LAPORAN MODUL

ARSITEKTUR JARINGAN TERKINI

DISUSUN OLEH:

MUHAMAD MIFTAHUR R	**155150200111174**
VITO KURNIAWAN S	**155150200111180**
DARMADANI KYAT M	**155150207111105**

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

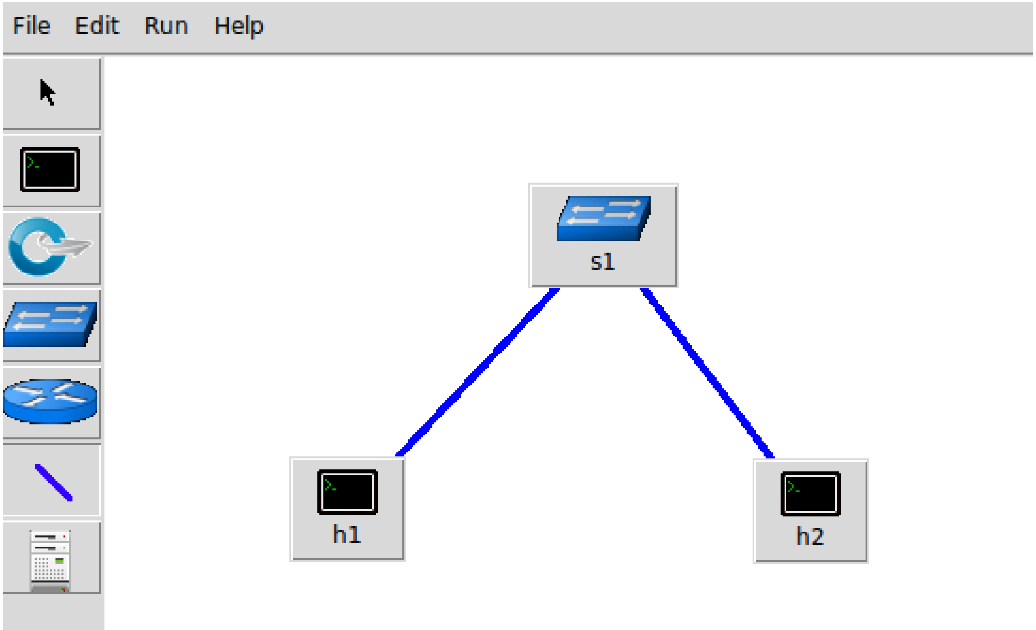
2018

ARSITEKTUR JARINGAN TERKINI

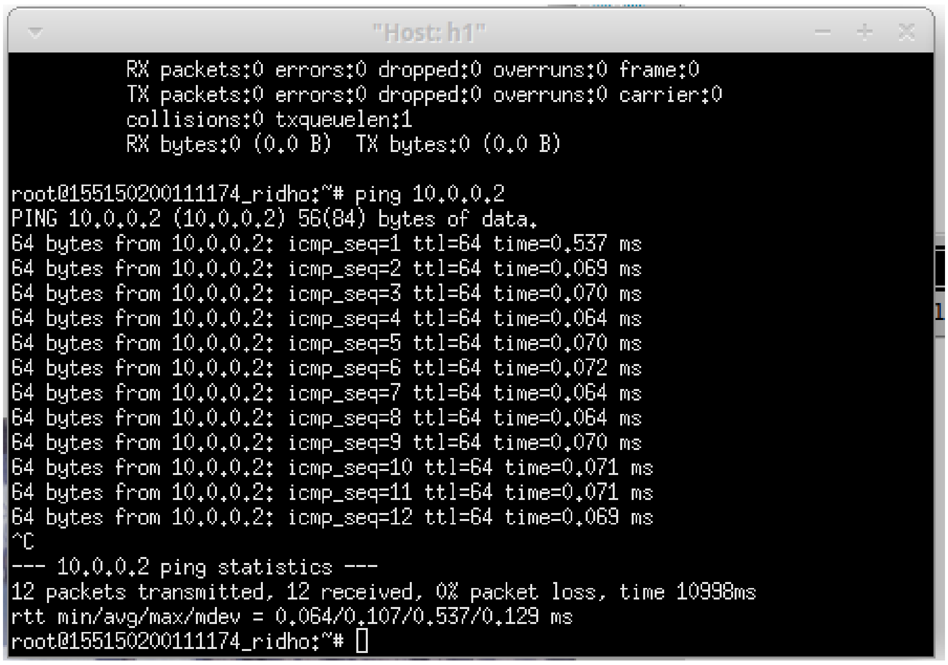
TUGAS : IMPLEMENTASI SOFTWARE DEFINED NETWORKING
NAMA : MUHAMAD MIFTAHUR RIDHOILAH
NIM : 155150200111174

Laporan Modul Implementasi Software Defined Networking(SDN) LAB

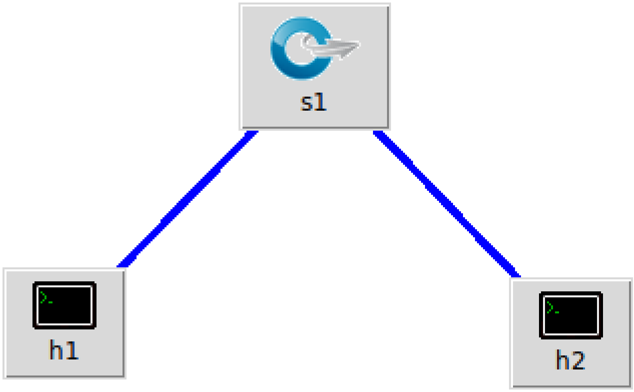
1. Bentuk Topologi



Ya kedua dapat terhubung. Terlihat dari h1 dapat melakukan ping ke h2. Hal ini terjadi karena pada switch tradisional control plane yang berfungsi untuk membuat forwarding table tidak terpisah dengan data plane switch, sehingga switch dapat melakukan forwarding dari h1 ke h2.



2. Bentuk Topologi dengan switch OpenFlow



Tidak dapat terhubung karena switch OpenFlow tidak memiliki Control Plane sehingga switch OpenFlow tidak bisa membentuk forwarding table, tanpa forwarding table, switch tidak bisa melakukan forwarding paket

```
root@155150200111174_ridho:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
8 packets transmitted, 0 received, +6 errors, 100% packet loss, time 7039ms
pipe 3
root@155150200111174_ridho:~#
```

3. Saat digunakan perintah sudo ovs-ofctl dump-flows s1 tidak ada data flow entries yang keluar

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
[sudo] password for ridou:
ovs-ofctl: s1 is not a bridge or a socket
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
ridou@155150200111174_ridho:~$
```

Fungsi dari dump-flows sendiri adalah untuk menampilkan flow entries dari switch. Flow entries sendiri digunakan sebagai panduan data plane untuk melakukan forwarding dari flow.

```
dump-flows SWITCH          print all flow entries
```

4. Bisa terhubung karena ovs-ofctl add-flow menambahkan flow pada flow entries, flow yang ditambahkan adalah pada switch s1 flow dari port 1 di forward ke port 2.

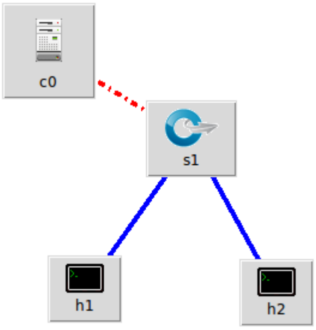
```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=outp
ut:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=outp
ut:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=38.203s, table=0, n_packets=0, n_bytes=0, idle_age=41, in_
port=1 actions=output:2
ridou@155150200111174_ridho:~$
```

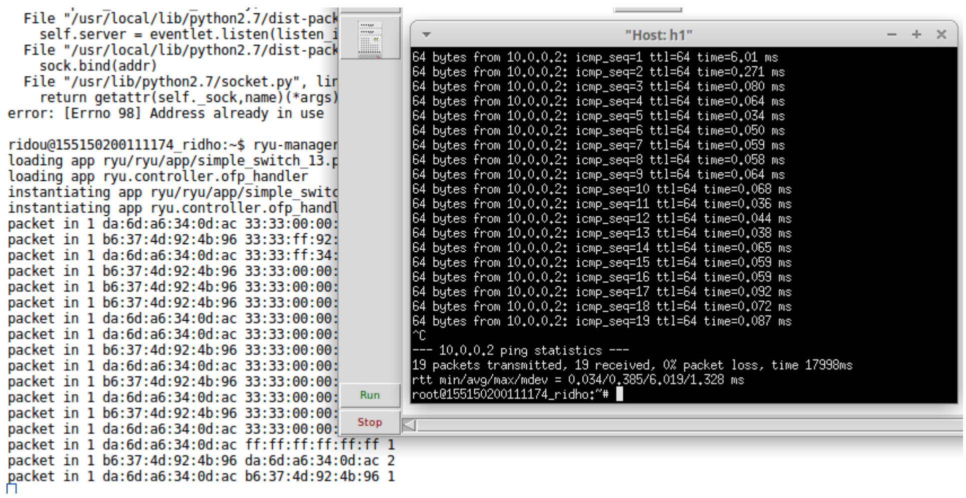
Bukti terhubungnya h1 ke h2 adalah dapat dilakukan ping h1 ke h2

```
mininet> h1 ping -c 4 h2
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=0.445 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.657 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.877 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.445/0.759/1.060/0.233 ms
mininet>
```

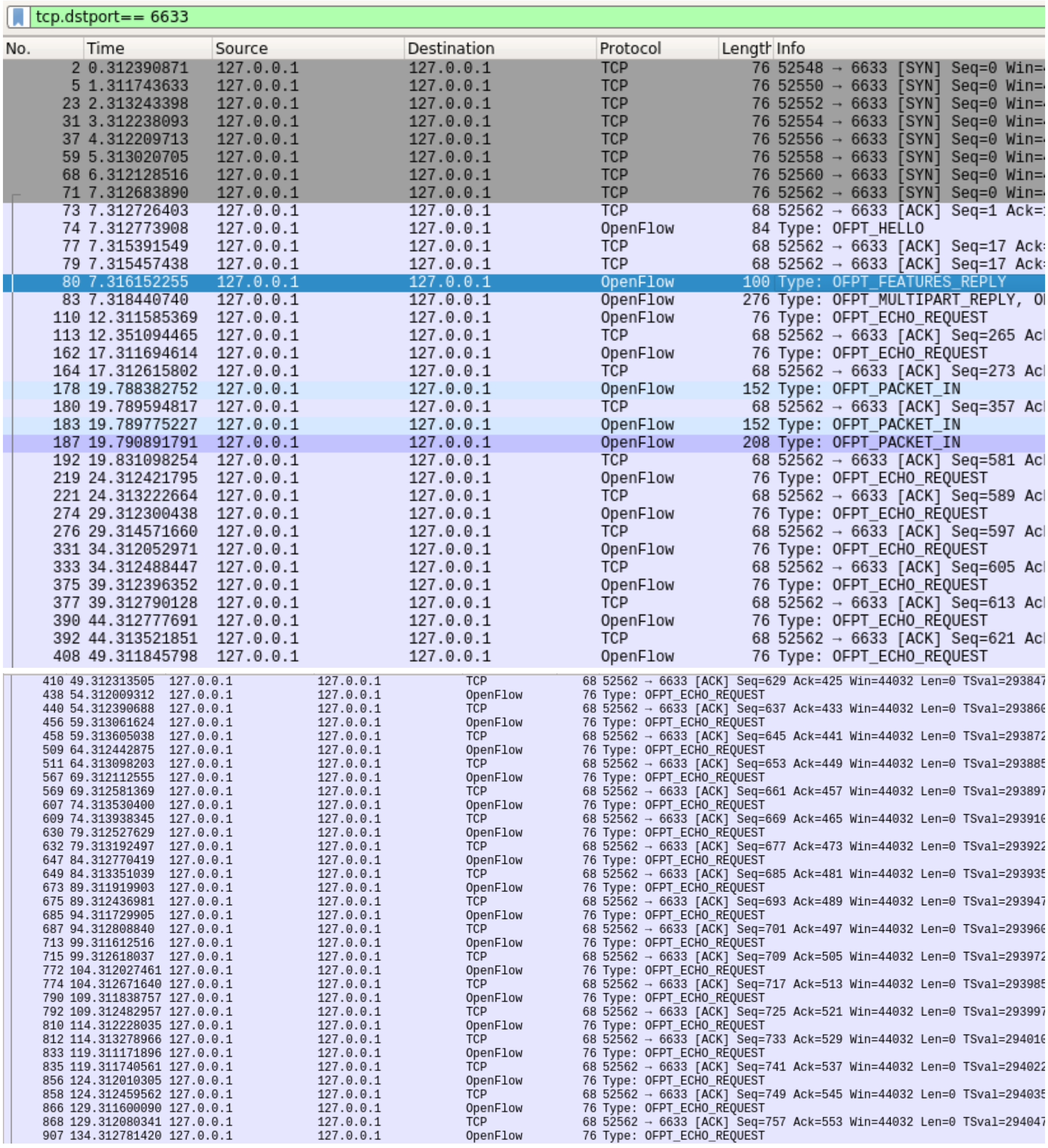
5. Setelah ditambah controller dan mengganti tipe controller dengan remote controller dan preferensi dari protocol, ping dari h1 ke h2 bisa dilakukan. Dalam hal ini sistem membuat flow entries sesuai dengan controller.





Perbedaan dengan yang sebelumnya adalah dengan adanya contoller dan dirubahnya preference. Karena adanya controller, data plane dapat menanyakan kepada controller apabila tidak ada flow entries yang cocok pada data plane.

6. Setelah dilakukan filter sesuai dst port 6633 (port default ryu-controller) protocol yang digunakan adalah protocol TCP dan OpenFlow.



tcp.dstport== 6633

No.	Time	Source	Destination	Protocol	Length	Info
2	0.312390871	127.0.0.1	127.0.0.1	TCP	76	52548 → 6633 [SYN] Seq=0 Win=
5	1.311743633	127.0.0.1	127.0.0.1	TCP	76	52550 → 6633 [SYN] Seq=0 Win=
23	2.313243398	127.0.0.1	127.0.0.1	TCP	76	52552 → 6633 [SYN] Seq=0 Win=
31	3.312238093	127.0.0.1	127.0.0.1	TCP	76	52554 → 6633 [SYN] Seq=0 Win=
37	4.312209713	127.0.0.1	127.0.0.1	TCP	76	52556 → 6633 [SYN] Seq=0 Win=
59	5.313020705	127.0.0.1	127.0.0.1	TCP	76	52558 → 6633 [SYN] Seq=0 Win=
68	6.312128516	127.0.0.1	127.0.0.1	TCP	76	52560 → 6633 [SYN] Seq=0 Win=
71	7.312683890	127.0.0.1	127.0.0.1	TCP	76	52562 → 6633 [SYN] Seq=0 Win=
73	7.312726403	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=1 Ack=
74	7.312773908	127.0.0.1	127.0.0.1	OpenFlow	84	Type: OFPT_HELLO
77	7.315391549	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=17 Ack
79	7.315457438	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=17 Ack
80	7.316152255	127.0.0.1	127.0.0.1	OpenFlow	100	Type: OFPT_FEATURES_REPLY
83	7.318440740	127.0.0.1	127.0.0.1	OpenFlow	276	Type: OFPT_MULTIPART_REPLY, O
110	12.311585369	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
113	12.351094465	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=265 Ac
162	17.311694614	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
164	17.312615802	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=273 Ac
178	19.788382752	127.0.0.1	127.0.0.1	OpenFlow	152	Type: OFPT_PACKET_IN
180	19.789594817	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=357 Ac
183	19.789775227	127.0.0.1	127.0.0.1	OpenFlow	152	Type: OFPT_PACKET_IN
187	19.790891791	127.0.0.1	127.0.0.1	OpenFlow	208	Type: OFPT_PACKET_IN
192	19.831098254	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=581 Ac
219	24.312421795	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
221	24.313222664	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=589 Ac
274	29.312300438	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
276	29.314571660	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=597 Ac
331	34.312052971	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
333	34.312488447	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=605 Ac
375	39.312396352	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
377	39.312790128	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=613 Ac
390	44.312777691	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
392	44.313521851	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=621 Ac
408	49.311845798	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
410	49.312313505	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=629 Ack=425 Win=44032 Len=0 TSval=293847
438	54.312009312	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
440	54.312390688	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=637 Ack=433 Win=44032 Len=0 TSval=293860
456	59.313061624	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
458	59.313605038	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=645 Ack=441 Win=44032 Len=0 TSval=293872
509	64.312442875	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
511	64.313098203	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=653 Ack=449 Win=44032 Len=0 TSval=293885
567	69.312112555	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
569	69.312581369	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=661 Ack=457 Win=44032 Len=0 TSval=293897
607	74.313530400	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
609	74.313938345	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=669 Ack=465 Win=44032 Len=0 TSval=293910
630	79.312527629	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
632	79.313192497	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=677 Ack=473 Win=44032 Len=0 TSval=293922
647	84.312770419	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
649	84.313351039	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=685 Ack=481 Win=44032 Len=0 TSval=293935
673	89.311919903	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
675	89.312436981	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=693 Ack=489 Win=44032 Len=0 TSval=293947
685	94.311729905	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
687	94.312808840	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=701 Ack=497 Win=44032 Len=0 TSval=293960
713	99.311612516	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
715	99.312618037	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=709 Ack=505 Win=44032 Len=0 TSval=293972
772	104.312027461	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
774	104.312671640	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=717 Ack=513 Win=44032 Len=0 TSval=293985
790	109.311838757	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
792	109.312482957	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=725 Ack=521 Win=44032 Len=0 TSval=293997
810	114.312228035	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
812	114.313278966	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=733 Ack=529 Win=44032 Len=0 TSval=294010
833	119.311171896	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
835	119.311740561	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=741 Ack=537 Win=44032 Len=0 TSval=294022
856	124.312010305	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
858	124.312459562	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=749 Ack=545 Win=44032 Len=0 TSval=294035
866	129.311600090	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
868	129.312080341	127.0.0.1	127.0.0.1	TCP	68	52562 → 6633 [ACK] Seq=757 Ack=553 Win=44032 Len=0 TSval=294047
907	134.312781420	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST

7. Berikut semua paket yang digunakan pada komunikasi antara switch OpenFlow dan controller:
Bukti wireshark (setelah filter berdasarkan protocol openflow_v4):

74	7.312773908	127.0.0.1	127.0.0.1	OpenFlow	84	Type: OFPT_HELLO
76	7.315373347	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_HELLO
78	7.315448008	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_FEATURES_REQUEST
80	7.316152255	127.0.0.1	127.0.0.1	OpenFlow	100	Type: OFPT_FEATURES_REPLY
81	7.318267578	127.0.0.1	127.0.0.1	OpenFlow	84	Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_DESC
82	7.318322812	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_FLOW_MOD
83	7.318440740	127.0.0.1	127.0.0.1	OpenFlow	276	Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC
110	12.311585369	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
112	12.312575494	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
162	17.311694614	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
163	17.312586640	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY
178	19.788382752	127.0.0.1	127.0.0.1	OpenFlow	152	Type: OFPT_PACKET_IN
179	19.789585028	127.0.0.1	127.0.0.1	OpenFlow	108	Type: OFPT_PACKET_OUT
183	19.789775227	127.0.0.1	127.0.0.1	OpenFlow	152	Type: OFPT_PACKET_IN
184	19.790689792	127.0.0.1	127.0.0.1	OpenFlow	172	Type: OFPT_FLOW_MOD
187	19.790891791	127.0.0.1	127.0.0.1	OpenFlow	208	Type: OFPT_PACKET_IN
188	19.793399029	127.0.0.1	127.0.0.1	OpenFlow	172	Type: OFPT_FLOW_MOD
219	24.312421795	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REQUEST
220	24.313196635	127.0.0.1	127.0.0.1	OpenFlow	76	Type: OFPT_ECHO_REPLY

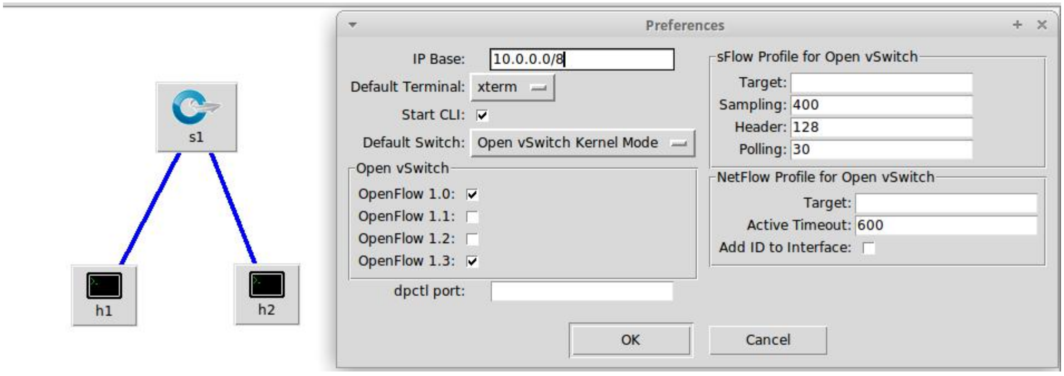
8. Berikut penjelasan semua paket yang digunakan pada komunikasi antara switch OpenFlow dan controller:
    a. Paket-paket 3 way handshake TCP untuk membuat jalur komunikasi.
    b. Paket OFPT_HELLO dari controller dan switch openflow untuk membuat secure channel.
    c. Paket OFPT_FEATURES_REQUEST dari controller untuk meminta daftar fitur yang didukung switch openflow.
    d. Paket OFPT_FEATURES_REPLY dari switch openflow untuk menginformasikan daftar fitur yang didukung switch openflow.
    e. Paket OFPT_PORT_STATUS dari switch openflow untuk menginformasikan status port.
    f. Paket OFPT_MULTIPART_REQUEST dari controller untuk meminta informasi switch, group, port, atau forwarding table switch tersebut.
    g. Paket OFPT_MULTIPART_REPLY dari switch openflow untuk menginformasikan informasi yang dibutuhkan meliputi switch, group, port, atau forwarding table switch tersebut.
    h. Paket OFPMP_PORT_DESC dari controller dan switch openflow untuk saling menginformasikan manufatur hardware dan software yang digunakan.
    i. Paket OFPT_FLOW_MOD dari controller untuk memodifikasi forwarding table.
    j. Paket OFPT_PACKET_IN dari switch controller untuk meneruskan paket jika ada paket yang tidak dikenali untuk diproses oleh controller.
    k. Paket OFPT_PACKET_OUT dari controller untuk meneruskan paket ke switch openflow setelah diproses oleh controller.
    l. Paket OFPT_ECHO_REQUEST dari switch openflow dan OFPT_ECHO_REPLY dari controller untuk saling menginformasikan informasi mengenai koneksi antara keduanya.

ARSITEKTUR JARINGAN TERKINI

TUGAS : BASIC FORWARDING
NAMA : MUHAMAD MIFTAHUR RIDHOILAH
VITO KURNIAWAN SAMPURNO
DARMADANI KYAT MADANA
NIM : 155150200111174
155150200111180
155150207111105

Laporan Basic Forwarding

Berikut topologi dan preferences yang kami gunakan pada percobaan



1. Forwarding bedasarkan port
   Command nya adalah sebagai berikut:

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1 ,actions=out
put:2
ovs-ofctl: 'add-flow' command takes at most 2 arguments
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=outp
ut:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=outp
ut:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=17.686s, table=0, n_packets=0, n_bytes=0, idle_age=17, in_
port=1 actions=output:2
 cookie=0x0, duration=4.235s, table=0, n_packets=0, n_bytes=0, idle_age=4, in_po
rt=2 actions=output:1
ridou@155150200111174_ridho:~$
```

   dump-flows berguna untuk mengecek flow, sementara add-flow berguna untuk mengisi aturan forwarding flow. In_port pada command add-flows menunjukkan port masuk, actions menunjukkan aksi yang akan dilakukan apabila data sesuai rule nya.
   Berikut ping h1 ke h2 setelah dilakukan add-flows

```
mininet> h1 ping h2
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=0.432 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.076 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.069/0.145/0.432/0.143 ms
```

Pada percobaan berikutnya, digunakan action=all pada data yang dikirim dari port 2.

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=17.686s, table=0, n_packets=0, n_bytes=0, idle_age=17, in_
port=1 actions=output:2
 cookie=0x0, duration=4.235s, table=0, n_packets=0, n_bytes=0, idle_age=4, in_po
rt=2 actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=all
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=261.569s, table=0, n_packets=62, n_bytes=5796, idle_age=12
9, in_port=1 actions=output:2
 cookie=0x0, duration=5.836s, table=0, n_packets=0, n_bytes=0, idle_age=129, in_
port=2 actions=ALL
ridou@155150200111174_ridho:~$
```

Fungsi action=all disini berarti untuk data dari port 2 maka akan dikirimkan ke semua yang terhubung ke s1

2. Forwarding bedasarkan ip

Commandnya adalah sebagai berikut

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl del-flows s1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,in_port=1,
ip,nw_dst=10.0.0.2,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,in_port=2,
ip,nw_dst=10.0.0.1,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=49.144s, table=0, n_packets=0, n_bytes=0, idle_age=49, pri
ority=10,ip,in_port=1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=30.603s, table=0, n_packets=0, n_bytes=0, idle_age=30, pri
ority=10,ip,in_port=2,nw_dst=10.0.0.1 actions=output:1
ridou@155150200111174_ridho:~$
```

Command del-flows digunakan untuk menghapus semua rule flow yang ada pada switch. Pada add-flow ditambahkan ip untuk menunjukkan rule menggunakan ip, nw_dst menunjukkan ip tujuan dari data yang dikirim. Untuk melihat ip dari host bisa dilihat dimasing-masing terminal host dengan command ifconfig

Hasil ping h1 ke h2 dan sebaliknya adalah:

```
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3999ms
pipe 3
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
15 packets transmitted, 0 received, +6 errors, 100% packet loss, time 14094ms
pipe 3
mininet>
```

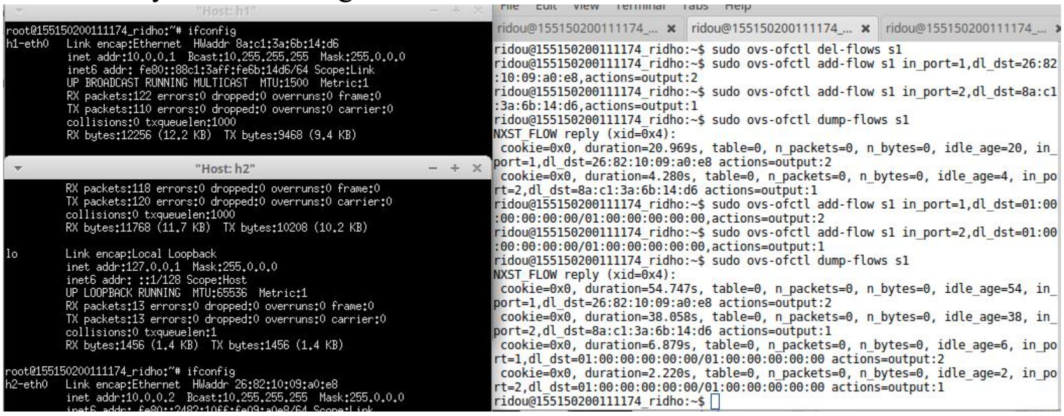
Unreachable terjadi karena switch tidak mengetahui ip dari host yang terhubung. Untuk mengetahui ip digunakan ARP.

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,in_port=1,
ip,arp,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,in_port=2,
ip,arp,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=557.127s, table=0, n_packets=13, n_bytes=1274, idle_age=38
5, priority=10,ip,in_port=1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=538.586s, table=0, n_packets=5, n_bytes=490, idle_age=426,
 priority=10,ip,in_port=2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=28.757s, table=0, n_packets=0, n_bytes=0, idle_age=28, pri
ority=10,arp,in_port=1 actions=output:2
 cookie=0x0, duration=12.220s, table=0, n_packets=0, n_bytes=0, idle_age=12, pri
ority=10,arp,in_port=2 actions=output:1
ridou@155150200111174_ridho:~$
```

Pada argument add-flow, digunakan argument ARP untuk menunjukan bahwa rule tersebut menggunakan ARP. Berikut hasil ping setelah ditambah rule untuk ARP.

```
mininet> h1 ping h2
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=0.793 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.062 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.062/0.309/0.793/0.342 ms
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.042 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.076 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.042/0.062/0.076/0.016 ms
mininet>
```

3. Forwarding menggunakan MAC address
   Commandnya adalah sebagai berikut:



   Pada command add-flow diatas, digunakan dl_dst yang menunjukkan rulenya menggunakan MAC address tujuan dari flow. MAC address dari masing-masing host bisa dilihat diterminal host dengan ifconfig. MAC address ditunjukan oleh HWaddr. Selain MAC address tujuan flow, ditambahkan MAC address untuk broadcast (dl_dst=01.00.00.00.00.00/01.00.00.00.00.00) yang digunakan pertama kali untuk mencari letak MAC address tujuan. Berikut hasil ping h1 ke h2 dan sebalikknya:

```
mininet> h1 ping h2
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=0.441 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.054 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.050/0.181/0.441/0.184 ms
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.063 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.043/0.053/0.063/0.008 ms
mininet>
```

4. Filter (port 80)
   Pada percobaan filter digunakan h2 sebagai server dengan port 80. Berikut command yang dipakai

```
mininet> h1 ping h2
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=0.485 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.070 ms
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.070/0.277/0.485/0.208 ms
mininet> h2 python -m SimpleHTTPServer 80 &
mininet> h1 curl 10.0.0.2
<!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=".bash_history">.bash_history</a>
<li><a href=".bash_logout">.bash_logout</a>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".cache/">.cache/</a>
<li><a href=".code.swp">.code.swp</a>
```

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl del-flows s1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=5,in_port=1,dl_dst=26:82:10:09:a0:e8,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=5,in_port=2,dl_dst=8a:c1:3a:6b:14:d6,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=20.543s, table=0, n_packets=0, n_bytes=0, idle_age=22, in_port=1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2
 cookie=0x0, duration=30.101s, table=0, n_packets=0, n_bytes=0, idle_age=30, priority=5,in_port=1,dl_dst=26:82:10:09:a0:e8 actions=output:2
 cookie=0x0, duration=27.229s, table=0, n_packets=0, n_bytes=0, idle_age=27, priority=5,in_port=2,dl_dst=8a:c1:3a:6b:14:d6 actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 in_port=2,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=45.618s, table=0, n_packets=0, n_bytes=0, idle_age=47, in_port=1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2
 cookie=0x0, duration=3.447s, table=0, n_packets=0, n_bytes=0, idle_age=3, in_port=2,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1
 cookie=0x0, duration=55.176s, table=0, n_packets=0, n_bytes=0, idle_age=55, priority=5,in_port=1,dl_dst=26:82:10:09:a0:e8 actions=output:2
 cookie=0x0, duration=52.304s, table=0, n_packets=0, n_bytes=0, idle_age=52, priority=5,in_port=2,dl_dst=8a:c1:3a:6b:14:d6 actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,tcp,tcp_dst=80,actions=drop
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=91.430s, table=0, n_packets=0, n_bytes=0, idle_age=93, in_port=1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2
 cookie=0x0, duration=49.259s, table=0, n_packets=0, n_bytes=0, idle_age=49, in_port=2,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1
 cookie=0x0, duration=2.566s, table=0, n_packets=0, n_bytes=0, idle_age=2, priority=10,tcp,tp_dst=80 actions=drop
 cookie=0x0, duration=100.988s, table=0, n_packets=0, n_bytes=0, idle_age=100, priority=5,in_port=1,dl_dst=26:82:10:09:a0:e8 actions=output:2
 cookie=0x0, duration=98.116s, table=0, n_packets=0, n_bytes=0, idle_age=98, priority=5,in_port=2,dl_dst=8a:c1:3a:6b:14:d6 actions=output:1
ridou@155150200111174_ridho:~$
```

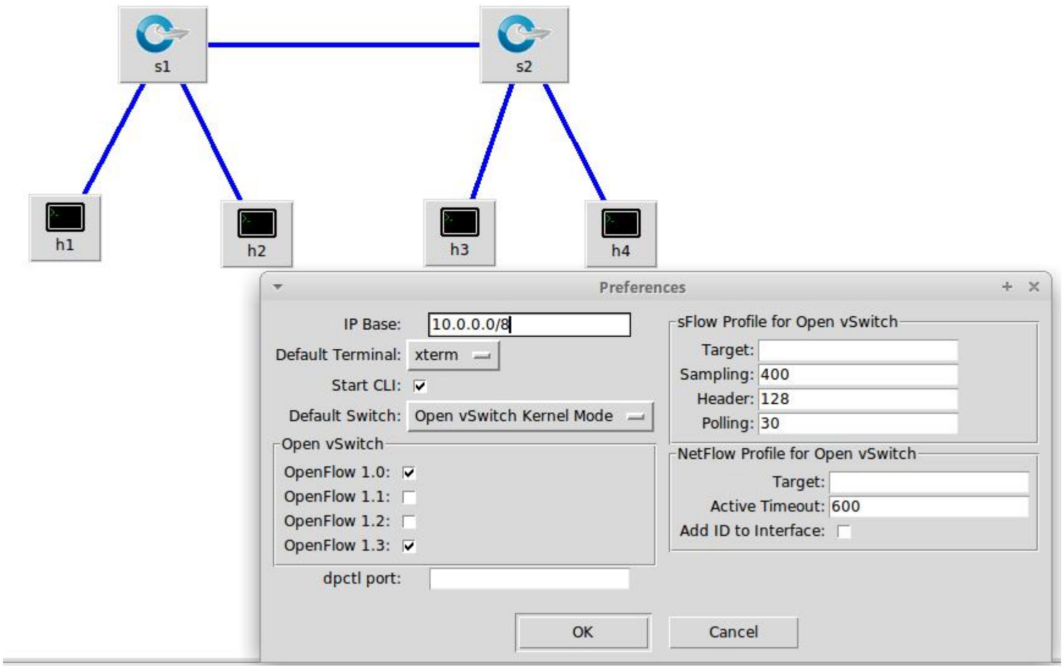
Rule yang dipakai kurang lebih sama dengan rule yang ada pada forwarding flow sesuai MAC address hanya saja dimodifikasi dibagian priority dan ditambah rule untuk mengdrop paket tcp yang bertujuan ke tcp dengan port 80 (server). Rule drop ini memiliki priority lebih tinggi daripada rule forwarding biasa agar rule ini dicek terlebih dahulu. Hasil setelah ditambah rule adalah:

```
mininet> h1 ping h2
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=0.518 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.075 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.066/0.183/0.518/0.193 ms
mininet> h1 curl 10.0.0.2
```

Ping tetap bisa dilakukan karena ping menggunakan protocol ICMP (bukan TCP) sehingga tidak memenuhi rule drop. Rulu drop ditest dengan command curl [ip server], jika tidak ada balasan, maka berhasil di filter.

Tugas Basic Forwarding

Topologi dan preferensi yang digunakan:



1. Untuk menhubungkan semua host, rule yang saya gunakan adalah rule forwarding dengan ip. Berikut command yang dipakai:

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl del-flows s1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl del-flows s2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.3,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.4,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.1,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.1,actions=output:3
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=15,arp,actions=ALL
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s2 priority=10,ip,nw_dst=10.0.0.1,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s2 priority=10,ip,nw_dst=10.0.0.2,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s2 priority=10,ip,nw_dst=10.0.0.3,actions=output:3
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s2 priority=10,ip,nw_dst=10.0.0.4,actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.1,actions=output:2
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=10,ip,nw_dst=10.0.0.2,actions=output:3
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=90.555s, table=0, n_packets=0, n_bytes=0, idle_age=90, priority=15,arp actions=ALL
 cookie=0x0, duration=138.708s, table=0, n_packets=0, n_bytes=0, idle_age=138, priority=10,ip,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=135.017s, table=0, n_packets=0, n_bytes=0, idle_age=135, priority=10,ip,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=12.941s, table=0, n_packets=0, n_bytes=0, idle_age=120, priority=10,ip,nw_dst=10.0.0.1 actions=output:2
 cookie=0x0, duration=6.418s, table=0, n_packets=0, n_bytes=0, idle_age=6, priority=10,ip,nw_dst=10.0.0.2 actions=output:3
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=78.614s, table=0, n_packets=0, n_bytes=0, idle_age=78, priority=10,ip,nw_dst=10.0.0.1 actions=output:2
 cookie=0x0, duration=75.173s, table=0, n_packets=0, n_bytes=0, idle_age=75, priority=10,ip,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=55.959s, table=0, n_packets=0, n_bytes=0, idle_age=55, priority=10,ip,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=42.379s, table=0, n_packets=0, n_bytes=0, idle_age=42, priority=10,ip,nw_dst=10.0.0.4 actions=output:1
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s2 priority=15,arp,actions=ALL
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=2.645s, table=0, n_packets=0, n_bytes=0, idle_age=2, priority=15,arp actions=ALL
 cookie=0x0, duration=96.813s, table=0, n_packets=0, n_bytes=0, idle_age=96, priority=10,ip,nw_dst=10.0.0.1 actions=output:2
 cookie=0x0, duration=93.372s, table=0, n_packets=0, n_bytes=0, idle_age=93, priority=10,ip,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=74.158s, table=0, n_packets=0, n_bytes=0, idle_age=74, priority=10,ip,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=60.578s, table=0, n_packets=0, n_bytes=0, idle_age=60, priority=10,ip,nw_dst=10.0.0.4 actions=output:1
ridou@155150200111174_ridho:~$
```

Forwarding dilakukan dengan melihat ip tujuannya saja. Untuk ip yang terhubung dengan switch yang lain, actions yang dilakukan adalah meneruskan ke port yang terhubung dengan switch lain tersebut. Pada add_flow in_port tidak digunakan karena jika menggunakan in_port dapat terjadi error. Selain rule forwarding dengan ip, ditambah rule ARP untuk mencari tau IP dari semua host yang terhubung (Oleh karena itu menggunakan action=ALL). Priority dari rule ARP ditinggikan untuk memastikan rule ARP terpenuhi. Berikut hasil ping :

```
mininet> h1 ping h3
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=0.693 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.065 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.085 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.065/0.229/0.693/0.267 ms
mininet> h3 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.059 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.059/0.069/0.081/0.011 ms
mininet>
```

2. Untuk melakukan filter pada topologi diatas (h2 tidak bisa akses server h1) yang perlu dilakukan tidak jauh berbeda dengan yang dilakukan pada filter yang sebelumnya. Berikut command yang dipakai:

```
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 curl h1
<!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=".bash_history">.bash_history</a>
<li><a href=".bash_logout">.bash_logout</a>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".cache/">.cache/</a>
<li><a href=".code.swp">.code.swp</a>
<li><a href=".config/">.config/</a>
<li><a href=".dbus/">.dbus/</a>
<li><a href=".dia/">.dia/</a>
<li><a href=".din_philo2.c.swp">.din_philo2.c.swp</a>
<li><a href=".dmrc">.dmrc</a>
<li><a href=".gconf/">.gconf/</a>
<li><a href=".gnome/">.gnome/</a>
<li><a href=".gnome2/">.gnome2/</a>
<li><a href=".gnome2_private/">.gnome2_private/</a>
<li><a href=".gnupg/">.gnupg/</a>
<li><a href=".ICEauthority">.ICEauthority</a>
```

Server ditest dengan menggunakan curl. Setelah itu ditambah rule drop:

```
ridou@155150200111174_ridho:~$ sudo ovs-ofctl add-flow s1 priority=20,in_port=3,tcp,tcp_dst=80,actions=drop
ridou@155150200111174_ridho:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=3.889s, table=0, n_packets=0, n_bytes=0, idle_age=3, priority=20,tcp,in_port=3,tp_dst=80 actions=drop
 cookie=0x0, duration=889.738s, table=0, n_packets=10, n_bytes=420, idle_age=183, priority=15,arp actions=ALL
 cookie=0x0, duration=937.891s, table=0, n_packets=12, n_bytes=1176, idle_age=368, priority=10,ip,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=934.200s, table=0, n_packets=0, n_bytes=0, idle_age=934, priority=10,ip,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=812.124s, table=0, n_packets=28, n_bytes=2472, idle_age=188, priority=10,ip,nw_dst=10.0.0.1 actions=output:2
 cookie=0x0, duration=805.601s, table=0, n_packets=16, n_bytes=6602, idle_age=188, priority=10,ip,nw_dst=10.0.0.2 actions=output:3
ridou@155150200111174_ridho:~$
```

Rule drop yang digunakan mirip dengan rule drop pada filter server yang sebelumnya. Perbedaannya pada rule drop ini ditambah in_port yang menunjukkan port asal paket tcp. Hal ini dilakukan untuk menjaga agar h3 dan h4 yang terhubung melalui port 1 pada s1 tidak ikut difilter. Command rule drop sebelumnya bersifat mengedrop semua paket tcp bertujuan tp port 80, pada filter yang sebelumnya tidak masalah karena topologi hanya berisi 1 server dan 1 client, akan tetapi pada filter kali ini hal tersebut tidak dapat dilakukan karena persyaratan dari tugas, yang di drop adalah yang berasal dari h2 saja. Berikut hasil testnya:

```
mininet> h2 curl h1
^C
mininet> h3 curl h1
<!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=".bash_history">.bash_history</a>
<li><a href=".bash_logout">.bash_logout</a>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".cache/">.cache/</a>
<li><a href=".code.swp">.code.swp</a>
<li><a href=".config/">.config/</a>
<li><a href=".dbus/">.dbus/</a>
<li><a href=".dia/">.dia/</a>
<li><a href=".din_philo2.c.swp">.din_philo2.c.swp</a>
<li><a href=".dmrc">.dmrc</a>
```

ARSITEKTUR JARINGAN TERKINI

TUGAS : DYNAMIC MAC ADDRESSING

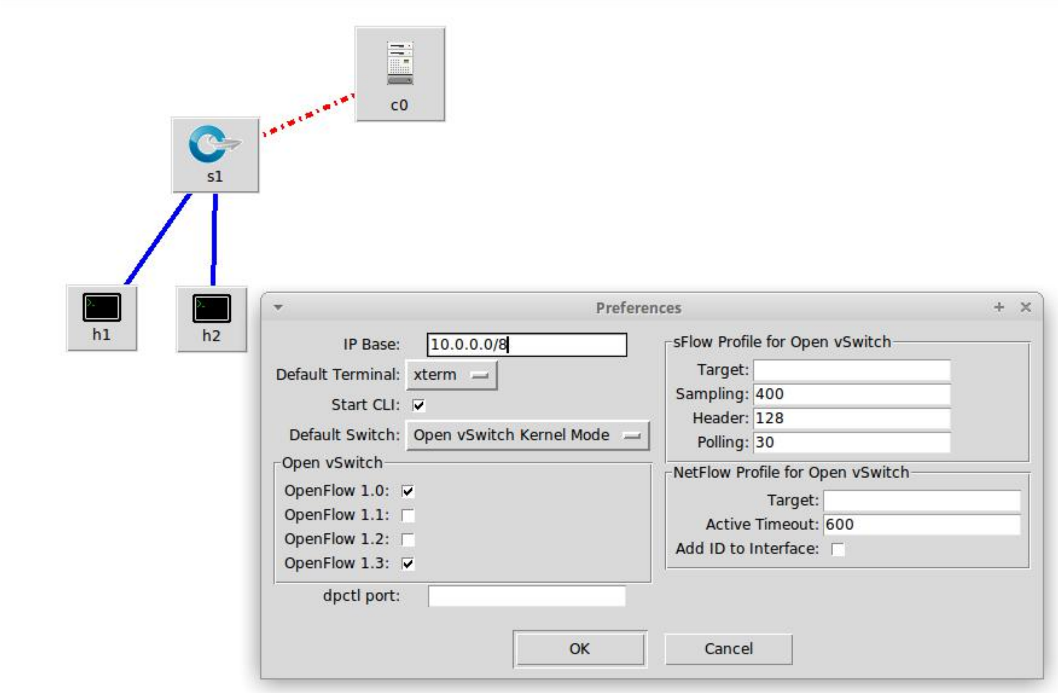
NAMA : MUHAMAD MIFTAHUR RIDHOILAH
VITO KURNIAWAN SAMPURNO
DARMADANI KYAT MADANA

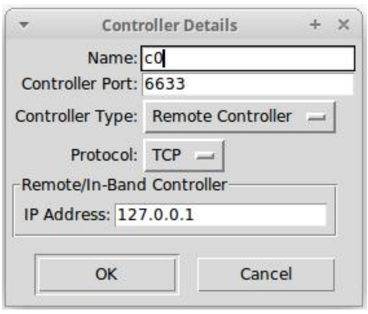
NIM : 155150200111174
155150200111180
155150207111105

Laporan Dynamic MAC Addressing

Percobaan Pertama

1. Berikut topologi dan preferences yang kami gunakan pada percobaan pertama:





Topologi terdiri dari 2 host 1 switch dan 1 remote controller dengan Openflow protocol versi 1.0 .

Source code dari control plane untuk dynamic mac addressing

```python
from ryu.ofproto import ofproto_v1_0
from ryu.lib import mac
from ryu.lib.packet import packet,ethernet

class hub(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(hub,self).__init__(*args, **kwargs)
        self.sat = {}

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        # src = eth.src
        ofproto = dp.ofproto
        data = msg.data

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocol(ethernet.ethernet)
        src = eth.src
        dst = eth.dst
        switch = dp.id

        self.sat.setdefault(switch,{})

        self.sat[switch][src] = msg.in_port

        if dst in self.sat[switch]:
            if dst != src:
                output = self.sat[switch][dst]
        else:
            output = ofproto.OFPP_FLOOD

        actions = [dp.ofproto_parser.OFPActionOutput(output)]
        out = dp.ofproto_parser.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
            in_port=msg.in_port, actions=actions)
        dp.send_msg(out)
```

2. Menjalankan source code dengan ryu-manager

```
ridou@155150200111174_ridho:~$ cd ~/Ryu-python
ridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub5.py
loading app hub5.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub5.py of hub
```

3. Hasil ping h1 ke h2

```
mininet> h1 ping h2
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=5.69 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.86 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.58 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=3.85 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 1.548/3.507/5.690/1.353 ms
mininet>
```

Hasil ping h1 ke h2 besar dengan rata 3.507 ms. Hal ini terjadi karena rule forwarding tidak disimpan pada flow table data plane. Sehingga setiap paket masuk harus di forward ke control plane menyebabkan response time yang tinggi.

4. Lihat dump-flows dari s1

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
ridou@155150200111174_ridho:~/Ryu-python$
```

Table flow dari s1 kosong karena tidak dilakukan add flow pada packet in handler

5. Menambahkan code pada source code
Pada packet in handler ditambah fungsi untuk melakukan add flow ke flow table

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    # src = eth.src
    ofproto = dp.ofproto
    data = msg.data

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocol(ethernet.ethernet)
    src = eth.src
    dst = eth.dst
    switch = dp.id

    self.sat.setdefault(switch,{})

    self.sat[switch][src] = msg.in_port

    if dst in self.sat[switch]:
        if dst != src:
            output = self.sat[switch][dst]
    else:
        output = ofproto.OFPP_FLOOD

    actions = [dp.ofproto_parser.OFPActionOutput(output)]
    out = dp.ofproto_parser.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
        in_port=msg.in_port, actions=actions)
    dp.send_msg(out)

    if output != ofproto.OFPP_FLOOD:
        self.addflow(dp,msg.in_port,actions,dst)
```

Lalu ditambah fungsi addflow

```
def addflow(self,dp,in_port,actions,dst):
    ofproto = dp.ofproto
    match = dp.ofproto_parser.OFPMatch(in_port=in_port, dl_dst=mac.haddr_to_bin(dst))
    mod = dp.ofproto_parser.OFPFlowMod(datapath=dp,match=match,
        priority=ofproto.OFP_DEFAULT_PRIORITY,actions=actions)
    dp.send_msg(mod)
```

Setelah itu dilakukan run source code dengan ryu-manager

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub5.py
loading app hub5.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub5.py of hub
```

Dengan ditambah code add flow maka rule flow akan dimasukkan ke flow table sehingga hasil ping dari h1 ke h2 akan lebih kecil dari sebelumnya.

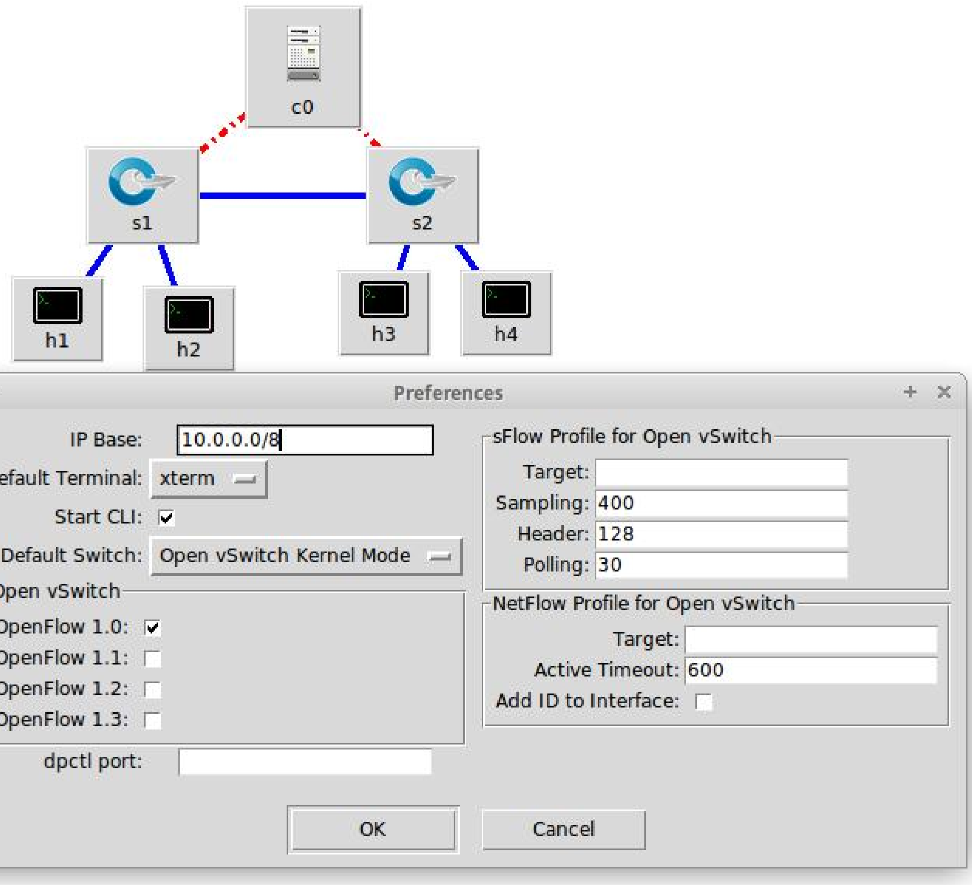
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.55 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.82 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.310 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.081 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.070/1.766/6.550/2.479 ms
mininet>
```

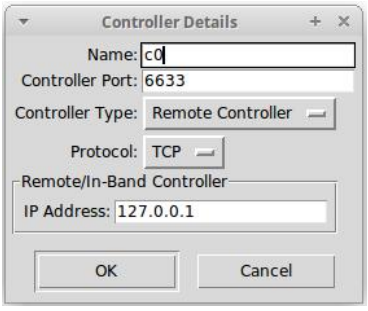
Hasil ping h1 ke h2 menjadi rendah karena rule forwarding disimpan pada flow table data plane. Saat ada paket masuk yang sesuai dengan rule forwarding, maka akan diforward sesuai flow table tanpa dikirim ke control plane. Berikut dump-flows dari s1 :

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=56.103s, table=0, n_packets=5, n_bytes=434, idle_age=51, i
n_port=2,dl_dst=36:ec:35:66:56:ca actions=output:1
 cookie=0x0, duration=55.106s, table=0, n_packets=5, n_bytes=434, idle_age=51, i
n_port=1,dl_dst=1e:c1:4d:d4:9a:da actions=output:2
ridou@155150200111174_ridho:~/Ryu-python$
```

Percobaan Kedua

1. Berikut topologi dan preferences yang kami gunakan pada percobaan kedua:





Topologi terdiri dari 4 host 2 switch dan 1 remote controller dengan Openflow protocol versi 1.0 .

Source code dari control plane untuk dynamic mac addressing 2 switch
Import dan Initialisasi

```python
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_0
from ryu.lib import mac
from ryu.lib.packet import packet,ethernet

class hub(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(hub,self).__init__(*args, **kwargs)
        self.sat = {}
```

Packet in Handler untuk menghandle packet yang masuk

```python
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    # src = eth.src
    ofproto = dp.ofproto
    data = msg.data

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocol(ethernet.ethernet)
    src = eth.src
    dst = eth.dst
    switch = dp.id

    self.sat.setdefault(switch,{})

    self.sat[switch][src] = msg.in_port

    if dst in self.sat[switch]:
        if dst != src:
            output = self.sat[switch][dst]
    else:
        output = ofproto.OFPP_FLOOD

    actions = [dp.ofproto_parser.OFPActionOutput(output)]
    out = dp.ofproto_parser.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
        in_port=msg.in_port, actions=actions)
    dp.send_msg(out)

    if output != ofproto.OFPP_FLOOD:
        self.addflow(dp,msg.in_port,actions,dst)
```

Fungsi untuk melakukan addflow sehingga aturan flow dimasukkan ke flow table

```
def addflow(self,dp,in_port,actions,dst):
    ofproto = dp.ofproto
    match = dp.ofproto_parser.OFPMatch(in_port=in_port, dl_dst=mac.haddr_to_bin(dst))
    mod = dp.ofproto_parser.OFPFlowMod(datapath=dp,match=match,
        priority=ofproto.OFP_DEFAULT_PRIORITY,actions=actions)
    dp.send_msg(mod)
```

2. Menjalankan source code dengan ryu-manager

```
ridou@155150200111174_ridho:~$ cd ~/Ryu-python
ridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub5.py
loading app hub5.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub5.py of hub
```

3. Hasil ping

dari s1 ke s2 (h1 ke h3 dan h2 ke h4)

```
*** Starting CLI:
mininet> h1 ping h3
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=12.4 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.483 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.085 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.073/3.260/12.402/5.280 ms
mininet> h2 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=8.33 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.523 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.053 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.046 ms
^C
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.046/2.240/8.338/3.525 ms
mininet>
```

Hasil kedua dari tiap ping memiliki response rendah karena rule forwarding disimpan pada flow table data plane. Saat ada paket masuk yang sesuai dengan rule forwarding, maka akan diforward sesuai flow table tanpa dikirim ke control plane.

Sebaliknya dari s2 ke s1 (h3 ke h2 dan h4 ke h1)

```
mininet> h3 ping h2
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=4.48 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.282 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.074 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.070/1.228/4.489/1.884 ms
mininet> h4 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=4.96 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.191 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.081 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.081/1.744/4.962/2.275 ms
mininet>
```

Sama seperti hasil ping dari s1 ke s2, hasil ping kedua dari tiap sesi memiliki response time yang rendah karena rule forwarding disimpan pada flow table data plane sehingga pada ping kedua dan seterusnya packet yang sesuai dengan rule flow pada flow table tidak perlu di forward ke control plane terlebih dahulu.

Berikut dump-flows dari s1 dan s2:

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=504.687s, table=0, n_packets=9, n_bytes=770, idle_age=223,
 in_port=3,dl_dst=ca:83:e2:fa:13:4a actions=output:1
 cookie=0x0, duration=504.685s, table=0, n_packets=4, n_bytes=336, idle_age=499,
 in_port=1,dl_dst=26:7b:94:e2:18:0c actions=output:3
 cookie=0x0, duration=493.771s, table=0, n_packets=10, n_bytes=868, idle_age=233
, in_port=3,dl_dst=4e:8d:7c:16:71:b0 actions=output:2
 cookie=0x0, duration=493.770s, table=0, n_packets=4, n_bytes=336, idle_age=488,
 in_port=2,dl_dst=62:20:58:77:e3:1f actions=output:3
 cookie=0x0, duration=238.308s, table=0, n_packets=5, n_bytes=434, idle_age=233,
 in_port=2,dl_dst=26:7b:94:e2:18:0c actions=output:3
 cookie=0x0, duration=228.973s, table=0, n_packets=4, n_bytes=336, idle_age=223,
 in_port=1,dl_dst=62:20:58:77:e3:1f actions=output:3
ridou@155150200111174_ridho:~/Ryu-python$
```

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=478.925s, table=0, n_packets=5, n_bytes=434, idle_age=473,
 in_port=1,dl_dst=ca:83:e2:fa:13:4a actions=output:3
 cookie=0x0, duration=478.919s, table=0, n_packets=10, n_bytes=812, idle_age=207
, in_port=3,dl_dst=26:7b:94:e2:18:0c actions=output:1
 cookie=0x0, duration=468.009s, table=0, n_packets=5, n_bytes=434, idle_age=463,
 in_port=2,dl_dst=4e:8d:7c:16:71:b0 actions=output:3
 cookie=0x0, duration=468.005s, table=0, n_packets=9, n_bytes=714, idle_age=198,
 in_port=3,dl_dst=62:20:58:77:e3:1f actions=output:2
 cookie=0x0, duration=212.543s, table=0, n_packets=4, n_bytes=336, idle_age=207,
 in_port=1,dl_dst=4e:8d:7c:16:71:b0 actions=output:3
 cookie=0x0, duration=203.208s, table=0, n_packets=3, n_bytes=238, idle_age=198,
 in_port=2,dl_dst=ca:83:e2:fa:13:4a actions=output:3
ridou@155150200111174_ridho:~/Ryu-python$
```

```
mininet> h1 ping h2
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=6.55 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.82 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.310 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.081 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.070/1.766/6.550/2.479 ms
mininet> 
```

Hasil ping h1 ke h2 menjadi rendah karena rule forwarding disimpan pada flow table data plane. Saat ada paket masuk yang sesuai dengan rule forwarding, maka akan diforward sesuai flow table tanpa dikirim ke control plane. Berikut dump-flows dari s1 :

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=56.103s, table=0, n_packets=5, n_bytes=434, idle_age=51, i
n_port=2,dl_dst=36:ec:35:66:56:ca actions=output:1
 cookie=0x0, duration=55.106s, table=0, n_packets=5, n_bytes=434, idle_age=51, i
n_port=1,dl_dst=1e:c1:4d:d4:9a:da actions=output:2
ridou@155150200111174_ridho:~/Ryu-python$ 
```

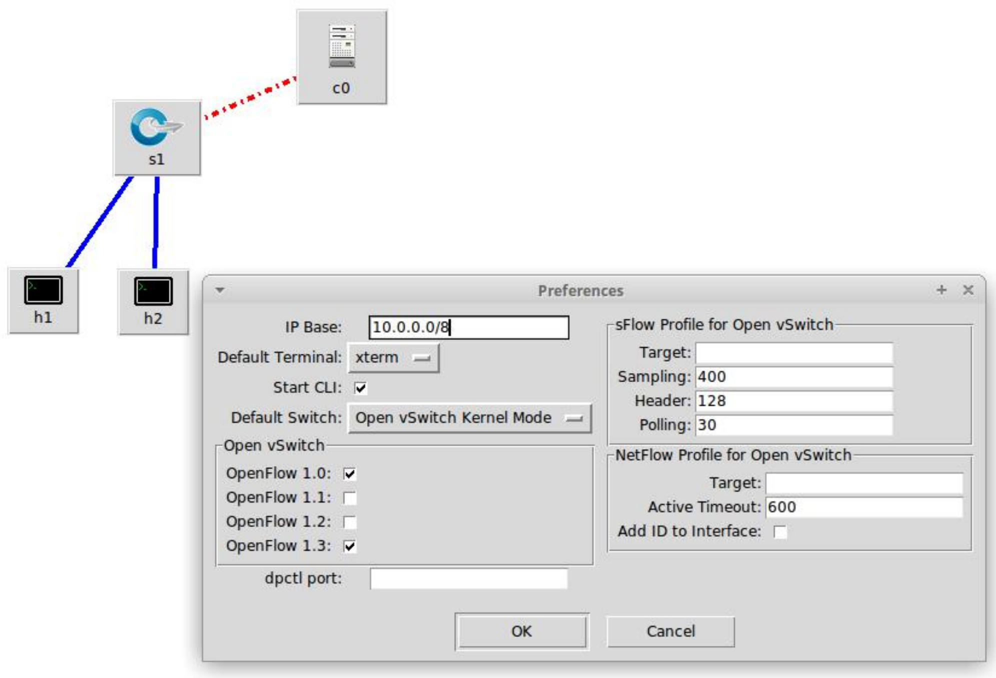
ARSITEKTUR JARINGAN TERKINI

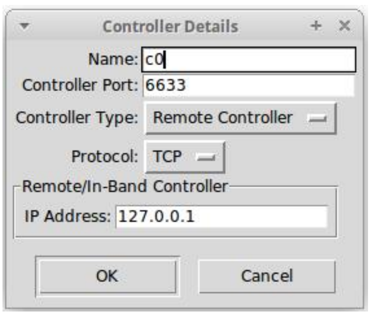
TUGAS	: OPENFLOW 1.3
NAMA	: MUHAMAD MIFTAHUR RIDHOILAH VITO KURNIAWAN SAMPURNO DARMADANI KYAT MADANA
NIM	: 155150200111174 155150200111180 155150207111105

Laporan Openflow 1.3

1. Penjelasan Openflow parser:
   a) OFPActionSetField digunakan untuk meng-set header field dari packet
   b) OFPInstructionActions digunakan untuk mendefinisikan /mengimplementasikan /menghapus actions pada openflow versi 1.3
   c) OFPIT_APPLY_ACTIONS digunakan sebagai parameter OFPInstructionActions menunjukkan bahwa actions diimplementasikan

2. Berikut topologi dan preferences yang kami gunakan pada percobaan pertama:





Topologi terdiri dari 2 host 1 switch dan 1 remote controller dengan Openflow protocol versi 1.3 .

Source code dari control plane untuk openflow 1.3:
Modifikasi dari source code forwarding dengan mac

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet,ethernet

class hub(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(hub,self).__init__(*args, **kwargs)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        data = msg.data

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocol(ethernet.ethernet)
        src = eth.src

        if src== '22:80:bb:da:35:68':
            out_port=2
        else:
            out_port=1

        #actions = [datapath.ofproto_parser.OFPActionOutput(ofproto.OFPP_FLOOD)] (Flooding)
        actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]
        out = datapath.ofproto_parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
        in_port=msg.in_port, actions=actions)
        datapath.send_msg(out)
```

3. Menjalankan source code dengan ryu-manager

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub2.py
loading app hub2.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub2.py of hub
```

4. Hasil ping h1 ke h2

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4006ms
pipe 3
mininet>
```

h1 tidak bisa melakukan ping ke h2 (unreachable) karena pada openflow versi 1.3 harus ada entri dasar untuk table miss flow yang berguna untuk forwarding paket yang tidak dikenali control plane.

5. Lihat dump-flows dari s1

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow1
3 s1
[sudo] password for ridou:
OFPST_FLOW reply (OF1.3) (xid=0x2):
ridou@155150200111174_ridho:~/Ryu-python$
```

6. Menambahkan code pada source code

Untuk handling table miss entry dengan CONFIG_DISPATCHER

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
def switch_features_handler(self,ev):
    msg = ev.msg
    dp = msg.datapath
    ofproto = dp.ofproto
    parser = dp.ofproto_parser

    match = parser.OFPMatch()
    actions =  [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER, ofproto.OFPCML_NO_BUFFER)]
    self.addflow(dp,0,match,actions)
```

Untuk add flow

```
def addflow(self,dp,priority,match,actions):
    ofproto = dp.ofproto
    parser = dp.ofproto_parser
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,actions)]
    mod = dp.ofproto_parser.OFPFlowMod(datapath=dp,match=match,priority=priority,instructions=inst)
    dp.send_msg(mod)
```

Setelah itu dilakukan run source code dengan ryu-manager

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub2.py
loading app hub2.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub2.py of hub
```

Dengan ditambah dua fungsi tersebut, maka switch bisa menghandle table miss entry sehinggi h1 bisa melakukan ping h2

```
mininet> h1 ping h2
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=5.71 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3.98 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.49 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.65 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 2.495/3.963/5.717/1.154 ms
mininet>
```

Hasil ping h1 ke h2 terlihat tinggi karena rule forwarding tidak disimpan pada flow table data plane. Yang ada di data plane hanya untuk menghandle table miss entry. Berikut dump-flows dari s1 :

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow1
3 s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow1
3 s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=3.709s, table=0, n_packets=0, n_bytes=0, priority=0 action
s=CONTROLLER:65535
ridou@155150200111174_ridho:~/Ryu-python$
```

7. Penambahan source code untuk menambahkan flow pada flow table

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    data = msg.data
    parser = datapath.ofproto_parser

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocol(ethernet.ethernet)
    src = eth.src

    if src == '92:dc:73:f7:e5:f2' :
        out_port=2
    else:
        out_port=1

    #actions = [datapath.ofproto_parser.OFPActionOutput(ofproto.OFPP_FLOOD)] (Flooding)
    match = parser.OFPMatch(in_port=msg.match['in_port'],eth_src=src)
    actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]
    out = datapath.ofproto_parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
     in_port=msg.match['in_port'], actions=actions)
    datapath.send_msg(out)
    self.addflow(datapath,ofproto.OFP_DEFAULT_PRIORITY,match,actions)
```

Setelah itu dilakukan run source code dengan ryu-manager

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager hub2.py
loading app hub2.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub2.py of hub
```

Dengan ditambah syntax untuk melakukan add-flow, flow yang baru akan dimasukkan ke flow table sehingga forwarding berikutnya dapat dilakukan dengan cepat

```
mininet> h1 ping h2
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=8.28 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.077 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.049/2.116/8.287/3.562 ms
mininet>
```

Hasil ping h1 ke h2 menjadi rendah karena rule forwarding disimpan pada flow table data plane. Saat ada paket masuk yang sesuai dengan rule forwarding, maka akan diforward sesuai flow table tanpa dikirim ke control plane. Berikut dump-flows dari s1 :

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow1
3 s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=84.298s, table=0, n_packets=5, n_bytes=434, in_port=1,dl_s
rc=92:dc:73:f7:e5:f2 actions=output:2
 cookie=0x0, duration=84.295s, table=0, n_packets=5, n_bytes=434, in_port=2,dl_s
rc=8e:a5:3a:da:53:48 actions=output:1
 cookie=0x0, duration=86.606s, table=0, n_packets=128, n_bytes=9968, priority=0
actions=CONTROLLER:65535
ridou@155150200111174_ridho:~/Ryu-python$
```

ARSITEKTUR JARINGAN TERKINI

TUGAS : LOAD BALANCER
NAMA : MUHAMAD MIFTAHUR RIDHOILAH
VITO KURNIAWAN SAMPURNO
DARMADANI KYAT MADANA
NIM : 155150200111174
155150200111180
155150207111105

Laporan Load Balancer

Berikut topologi dan preferences yang kami gunakan pada percobaan

```
ridou@155150200111174_ridho:~$ sudo mn --topo single,7 --mac --controller=remote
 --switch ovs,protocols=OpenFlow13
[sudo] password for ridou:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Topologi terdiri dari 7 host 1 switch dan 1 remote controller dengan Openflow protocol versi 1.3 . Dalam semua percobaan kali ini, h1, h2 dan h3 berperan sebagai simple HTTP server dengan port 80

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
h6 h6-eth0:s1-eth6
h7 h7-eth0:s1-eth7
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0 s1-eth5:
h5-eth0 s1-eth6:h6-eth0 s1-eth7:h7-eth0
c0
mininet>
```

```
*** Starting CLI:
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 python -m SimpleHTTPServer 80 &
mininet> h3 python -m SimpleHTTPServer 80 &
mininet>
```

Percobaan pertama (load balancer statis)

1. Source code dari control plane untuk load balancer statis:

Initialisasi dan fungsi reply arp

```
import random
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER,CONFIG_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3,ether
from ryu.lib import mac
from ryu.lib.packet import packet,ethernet,arp,ipv4,tcp,ether_types

class loadBalancer(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(loadBalancer,self).__init__(*args, **kwargs)
        self.mac_to_port = {}
        self.serverlist = []
        self.virtual_lb_ip = "10.0.0.100"
        self.virtual_lb_mac = "AB:BC:CD:EF:AB:BC"
        self.serverlist.append({'ip': "10.0.0.1", 'mac': "00:00:00:00:00:01","outport": "1"})
        self.serverlist.append({'ip': "10.0.0.2", 'mac': "00:00:00:00:00:02", "outport": "2"})
        self.serverlist.append({'ip': "10.0.0.3", 'mac': "00:00:00:00:00:03", "outport": "3"})

    def function_for_arp_reply(self, dst_ip,dst_mac):
        arp_target_mac = dst_mac # Menjadikan IP dan MAC virtual LB sebagai IP Src dan MAC dari
        src_ip = self.virtual_lb_ip
        src_mac = self.virtual_lb_mac
        arp_opcode = 2 # ARP opcode (Operationcode)2 untuk ARP reply
        hardware_type = 1 # 1 = Ethernet ie 10Mb
        arp_protocol = 2048 # 2048 = IPv4 packet
        ether_protocol = 2054 # 2054 = ARP protocol
        len_of_mac = 6 # length of MAC in bytes
        len_of_ip = 4 # length of IP in bytes
        pkt = packet.Packet()
        ether_frame = ethernet.ethernet(dst_mac, src_mac, ether_protocol) # Dealing with only l
        arp_reply_pkt = arp.arp(hardware_type, arp_protocol, len_of_mac, len_of_ip,arp_opcode,
        src_mac, src_ip,arp_target_mac, dst_ip) # Building the ARP reply packet,
        pkt.add_protocol(ether_frame)
        pkt.add_protocol(arp_reply_pkt)
        pkt.serialize()
        return pkt
```

Fungsi config dan main dispatcher

```python
@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
def switch_features_handler(self,ev):
    msg = ev.msg
    dp = msg.datapath
    ofproto = dp.ofproto
    parser = dp.ofproto_parser

    match = parser.OFPMatch()
    actions =  [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER, ofproto.OFPCML_NO_BUFFER)]
    self.addflow(dp,0,match,actions)


@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofproto = dp.ofproto
    parser = dp.ofproto_parser

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]
    in_port = msg.match['in_port']

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        return

    if eth.ethertype == ether.ETH_TYPE_ARP:
        arp_header = pkt.get_protocols(arp.arp)[0]
        if arp_header.dst_ip == self.virtual_lb_ip and arp_header.opcode == arp.ARP_REQUEST:
            reply_pkt = self.function_for_arp_reply(arp_header.src_ip,arp_header.src_mac)
            actions = [parser.OFPActionOutput(in_port)]
            packet_out = parser.OFPPacketOut(datapath=dp,in_port=ofproto.OFPP_ANY,data=reply_pkt.data,
                actions=actions,buffer_id=0xffffffff)
            dp.send_msg(packet_out)
            return

    ip_header = pkt.get_protocols(ipv4.ipv4)[0]
    tcp_header = pkt.get_protocols(tcp.tcp)[0]
    match = parser.OFPMatch(in_port=in_port, eth_type=eth.ethertype,eth_src=eth.src, eth_dst=eth.dst,
        ip_proto=ip_header.proto,ipv4_src=ip_header.src, ipv4_dst=ip_header.dst,
        tcp_src=tcp_header.src_port,tcp_dst=tcp_header.dst_port)
```

Algoritma pembagian load secara statis

```python
if ip_header.src == "10.0.0.4" or ip_header.src == "10.0.0.5":
    server_mac_selected = self.serverlist[0]['mac']
    server_ip_selected = self.serverlist[0]['ip']
    server_outport_selected = int(self.serverlist[0]['outport'])
elif ip_header.src == "10.0.0.6":
    server_mac_selected = self.serverlist[1]['mac']
    server_ip_selected = self.serverlist[1]['ip']
    server_outport_selected = int(self.serverlist[1]['outport'])
else:
    server_mac_selected = self.serverlist[2]['mac']
    server_ip_selected = self.serverlist[2]['ip']
    server_outport_selected = int(self.serverlist[2]['outport'])
```

Mengganti source atau destination dari paket sesuai dengan selected server, ip dan output port

```python
    actions = [parser.OFPActionSetField(ipv4_src=self.virtual_lb_ip),parser.OFPActionSetField(eth_src=self.virtual_lb_mac),
    parser.OFPActionSetField(eth_dst=server_mac_selected),parser.OFPActionSetField(ipv4_dst=server_ip_selected),
    parser.OFPActionOutput(server_outport_selected)]

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
    cookie = random.randint(0, 0xffffffffffffffff)
    flow_mod = parser.OFPFlowMod(datapath=dp, match=match, idle_timeout=7, instructions=inst,buffer_id=msg.buffer_id, cookie=cookie)
    dp.send_msg(flow_mod)

    match = parser.OFPMatch(in_port=server_outport_selected, eth_type=eth.ethertype, eth_src=server_mac_selected,
        eth_dst=self.virtual_lb_mac, ip_proto=ip_header.proto, ipv4_src=server_ip_selected,
        ipv4_dst=self.virtual_lb_ip, tcp_src=tcp_header.dst_port, tcp_dst=tcp_header.src_port)
    actions = [parser.OFPActionSetField(eth_src=self.virtual_lb_mac),parser.OFPActionSetField(ipv4_src=self.virtual_lb_ip),
    parser.OFPActionSetField(ipv4_dst=ip_header.src), parser.OFPActionSetField(eth_dst=eth.src),parser.OFPActionOutput(in_port)]
    inst2 = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
    cookie = random.randint(0, 0xffffffffffffffff)
    flow_mod2 = parser.OFPFlowMod(datapath=dp, match=match, idle_timeout=7, instructions=inst2, cookie=cookie)
    dp.send_msg(flow_mod2)

def addflow(self,dp,priority,match,actions,buffer_id=None):
    ofproto = dp.ofproto
    parser = dp.ofproto_parser
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,actions)]
    if buffer_id:
        mod = dp.ofproto_parser.OFPFlowMod(datapath=dp,buffer_id=buffer_id,match=match,priority=priority,instructions=inst)
    else:
        mod = dp.ofproto_parser.OFPFlowMod(datapath=dp,match=match,priority=priority,instructions=inst)
    dp.send_msg(mod)
```

Sesuai dengan algotitma pembagian load diatas, apabila ip source berupa 10.0.0.4 atau 10.0.0.5 (h4 atau h5) akan dilayani oleh h1, sementara untuk 10.0.0.6 (h6) akan dilayani oleh h2, dan sisanya (h7) akan dilayani oleh h3.

2. Source code loadbalancer.py dijalankan dengan ryu-manager dan dilakukan curl dari h4,h5,h6,h7 ke 10.0.0.100

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager loadbalancer.py
loading app loadbalancer.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app loadbalancer.py of loadBalancer
```

```
mininet> h4 curl 10.0.0.100 &
mininet> h5 curl 10.0.0.100 &
mininet> h6 curl 10.0.0.100 &
mininet> h7 curl 10.0.0.100 &
mininet>
```

3. Berikut hasil wiresharknya

http

No.	Time	Source	Destination	Protocol	Length	Info
552	40.519135205	10.0.0.4	10.0.0.100	HTTP	142	GET / HTTP/1.1
554	40.519250766	10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
568	40.523398646	10.0.0.1	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
569	40.523404841	10.0.0.100	10.0.0.4	HTTP	198	HTTP/1.0 200 OK (text/html)
828	74.006302694	10.0.0.6	10.0.0.100	HTTP	142	GET / HTTP/1.1
830	74.006457406	10.0.0.100	10.0.0.2	HTTP	142	GET / HTTP/1.1
844	74.010790012	10.0.0.2	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
845	74.010797734	10.0.0.100	10.0.0.6	HTTP	198	HTTP/1.0 200 OK (text/html)
1164	143.753532304	10.0.0.5	10.0.0.100	HTTP	142	GET / HTTP/1.1
1165	143.753719617	10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
1179	143.757067462	10.0.0.1	10.0.0.100	HTTP	158	HTTP/1.0 200 OK (text/html)
1180	143.757072998	10.0.0.100	10.0.0.5	HTTP	158	HTTP/1.0 200 OK (text/html)
1204	151.806997021	10.0.0.7	10.0.0.100	HTTP	142	GET / HTTP/1.1
1205	151.807085183	10.0.0.100	10.0.0.3	HTTP	142	GET / HTTP/1.1
1219	151.809052845	10.0.0.3	10.0.0.100	HTTP	158	HTTP/1.0 200 OK (text/html)
1220	151.809055589	10.0.0.100	10.0.0.7	HTTP	158	HTTP/1.0 200 OK (text/html)

Dari hasil wireshark, dapat dilihat load balancer sudah membagi request dari client sesuai dengan algoritma nya

Dump-flows s1 awal :

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow13 s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=40.816s, table=0, n_packets=48, n_bytes=3816, priority=0 actions=CONTROLLER:65535
ridou@155150200111174 ridho:~/Ryu-python$
```

Dump-flows s1 setelah dilakukan request dari h4,h5,h6,h7:

```
ridou@155150200111174_ridho:~/Ryu-python$ sudo ovs-ofctl dump-flows -O openflow13 s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x17365b1a59521652, duration=5.181s, table=0, n_packets=8, n_bytes=610, idle_timeout=7, tcp,in_p
ort=6,dl_src=00:00:00:00:00:06,dl_dst=ab:bc:cd:ef:ab:bc,nw_src=10.0.0.6,nw_dst=10.0.0.100,tp_src=38432,t
p_dst=80 actions=set_field:10.0.0.100->ip_src,set_field:ab:bc:cd:ef:ab:bc->eth_src,set_field:00:00:00:00
:00:02->eth_dst,set_field:10.0.0.2->ip_dst,output:2
 cookie=0x1f5569569168d61f, duration=5.181s, table=0, n_packets=8, n_bytes=6550, idle_timeout=7, tcp,in_
port=2,dl_src=00:00:00:00:00:02,dl_dst=ab:bc:cd:ef:ab:bc,nw_src=10.0.0.2,nw_dst=10.0.0.100,tp_src=80,tp_
dst=38432 actions=set_field:ab:bc:cd:ef:ab:bc->eth_src,set_field:10.0.0.100->ip_src,set_field:10.0.0.6->
ip_dst,set_field:00:00:00:00:00:06->eth_dst,output:6
 cookie=0x1931b73aed14b5f3, duration=2.534s, table=0, n_packets=8, n_bytes=610, idle_timeout=7, tcp,in_p
ort=7,dl_src=00:00:00:00:00:07,dl_dst=ab:bc:cd:ef:ab:bc,nw_src=10.0.0.7,nw_dst=10.0.0.100,tp_src=44664,t
p_dst=80 actions=set_field:10.0.0.100->ip_src,set_field:ab:bc:cd:ef:ab:bc->eth_src,set_field:00:00:00:00
:00:03->eth_dst,set_field:10.0.0.3->ip_dst,output:3
 cookie=0x255460338e119661, duration=2.534s, table=0, n_packets=8, n_bytes=6550, idle_timeout=7, tcp,in_
port=3,dl_src=00:00:00:00:00:03,dl_dst=ab:bc:cd:ef:ab:bc,nw_src=10.0.0.3,nw_dst=10.0.0.100,tp_src=80,tp_
dst=44664 actions=set_field:ab:bc:cd:ef:ab:bc->eth_src,set_field:10.0.0.100->ip_src,set_field:10.0.0.7->
ip_dst,set_field:00:00:00:00:00:07->eth_dst,output:7
 cookie=0x0, duration=136.005s, table=0, n_packets=59, n_bytes=4406, priority=0 actions=CONTROLLER:65535
ridou@155150200111174_ridho:~/Ryu-python$
```

Dapat dilihat actions nya adalah mengganti tujuan atau source dari paket.

Percobaan kedua dengan load balancer dengan algoritma Round Robin

1. Source code yang digunakan pada load balancer RR kurang lebih sama dengan yang statis hanya saja bagian algotitma diganti dan ditambah variable request yang digunakan untuk menentukan server yang dipakai sesuai dengan urutan request

Initialisasi variable awal

```
def __init__(self, *args, **kwargs):
    super(loadBalancer,self).__init__(*args, **kwargs)
    self.mac_to_port = {}
    self.serverlist = []
    self.virtual_lb_ip = "10.0.0.100"
    self.virtual_lb_mac = "AB:BC:CD:EF:AB:BC"
    self.serverlist.append({'ip': "10.0.0.1", 'mac': "00:00:00:00:00:01","outport": "1"})
    self.serverlist.append({'ip': "10.0.0.2", 'mac': "00:00:00:00:00:02", "outport": "2"})
    self.serverlist.append({'ip': "10.0.0.3", 'mac': "00:00:00:00:00:03", "outport": "3"})
    self.req = 0
```

Algoritma pembagian load secara round robin

```
#algoritma RR
server = self.req % 3
print("request {} , server index {}".format(self.req+1,server))
server_mac_selected = self.serverlist[server]['mac']
server_ip_selected = self.serverlist[server]['ip']
server_outport_selected = int(self.serverlist[server]['outport'])
self.req += 1
```

Sesuai dengan algortima diatas, server yang dipakai sesuai dengan urutan request di mod 3, dimana 3 merupakan jumlah server yang ada dalam topologi. Pada terminal ryu ditampilkan request ke berapa dan index server yang dipakai. Angka request yang ditampilkan ditambah 1 sementara index server ditampilkan apa adanya.

2. Berikut saat source code dijalankan dengan ryu-manager dan dilakukan curl dari client. Urutan requestnya adalah : h4,h4,h4,h4,h5,h6,h6,h7,h7,h5

```
^Cridou@155150200111174_ridho:~/Ryu-python$ ryu-manager loadbalancerRR.py
loading app loadbalancerRR.py
loading app ryu.controller.ofp_handler
instantiating app loadbalancerRR.py of loadBalancer
instantiating app ryu.controller.ofp_handler of OFPHandler
request 1 , server index 0
request 2 , server index 1
request 3 , server index 2
request 4 , server index 0
request 5 , server index 1
request 6 , server index 2
request 7 , server index 0
request 8 , server index 1
request 9 , server index 2
request 10 , server index 0
```

Dapat dilihat, server yang melayani request digilir dari index 0 sampai index 2 bedasarkan urutan request nya, bukan bedasarkan ip source dari paket. Angka request yang ditampilkan diatas merupakan request + 1.

3. Berikut hasil wireshark dari load balancer RR

Source	Destination	Protocol	Lengtl	Info
10.0.0.4	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
10.0.0.1	10.0.0.100	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.4	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.4	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.2	HTTP	142	GET / HTTP/1.1
10.0.0.2	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.4	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.4	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.3	HTTP	142	GET / HTTP/1.1
10.0.0.3	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.4	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.4	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
10.0.0.1	10.0.0.100	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.4	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.5	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.2	HTTP	142	GET / HTTP/1.1
10.0.0.2	10.0.0.100	HTTP	5926	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.5	HTTP	5926	HTTP/1.0 200 OK (text/html)
10.0.0.6	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.3	HTTP	142	GET / HTTP/1.1
10.0.0.3	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.6	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.6	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
10.0.0.1	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.6	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.7	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.2	HTTP	142	GET / HTTP/1.1
10.0.0.2	10.0.0.100	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.7	HTTP	134	HTTP/1.0 200 OK (text/html)
10.0.0.7	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.3	HTTP	142	GET / HTTP/1.1
10.0.0.3	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.7	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.5	10.0.0.100	HTTP	142	GET / HTTP/1.1
10.0.0.100	10.0.0.1	HTTP	142	GET / HTTP/1.1
10.0.0.1	10.0.0.100	HTTP	198	HTTP/1.0 200 OK (text/html)
10.0.0.100	10.0.0.5	HTTP	198	HTTP/1.0 200 OK (text/html)

No	Client	Server
1	h4	Server index 0 / h1
2	h4	Server index 1 / h2
3	h4	Server index 2 / h3
4	h4	Server index 0 / h1
5	h5	Server index 1 / h2
6	h6	Server index 2 / h3
7	h6	Server index 0 / h1
8	h7	Server index 1 / h2
9	h7	Server index 2 / h3
10	h5	Server index 0 / h1