Workshop OpenFlow

4. Load balancing

Akbari Indra Basuki
Pusat Penelitian Informatika, LIPI

Daftar Materi

Topik	Keterangan
Basic forwarding	Dasar-dasar OpenFlow
Routing & Monitoring	Program Controller: Shortest-path routing Monitor node and link status Integrasi NetworkX dan matplotlib
Packet Filtering (Firewall + Web Interface)	Program Controller: Bloom Filter, Flask
Load balancing	Group bucket and group tables Round robin load balancing Main-backup path protection
Rate limiting	Meter tables
Stateless vs Stateful data plane Stateful data plane	Jenis data plane dalam memproses paket OpenState SDN Arp handling Port Knocking

Load balancing

Keguanaan Load balancing:

- Membagi trafik jaringan sesuai kapasitas bandwidth jalur pengiriman
- Mencegah overload pada salah datu jalur/server
- Meminimalisir latency/delay pengiriman

OpenFLow: Load balancing menggunakan **Group table**. Jenis group table:

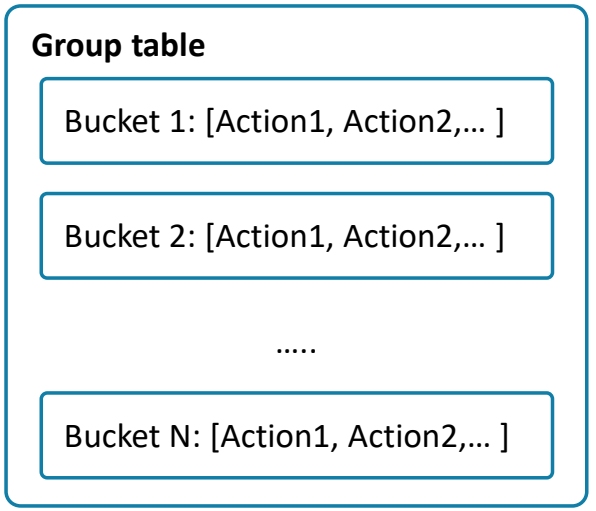
- All : Menduplikasi paket untuk dikirim ke seluruh bucket di dalam group table
- Select : Mengirim paket ke salah satu bucket saja menggunakan probabilitas
- Fast Failover (FF) : Mengirim paket ke bucket urutan pertama yang masih aktif

Daftar topik:

- Weighted Round robin load balancing (Group select)
- Main-backup path balancing (Group FF)

Latihan

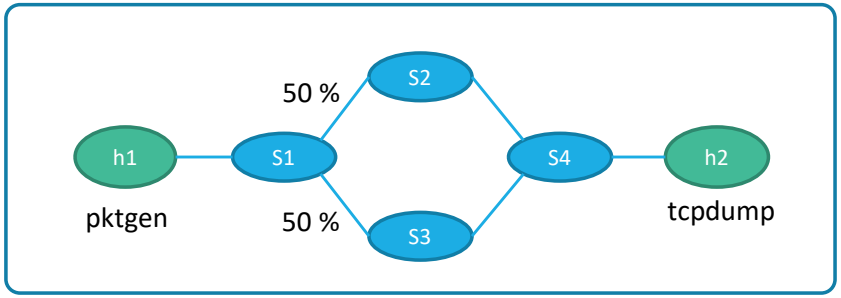
- *Packet hijacking* menggunakan Group All



Studi kasus 1 – Weighted round robin

Skenario:

- Switch S1 membagi trafik dari H1 ke H2 secara rata (50:50) melalui Switch S2 dan S3.
- Trafik akan kembali disatukan di Switch S4 dan dikirim ke host2.



Implementasi:

- Switch 1: Group table SELECT dengan 2 bucket, @weight = 50 (%)
- Switch S2 dan S3 memforward paket dari port 1 ke port 2 (Switch S4)
- Switch S4 forward paket ke Host 2
- Host 1 mengirim paket dengan pktgen
- Host 2 menjalankan sniffer (tcpdump)
- Controller menginstall flow rule secara statis ketika switch terkoneksi

Source code

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

        if (dp.id ==1):
                #50-50 balancing
                buckets = []
                weight = 50
                actions = [dp.ofproto_parser.OFPActionOutput(2, 0)]
                buckets.append(ofparser.OFPBucket(weight=weight,actions=actions))
                weight = 50
                actions = [dp.ofproto_parser.OFPActionOutput(3, 0)]
                buckets.append(ofparser.OFPBucket(weight=weight,actions=actions))
                req = ofparser.OFPGroupMod(datapath=dp, command=ofproto.OFPGC_ADD, type_=ofproto.OFPGT_SELECT, group_id=1, buckets=buckets)
                dp.send_msg(req)

                match = ofparser.OFPMatch(in_port = 1, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionGroup(1)]
                inst = [ofparser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
                mod = ofparser.OFPFlowMod(datapath=dp, table_id=0,priority=10, match=match, instructions=inst)
                dp.send_msg(mod)

        elif (dp.id == 4):
                #from s2 and s3 forward to host 2 (port 1)
                match = dp.ofproto_parser.OFPMatch(in_port=2, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(1, 0)]
                self.add_flow(dp, match, actions,1)
                match = dp.ofproto_parser.OFPMatch(in_port=3, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(1, 0)]
                self.add_flow(dp, match, actions,1)
        else:
                #from in_port=1 (s1) forward to port 2 (s4)
                match = dp.ofproto_parser.OFPMatch(in_port=1, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(2, 0)]
                self.add_flow(dp, match, actions,1)
```

Switch S1

Bucket 1

Bucket 2

Group table 1, mode = Select

Kirim paket IP dari Host 1 ke group table 1

Switch S4

Switch S2 dan S3

Langkah pengujian

Pengujian:

- Jalankan aplikasi controller (simpleBalancer.py)

  Ryu-manager simpleBalancer.py
- Jalankan mininet, topologi: topoBalancer.py

  Sudo python topoBalancer.py
- H2, S2, dan S3 menjalankan program sniffer (tcpdump)

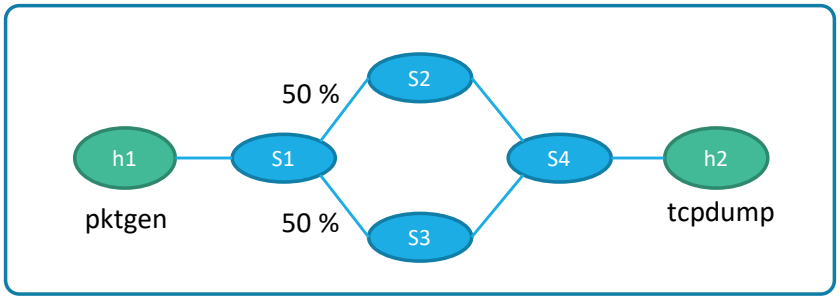
  Tcpdump –ni h2-eth0

  Tcpdump –ni s2-eth1

  Tcpdump –ni s3-eth1
- Kirim paket dari H1 dengan pktgen ke H2

  ./pktgen.sh

  Ctrl+C untuk mengakhiri program tcpdump
- Amati jumlah paket total yang diterima oleh H2
  - Jumlah total paket H2 = paket yang dterima S2 + S3
  - Apakah jumlah paket yang diterima oleh S2 dan S3 sama persis jumlahnya?
  - Variasi nilai weight dari bucket 1 dan bucket 2, dengan skema: (x) dan (100-x)



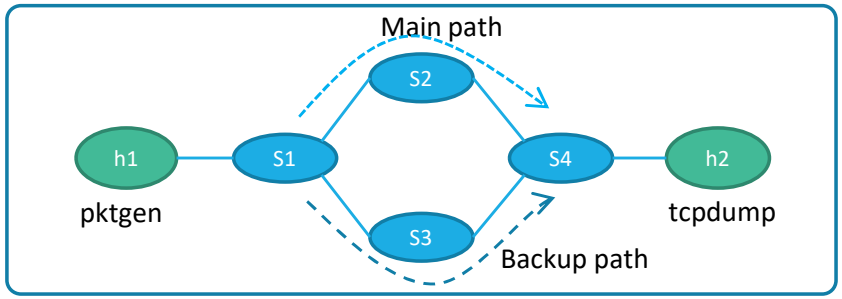
Studi kasus 2 – Main-backup path balancing

Skenario:

- Jaringan menggunakan jalur S1-S2-S4 sebagai jalur utama, dan

  S1-S3-S4 sebagai jalur cadangan

- Trafik akan dibelokkan ke jalur cadangan apabila jalur utama terputus



Implementasi:

- Switch 1: Group table FF dengan 2 bucket,
  - Bucket 1: watch port = 2 , status bucket aktif apabila port 2 UP (link s1-s2 UP)
  - Bucket 2: watch port = 3 , jika status bucket 1 non-aktif, bucket 2 akan dieksekusi (sesuai urutan list/array bucket)
- Konfigurasi H1, H2, S2, S3, dan S4 sama seperti studi kasus 1
- Controller menginstall flow rule secara statis ketika switch terkoneksi

Source code

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

        if (dp.id ==1):
                #main-backup schema
                buckets = []
                watch_port = 2
                actions = [dp.ofproto_parser.OFPActionOutput(2, 0)]
                buckets.append(ofparser.OFPBucket(watch_port=watch_port,actions=actions))
                watch_port = 3
                actions = [dp.ofproto_parser.OFPActionOutput(3, 0)]
                buckets.append(ofparser.OFPBucket(watch_port=watch_port,actions=actions))
                req = ofparser.OFPGroupMod(datapath=dp, command=ofproto.OFPGC_ADD, type_=ofproto.OFPGT_FF, group_id=2, buckets=buckets)
                dp.send_msg(req)

                match = ofparser.OFPMatch(in_port = 1, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionGroup(2)]
                inst = [ofparser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
                mod = ofparser.OFPFlowMod(datapath=dp, table_id=0,priority=10, match=match, instructions=inst)
                dp.send_msg(mod)
        elif (dp.id == 4):
                #from s2 and s3 forward to host 2 (port 1)
                match = dp.ofproto_parser.OFPMatch(in_port=2, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(1, 0)]
                self.add_flow(dp, match, actions,1)
                match = dp.ofproto_parser.OFPMatch(in_port=3, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(1, 0)]
                self.add_flow(dp, match, actions,1)
        else:
                #from in_port=1 (s1) forward to port 2 (s4)
                match = dp.ofproto_parser.OFPMatch(in_port=1, eth_type=0x800)
                actions = [dp.ofproto_parser.OFPActionOutput(2, 0)]
                self.add_flow(dp, match, actions,1)
```

Bucket 1, watch port: 2 (link ke S2)

Bucket 2, watch port: 3 (link ke S3)

Group table 2, mode FF

Langkah pengujian

Pengujian:

- Jalankan aplikasi controller (simpleBackup.py)

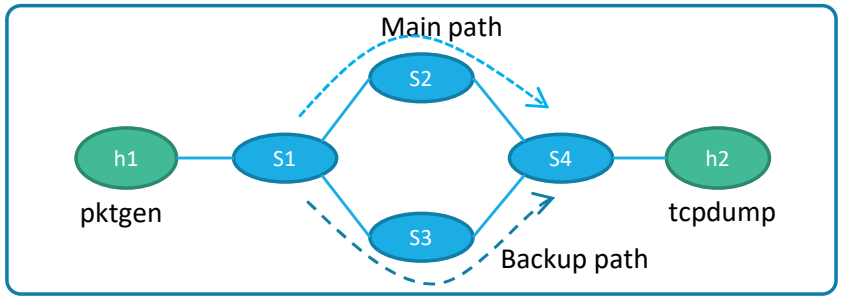
  Ryu-manager simpleBackup.py
- Jalankan mininet, topologi: topoBalancer.py

  Sudo python topoBalancer.py
- H2, menjalankan program sniffer (tcpdump)

  Tcpdump –ni h2-eth0
- Kirim paket dari H1 dengan pktgen ke H2

  ./pktgen.sh
- Matikan link S1-S2 dari jendela mininet

  Link s1 s2 down
- Amati jumlah paket total yang diterima oleh H2
  - Apakah semua paket berhasil dikirim, atau ada yang hilang?
  - Switch membutuhkan waktu untuk mendeteksi sebuah port/link telah mati
  - Berapa kisaran waktu yang dibutuhkan oleh s1?



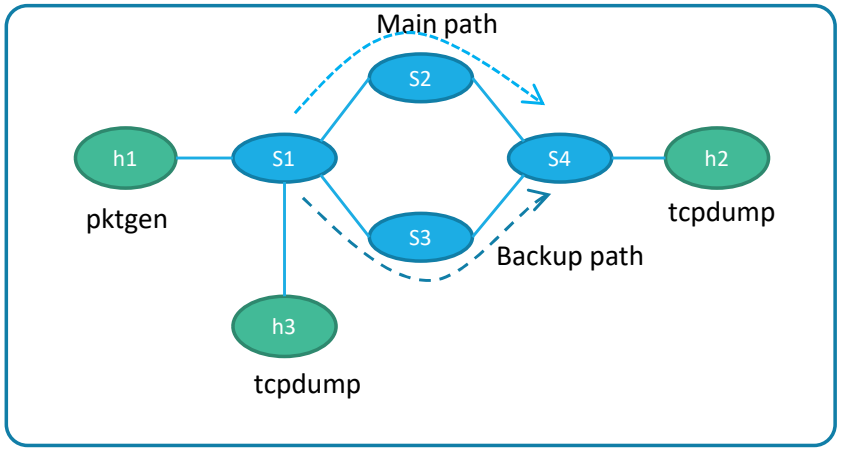
Latihan: *Packet hijacking* menggunakan Group All

1. Host 3 berusaha menyadap semua trafik Host 1 yang melalui switch S1.

Tips:

Install flow rule di Switch 1 untuk mengirim kopian paket ke h3 menggunakan group table mode All.

- Bucket 1: kirim ke main path (port 2/Switch S2)
- Bucket 2: kirim ke Host 3 (port 4)



Daftar pustaka

- https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/7995427/How+to+Work+with+Fast-Failover+OpenFlow+Groups