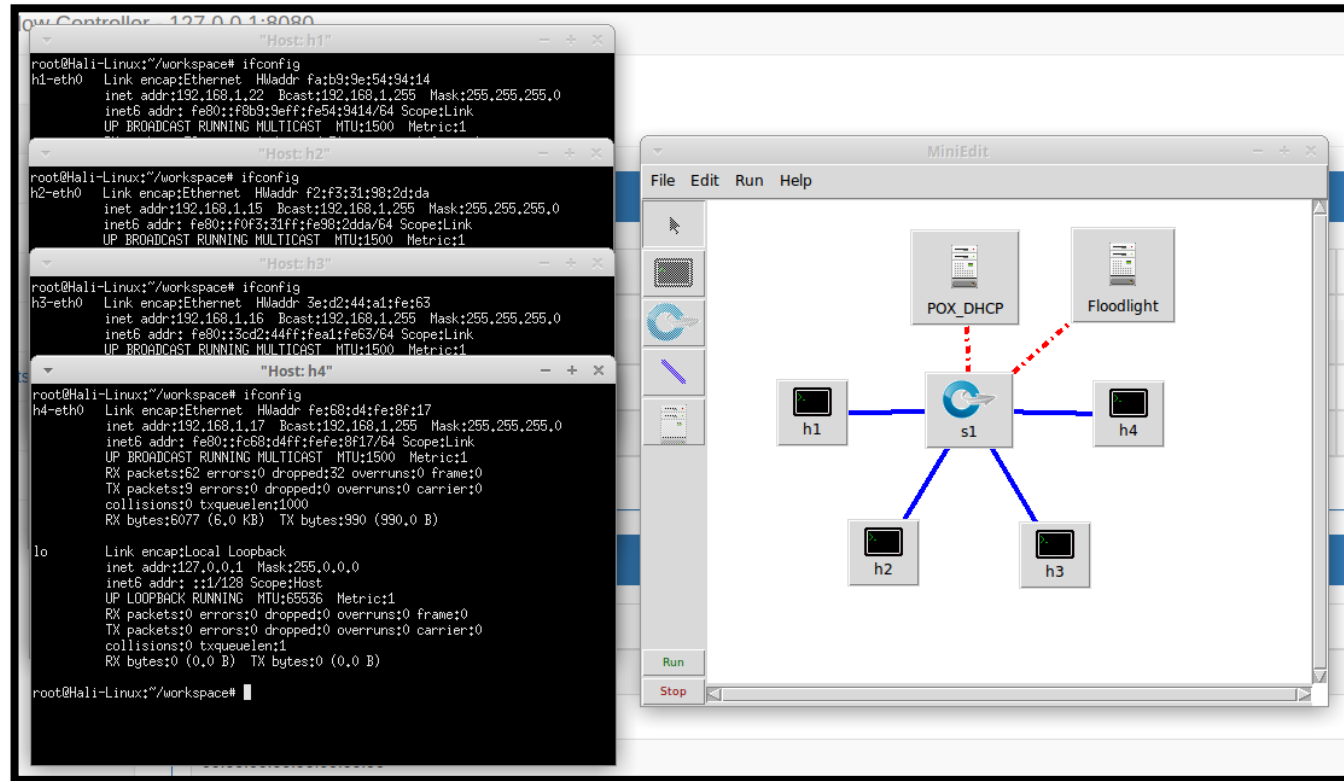

Miniedit 구동 및 Controller 연동하기

3) Miniedit Terminal에서 DHCP로 각각 IP 할당해주기

- POX와 연동된 Mininet CLI 창에서 dhclient 명령어를 통해 h1,h2,h3,h4 에 각각 ip를 할당해주기

➤ h1 : 192.168.1.22 / h2 : 192.168.1.15 / h3 : 192.168.1.16 / h4 : 192.168.1.17

```
*** Starting CLI:  
mininet> h1 dhclient  
mininet> h2 dhclient  
mininet> h3 dhclient  
mininet> h4 dhclient
```



Miniedit 구동 및 Controller 연동하기

3) Miniedit Terminal에서 DHCP로 각각 IP 할당해주기

- 각각 연동된 Mininet CLI 창에서 dhclient 명령어를 통해 h1,h2,h3,h4 에 각각 ip를 할당해주기

➢ [POX_서버1] h1 : 192.168.1.22 / h2 : 192.168.1.15 / h3 : 192.168.1.16 / h4 : 192.168.1.17

➢ [Flood_서버2] h1 : 192.168.1.30 / h2: 192.168.1.29 / h3 : 192.168.1.28 / h4 : 192.168.1.27

```
*** Starting CLI:
mininet> h1 dhclient
mininet> h2 dhclient
mininet> h3 dhclient
mininet> h4 dhclient
```

```
POX_Controller_127.0.0.1:8080
"Host:h1"
root@Hali-Linux:~/workspace# ifconfig
h1-eth0  Link encap:Ethernet  HWaddr fa:b9:9e:54:94:14
          inet addr:192.168.1.22  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f8b3:9eff:fe54:9414/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

"Host:h2"
root@Hali-Linux:~/workspace# ifconfig
h2-eth0  Link encap:Ethernet  HWaddr f2:f3:31:98:2d:da
          inet addr:192.168.1.15  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f0f3:31ff:fe98:2dda/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

"Host:h3"
root@Hali-Linux:~/workspace# ifconfig
h3-eth0  Link encap:Ethernet  HWaddr Seid2:44:a1:fe:63
          inet addr:192.168.1.16  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::3cd2:44ff:feal:fe63/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

"Host:h4"
root@Hali-Linux:~/workspace# ifconfig
h4-eth0  Link encap:Ethernet  HWaddr fe:68:d4:fe:8f:17
          inet addr:192.168.1.17  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::fc68:d4ff:fe8f:17/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:32 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6077 (6.0 KB)  TX bytes:990 (990.0 B)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@Hali-Linux:~/workspace#
```

POX_서버1

```
"Host:h4"
root@Hali-Linux:~/workspace# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: h4-eth0:f11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether d8:a0:76:9d:17:a0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.1.30/24 brd 192.168.1.255 scope global h4-eth0

"Host:h3"
root@Hali-Linux:~/workspace# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: h3-eth0:f10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether b2:00:06:18:bc:4f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.1.29/24 brd 192.168.1.255 scope global h3-eth0

"Host:h2"
root@Hali-Linux:~/workspace# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: h2-eth0:f8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether 1a:0d:0b:9f:71:29 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.1.28/24 brd 192.168.1.255 scope global h2-eth0

"Host:h1"
root@Hali-Linux:~/workspace# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: h1-eth0:f8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether 9e:ef:23:b5:12:df brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.1.27/24 brd 192.168.1.255 scope global h1-eth0
    inet6 fe80::9ef:23bf:feb5:12df/64 scope link
        valid_lft forever preferred_lft forever
root@Hali-Linux:~/workspace#
```

Floodlight_서버2

Miniedit 구동 및 Controller 연동하기

4-1) 각 Host 별로 웹 서버를 하나 만들어서 통신이 가능한지 확인하기 (POX의 경우)

- POX controller를 사용하는 서버에서 h2를 웹서버로 만들고, h1을 통해 curl과 ping을 확인한다.

- SimpleHTTPServer 80 명령어를 통해 h2를 http 테스트 웹서버로 만든다.
- 결과 : h1에서 curl 192.168.1.26(h2 ip)와 ping 192.168.1.26을 시도해본 결과,

```
root@Hali-Linux:~/workspace# ifconfig
h2-eth0: Link encap:Ethernet  HWaddr 6a:cc:ad:55:72:27
        inet addr:192.168.1.26  Bcast:192.168.1.255  Mask:255.255.255.0
        inet6 addr: fe80::6a:cc:ad:55:72:27/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:45 errors:0 dropped:6 overruns:0 frame:0
        TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:4619 (4.6 KB)  TX bytes:990 (990.0 B)

lo: Link encap:Local Loopback
    inet addr:127.0.0.1  Mask:255.0.0.0

root@Hali-Linux:~/workspace# ping 192.168.1.26
PING 192.168.1.26 (192.168.1.26) 56(84) bytes of data:
From 192.168.1.25 icmp_seq=1 Destination Host Unreachable
From 192.168.1.25 icmp_seq=2 Destination Host Unreachable
From 192.168.1.25 icmp_seq=3 Destination Host Unreachable
From 192.168.1.25 icmp_seq=4 Destination Host Unreachable
From 192.168.1.25 icmp_seq=5 Destination Host Unreachable
From 192.168.1.25 icmp_seq=6 Destination Host Unreachable
From 192.168.1.25 icmp_seq=7 Destination Host Unreachable
From 192.168.1.25 icmp_seq=8 Destination Host Unreachable
From 192.168.1.25 icmp_seq=9 Destination Host Unreachable
From 192.168.1.25 icmp_seq=10 Destination Host Unreachable
From 192.168.1.25 icmp_seq=11 Destination Host Unreachable
From 192.168.1.25 icmp_seq=12 Destination Host Unreachable
From 192.168.1.25 icmp_seq=13 Destination Host Unreachable
From 192.168.1.25 icmp_seq=14 Destination Host Unreachable
From 192.168.1.25 icmp_seq=15 Destination Host Unreachable
^C
--- 192.168.1.26 ping statistics ---
 6 packets transmitted, 0 received, 100% packet loss, time 15030ms
root@Hali-Linux:~/workspace# curl 192.168.1.26
curl: (7) Failed to connect to 192.168.1.26 port 80: No route to host
root@Hali-Linux:~/workspace#
```

1. Ping 192.168.1.26 을 했을 경우, ping을 잡지 못한다는 것을 알 수 있음.

2. Curl 192.168.1.26을 했을 경우, curl을 잡지 못한다는 것을 알 수 있음.

Why?

“L2 스위치 설정이 없기 때문이다.”

- MAC 주소와 IP를 매핑해줘야 되는데 Miniedit끼리 통신이 안돼서 가져오기 못하기 때문에 Remote로 사용하는 쪽의 IP 설정이 불가능하다.
- Pox 코딩 시 DHCP 기능만 살리고 다른 기능들을 다 날려버려서 Remote 받아서 사용하는 쪽에서 정보를 받아올 수 없는 것이다.

Miniedit 구동 및 Controller 연동하기

4-1) 각 Host 별로 웹 서버를 하나 만들어서 통신이 가능한지 확인하기 (floodlight의 경우)

- floodlight controller를 사용하는 서버에서 h4를 웹서버로 만들고, h1을 통해 curl과 ping을 확인한다.

- SimpleHTTPServer 80 명령어를 통해 h4를 http 테스트 웹서버로 만든다.
- 결과 : h1에서 curl 192.168.1.30(h4 ip)와 ping 192.168.1.30을 시도해본 결과,

```
root@Hali-Linux:~/workspace# ping 192.168.1.30
PING 192.168.1.30 (192.168.1.30) 56(84) bytes of data:
64 bytes from 192.168.1.30: icmp_seq=1 ttl=64 time=6.66 ms
64 bytes from 192.168.1.30: icmp_seq=2 ttl=64 time=0.248 ms
64 bytes from 192.168.1.30: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 192.168.1.30: icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from 192.168.1.30: icmp_seq=5 ttl=64 time=0.041 ms
--- 192.168.1.30 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 0.034/1.405/6.663/2.630 ms
root@Hali-Linux:~/workspace# curl 192.168.1.30
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="Floodlight.mn">Floodlight.mn</a>
<li><a href="miniedit.py">miniedit.py</a>
<li><a href="Mininet.mn">Mininet.mn</a>
<li><a href="mininet_1.mn">mininet_1.mn</a>
<li><a href="Mininet_ajp.mn">Mininet_ajp.mn</a>
<li><a href="Opendaylight.mn">Opendaylight.mn</a>
<li><a href="POX.mn">POX.mn</a>
<li><a href="POX_DHCP.mn">POX_DHCP.mn</a>
<li><a href="POX_DHCP_floodlight.mn">POX_DHCP_floodlight.mn</a>
<li><a href="POX_L2_Multi.mn">POX_L2_Multi.mn</a>
<li><a href="POX_L3.mn">POX_L3.mn</a>
<li><a href="POX_SLB.mn">POX_SLB.mn</a>
<li><a href="POX_SSL.mn">POX_SSL.mn</a>
<li><a href="RYU.mn">RYU.mn</a>
</ul>
<hr>
</body>
</html>
```

```
root@Hali-Linux:~/workspace# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc fq_codel state UP qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: h4-eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether d8:a0:76:9d:17:a0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.30/24 brd 192.168.1.255 scope global h4-eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::d4a0:76ff:fe9d:17a0/64 scope link
        valid_lft forever preferred_lft forever
root@Hali-Linux:~/workspace# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.1.27 - - [28/Jan/2019 11:40:40] "GET / HTTP/1.1"
```

1. Ping 192.168.1.30 을 했을 경우, ping을 잡을 수 있음.
2. Curl 192.168.1.30을 했을 경우, curl을 할 수 있음.

Why?

"L2 스위치 설정이 되어 있기 때문이다."

→ MAC 주소와 IP를 매핑해줘야 되는데 Floodlight쪽에서는 그럴 수 있는 기능들이 갖춰져 있다. 그러므로 MAC주소나 IP를 받아올 수 있기 때문에 통신이 가능하다.

4. Floodlight 결과 측정

Floodlight Web에서 결과 측정하기

1) Floodlight Web을 켜서 Dashboard 들어가기

- Floodlight Controller를 실행하고 있는 서버2에서 진행해야 함. Floodlight Web을 들어가면 아래와 같은 대시보드가 뜬
 - 서버1과 서버2의 호스트, 스위치가 모두 다 나와야 정상임 (현재 POX와 Floodlight controller 둘 다 연동되어 있기 때문)

The image shows the Floodlight Web dashboard and a terminal window. The dashboard displays the following information:

- Controller Status:** Active (Green checkmark)
- Uptime:** 00:09:44 (Orange clock icon)
- Controller Role:** ACTIVE (Blue user icon)
- Switches:** 2 (Red box with double arrows icon)
- Hosts:** 8 (Red box with laptop icon)
- Connections (Links):** 0 (Red box with double arrows icon)
- Reserved Ports:** 0 (Red box with checkmark icon)
- JVM Memory Bloat:** 128.13 MB (Green gauge icon)
- Consumption Detail:**

Total:	369.62 MB
Used:	128.65 MB
Free:	240.97 MB
- Storage Tables:**

controller_controller
controller_controllerinterface
controller_switchconfig
controller_forwardingconfig
controller_staticentrytable
controller_topologyconfig
controller_link
controller_firewallrules

The terminal window shows the installation of Floodlight Web:

```
Openflow Miniedit
POX Controller
RYU Controller
Snort IDS
Floodlight Controller
Floodlight Web
Opendaylight Controller
Opendaylight Web
```

Floodlight Web에서 결과 측정하기

2) Floodlight Web에서 스위치 확인하기.

- Switches 란에 들어가면 앞서서 설정해둔 서버1인 91과 서버2인 90이 현재 연동되었음을 확인가능하다.

➤ 해당 스위치에 들어가면 더 자세한 정보를 볼 수 있다.

Floodlight OpenFlow Controller - 127.0.0.1:8080

Controller (Home)
Switches
Hosts
Links
Topology
Firewall
Access Control Lists
Statistics
Change Controllers

Switches

Switches Connected

Switch ID	IPv4 Address	Connected Since
00:00:00:00:00:00:90	/127.0.0.1:38128	Mon Jan 28 2019 02:36:19 GMT+0900 (Korean Standard Time)
00:00:00:00:00:00:91	/192.168.0.185:50618	Mon Jan 28 2019 02:38:12 GMT+0900 (Korean Standard Time)

Showing 1 to 2 of 2 entries

Switch Roles

Switch MAC
00:00:00:00:00:00:91
00:00:00:00:00:00:90

Switch Detail

Switch Detail

MAC	: 00:00:00:00:00:00:90
Version	: OF_13
Vendor	: Nicira, Inc.
Hardware Info	: Open vSwitch
Software Version	: 2.5.4
Serial Number	: None
Datapath	: s1

Flow Summary

Flow Count	: 1
Packet Count	: 58
Byte	: 5164
Flag	:
Buffer	: 256
Table Count	: 254

Role Info

MASTER
SLAVE
EQUAL

Change

Port Table

Show 10 entries Search:

No	R. Packets	Tran. Packets	R. Bytes	Tran. Bytes	R. Dropped	Tran. Dropped	Coll.	Duration(s)
local	0	0	0	0	35	0	0	590

Showing 1 to 1 of 1 entries.

Floodlight Web에서 결과 측정하기

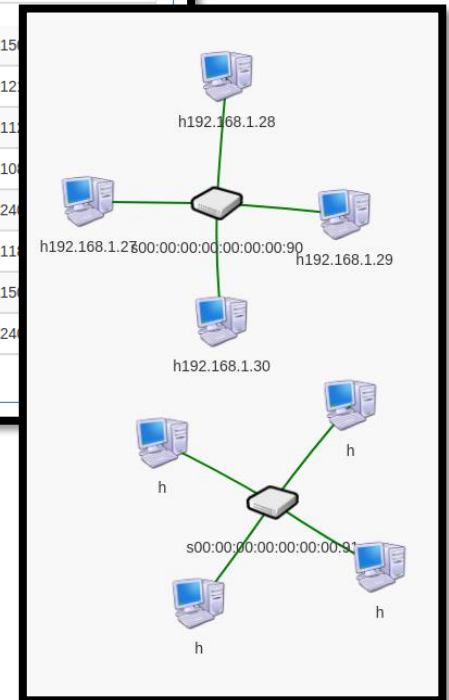
2) Floodlight Web에서 Host와 네트워크 토폴로지 확인하기.

- Hosts, Topology 란에 들어가면 앞서서 설정해둔 서버1인 91과 서버2인 90이 현재 연동되었음을 확인 가능하다.

➤ 그러나 어디까지나 해당 로컬인 90의 host들만 나타나고, 91의 host ip는 나타나지 않는다. (그러나 실제로 연동되긴 했음)

Hosts						
Hosts Connected						
MAC	IPv4 Address	IPv6 Address	Switch	Port	Last Seen	
1a:0d:0b:9f:71:29	192.168.1.28	fe80::180d:bff:fe9f:7129	00:00:00:00:00:00:90	2	154864315	
36:4e:00:07:90:12			00:00:00:00:00:00:91	4	154864312	
6a:cc:ad:55:72:27			00:00:00:00:00:00:91	2	154864311	
82:53:96:34:41:73			00:00:00:00:00:00:91	1	154864310	
9e:ef:23:b5:12:df	192.168.1.27	fe80::9cef:23ff:feb5:12df	00:00:00:00:00:00:90	1	154864324	
a2:f7:14:1c:d2:d1		fe80::a0f7:14ff:fe1c:d2d1	00:00:00:00:00:00:91	3	154864311	
b2:00:06:18:bc:4f	192.168.1.29	fe80::b000:6ff:fe18:bc4f	00:00:00:00:00:00:90	3	154864315	
d6:a0:76:9d:17:a0	192.168.1.30	fe80::d4a0:76ff:fe9d:17a0	00:00:00:00:00:00:90	4	154864324	

Showing 1 to 8 of 8 entries



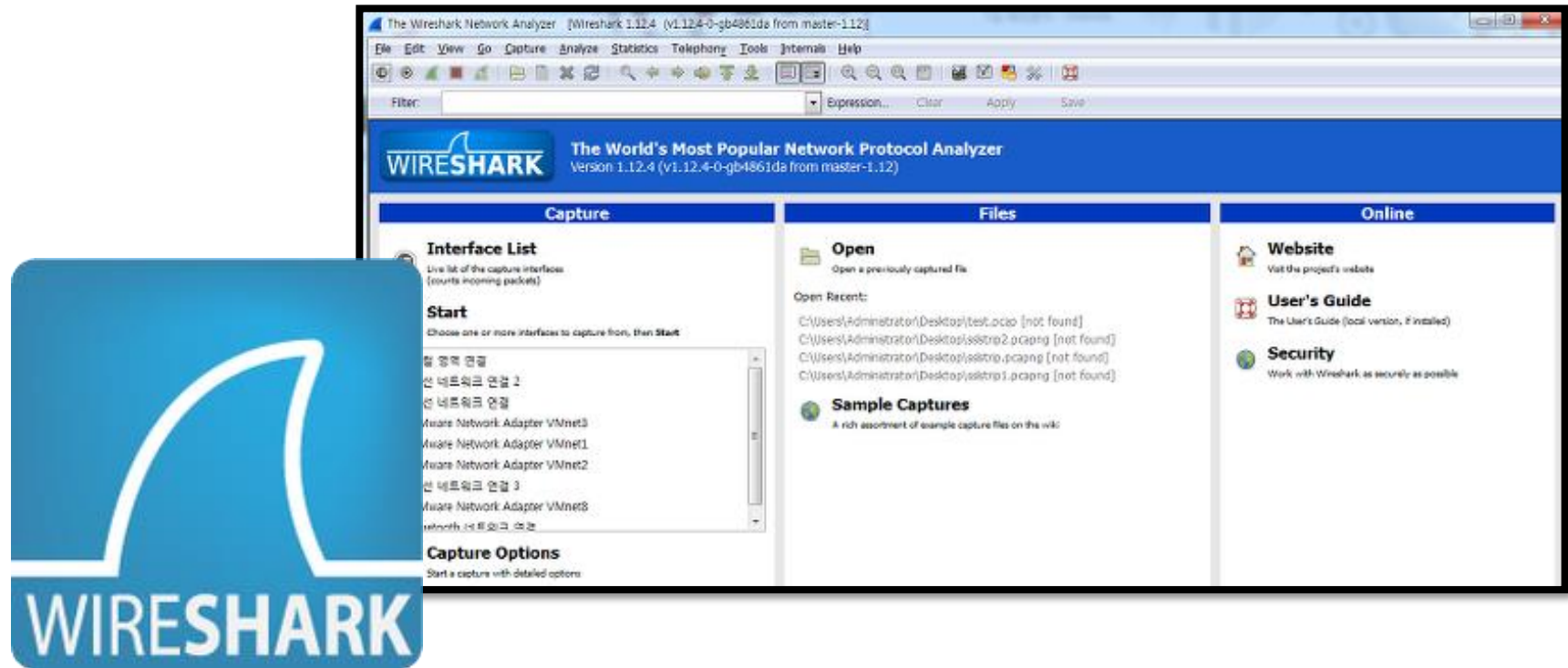
➤ 토폴로지에서 91(서버1의 스위치)의 호스트 IP는 나오지 않는다.

5. WireShark 결과 측정 및 분석

Wireshark로 패킷 측정하기

Wireshark 란?

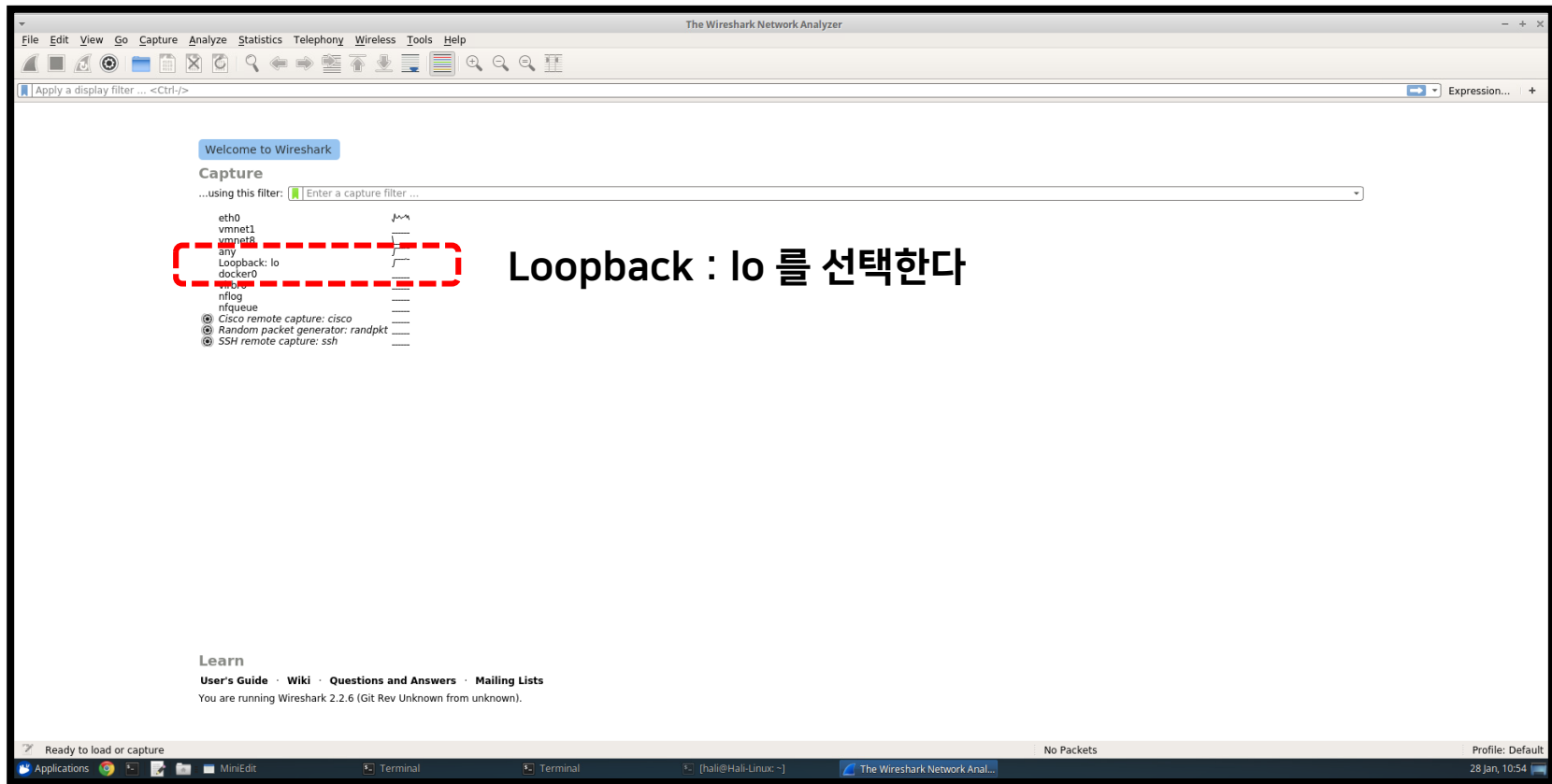
- Wireshark란 네트워크를 분석하는데 사용되는 공개된 패킷 스나핑(packet sniffing) 프로그램이다. 제럴드 콤즈라는 사람에 의해 만들어졌으며 Ehtereal 프로젝트에서 Wireshark로 이름이 바뀌었다. 세계에서 가장 널리 쓰이는 네트워크 분석 프로그램으로 네트워크 상에서 캡처한 데이터에 대한 네트워크 / 상위 레이어 프로토콜의 정보를 제공해준다. 패킷캡처를 위해 pcap 네트워크 라이브러리를 이용한다.



Wireshark로 패킷 측정 및 분석하기

1) Wireshark를 먼저 들어가서 켜다

- 각각의 호스트에서 Loopback : lo 를 선택하여 해당 호스트로 흐르는 패킷들을 확인한다.
 - Openflow가 어떻게 흐르고 있는지를 확인하고자 함이기에 wireshark로 측정하는 것이다.



Wireshark로 패킷 측정 및 분석하기

2-1) Wireshark에서 측정결과 확인하기 (pox controller가 local인 경우)

- 각각의 연동된 호스트의 결과가 어떻게 나타나고 있는지 확인한다. (185에서 63으로 흐르는 패킷을 확인할 경우)

➤ 185(pox)에서 63(floodlight)로 흐르는 패킷이 OpenFlow로 나타남을 확인할 수 있다.

The screenshot shows the Wireshark interface with a packet capture on the interface 'Loopback: lo'. The filter bar shows 'ip.addr==192.168.0.63'. The packet list pane displays several packets, with packet 34 highlighted. The details pane for packet 34 shows the following structure:

- Frame 34: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
- Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 36514, Dst Port: 6633, Seq: 65, Ack: 65, Len: 0

The bottom status bar indicates 'Packets: 65 · Displayed: 65 (100.0%)'.

Wireshark로 패킷 측정 및 분석하기

2-2) Wireshark에서 측정결과 확인하기 (pox controller가 로컬인 경우)

- 각각의 연동된 호스트의 결과가 어떻게 나타나고 있는지 확인한다. (63에서 185으로 흐르는 패킷을 확인할 경우)

➤ 63(floodlight)에서 185(pox)로 흐르는 패킷이 OpenFlow로 나타남을 확인할 수 있다.

The screenshot displays the Wireshark interface within a VMware Workstation 15 Player. The main window shows a packet capture on the *eth0 interface. The packet list pane at the top shows a list of captured packets, with the filter 'ip.addr==192.168.0.185' applied. The selected packet (No. 11067) is a TCP packet from 192.168.0.185 to 192.168.0.114. The packet details pane shows the structure of the selected packet, including the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
9467	38.978547960	192.168.0.185	192.168.0.114	TCP	64	6653 -> 41238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9700	39.511517158	192.168.0.63	192.168.0.185	TCP	66	6653 -> 50618 [ACK] Seq=2057 Ack=153 Win=1902 Len=0 TSval=75985 TSecr=72533
9716	39.586247157	192.168.0.63	192.168.0.185	TCP	66	48674 -> 6653 [ACK] Seq=81 Ack=81 Win=66 Len=0 TSval=76003 TSecr=72552
9808	39.990852810	192.168.0.114	192.168.0.185	TCP	74	41238 -> 6653 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=130742250 TSecr=0 WS=512
9899	39.980109017	192.168.0.185	192.168.0.114	TCP	54	6653 -> 41238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10345	40.978245292	192.168.0.114	192.168.0.185	TCP	74	41238 -> 6653 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=130742500 TSecr=0 WS=512
10346	40.979202419	192.168.0.114	192.168.0.185	UDP	54	6653 -> 41238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10410	41.511758122	192.168.0.63	192.168.0.185	OpenFlow	74	Type: OFPT_ECHO_REQUEST
10417	41.511897795	192.168.0.185	192.168.0.63	OpenFlow	74	Type: OFPT_ECHO_REPLY
10418	41.512178301	192.168.0.63	192.168.0.185	TCP	66	6653 -> 50618 [ACK] Seq=2065 Ack=161 Win=1902 Len=0 TSval=76485 TSecr=73033
10683	41.970203006	192.168.0.114	192.168.0.185	TCP	74	41238 -> 6653 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=130742750 TSecr=0 WS=512
10684	41.970352027	192.168.0.185	192.168.0.114	TCP	54	6653 -> 41238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10811	42.978612995	192.168.0.114	192.168.0.185	TCP	74	41238 -> 6653 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=130743000 TSecr=0 WS=512
10812	42.978650161	192.168.0.114	192.168.0.185	UDP	54	6653 -> 41238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10955	43.513450890	192.168.0.63	192.168.0.185	OpenFlow	74	Type: OFPT_ECHO_REQUEST
10956	43.513553663	192.168.0.185	192.168.0.63	OpenFlow	74	Type: OFPT_ECHO_REPLY
10957	43.513706099	192.168.0.63	192.168.0.185	TCP	66	6653 -> 50618 [ACK] Seq=2073 Ack=169 Win=1902 Len=0 TSval=76985 TSecr=73534
11067	43.978438944	192.168.0.114	192.168.0.185	TCP	74	41238 -> 6653 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=130743250 TSecr=0 WS=512

Frame 2263: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: Vmware_b9:f7:20 (00:0c:29:b9:f7:20), Dst: Vmware_5f:1e:8a (00:0c:29:5f:1e:8a)
Internet Protocol Version 4, Src: 192.168.0.185, Dst: 192.168.0.114
Transmission Control Protocol, Src Port: 6653, Dst Port: 41166, Seq: 1, Ack: 1, Len: 0

0000 00 0c 29 5f 1e 8a 00 0c 29 b9 f7 20 00 00 45 c0 ..).....E.
0010 00 29 44 eb 40 00 40 06 72 a9 c0 a8 00 b9 c0 a8 (D.0.0.f.....
0020 00 72 19 fd a0 ce 00 00 00 ed de 29 bc 50 14 .f.....).P.
0030 00 00 5a ee 00 00 ..Z...

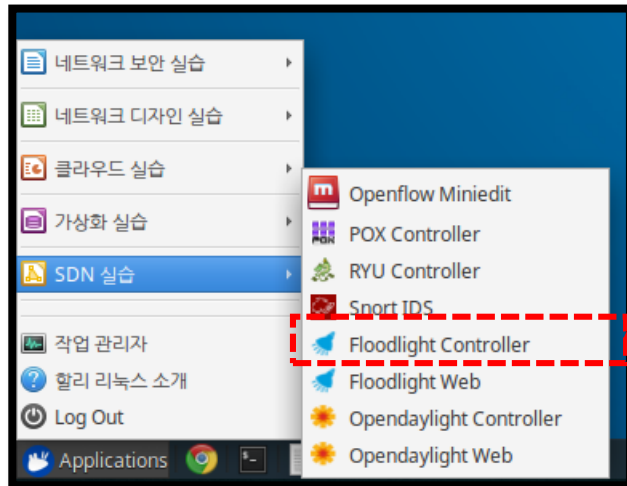
1. Floodlight 구동

Floodlight Controller 설정

1) Floodlight Controller를 구동한다.

- 미리 구성한 서버2(90)의 ESXi 환경에 설치된 Floodlight Controller를 연다.

- 파일 위치 : Applications > SDN 실습 > Floodlight Controller
- 열면 하단 그림과 같이 터미널 하나가 생기면서 Floodlight Controller가 켜졌음을 보여준다.

A screenshot of a terminal window titled 'Terminal'. It displays the startup logs of the Floodlight Controller. The logs show the controller starting on port 8080, receiving a new switch connection from 192.168.0.114:39704, negotiating the OpenFlow version (OF_13), and binding the switch to the controller. There is an error message about a failed OFGroupMod operation, followed by a topology recomputation. The logs end with the debug server starting on port 6655.

```
Server on port 8080
2019-01-28 10:48:31.650 INFO [org.restlet] Starting net.floodlightcontroller.re
stserver.RestApiServer$RestApplication application
2019-01-28 10:48:31.694 INFO [n.f.c.i.OFChannelHandler] New switch connection f
rom /192.168.0.114:39704
2019-01-28 10:48:31.707 INFO [n.f.c.i.OFChannelHandler] Negotiated down to swit
ch OpenFlow version of OF_13 for /192.168.0.114:39704 using lesser hello header
algorithm.
2019-01-28 10:48:31.741 INFO [n.f.c.i.OFSwitchHandshakeHandler] Switch OFSwitch
DPID[00:00:00:00:00:00:10] bound to class class net.floodlightcontroller.cor
e.internal.OFSwitch, description SwitchDescription [manufacturerDescription=Nici
ra, Inc., hardwareDescription=Open vSwitch, softwareDescription=2.5.4, serialNum
ber=None, datapathDescription=s1]
2019-01-28 10:48:31.919 INFO [n.f.c.i.OFSwitchHandshakeHandler] Clearing flow t
ables of 00:00:00:00:00:00:10 on upcoming transition to MASTER.
2019-01-28 10:48:31.985 ERROR [n.f.c.i.OFSwitchHandshakeHandler] OFGroupModFaile
dErrorMsgVer13(xid=12, code=INVALID GROUP, data=OFGroupDeleteVer13(xid=12, group
Type=INDIRECT, group=all, buckets=[])) from switch OFSwitch DPID[00:00:00:00:
00:00:10] in state net.floodlightcontroller.core.internal.OFSwitchHandshakeHandl
er$MasterState@32644e23
2019-01-28 10:48:32.78 INFO [n.f.t.TopologyManager] Recomputing topology due to
: link-discovery-updates
2019-01-28 10:48:32.997 INFO [n.f.j.JythonServer] Starting DebugServer on :6655
```

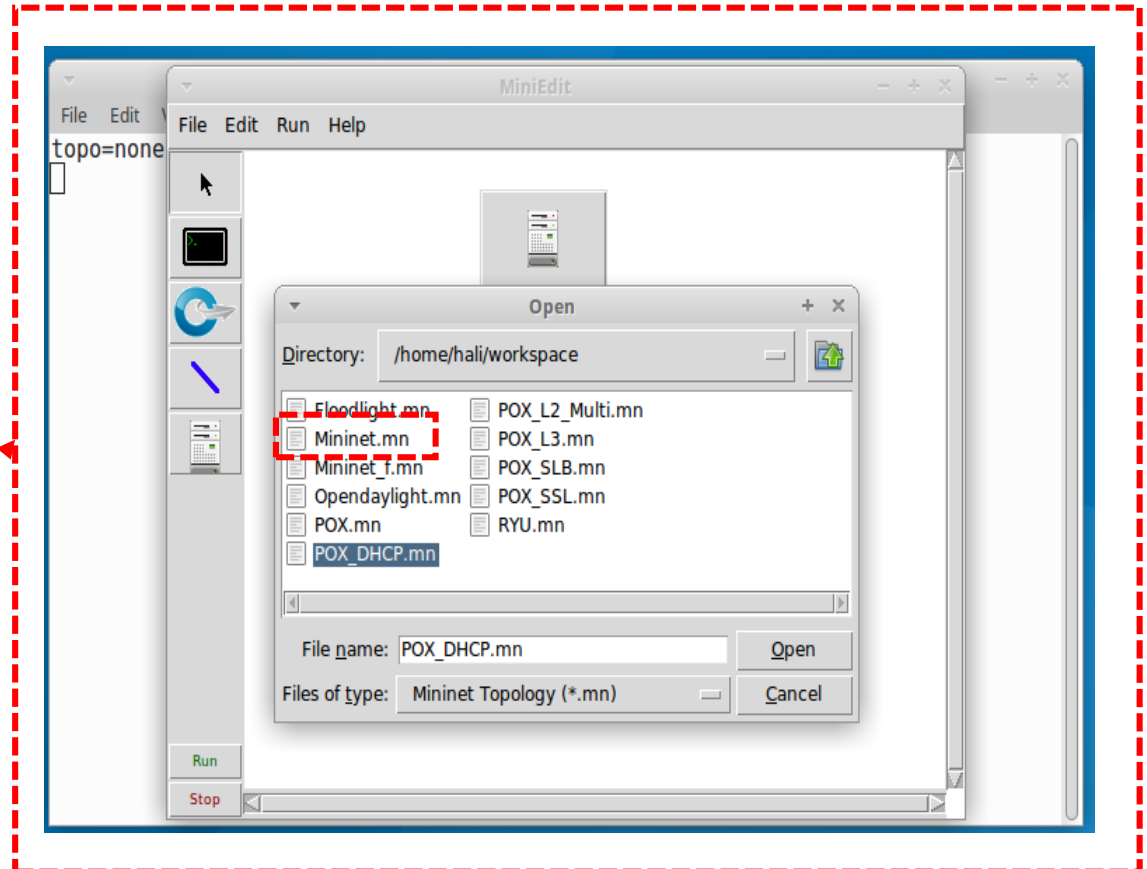
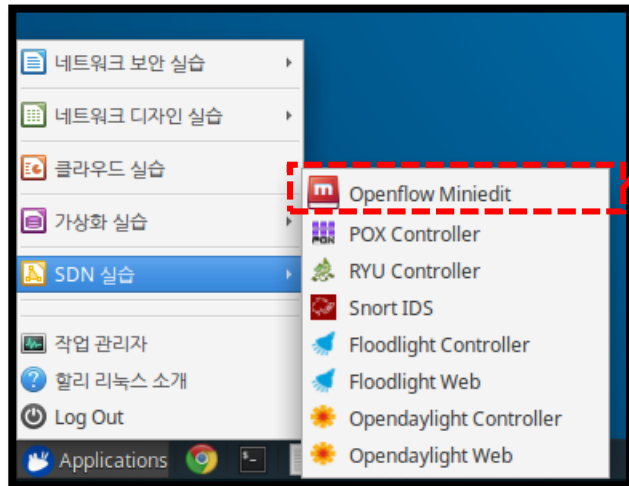
2. Miniedit 구동

Miniedit 구동 및 Controller 연동하기

1) Miniedit에서 mininet.mn 실행

- 미리 구성한 서버1의 hali-linux에 설치된 Openflow Miniedit을 연다.

- 파일 위치 : Applications > SDN 실습 > Openflow Miniedit
- File > Open > mininet.mn을 불러온다. (기본 틀이 되는 예시를 가져와서 쓸 것)



Miniedit 구동 및 Controller 연동하기

2-1) Miniedit 기본 틀 만들기 (서버2,3의 경우)

- mininet.mn 기본 틀에서 Mininet Controller를 설정해준다.

- Controller Name : Mininet / Controller Port : 6633 / IP : 192.168.0.63 / Controller Type: Remote Controller
- Remote Controller : 서버2가 되므로 서버2의 ip를 써준다.

The screenshot displays the Mininet GUI interface. On the left, a terminal window shows the command `topo=None`. The main area shows a network topology with a central switch `s1` connected to three hosts `h1`, `h2`, and `h3`. A Mininet controller icon is shown at the top, connected to the switch via a dashed red line. A 'Controller Details' dialog box is open, showing the following configuration:

- Name: Mininet
- Controller Port: 6633
- Controller Type: Remote Controller
- Protocol: TCP
- Remote/In-Band Controller: IP Address: 192.168.0.63

On the right side of the image, there is a text box with the following content:

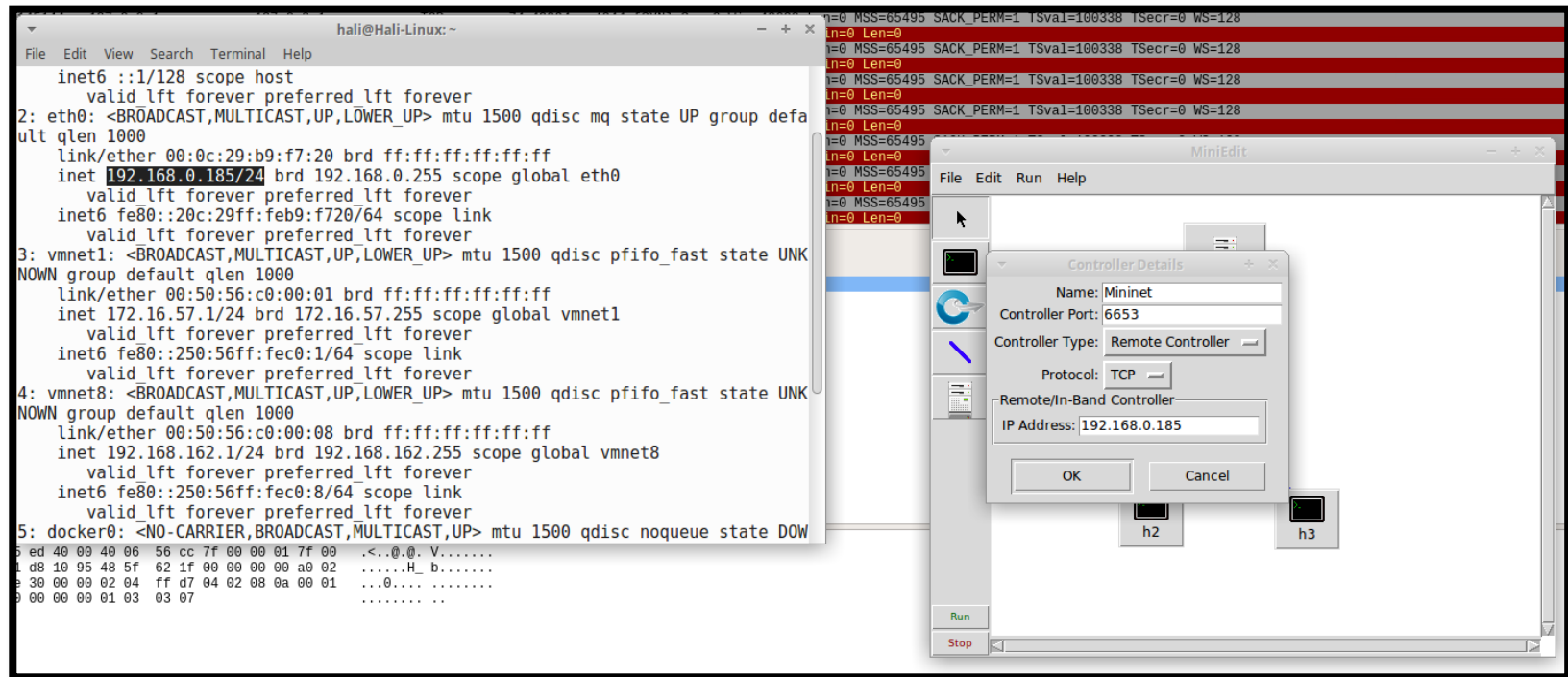
서버2,3에서는 floodlight controller가 없다!

→ 해당 그림은 예시로 **서버3**을 설정한 것
(ip : 192.168.0.63)
(Switch ID : 00:00:00:00:00:00:00:11)

Miniedit 구동 및 Controller 연동하기

2-2) Miniedit 기본 틀 만들기 (서버1의 경우)

- mininet.mn 기본 틀에서 Mininet Controller를 설정해준다. (여긴 floodlight 연동하는 것과 같다)
 - Controller Name : Mininet / Controller Port : 6653 / IP : 192.168.0.185 / Controller Type: Remote Controller
 - Remote Controller : 서버1가 되므로 서버1의 ip를 써준다.



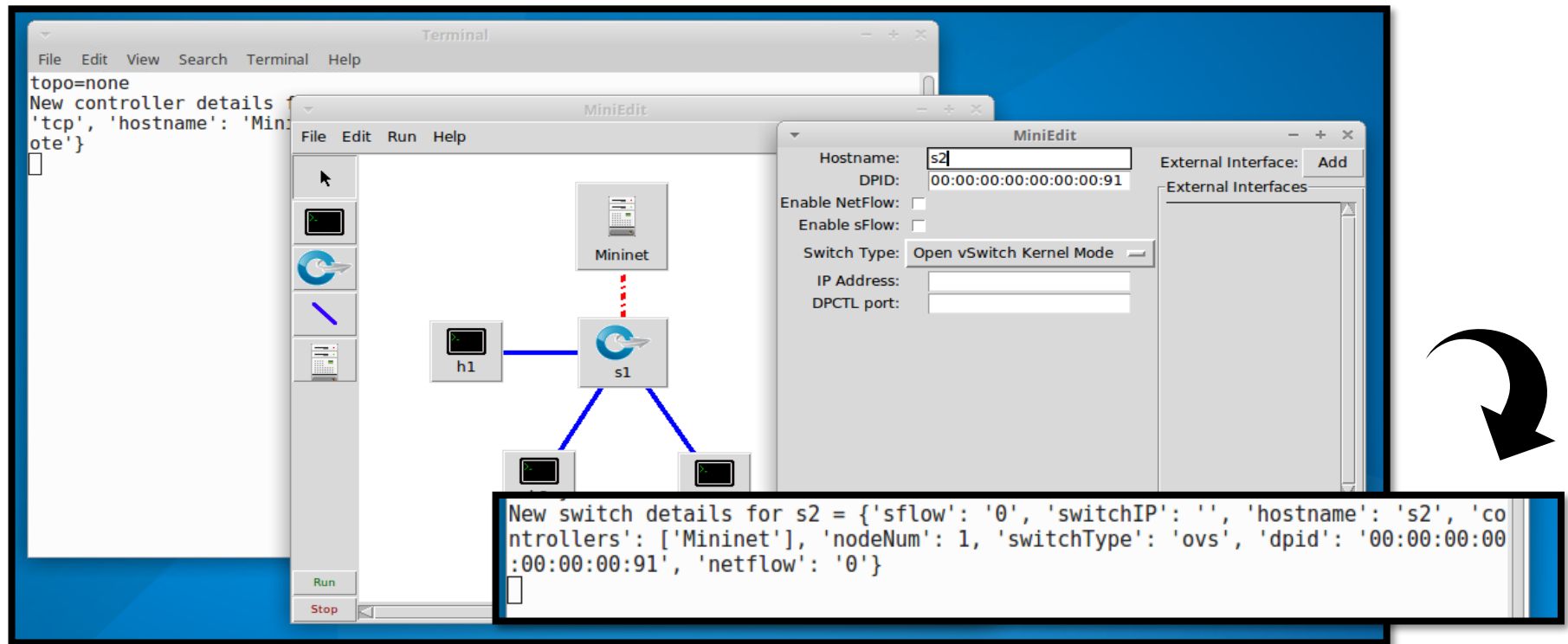
서버1에서는 floodlight controller를 연결하기에, mininet을 올리기 전에 floodlight를 켜야 한다.

Miniedit 구동 및 Controller 연동하기

2-3) Miniedit 기본 틀 만들기 (서버1, 2,3 공통의 경우)

- mininet.mn 기본 틀에서 Switch를 설정해준다.

➤ 그림 예시) Switch Hostname : s2 / DPID : 00:00:00:00:00:00:91 -> 실제로는 서버1은 12, 서버2는 10, 서버3은 11로 잡아준다.



* 실제 각 서버 Switch ID

서버1 : 00:00:00:00:00:00:00:12 / 서버2 : 00:00:00:00:00:00:00:10 / 서버3: 00:00:00:00:00:00:00:11

Miniedit 구동 및 Controller 연동하기

3) Miniedit 구동하기 (서버2,3의 경우)

- 다시 설정된 mininet.mn 를 run을 눌러 실행해준다.

- 그 다음 Terminal에서 pingall을 통해 호스트들끼리 통신이 가능하도록 만들어준다.
- h1에서 h2로 ping 했을 시 가능하다는 것을 볼 수 있다.

```
Terminal
File Edit View Search Terminal Help
Getting controller selection:remote
Getting Links.
*** Configuring hosts
h3 h2 h1
**** Starting 1 controllers
Mininet
**** Starting 1 switches
s2
No NetFlow targets specified.
No sFlow targets specified.

NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exiting will prevent MiniEdit from quitting and will prevent you from starting the network again during this session.

*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h3 -> h2 h1
h2 -> h3 h1
h1 -> h3 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Ping이 제대로 되고 있음!

Floodlight있는 서버1의 경우에도 마찬가지로
진행해주되, controller를 **floodlight**로 바꾼다.

```
"Host: h1"
collisions:0 txqueuelen:1000
RX bytes:86584 (86.5 KB) TX bytes:76262 (76.2 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1

root@Hali-Linux:~/workspace# h1 ping h2
h1: command not found
root@Hali-Linux:~/workspace# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=168 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.182 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.044 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.044/56.210/168.406/79.334 ms
root@Hali-Linux:~/workspace#

"Host: h2"
root@Hali-Linux:~/workspace# ifconfig
eth0: Link encap:Ethernet HWaddr 6e:62:17:ac:90:c3
inet addr:10.0.0.2 Bcast:10.255.255.255 Mask:255.0.0.0
inet6 addr: fe80::6e62:17ff:feac:90c3/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:938 errors:0 dropped:124 overruns:0 frame:0
TX packets:799 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:87552 (87.5 KB) TX bytes:76598 (76.5 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

root@Hali-Linux:~/workspace#
```

3. floodlight ACL, Firewall 시험

floodlight ACL, Firewall 시험

1) Floodlight Web을 접속해서 연동된 switch확인하기

- 미리 구성한 서버1의 hali-linux에 설치된 Floodlight Web을 열어 서버 3개가 switch 연동되었는지 확인한다.

- 서버1,2,3이 전부 Floodlight Controller에 연결되었음을 알 수 있다.
- 서버1 - floodlight controller (192.168.0.63)
- 서버2,3 - remote (192.168.0.63 / 192.168.0.114)

The screenshot displays the Floodlight OpenFlow Controller web interface. The main content area is titled 'Switches' and contains a table of 'Switches Connected'. A red dashed box highlights the first three rows of this table, which correspond to the servers mentioned in the text. A red arrow points from the 'Switches' menu item in the left sidebar to the table. Below the switches table is a 'Switch Roles' table showing the roles assigned to the switches.

Switch ID	IPv4 Address	Connected Since
00:00:00:00:00:00:10	/192.168.0.114:60080	Tue Jan 22 2019 01:37:01 GMT+0900 (Korean Standard Time)
00:00:00:00:00:00:11	/192.168.0.63:35966	Tue Jan 22 2019 01:36:48 GMT+0900 (Korean Standard Time)
00:00:00:00:00:00:12	/192.168.0.185:38900	Tue Jan 22 2019 01:36:39 GMT+0900 (Korean Standard Time)

Showing 1 to 3 of 3 entries

Switch MAC	Role
00:00:00:00:00:00:11	MASTER
00:00:00:00:00:00:12	MASTER
00:00:00:00:00:00:10	MASTER

floodlight ACL, Firewall 시험

2-1) Floodlight Web에서 Firewall을 설정해서 시험 및 검증하기 (서버1 예시)

-Firewall 을 들어가기 전에 먼저 terminal에서 방화벽 규칙을 만들어준다. (CLI환경으로 진행)

- 한 IP당 총 3가지 종류의 규칙을 추가해준다.
- 10.0.0.1, 10.0.0.2, 10.0.0.3 에 해당 규칙을 모두 적용해준다. (총 9가지 규칙)

```
curl -X POST -d '{"src-ip": "10.0.0.1/32", "dst-ip": "10.0.0.1/32", "dl-type":"ARP" }'  
http://localhost:8080/wm/firewall/rules/json  
curl -X POST -d '{"src-ip": "10.0.0.1/32", "dst-ip": "10.0.0.1/32", "nw-proto":"TCP" }'  
http://localhost:8080/wm/firewall/rules/json  
curl -X POST -d '{"src-ip": "10.0.0.1/32", "dst-ip": "10.0.0.1/32", "nw-proto":"TCP", "tp-dst":"80",  
"action":"ALLOW" }' http://localhost:8080/wm/firewall/rules/json
```

1. ARP 허용
2. TCP 허용
3. 80 TCP 허용

ID	Chain	InPort	Source	Dest	DL	Source	MaskBit	Dest	MaskBit	Protocol	Source	Dest	Pri	Act	Delete
58439643		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.2	32	10.0.0.1	32	6	0	0	0	ALLOW	Delete
112927382		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.3	32	10.0.0.1	32	6	0	0	0	ALLOW	Delete
606601344		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.3	32	6	0	80	0	ALLOW	Delete
673892316		-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.1	32	10.0.0.3	32	0	0	0	0	ALLOW	Delete
827061531		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.2	32	6	0	0	0	ALLOW	Delete
1320057669		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.2	32	10.0.0.1	32	6	0	80	0	ALLOW	Delete
1374545408		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.3	32	10.0.0.1	32	6	0	80	0	ALLOW	Delete
1650171158		-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.3	32	6	0	0	0	ALLOW	Delete
1770788769		-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.2	32	10.0.0.1	32	0	0	0	0	ALLOW	Delete
1825276508		-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.3	32	10.0.0.1	32	0	0	0	0	ALLOW	Delete

규칙이 만들어짐!

floodlight ACL, Firewall 시험

2-1) Floodlight Web에서 Firewall을 설정해서 시험 및 검증하기 (서버1 예시)

- h3(10.0.0.3)을 http 테스트용 웹서버로 만들어 두고, h1(10.0.0.1)에서 접속이 되는지 확인한다.

➢ 방화벽에서 10.0.0.3에 대한 접근을 허용하는 규칙이 Firewall에 있으므로 h1에서 h3로 curl이 가능하다.

The screenshot shows the Floodlight Firewall configuration interface. The Firewall status is 'Active'. A table of Firewall Rules is displayed, showing a rule that allows traffic from 10.0.0.2 to 10.0.0.1. A terminal window on the right shows the execution of a curl command from h1 to h3, which is successful. Another terminal window shows the execution of a ping command from h1 to h3, which is also successful. A third terminal window shows the execution of a python command to start a SimpleHTTPServer on h3, which is also successful.

Firewall

Active
Firewall Status [Change](#)

Add New Rule

Firewall Rules Table

ID	Switch	InPort	Source	Dest.	DL	Source IP	MaskBit	Dest. IP	MaskBit
58439643	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.2	32	10.0.0.1	32
112927382	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.3	32	10.0.0.1	32
606601344	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.3	32
673892316	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.1	32	10.0.0.3	32
827061531	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.2	32
1320057669	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.2	32	10.0.0.1	32
1374545408	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.3	32	10.0.0.1	32
1650171158	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2048	10.0.0.1	32	10.0.0.3	32
1770788769	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.2	32	10.0.0.1	32
1825276508	00:00:00:00:00:00:00	-1	00:00:00:00:00:00	00:00:00:00:00:00	2054	10.0.0.3	32	10.0.0.1	32

가능!

```
root@h1:~# curl 10.0.0.3
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="Floodlight.mn">Floodlight.mn</a>
<li><a href="minidit.py">minidit.py</a>
<li><a href="Mininet.mn">Mininet.mn</a>
<li><a href="Opendaylight.mn">Opendaylight.mn</a>
<li><a href="POX.mn">POX.mn</a>
<li><a href="POX_DHCP.mn">POX_DHCP.mn</a>
<li><a href="POX_L2_Multi.mn">POX_L2_Multi.mn</a>
<li><a href="POX_L3.mn">POX_L3.mn</a>
<li><a href="POX_SLB.mn">POX_SLB.mn</a>
<li><a href="POX_SSL.mn">POX_SSL.mn</a>
<li><a href="RYU.mn">RYU.mn</a>
</ul>
</body>
</html>
root@h1:~#
```

```
File "/usr/lib/python2.7/runpy.py", line 72, in _run_code
exec code in run_globals
File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <module>
test()
root@h1:~#
```

```
root@h1:~# ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data:
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.026 ms
^C
--- 10.0.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.026/0.041/0.057/0.016 ms
root@h1:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
^C
--- 10.0.0.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1007ms
root@h1:~# python -m SimpleHTTPServer 80
No command 'python' found, did you mean:
Command 'python' from package 'python-minimal' (main)
Command 'python' from package 'python3' (main)
python command not found
root@h1:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.1 - - [16/Jan/2019 14:44:05] "GET / HTTP/1.1" 200 -
```

floodlight ACL, Firewall 시험

2-2) Floodlight Web에서 Access Control List를 설정해서 시험 및 검증하기 (서버1 예시)

- ACL에 규칙 하나를 추가해서 웹서버로 만든 host2를 차단한 후, 이 후에 host1 터미널에서 curl이 되는지 시험해본다.
 - 먼저 h2, h3를 둘다 테스트용 http 웹서버로 만든 후, Floodlight ACL에서 [Add New]를 눌러 h2으로의 80 포트 접근을 차단하는 규칙을 추가.
 - 추가 후에 차단되지 않은 h3(10.0.0.3)과 차단된 h2(10.0.0.2)를 curl 명령어로 시험해본다.
 - 결과 : 차단 후, h3는 여전히 curl 명령어가 들지만, h2는 curl 명령어가 들지 않는다.

The screenshot shows the Floodlight OpenFlow Controller web interface. The left sidebar contains navigation links: Controller (Home), Switches, Hosts, Links, Topology, Firewall, Access Control Lists, Statistics, and Change Controllers. The main content area is titled 'Access Control Lists' and features an 'Add New' button and a 'Delete All' button. Below these is a table with the following data:

ID	Source	Dest	Source IP	Mask	Dest IP	Mask	Port	Protocol	Action
1		10.0.0.2/32	0	0	167772162	32	6	80	DENY

Below the table, it says 'Showing 1 to 1 of 1 entries'. To the right of the table, there is a terminal window titled 'Host: h1' showing the following commands and output:

```
root@Hali-Linux:~/workspace# curl 10.0.0.3
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="Floodlight.mn">Floodlight.mn</a>
<li><a href="miniedit.py">miniedit.py</a>
<li><a href="Mininet.mn">Mininet.mn</a>
<li><a href="Opendaylight.mn">Opendaylight.mn</a>
<li><a href="POX.mn">POX.mn</a>
<li><a href="POX_DHCP.mn">POX_DHCP.mn</a>
<li><a href="POX_L2_Multi.mn">POX_L2_Multi.mn</a>
<li><a href="POX_L3.mn">POX_L3.mn</a>
<li><a href="POX_SLB.mn">POX_SLB.mn</a>
<li><a href="POX_SSL.mn">POX_SSL.mn</a>
<li><a href="RYU.mn">RYU.mn</a>
</ul>
</body>
</html>
root@Hali-Linux:~/workspace# curl 10.0.0.2
```

The terminal output for the second curl command is cut off, and a red dashed box highlights the command and the text 'curl이 안됨' (curl is not working) next to it.

floodlight ACL, Firewall 시험

2-3) Floodlight Web에서 Access Control List를 설정해서 시험 및 검증하기 (서버1)

- ACL에서 h1인 10.12.0.1에서 80포트로 나가는 걸 차단한 후, 이 후에 h1에서 ping과 curl이 되는지 확인한다.
 - 먼저 10.12.0.3를 테스트용 http 웹서버로 만든 후, Floodlight ACL에서 Source IP에서 80 포트로 나가는 걸 차단하는 규칙을 추가.
 - 추가 후에 전과 후를 ping과 curl 명령어로 시험해본다.

The screenshot shows the Floodlight ACL configuration interface and a terminal window. The ACL configuration is for a rule that denies traffic from source IP 10.12.0.1/32 to destination IP 10.12.0.3 on port 80. The terminal output shows the results of a ping test from h1 to 10.12.0.3, which fails with 100% packet loss.

Access Control Lists

127.0.0.1:8080

Control List

ID	Source	Dest	Source IP	Mask	Dest IP	Mask
No data available in table						

Showing 0 to 0 of 0 entries

Create Rule

Protocol: ☒ TCP ☐ UDP ☐ ICMP

Source IPv4: 10.12.0.1/32

Dest IPv4: A.B.C.D/M

Dest TP: 80

Action: ☐ ALLOW ☒ DENY

Cancel Create

Host: h1

```
root@h1:~# ping -c 1 10.12.0.3
PING 10.12.0.3 (10.12.0.3) 56(84) bytes of data:
2 packets transmitted, 0 received, 100% packet loss, time 1007ms
```

규칙 없을 시, 80포트를 통해 10.12.0.3으로 Ping이 나간다.

이번에는 Dest IPv4를 차단하지 않고 Source IPv4를 차단하기로 한다.

floodlight ACL, Firewall 시험

2-3) Floodlight Web에서 Access Control List를 설정해서 시험 및 검증하기 (서버1)

- ACL 추가 후, 10.12.0.3으로 나가기 위해 h1에서 ping할 경우, ping 은 되지만 curl은 불가하다.

➤ 그러나 규칙에 해당되지 않는 h2에서는 10.12.0.3으로 ping과 curl이 가능하다.

The screenshot displays the Floodlight web interface for configuring Access Control Lists (ACLs) and two terminal windows showing network tests.

Access Control Lists Interface:

- Buttons: Add New, Delete All
- Control List Table:

ID	Source	Dest	Source IP	Mask	Dest IP	Mask	Prot.	Dest TP	Act.	Delete
2	10.12.0.1/32		168558593	32	0	0	6	80	DENY	Delete

Showing 1 to 1

Host: h2 Terminal:

```
root@Hali-Linux:~/workspace# curl 10.12.0.3
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for </h2>
<hr>
<ul>
<li><a href="Floodlight.mn">Floodlight.mn</a>
<li><a href="miniedit.py">miniedit.py</a>
<li><a href="Mininet.mn">Mininet.mn</a>
<li><a href="Mininet_f.mn">Mininet_f.mn</a>
<li><a href="Opendaylight.mn">Opendaylight.mn</a>
<li><a href="POX.mn">POX.mn</a>
<li><a href="POX_DHCP.mn">POX_DHCP.mn</a>
<li><a href="POX_L2_Multi.mn">POX_L2_Multi.mn</a>
<li><a href="POX_L3.mn">POX_L3.mn</a>
<li><a href="POX_SLB.mn">POX_SLB.mn</a>
<li><a href="POX_SSL.mn">POX_SSL.mn</a>
<li><a href="RYU.mn">RYU.mn</a>
</ul>
<hr>
</body>
</html>
root@Hali-Linux:~/workspace#
```

Host: h1 Terminal:

```
root@Hali-Linux:~/workspace# ping 10.12.0.3
PING 10.12.0.3 (10.12.0.3) 56(84) bytes of data:
64 bytes from 10.12.0.3: icmp_seq=1 ttl=64 time=3.58 ms
64 bytes from 10.12.0.3: icmp_seq=2 ttl=64 time=0.292 ms
64 bytes from 10.12.0.3: icmp_seq=3 ttl=64 time=0.038 ms
64 bytes from 10.12.0.3: icmp_seq=4 ttl=64 time=0.039 ms
64 bytes from 10.12.0.3: icmp_seq=5 ttl=64 time=0.040 ms
^C
--- 10.12.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.038/0.798/3.583/1.395 ms
root@Hali-Linux:~/workspace# curl 10.12.0.3
```

floodlight ACL, Firewall 시험

2-3) Floodlight Web에서 Access Control List를 설정해서 시험 및 검증하기 (서버2)

- ACL에서 h1인 10.11.0.1에서 80포트로 나가는 걸 차단한 후, 이 후에 h1에서 ping과 curl이 되는지 확인한다.
 - 먼저 h3(10.11.0.3)를 테스트용 http 웹서버로 만든 후, Floodlight ACL에서 source IP에서 80 포트로 나가는 걸 차단하는 규칙을 추가.
 - 추가 후에 전과 후를 curl 명령어로 시험해보면 규칙 추가 시, h1에서 더 이상 다른 호스트인 h3(10.11.0.3)로 curl할 수 없게 된다.

Access Control Lists

[Add New](#) [Delete All](#)

Control List

ID	Source	Dest	Source IP	Mask	Dest IP	Mask	Prot.	Dest TP	Act.	Delete
4	10.11.0.1/32		168493057	32	0	0	1	0	DENY	Delete

Showing 1 to 1 of 1 entries

규칙 추가 전

규칙 추가 후

```
Host: h1
root@Hali-Linux:~/workspace# curl 10.11.0.3
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="Floodlight_mn">Floodlight_mn</a>
<li><a href="miniedit.py">miniedit.py</a>
<li><a href="Mininet_mn">Mininet_mn</a>
<li><a href="mininet_1_mn">mininet_1_mn</a>
<li><a href="Mininet_ajp_mn">Mininet_ajp_mn</a>
<li><a href="Opendaylight_mn">Opendaylight_mn</a>
<li><a href="POX_mn">POX_mn</a>
<li><a href="POX_DHCP_mn">POX_DHCP_mn</a>
<li><a href="POX_L2_Multi_mn">POX_L2_Multi_mn</a>
<li><a href="POX_L3_mn">POX_L3_mn</a>
<li><a href="POX_SLB_mn">POX_SLB_mn</a>
<li><a href="POX_SSL_mn">POX_SSL_mn</a>
<li><a href="RYU_mn">RYU_mn</a>
</ul>
</body>
</html>
root@Hali-Linux:~/workspace# curl 10.11.0.3
^C
root@Hali-Linux:~/workspace#
```

```
Host: h3
root@Hali-Linux:~/workspace# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.11.0.1 - - [22/Jun/2019 10:54:59] "GET / HTTP/1.1" 200 -
[]
```

floodlight ACL, Firewall 시험

2-3) Floodlight Web에서 Access Control List를 설정해서 시험 및 검증하기 (서버3)

- ACL에서 h1인 10.10.0.1에서 80포트로 나가는 걸 차단한 후, 이 후에 h1에서 ping과 curl이 되는지 확인한다.
 - 먼저 h3(10.10.0.3)를 테스트용 http 웹서버로 만든 후, Floodlight ACL에서 source IP에서 80 포트로 나가는 걸 차단하는 규칙을 추가.
 - 추가 후에 전과 후를 curl 명령어로 시험해보면 규칙 추가 시, h1에서 더 이상 다른 호스트인 h3(10.10.0.3)로 curl할 수 없게 된다.

The screenshot displays the Floodlight Web interface for configuring Access Control Lists (ACLs). A green notification box in the top right corner states "Rule Added! Rule successfully added." Below this, the "Access Control Lists" section shows a table with one entry:

ID	Source	Dest	Source IP	Mask	Dest IP	Mask	Prot.	Dest TP	Act.	Delete
7	10.10.0.1/32		168427521	32	0	0	6	80	DENY	Delete

Below the table, it says "Showing 1 to 1 of 1 entries".

Overlaid on the bottom right are two terminal windows. The left window, titled "Host: h1", shows the command `curl 10.10.0.3` being executed, resulting in a directory listing for `10.10.0.3`. The right window, titled "Host: h3", shows the command `python -m SimpleHTTPServer 80` being executed, resulting in a message: `Serving HTTP on 0.0.0.0 port 80 ... 10.10.0.1 - - [22/Jan/2019 11:04:03] "GET / HTTP/1.1" 200 -`.

Blue dashed boxes and red dashed boxes highlight the "Host: h1" terminal window before and after the rule addition, respectively.

규칙 추가 전

규칙 추가 후

4. WireShark 결과 측정 및 분석

Wireshark 결과 측정 및 분석

1) Wireshark에서 각 서버의 측정결과 확인하기 (서버1 예시)

- 위의 일련의 과정들을 모두 완료 후에 Remote서버에서 wireshark를 확인하기.

- 원래는 아래의 사진과 같이 icmp Openflow가 나오면 안된다! (불안정한 환경으로 인해 간혹 측정되는 희귀한 경우가 있음-오류)
- Floodlight controller에서는 SSL통신을 하기 때문에 icmp가 wireshark에서 측정될 수 없다!

The image shows a Wireshark capture window. The top pane displays a list of captured packets. A red dashed box highlights a specific entry in the packet list:

No.	Time	Source	Destination	Protocol	Length	Info
18990	52.469767076	192.168.0.185	192.168.0.63	OpenFlow	206	Type: OFPT_PACKET_IN
18993	52.471546876	192.168.0.63	192.168.0.185	OpenFlow	204	Type: OFPT_PACKET_OUT
18995	52.471752795	192.168.0.185	192.168.0.63	OpenFlow	206	Type: OFPT_PACKET_IN
18997	52.474681464	192.168.0.63	192.168.0.185	OpenFlow	204	Type: OFPT_PACKET_OUT

Below the packet list, the packet details pane shows the selected packet (No. 18995) selected. The details are as follows:

- Frame 18995: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface 0
- Ethernet II, Src: Vmware_b9:f7:20 (00:0c:29:b9:f7:20), Dst: Vmware_b6:71:3c (00:0c:29:b6:71:3c)
- Internet Protocol Version 4, Src: 192.168.0.185, Dst: 192.168.0.63
- ICMP Echo (ping) request, Id: 0x1777 (6007)
- Type: 8 (Echo (ping))
- Code: 0 (No error)
- Checksum: 0x9fb8 [validation disabled]
- Header checksum status: Unverified
- Source: 192.168.0.185
- Destination: 192.168.0.63
- [Source GeoIP: Unknown]
- [Destination GeoIP: Unknown]
- Transmission Control Protocol, Src Port: 51136, Dst Port: 6653, Seq: 333, Ack: 3315, Len: 140

The packet bytes pane shows the raw data of the packet, including the Ethernet II header, Internet Protocol header, and ICMP Echo request data.

원래는 나올 수 없는 icmp측정값!

감사합니다