Workshop OpenFlow

6. Stateful forwarding

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
Stateful forwarding	Jenis data plane berdasarkan cara pemrosesan paket Stateless vs stateful data plane OpenState SDN Arp handling Port Knocking

Stateless vs stateful forwarding

Stateless dataplane: switch pada umumnya yang bekerja berdasarkan flow rule yang terinstall.

Stateful dataplane: switch yang memiliki kemampuan untuk menyimpan state, suatu nilai berdasarkan kondisi terntentu.

- Misal: state telah menerima paket, atau telah mengirim paket tertentu.
- Dapat bekerja tanpa bantuan controller
- Controller cukup menginstall flow rule statis ketika start up
- Berikutnya switch dapat beroperasi secara mandiri menangani forwarding jenis paket tertentu

Cara penyimpanan state tergantung dari jenis switch.

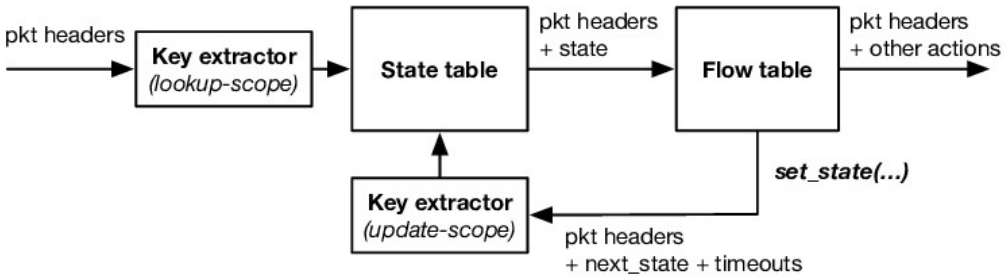
- OpenVswitch: menggunakan learn action,
  - State di instal dalam bentuk sebuah flow rule baru di dalam flow table
- Userswitch: menggunakan state table
  - State table dapat di *look up* dan di *update* berdasarkan operasi m*atch-actio*n
  - State berupa nilai integer
- P4-compatible Switch: menggunakan hashable register
  - State disimpan ke dalam register dengan alamat M, dimana M diperoleh dengan melakukan hash table pada header paket yang di match.

Openstate SDN

OpenState SDN: proyek open source untuk stateful SDN.

Berbasis user switch, dengan tambahan:

- Lookup key extractor: header field yang digunakan untuk lookup (stateful match)
- Update key extractor: header field yang digunakan untuk update (state update)
- State table: tempat penyimpanan state

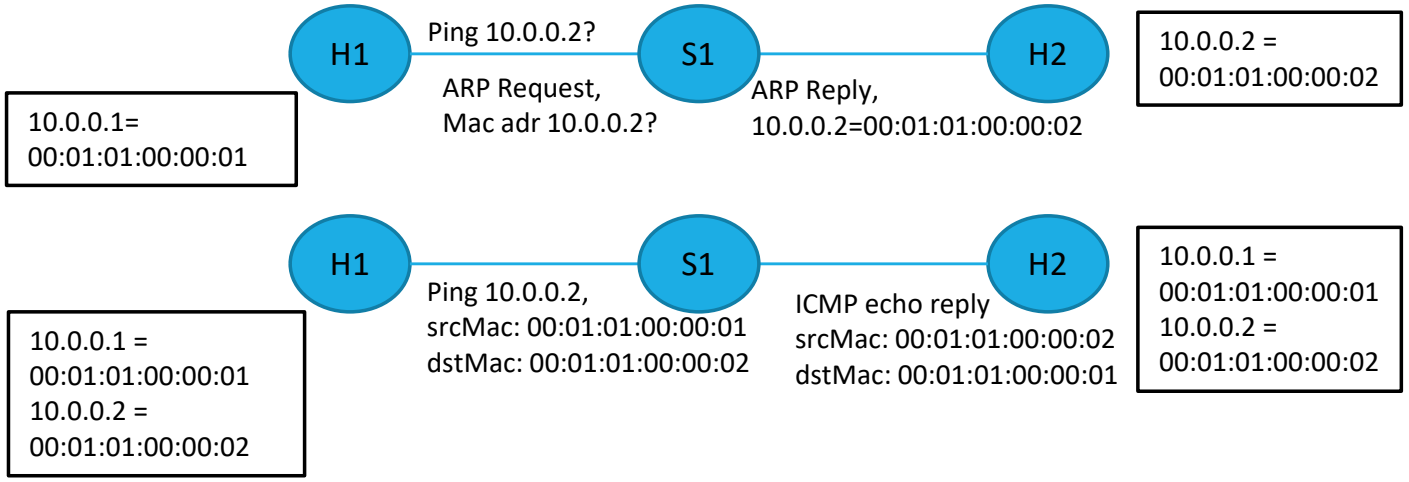


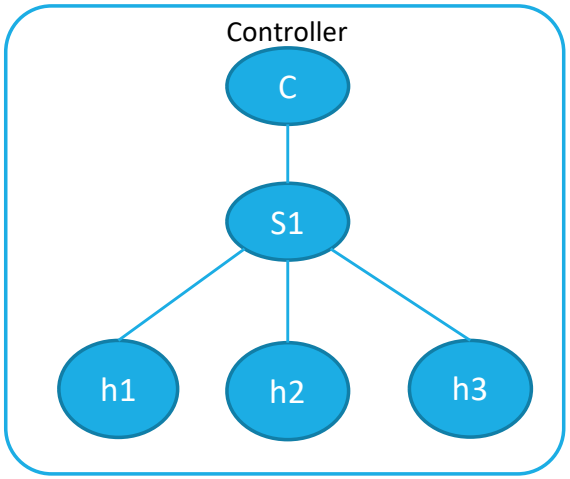
Studi kasus:

- ARP handling
- Port Knocking

Studi kasus 1 - ARP forwarding

- Stateless switch: membutuhkan bantuan controller untuk menginstall ARP flow rule pada s1 (non-flooding)





- *Stateful* switch: *controller* dapat menginstall flow rule statis untuk mengautomatisasi penginstalan flow rule paket ARP berdasarkan paket ARP request dan reply yang diterima tersebut.
- Tidak diperlukan bantuan *controller* lagi untuk memforward paket ARP ke host tujuan
- Tidak perlu selalu mem flood ARP request → dapat bekerja pada cyclic topology
- Dapat mengurangi *controller overhead* dan menurunkan *latency* pengiriman berkat *local process* di switch

Source code

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, event):

        """ Switche sent his features, check if OpenState supported """
        msg = event.msg
        datapath = msg.datapath

        LOG.info("Configuring switch %d..." % datapath.id)

        """ Set table 0 as stateful """
        req = osparser.OFPExpMsgConfigureStatefulTable(
                        datapath=datapath,
                        table_id=0,
                        stateful=1)
        datapath.send_msg(req)

        """ Set lookup extractor = {eth_dst} """
        req = osparser.OFPExpMsgKeyExtract(datapath=datapath,
                        command=osproto.OFPSC_EXP_SET_L_EXTRACTOR,
                        fields=[ofproto.OXM_OF_ETH_DST],
                        table_id=0)
        datapath.send_msg(req)

        """ Set update extractor = {eth_src}  """
        req = osparser.OFPExpMsgKeyExtract(datapath=datapath,
                        command=osproto.OFPSC_EXP_SET_U_EXTRACTOR,
                        fields=[ofproto.OXM_OF_ETH_SRC],
                        table_id=0)
        datapath.send_msg(req)

        # for each input port, for each state
        for i in range(1, N+1):
                for s in range(N+1):
                        match = ofparser.OFPMatch(in_port=i, state=s)
                        if s == 0:
                                out_port = ofproto.OFPP_FLOOD
                        else:
                                out_port = s
                        actions = [osparser.OFPExpActionSetState(state=i, table_id=0, hard_timeout=10),
                                        ofparser.OFPActionOutput(out_port)]
                        self.add_flow(datapath=datapath, table_id=0, priority=0,
                                        match=match, actions=actions)
```

Stateful operation hanya berlaku di flow table 0

Key lookup = ETH_DST (mac address tujuan)

Key update = ETH_SRC (mac address tujuan)

Set nilai state berdasarkan nilai input port

Forward paket ARP berdasarkan nilai state

Langkah pengujian

Pengujian:

- Jalankan mininet dengan jenis topologi Tree, dua host (PC):

  sudo python ~/Workshop/OpenFlow/Mininet/simpleTree.py
- Buka host1 (h1) dan host2 (h2) dari jendela mininet:

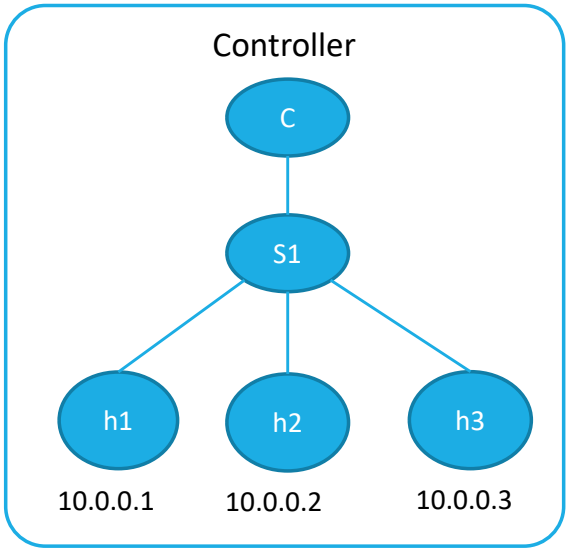
  Xterm h1 h2
- Jalankan program controller macLearning.py:

  Ryu-manager ryu/ryu/app/openstate/macLearning.py
- Pastikan semua flow rule telah terinstall di switch S1

  Sudo ovs-vsctl –protocol=OpenFlow13 dump-flows S1
- Matikan controller untuk mengetest kemampuan stateful forwarding

  Tekan ctrl + c di jendela controller
- Ping dari h1 ke h2:

  Ping 10.0.0.2



Latihan

1. Tambahkan operasi untuk memforward paket IP dengan cara yang hampir sama, dengan merubah key lookup dan update dengan Ip destination dan Ip source

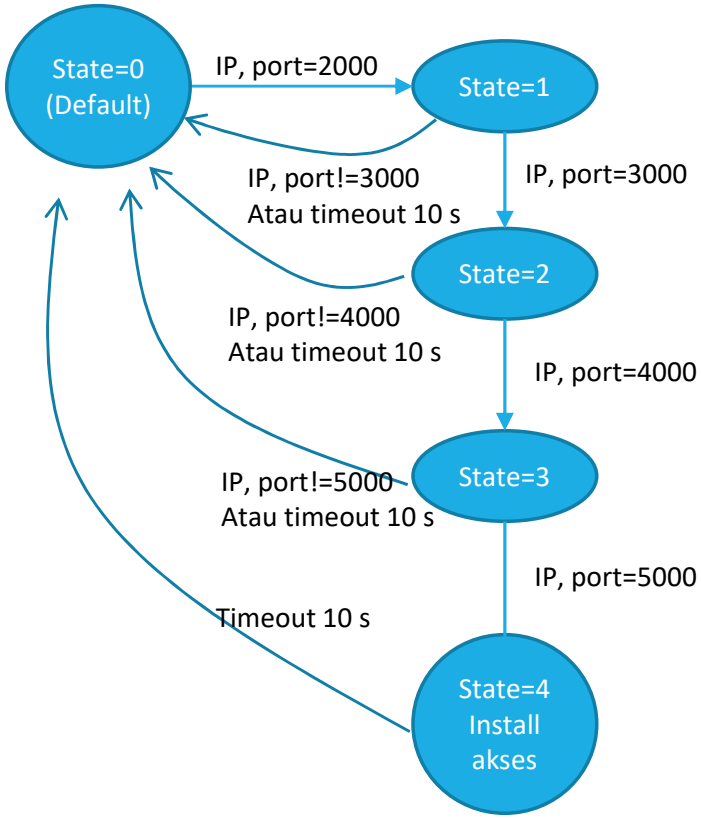
Studi kasus 2 – Port knocking

Port knocking: akses hanya diberikan apabila pihak yang akan mengajak berkomunikasi mampu mengirimkan urutan paket dengan port yang sesuai

Contoh urutan port untuk knocking: [2000, 3000, 4000, 5000]

Implementasi:

- Controller menginstall flow rule untuk port knocking
- User harus mengirimkan paket IP secara beruturan dengan konfigurasi port sesuai yang telah ditentukan
- Switch akan mengeset nilai state secara increment apabila paket yang dikirim sesuai urutan
- Apabila paket yang dikirim tidak sesuai, maka nilai state direset menjadi default (0)
- Apabila pengguna tidak mengirim paket dalam waktu 5 sampai 10 detik, maka nilai state akan direset ke default (0)
- Apabila pengguna suskses mengirimkan paket sampai selesai, switch akan memforward paket dari IP pengguna ke switch/host tujuan.



Source code

```
port_list = [2000, 3000, 4000, 5000]
final_port = port_list[-1]
second_last_port =  port_list[-2]
```

Daftar urutan port tujuan yang benar

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

        LOG.info("Configuring switch %d..." % datapath.id)

        """ Set table 0 as stateful """
        req = osparser.OFPExpMsgConfigureStatefulTable(datapath=datapath, table_id=0, stateful=1)
        datapath.send_msg(req)

        """ Set lookup extractor = {ip_src} """
        req = osparser.OFPExpMsgKeyExtract(datapath=datapath, command=osproto.OFPSC_EXP_SET_L_EXTRACTOR,
                                           fields=[ofproto.OXM_OF_IPV4_SRC], table_id=0)
        datapath.send_msg(req)

        """ Set update extractor = {ip_src} (same as lookup) """
        req = osparser.OFPExpMsgKeyExtract(datapath=datapath, command=osproto.OFPSC_EXP_SET_U_EXTRACTOR,
                                           fields=[ofproto.OXM_OF_IPV4_SRC], table_id=0)
        datapath.send_msg(req)

        """ ARP packets flooding """
        match = ofparser.OFPMatch(eth_type=0x0806)
        actions = [ofparser.OFPActionOutput(ofproto.OFPP_FLOOD)]
        self.add_flow(datapath=datapath, table_id=0, priority=100, match=match, actions=actions)

        """ Flow entries for port knocking """
        for i in range(len(port_list)):
                match = ofparser.OFPMatch(eth_type=0x0800, ip_proto=17, state=i, udp_dst=port_list[i])

                if port_list[i] != final_port and port_list[i] != second_last_port:
                        # If state not OPEN, set state and drop (implicit)
                        actions = [osparser.OFPExpActionSetState(state=i+1, table_id=0, idle_timeout=5)]
                elif port_list[i] == second_last_port:
                        # In the transaction to the OPEN state, the timeout is set to 10 sec
                        actions = [osparser.OFPExpActionSetState(state=i+1, table_id=0, idle_timeout=10)]
                else:
                        actions = [ofparser.OFPActionOutput(2)]
                self.add_flow(datapath=datapath, table_id=0, priority=10, match=match, actions=actions)

        """ Get back to DEFAULT if wrong knock (UDP match, lowest priority) """
        match = ofparser.OFPMatch(eth_type=0x0800, ip_proto=17)
        actions = [osparser.OFPExpActionSetState(state=0, table_id=0)]
        self.add_flow(datapath=datapath, table_id=0, priority=0, match=match, actions=actions)
```

Stateful operation hanya berlaku di flow table 0

Key lookup = IPV4_SRC(IPv4 source address)

Key update = IPV4_SRC(IPv4 source address)

Paket pentrigger: UDP, (match: port tujuan)

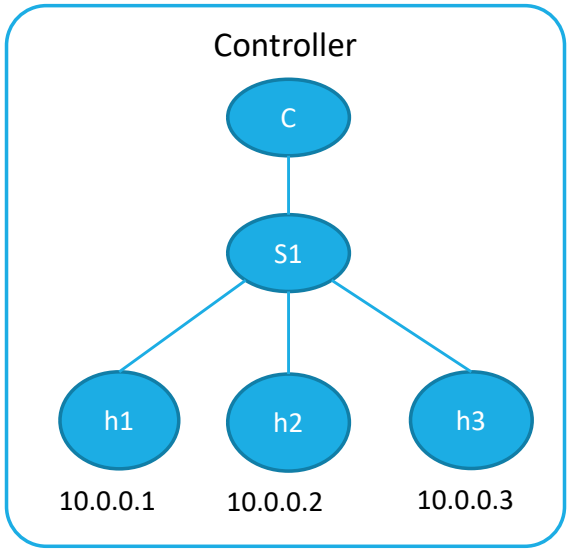
Implementasi port knocking

Default flow rule: reset nilai state jika tidak paket yang dikirim tidak match dengan aturan port knocking

Langkah pengujian

Pengujian:

- Jalankan mininet dengan jenis topologi Tree, dua host (PC):
  sudo python ~/Workshop/OpenFlow/Mininet/simpleTree.py
- Buka host1 (h1) dan host2 (h2) dari jendela mininet:
  Xterm h1 h2
- Jalankan program controller portKnocking.py:
  Ryu-manager ryu/ryu/app/openstate/portKnocking.py
- Pastikan semua flow rule telah terinstall di switch S1
  Sudo ovs-vsctl –protocol=OpenFlow13 dump-flows S1
- Matikan controller untuk mengetest kemampuan stateful forwarding
  Tekan ctrl + c di jendela controller
- Amati perilaku switch S1 untuk pengiriman paket dengan urutan acak dan yang sesuai aturan
  1. Kirim paket dengan port tujuan terpilih secara acak:
     ./random_port_knocking.sh
  2. Kirim paket dengan port tujuan yang benar:
     ./test_port_knocking.sh
- Amati respon yang dikirim oleh host 2 ketika berhasil



Daftar pustaka

1. http://openstate-sdn.org
2. https://github.com/OpenState-SDN/ryu/wiki/MAC-Learning-Tutorial
3. https://github.com/OpenState-SDN/ryu/wiki/Port-Knocking