Workshop OpenFlow

1. Basic Forwarding

Akbari Indra Basuki
Pusat Penelitian Informatika, LIPI

Daftar Materi

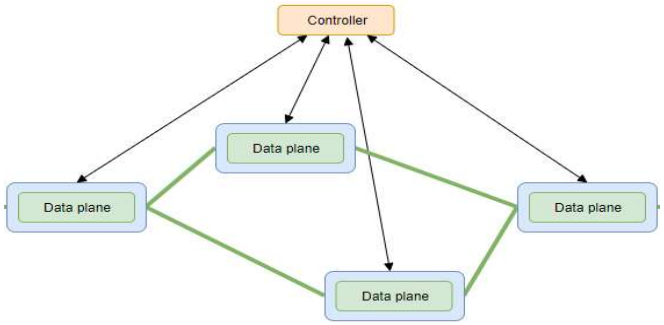
Topik	Keterangan
Basic forwarding	Dasar-dasar OpenFlow
Routing & Monitoring	Program Controller: Shortest-path routing Monitor node and link status NetworkX integration
Packet Filtering (Firewall + Web Interface)	Program Controller: Bloom Filter, Flask integration
Load balancing	Group bucket and group 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

Dasar-dasar OpenFlow

Controller:

- Bertugas menginstall dan memonitor OpenFlow *rules* pada setiap *switch (Data plane).*
- Pengguna dapat membuat aplikasi yang dijalan diatas *controller* untuk mengatur kerja *switch.*

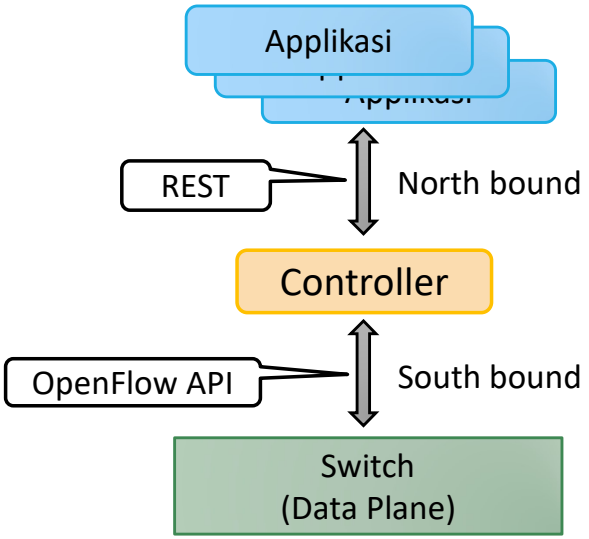
Jenis: Onos (Java), OpenDaylight (Java), **Ryu (Python)**, dll



OpenFlow *Switch*:

- *Data plane* yang bertugas untuk mengirim paket antar perangkat jaringan menggunalan OpenFlow *rules*.
- Workshop: menggunakan jenis *software switch.*

Jenis: **OpenVswitch**, ofsoftswitch, dll



Emulator:

- Mengemulasikan konfigurasi dan topologi jaringan pada suatu PC.
- Workshop: menggunakan **Mininet.**

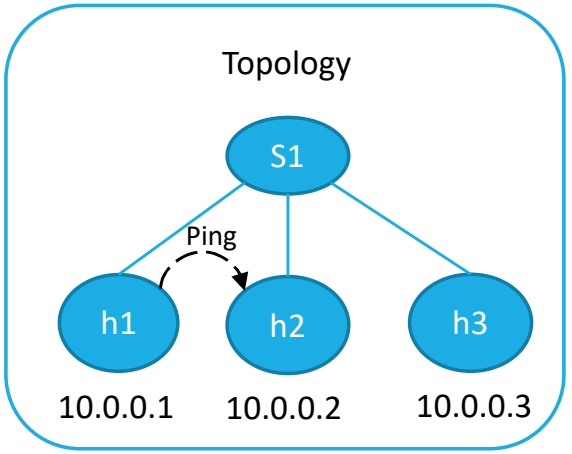
Persiapan testbed (*hello world*)

Semua tools sudah terinstall di dalam VM Virtualbox:

- Mininet2.2-VM_Workshop_OpenFlow
- Untuk menginstall semua *tool* dari awal → Lampiran 1

Test VM sudah berjalan normal,

- 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
- Buka jendela h1 dan ping ke h2:
  Ping 10.0.0.2
- Jalankan program controller simpleswitch.py:
  Ryu-manager simpleswitch.py
- Ping ulang h1 ke h2:
  Ping 10.0.0.2



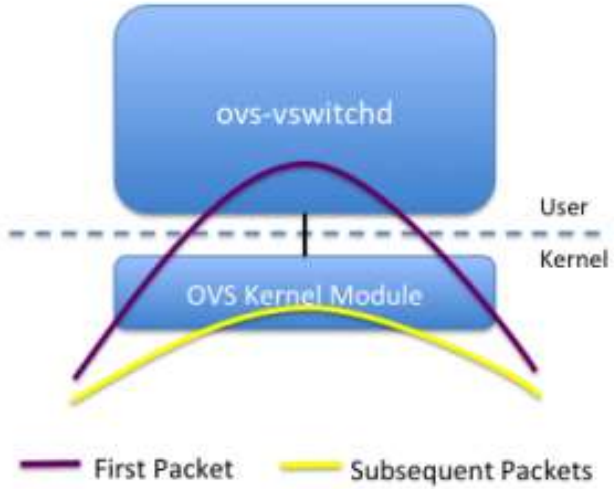
OpenFlow (Software) Switch

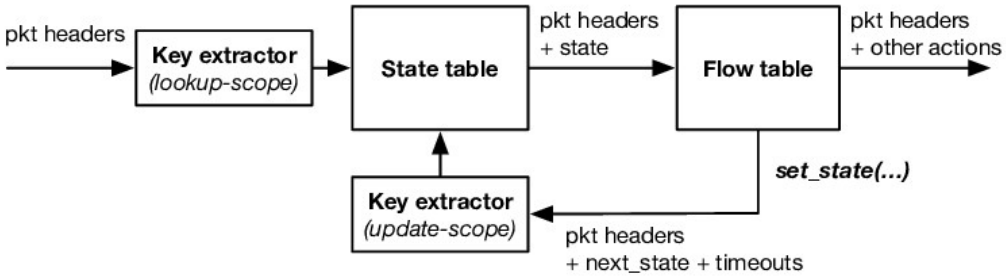
OpenVSwitch

- Kernel mode (mega cache)
- Software switch de facto

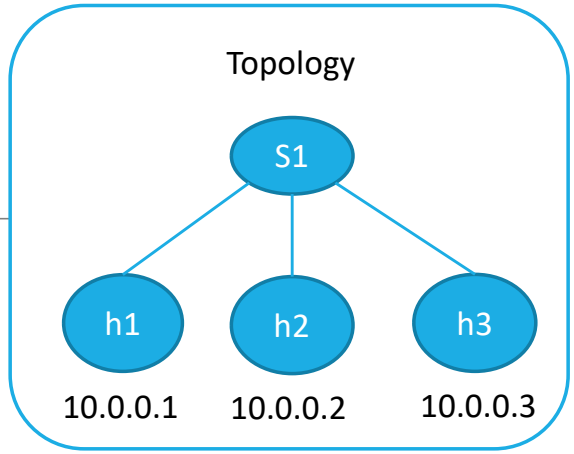
UserSwitch

- Mendukung stateful mode
- Kode lebih sederhana, mudah dikembangkan untuk proof of concept





Topologi dengan Mininet



Implementasi topologi jaringan menggunakan mininet:

- via Python → mendefinisikan topologi, jenis controller, jenis switch, dan konfigurasi link.
- via mininet custom → hanya mendefinisikan topologi.

*simpleTree.py

File Edit Search Options Help

```
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch
from mininet.link import TCLink

if '__main__' == __name__:

        net = Mininet(controller=RemoteController, switch=OVSSwitch,
                                link=TCLink, autoSetMacs = True)
        #OVSSwitch
        c0 = net.addController('c0', controller=RemoteController, port=6633)
        h1 = net.addHost('h1', ip='10.0.0.1', MAC='01:01:01:00:00:01')
        h2 = net.addHost('h2', ip='10.0.0.2', MAC='01:01:01:00:00:02')
        h3 = net.addHost('h3', ip='10.0.0.3', MAC='01:01:01:00:00:03')

        s1 = net.addSwitch('s1')

        #add link
        net.addLink( h1, s1, delay='1ms')
        net.addLink( h2, s1, delay='1ms')
        net.addLink( h3, s1, delay='1ms')

        net.build()
        c0.start()
        s1.start([c0])
        s1.cmd( 'ovs-vsctl set Bridge s1 protocols=OpenFlow13')
        CLI(net)
        net.stop()

#Run:
#sudo python simpleTree.py
```

simpleTree_custom.py

File Edit Search Options Help

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
        h3 = self.addHost( 'h3' )
        s1 = self.addSwitch( 's1' )

        # Add links
        self.addLink( h1, s1 )
        self.addLink( h2, s1 )
        self.addLink( h3, s1 )


topos = { 'mytopo': ( lambda: MyTopo() ) }

#how to run
#sudo mn --custom ~/mininet/custom/simpleTree_custom.py --topo=mytopo
```

Aplikasi Controller (*Ryu*)

Template program aplikasi di controller

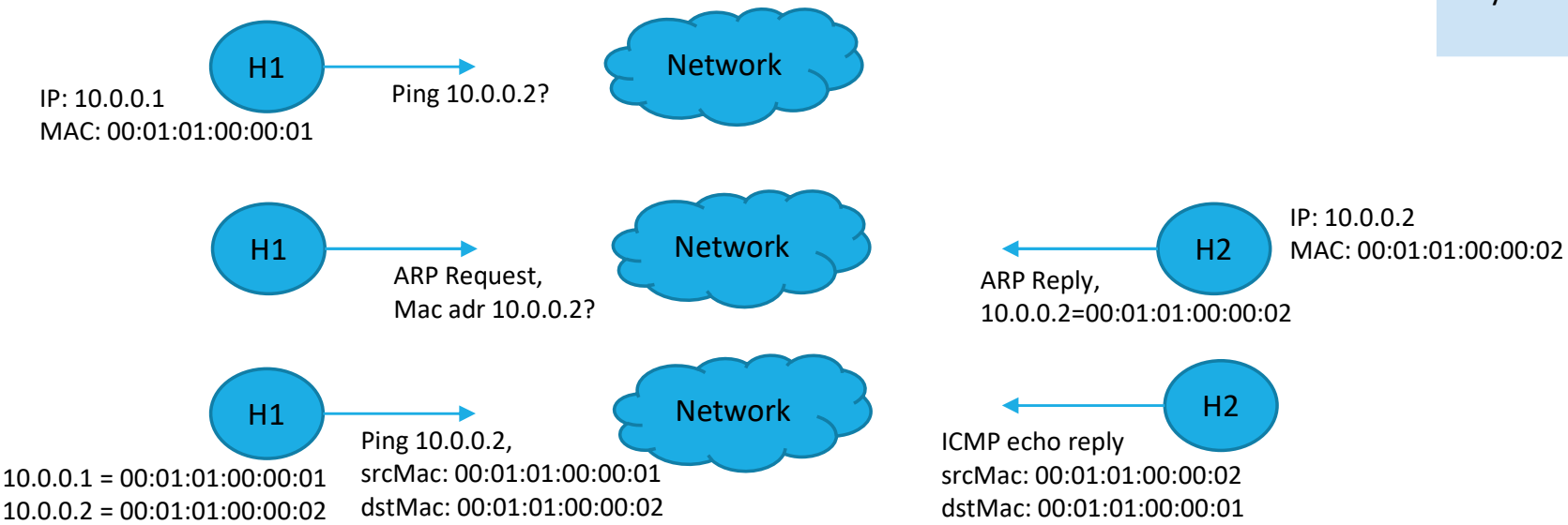
```
from ryu.base import app_manager                                    # digunakan untuk tipe kelas (inheritance-OOP)
from ryu.controller import ofp_event                                # library jenis-jenis event OpenFlow yang dikenali oleh controller
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER   # library jenis-jenis status traksaksi antara controller-switch
from ryu.controller.handler import set_ev_cls                       # listener untuk penghubung event, status dan fungsi
from ryu.ofproto import ofproto_v1_3                                # library berisi versi openflow
from ryu.lib.packet import packet                                   # library untuk penanganan paket
from ryu.lib.packet import ethernet                                 # library untuk protokol ethernet

class ExampleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
    ### OFP_VERSIONS: menentukan versi OpenFlow yang dipakai, disini memakai versi OF 1.3

    def __init__(self, *args, **kwargs):
        ### Fungsi yang pertama kali dijalankan oleh controller ketika pertama kali mulai
        ### contoh: initialisasi variabel, pooling data konfigurasi switch, dll.

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)    # config_dispatcher: true jika switch berhasil terkoneksi
    def switch_features_handler(self, ev):                              # ev: berisi pesan/data yang dikirim oleh pentrigger
        ### Fungsi yang otomatis dipanggil ketika switch terkoneksi ke controller untuk pertama kali
        ### contoh: Menhapus flow rule ygn sudah ada, atau menginstall flow rule default
        ### datapath = ev.msg.datapath      <<< ekstrak switch pengirim dari ev (event)

    def add_flow(self, datapath, priority, match, actions):
        ### Fungsi yang dipanggil untuk menginstall flow rule ke switch,
        ### Flow rule dapat juga dikirim langsung tanpa melalui fungsi ini, <dibuat fungsi supaya lebih rapi>

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)            # main_dispatcher: kondisi normal/switch sudah terkoneksi
    def _packet_in_handler(self, ev):
        ### Fungsi yang otomatis dipanggil ketika controller menerima kiriman packet dari switch
        ### Contoh:
        ### msg = ev.msg                    <<< baca pesan dari event yg diterima, event: EventOFPPacketIn (paket masuk)
        ### datapath = msg.datapath         <<< ekstrak switch pengirim dari pesan
        ### dpid = datapath.id              <<< contoh membaca switch ID
        ### pkt = packet.Packet(msg.data)   <<< ekstrak header paket yang dikirim
        ### eth_pkt = pkt.get_protocol(ethernet.ethernet)
        ### dst = eth_pkt.dst               <<< contoh membaca MAC address destination dari header packet
```

Hello World – Simple Switch (L2)

L2 witch: Protokol ARP dan ICMP

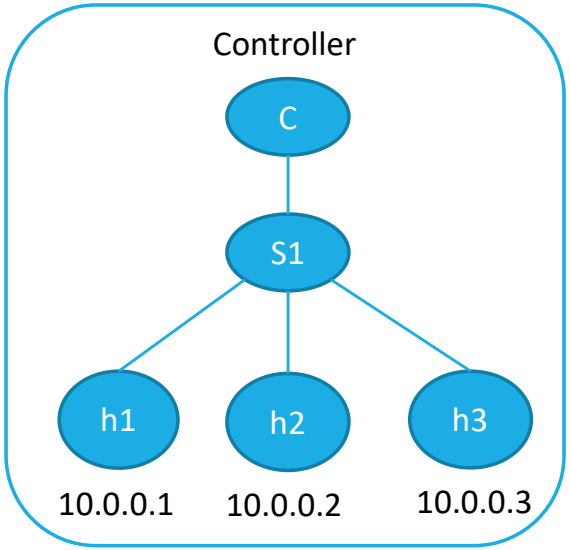
- ARP Table: Tabel pasangan MAC address dan IP address
- ICMP protokol: ping antar host,
  - IP dan Mac address harus diketahui sebelum dapat melakukan ping
  - Host akan mengirim ARP request untuk mengetahui IP address host tujuan

Internet Layer	Contoh protokol
Application layer	HTTP, dll
Transport layer	**ICMP**, TCP, UDP, dll
Network layer	**IP**
Data link layer	**ARP** (machine Addreess)
Physical layer	Ethernet Kabel, Fiber optic



Implementasi

- Default rule switch S1: Flood paket ARP (request & reply) ke semua port termasuk controller.
- Setiap hosts akan otomatis menginstall arp table sesuai paket ARP yang diterima
- Controller menggunakan infomrasi dari paket ARP untuk: menginstall flow rule untuk memforward ICMP packet di switch S1
- Antar host dapat saling melakukan perintah Ping



Source Code

Pertamakali switch terkoneksi ke controller, install static rule

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

        self.swList.append(dp)                  #daftar switch yang terkoneksi

        match = dp.ofproto_parser.OFPMatch(eth_type=0x806)  #0x806 -> ARP ethertype
        actions = [dp.ofproto_parser.OFPActionOutput(ofproto.OFPP_FLOOD, 0),
                           dp.ofproto_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER
                           ofproto.OFPCML_NO_BUFFER)]
        #FLOOD --> khusus topologi acyclic.
        #Pada topologi cyclic --> endless flooding --> overhead

        #install flow rule, pada switch dp, dengan skema match-actions
        self.add_flow(dp, match, actions)
```

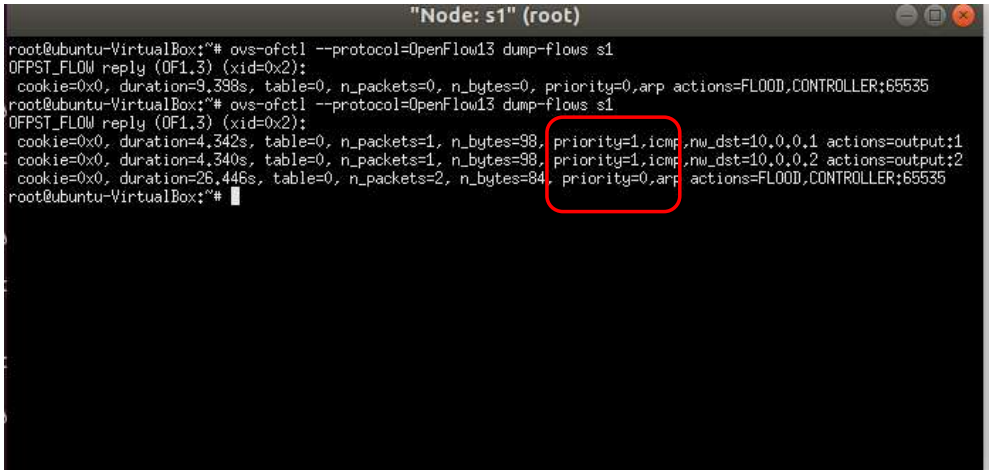
Flood paket ARP dan kirim ke controller

Untuk semua paket yang dikirim ke controller, cek apakah ada paket ARP

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
        msg = ev.msg
        in_port = msg.match['in_port']
        dp = msg.datapath
        ofproto = dp.ofproto
        dpid = dp.id
        pkt = packet.Packet(msg.data)
        pkt_arp = pkt.get_protocols(arp.arp)[0]

        if pkt_arp:
                print ("receiving arp packet")
                #ambil IP pengirim dan asosiasikan dengan input port.
                pkt_arp = pkt.get_protocol(arp.arp)
                sipv4 =  pkt_arp.src_ip
                self.hostDB[sipv4] = in_port

                #pasang flow rule, jika menerima paket ICMP dengan tujuan IP pengirim, kirim ke input_port
                match = dp.ofproto_parser.OFPMatch(eth_type=0x800,ip_proto=0x01,ipv4_dst=sipv4)
                actions = [dp.ofproto_parser.OFPActionOutput(self.hostDB[sipv4] , 0)]
                self.add_flow(dp, match, actions, 1)
                print ("install flow rule: match: ICMP to "+sipv4+" output:"+str(in_port))
```

Install flow rule untuk ICMP berdasarkan informasi dari paket ARP

```
def add_flow(self, datapath, match, actions,priority=0):
        ofproto = datapath.ofproto

        #instruksi dasar untuk mengeksekusi semua perintah di daftar actions
        inst = [datapath.ofproto_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
        mod = datapath.ofproto_parser.OFPFlowMod(
                datapath=datapath,                  #switch id
                cookie=0, cookie_mask=0,
                table_id=0,                         #nomor Flow table dimana flow rule di install
                command=ofproto.OFPFC_ADD,
                idle_timeout=0, hard_timeout=0,     #timeout = 0 -> tidak memiliki timeout
                priority=priority,                  #menentukan urutan matching
                buffer_id=ofproto.OFP_NO_BUFFER,
                out_port=ofproto.OFPP_ANY,
                out_group=ofproto.OFPG_ANY,
                flags=0,
                match=match,                        #perintah match
                instructions=inst)                  #perintah actions
        datapath.send_msg(mod)
```



Latihan

1. Tambah kemampuan forwarding protokol IP TCP/UDP
   - Tips: Ubah kode protokol ICMP menjadi kode TCP/UDP

Daftar pustaka

- RyuBook. https://osrg.github.io/ryu-book/en/Ryubook.pdf
- OpenVSwitch. https://docs.openvswitch.org/en/latest/
- Mininet. http://mininet.org/walkthrough/
- Networkx. https://networkx.github.io/documentation/stable/tutorial.html

Lampiran 1. Instalasi *tools* secara manual

Controller Ryu:

- sudo apt-get install git python-dev python-setuptools python-pip
- git clone https://github.com/osrg/ryu.git
- cd ryu
- sudo pip install .

Mininet from source code:

- git clone git://github.com/mininet/mininet
- cd mininet
- git tag # list available versions
- git checkout -b 2.2.1 2.2.1 # or whatever version you wish to install
- cd ..
- mininet/util/install.sh [options]
- Daftar opsi:
  - install.sh –a , install semua paket
  - install.sh –nfv , install Mininet + user switch + OpenVswitch

OpenVswitch:

Dapat di install bersamaan dengan mininet

Atau dari source code.

- git clone https://github.com/openvswitch/ovs.git
- git checkout v2.7.0
- git checkout origin/branch-2.7
- ./boot.sh
- ./configure --with-linux=/lib/modules/$(uname -r)/build
- make
- make install
- make modules_install
- config_file="/etc/depmod.d/openvswitch.conf"
- for module in datapath/linux/*.ko; do
- modname="$(basename ${module})"
- echo "override ${modname%.ko} * extra" >> "$config_file"
- echo "override ${modname%.ko} * weak-updates" >> "$config_file"
- done
- depmod -a
- /sbin/modprobe openvswitch
- export PATH=$PATH:/usr/local/share/openvswitch/scripts
- mkdir -p /usr/local/etc/openvswitch
- ovsdb-tool create /usr/local/etc/openvswitch/conf.db \
- vswitchd/vswitch.ovsschema
- mkdir -p /usr/local/var/run/openvswitch
- ovsdb-server --remote=punix:/usr/local/var/run/openvswitch/db.sock \
- --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
- --private-key=db:Open_vSwitch,SSL,private_key \
- --certificate=db:Open_vSwitch,SSL,certificate \
- --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
- --pidfile --detach --log-file
- ovs-vsctl --no-wait init
- ovs-ctl start