

PRAKTIKUM DESAIN DAN MANAJEMEN JARINGAN KOMPUTER

Nama	Aliyah Rizky Al-Afifah Polanda	No. Modul	08
NPM	2206024682	Tipe	Tugas Tambahan

A. Mendownload Package yang Dibutuhkan


1. *Installing git and python-pil.*

```
rizky@rizky-VirtualBox:~$ sudo apt install git python-pil python3-pil
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

2. *Installing ryu-manager.*

```
rizky@rizky-VirtualBox:~$ pip install ryu
Collecting ryu
  Downloading ryu-4.34.tar.gz (1.1 MB)
    | 1.1 MB 1.3 MB/s
rizky@rizky-VirtualBox:~$ sudo apt install python3-ryu
[sudo] password for rizky:
rizky@rizky-VirtualBox:~$ sudo apt install gcc python-dev libffi-dev libxml2-dev
libxslt1-dev zlib1g-dev
Reading package lists... Done
```

3. Penggunaan MiniNAM.
4. *Downloading resource.*

 **Resource Tugas Tambahan Modul 8.zip**
57.0 KB • 19 minutes ago

B. Menjalankan Topologi SDN

5. Menjalankan *file* simple_switch_stp_13.py.

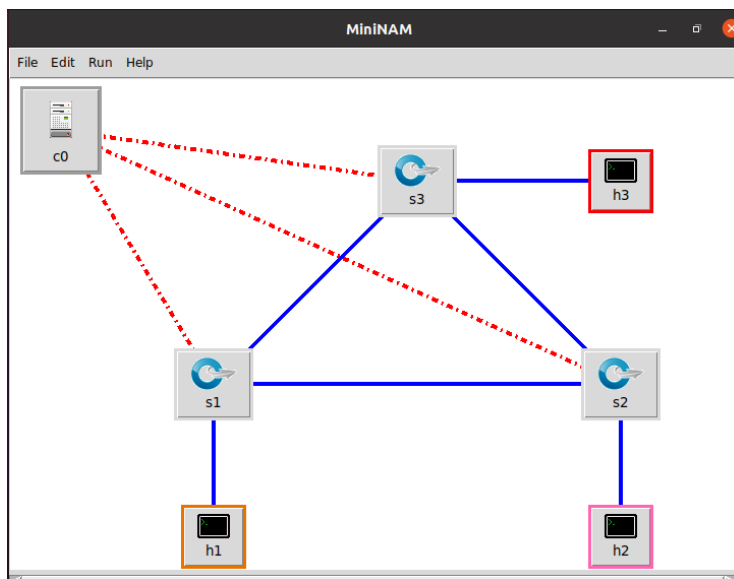
```
rizky@rizky-VirtualBox:~/Documents/tutan8$ ryu-manager simple_switch_stp_13.py
loading app simple_switch_stp_13.py
loading app ryu.controller.ofp_handler
instantiating app None of Stp
creating context stplib
instantiating app simple_switch_stp_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

6. Menjalankan *file* spanning_tree.py.

```

rizky@rizky-VirtualBox:~/Documents/tutan8$ sudo python MiniNAM.py --custom span
ning_tree.py --topo=mytopo --controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h1) (s1, s2) (s2, h2) (s2, s3) (s3, h3) (s3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>

```



7. Perintah net di mininet.

```

mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo: s1-eth1:h1-eth0 s1-eth2:s2-eth2 s1-eth3:s3-eth3
s2 lo: s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s1-eth3
c0

```

Perintah `net` digunakan untuk menampilkan informasi umum mengenai topologi jaringan. Dari *output* diatas, dapat dilihat koneksi yang terjadi antara masing-masing *node*. Misalnya `h1` (*interface* `eth0`) terhubung dengan `s1` (*interface* `eth1`) serta `s1` memiliki koneksi dengan `h1`, `s2`, dan `s3`. Selain itu, `c0` (*controller*) tidak memiliki koneksi secara langsung dengan *node* lain dalam topologi, karena `c0` berfungsi untuk mengontrol jaringan, bukan untuk berkomunikasi.

8. Mengubah *bridge option* dari switch.

```

root@rizky-VirtualBox:~/Documents/tutan8# ovs-vsctl set Bridge s1 protocols=OpenFlow13

```

```
root@rizky-VirtualBox:/home/rizky/Documents/tutam8# ovs-vsctl set bridge s2 protocols=OpenFlow13
```

```
root@rizky-VirtualBox:/home/rizky/Documents/tutam8# ovs-vsctl set bridge s3 protocols=OpenFlow13
```

Hasil:

```
root@rizky-VirtualBox:/home/rizky/Documents/tutam8# ovs-vsctl show
ec443b31-3fd4-4371-8634-fe911c354cd8
  Bridge s3
    Controller "tcp:127.0.0.1:6653"
      is_connected: true
    Controller "ptcp:6636"
      fail_mode: secure
    Port s3
      Interface s3
        type: internal
    Port s3-eth1
      Interface s3-eth1
    Port s3-eth2
      Interface s3-eth2
    Port s3-eth3
      Interface s3-eth3
  Bridge s2
    Controller "tcp:127.0.0.1:6653"
      is_connected: true
    Controller "ptcp:6635"
      fail_mode: secure
    Port s2-eth2
      Interface s2-eth2
    Port s2-eth1
      Interface s2-eth1
    Port s2-eth3
      Interface s2-eth3
    Port s2
      Interface s2
        type: internal
  Bridge s1
    Controller "ptcp:6634"
    Controller "tcp:127.0.0.1:6653"
      is_connected: true
    fail_mode: secure
    Port s1-eth1
      Interface s1-eth1
    Port s1-eth2
      Interface s1-eth2
    Port s1-eth3
      Interface s1-eth3
    Port s1
      Interface s1
        type: internal
    ovs_version: "2.13.8"
```

9. Menentukan *root bridge* dan *port status*.

- s1: root bridge

```
[STP][INFO] dpid=0000000000000001: Root bridge.
[STP][INFO] dpid=0000000000000001: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT / FORWARD
```

Semua port melakukan *forwarding*.

- s2: non-root bridge

```
[STP][INFO] dpid=0000000000000002: Non root bridge.
[STP][INFO] dpid=0000000000000002: [port=2] ROOT_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT / FORWARD
```

Semua port melakukan *forwarding*.

- s3: non-root bridge

```
[STP][INFO] dpid=0000000000000003: Non root bridge.
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / BLOCK
```

Port 1 (yang terkoneksi dengan h3) dan port 3 (yang terkoneksi dengan s1) melakukan *forwarding*. Sedangkan port 2 (yang terkoneksi dengan s2) melakukan *blocking*.

10. Verifikasi *looping* dengan melakukan ping antara h1 dan h2.

```
mininet> h1 ping -c3 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=0.582 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.150 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.075 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 0.075/0.269/0.582/0.223 ms
```

- s1:

```
root@rizky-VirtualBox:/home/rizky/Documents/tutan8# tcpdump -i s1-eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s1-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
13:03:10.460877 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:12.464318 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:14.466145 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:15.929021 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 29227, seq 1
, length 64
13:03:15.929104 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 29227, seq 1,
length 64
13:03:16.466584 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:16.930310 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 29227, seq 2c9
, length 64
13:03:16.930349 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 29227, seq 2, c9
length 64
13:03:17.936139 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 29227, seq 3c9
, length 64
13:03:17.936168 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 29227, seq 3, c9
length 64
13:03:18.466532 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:20.468331 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
c9:20:41.8002, length 43
13:03:21.006508 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
13:03:21.006525 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
13:03:21.006578 ARP, Reply 10.0.0.1 is-at 8a:d8:f9:ec:c2:8c (oui Unknown),
length 28
13:03:21.006606 ARP, Reply 10.0.0.2 is-at 72:14:38:ae:61:a9 (oui Unknown),
length 28
```

- s2:

```
root@rizky-VirtualBox:/home/rizky/Documents/tutan8# tcpdump -i s2-eth3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s2-eth3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:03:12.464427 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:14.466346 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:16.466794 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:18.466988 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:20.468530 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:22.471194 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:24.474203 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:26.479917 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
13:03:28.485832 STP 802.1d, Config, Flags [none], bridge-id 9000,8e:13:aa:61:6
1:45.8003, length 43
```

- s3:

```

root@rizky-VirtualBox: /home/rizky/Documents/tutam8# tcpdump -i s3-eth3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s3-eth3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:03:14.466212 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:16.466685 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:18.466563 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:20.468399 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:22.471065 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:24.474081 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:26.479790 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:28.485721 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43
13:03:30.503015 STP 802.1d, Config, Flags [none], bridge-id 8000,c2:c1:5f:c9
:20:41.8003, length 43

```

Dari hasil tcpdump di setiap *switch*, dapat dilihat bahwa paket ICMP hanya terdapat di s1. Hal ini menunjukkan, paket tidak mengalami *looping* (tidak berputar-putar pada setiap *switch*). Menunjukkan bahwa STP telah berhasil diterapkan dalam jaringan.

11. Mematikan salah satu *port* di *switch*.

```

mininet> s1 ifconfig s1-eth2 down
[STP][INFO] dpid=0000000000000001: [port=2] Link up.
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / LISTEN
[STP][INFO] dpid=0000000000000001: [port=2] Link down.
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / DISABLE
[STP][INFO] dpid=0000000000000002: [port=2] Link up.
[STP][INFO] dpid=0000000000000002: [port=2] DESIGNATED_PORT / LISTEN
[STP][INFO] dpid=0000000000000002: [port=2] Link down.
[STP][INFO] dpid=0000000000000002: [port=2] DESIGNATED_PORT / DISABLE
[STP][INFO] dpid=0000000000000002: [port=2] Link up.
[STP][INFO] dpid=0000000000000002: [port=2] DESIGNATED_PORT / LISTEN

```

Setelah mematikan *port* 2 (eth2) di s1, terjadi perubahan pada topologi (*link* antara s1 dan s2 terputus), sehingga akan terjadi pemilihan *root bridge* dengan STP. Pemilihan *root bridge* dan perubahan *status* dari *port* di setiap *switch* dapat diamati pada terminal *ryu manager*. Namun hasil pemilihan akan tetap sama seperti sebelumnya, dimana s1 yang menjadi *root bridge*. Hal ini karena s1 memiliki nilai prioritas *bridge* yang paling rendah (diamati dalam *file* `simple_switch_stp_13.py`).

```
[STP][INFO] dpid=0000000000000001: Root bridge.
```

```

config = {dpid_lib.str_to_dpid('0000000000000001'):
           {'bridge': {'priority': 0x8000}},
          dpid_lib.str_to_dpid('0000000000000002'):
           {'bridge': {'priority': 0x9000}},
          dpid_lib.str_to_dpid('0000000000000003'):
           {'bridge': {'priority': 0xa000}}}

```

```

mininet> s1 ifconfig s1-eth2 down
mininet> h1 ping -c3 -R h2
PING 10.0.0.2 (10.0.0.2) 56(124) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2053ms

mininet> h1 ping -c3 -R h2
PING 10.0.0.2 (10.0.0.2) 56(124) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=7.91 ms
RR:  10.0.0.1
     10.0.0.2
     10.0.0.2
     10.0.0.1

64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.169 ms (same route)
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.062 ms (same route)

```

Selain itu, saat dilakukan ping antara h1 dan h2 tepat setelah *link* terputus, ping tersebut gagal. Disebabkan karena proses yang dilakukan STP belum selesai. Namun saat pemilihan *root bridge* telah selesai, ping berhasil dilakukan.

12. Menyalakan kembali *port*.

```

mininet> s1 ifconfig s1-eth2 up
mininet> h1 ping -c3 -R h2
PING 10.0.0.2 (10.0.0.2) 56(124) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=4.48 ms
RR:  10.0.0.1
     10.0.0.2
     10.0.0.2
     10.0.0.1

64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3.81 ms (same route)
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.238 ms (same route)

```

Setelah eth2 pada s1 kembali dihidupkan, tidak terjadi pemilihan *root bridge*. Hal ini karena STP tidak mendeteksi perubahan yang signifikan pada topologi, seperti penambahan link. Ping yang dilakukan dari h1 dan h2 juga berhasil.

C. Analisis

13. Kegunaan *method*.

a. File `spanning_tree.py`:

- `MyTopo class`: Merupakan kelas turunan dari `Topo` yang disediakan oleh mininet. Digunakan untuk membuat topologi *custom*.
- `__init__ method`: Berisi deskripsi topologi *custom* yang akan dibuat.
- `addSwitch method`: Digunakan untuk menambahkan *switch* ke topologi.
- `addHost method`: Digunakan untuk menambahkan *host* ke topologi.
- `addLink method`: Digunakan untuk menambahkan koneksi antara dua *node* dalam

topologi.

- *topos*: Digunakan untuk menyimpan nama topologi *custom* yang dibuat, sehingga dapat dijalankan menggunakan mininet.
- *locations*: Digunakan untuk menentukan koordinat/lokasi dari setiap *nodes* dalam topologi.

b. File `simple_switch_stp_13.py`:

- *__init__ method*: Digunakan untuk melakukan inisialisasi protokol OpenFlow dan menetapkan prioritas *bridge* untuk STP.
- *delete_flow method*: Digunakan untuk menghapus aliran pada *switch*, saat aliran tersebut tidak lagi relevan/dibutuhkan.
- *_packet_in_handler method*: Digunakan untuk menangani paket yang diterima oleh suatu switch. Paket akan diperiksa dan akan mengirimkan paket ke tujuan jika tujuan diketahui.
- *_topology_change_handler method*: Digunakan untuk menangani perubahan topologi yang dipengaruhi oleh STP.
- *_port_change_handler method*: Digunakan untuk menangani perubahan *port status* pada *switch* yang dipengaruhi oleh STP. Akan mengembalikan *status* dari *port* tersebut.

14. Ryu *controller* merupakan *framework* OpenFlow dan termasuk *controller* SDN yang diatur untuk meningkatkan kemampuan jaringan dalam menangani *traffic* dan beradaptasi dengan perubahan topologi jaringan. Penggunaan ryu *controller* memudahkan pengembang untuk membuat aplikasi manajemen jaringan dan membantu untuk memenuhi kebutuhan jaringan yang lebih spesifik. Perilaku jaringan terhadap perubahan yang terjadi dapat diamati dengan lebih detail dengan menggunakan ryu *manager*.

Berikut adalah beberapa fungsi dari ryu *controller*:

- Ryu *controller* berfungsi sebagai pusat kontrol dalam SDN.
- Ryu *controller* dapat diprogram, sehingga aplikasinya dapat disesuaikan dengan kebutuhan spesifik.
- Ryu *controller* dapat digunakan untuk mengumpulkan data mengenai perilaku jaringan dan pengguna.

Referensi:

- “What Is a Ryu Controller?” sdxcentral.com. [Online]. Available: <https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/what-is-sdn-controller/openflow-controller/what-is-ryu-controller/>. [Accessed May 11, 2024].

15. Beberapa alasan untuk lebih baik mengimplementasikan SDN:

- a. Kontrol jaringan terpusat. SDN memusatkan kendali jaringan dalam satu *controller*, sehingga konfigurasi jaringan dapat dilakukan dengan lebih mudah.
- b. Jaringan dapat diprogram. Dengan SDN, perangkat jaringan dapat diprogram dan dikonfigurasi dengan cepat untuk beradaptasi dengan perubahan kebutuhan akan jaringan.
- c. Skalabilitas. SDN mempermudah perubahan skala jaringan untuk memenuhi permintaan lalu lintas yang berubah-ubah. Dengan adanya kemampuan kontrol jaringan yang disediakan oleh SDN, *administrator* dapat menyesuaikan jaringan sesuai kebutuhan dengan lebih cepat.
- d. Keamanan yang ditingkatkan. SDN menerapkan kontrol jaringan terpusat, sehingga lebih memudahkan deteksi terhadap ancaman keamanan yang muncul. Selain itu, *administrator* dapat menerapkan kontrol keamanan yang lebih detail untuk mengurangi risiko keamanan.
- e. Penghematan biaya. SDN memungkinkan penggunaan perangkat keras komoditas (yang tersedia secara luas di pasaran) untuk membangun jaringan. Kontrol jaringan terpusat juga dapat mengurangi kebutuhan pengelolaan jaringan secara manual, sehingga dapat mengurangi biaya tenaga kerja.

Referensi:

- “Software Defined Networking (SDN): Benefits and Challenges of Network Virtualization,” javatpoint.com. [Online]. Available: <https://www.javatpoint.com/software-defined-networking-sdn-benefits-and-challenges-of-network-virtualization>. [Accessed May 11, 2024].

16. Kesimpulan:

- Ryu *controller* dapat digunakan untuk mengumpulkan data mengenai perilaku jaringan. hal

ini berguna dalam analisis dan manajemen jaringan yang lebih baik.

- MiniNAM menyediakan visualisasi jaringan, sehingga dapat digunakan bersamaan dengan miniNet untuk memahami topologi jaringan dengan lebih baik.
- Dengan SDN, perangkat jaringan dapat diprogram, sehingga dapat disesuaikan dengan kebutuhan yang lebih spesifik. Selain itu, jaringan dapat beradaptasi secara lebih cepat terhadap perubahan topologi.
- Penggunaan STP (*Spanning Tree Protocol*) dapat mencegah terjadinya *looping* dalam jaringan, dengan pemilihan *root bridge* yang berfungsi sebagai pusat kendali.