# ONOS-P4 Tutorial
# Hands-on Activity

P4 Brigade Work Days, Seoul (Korea)

September 18-29, 2017

# Tutorial VM

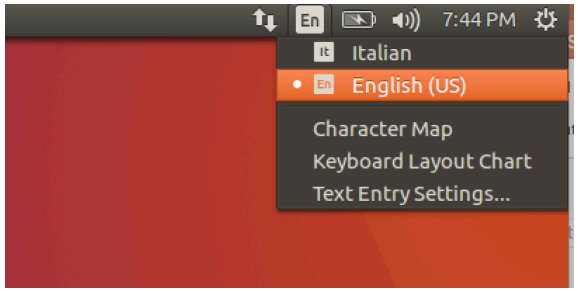
- Download (~4GB)
  - http://bit.ly/onos-p4-dev-vm
- Run
  - The VM is in .ova format and has been created using VirtualBox v5.1.
  - To run the VM you can use any modern virtualization system, although we recommend using VirtualBox.
  - To download VirtualBox and import the VM use the following links:
    - https://www.virtualbox.org/wiki/Downloads
    - https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html

# System requirements

- The current configuration of the VM is 6 GB of RAM and 4 core CPU, although this is the recommended configuration, it can be reduced if needed.
- Minimum system requirements to build and run ONOS are 2 core CPU and 2 GB of RAM.
- When imported, the VM takes approx. 8 GB of HDD space, however, you might need up to 15 GB to build and run ONOS.

# Tutorial VM content

- VM content
  - ONOS
  - P4 tools (p4c, BMv2, P4Runtime)
  - IntelliJ IDEA (Java IDE for ONOS)
- Credentials
  - User: “sdn”
  - Password: “rocks”
- Important: remember to set your keyboard settings:



# Update onos

- To complete the tutorial exercises you will need to download, build and run the latest development version of ONOS.
- Once you are able to run the VM, open a terminal window or SSH into it and type the following commands:

```
cd ~/onos
git pull origin master
buck build onos
```

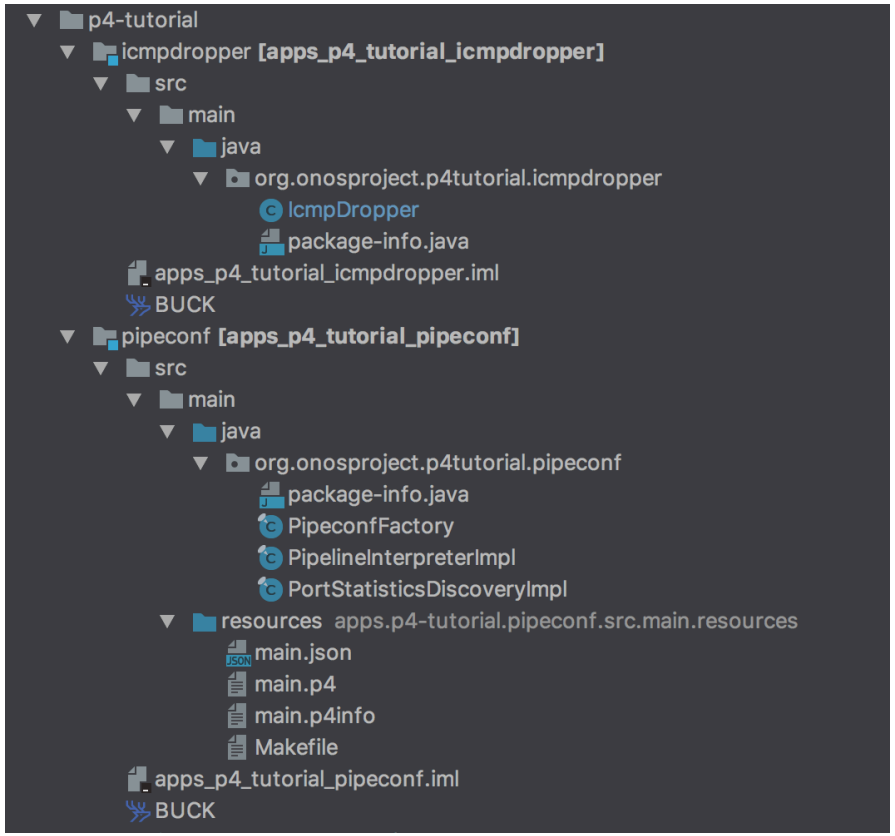
Building ONOS with the recommended system configuration takes approx. 10 minutes.

# Exercise instructions

http://bit.ly/onos-p4-tutorial

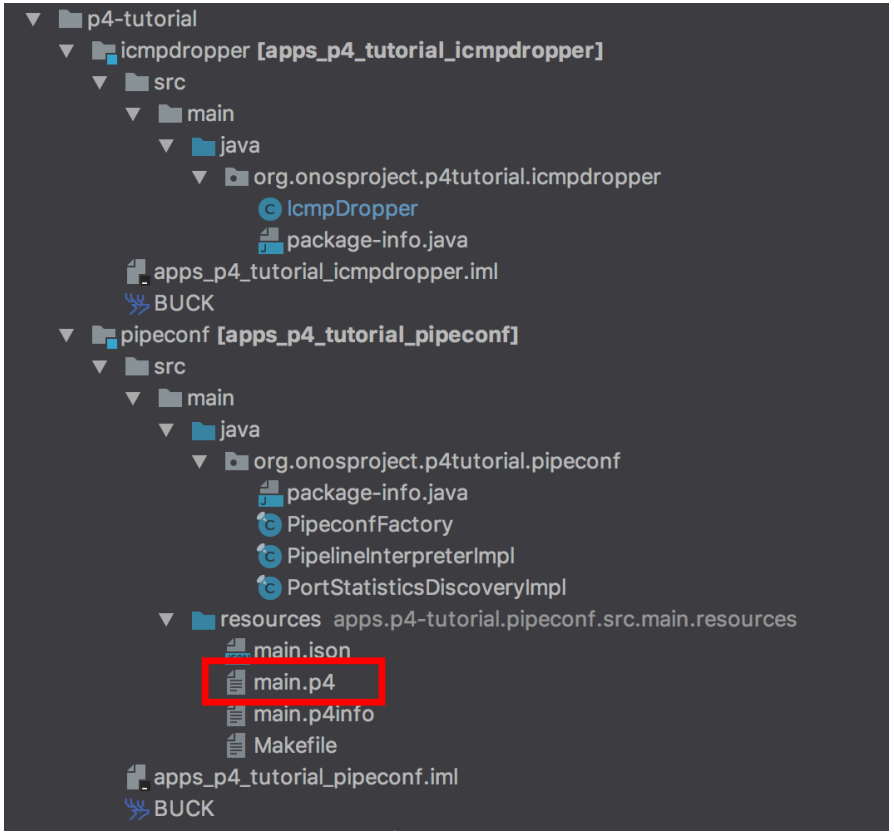
# Tutorial application

- Under ~/onos/apps/p4-tutorial
- Example pipeconf
- Example pipeline-aware application
  - IcmpDropper



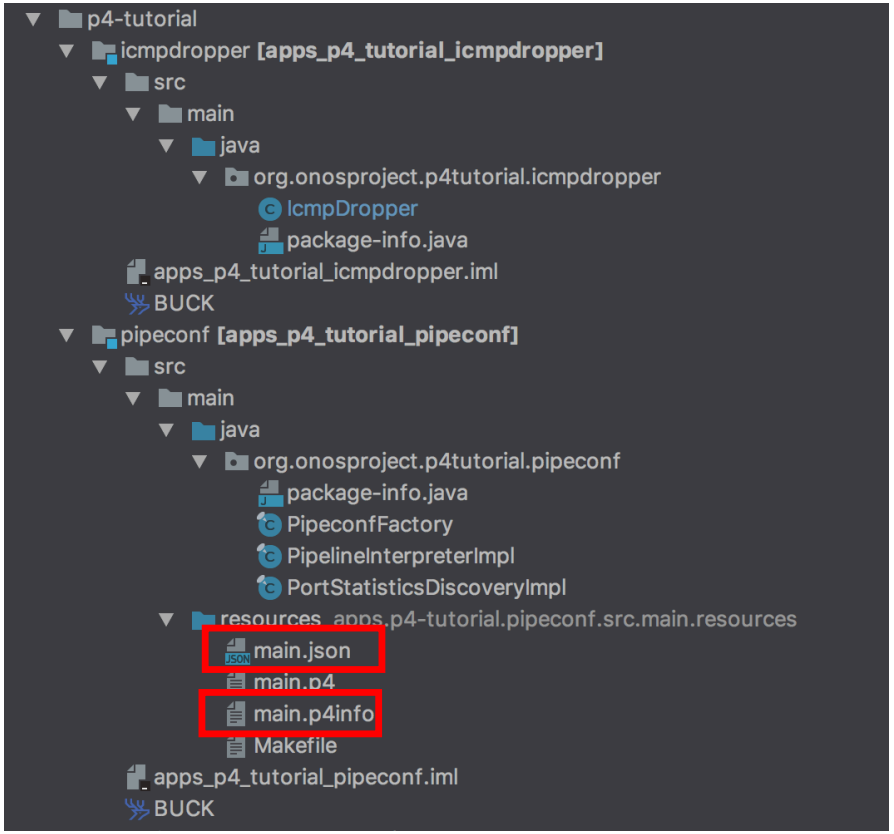
# Main.p4

- Parsed headers:
  - Ethernet, IPv4
- Ingress pipeline with 2 tables
  - Table0
    - Basic L2 forwarding capabilities
    - Actions
      - set_egress_port
      - send_to_cpu (packet-in)
      - drop
  - Ip_proto_filter_table
    - Filter (drop) packets based on IPv4 protocol code
      - ICMP, TCP, UDP, etc.
- Port counters
- Packet-in/out



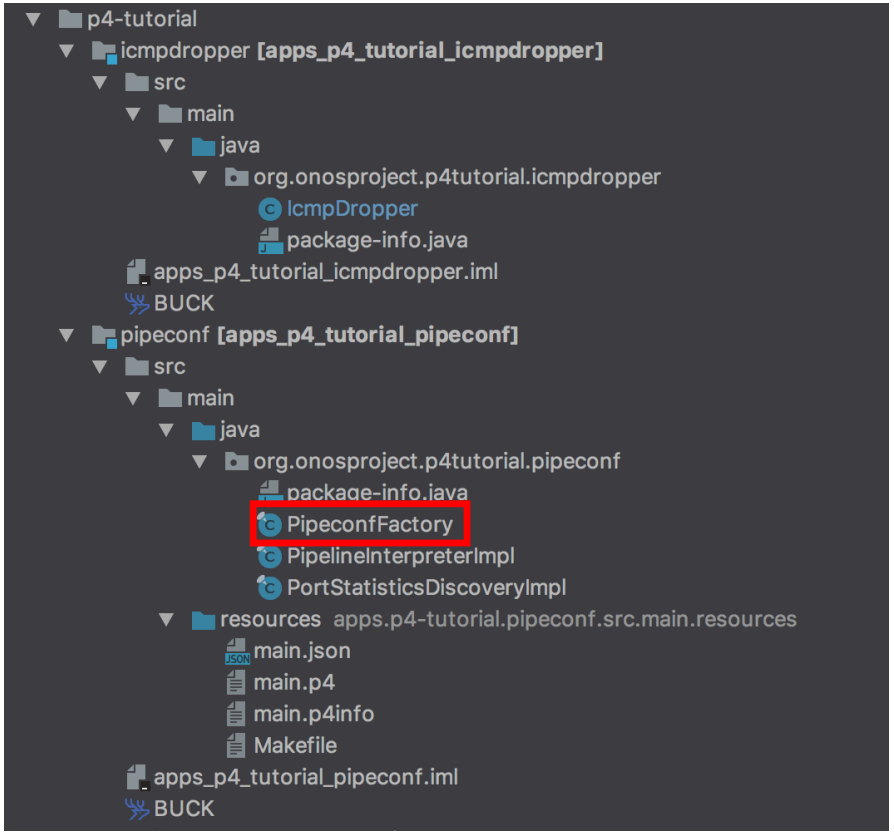
# Main.json/p4info

- Generated using p4c
- Makefile available with command to generate them
  - Contains workaround for p4c/bmv2 bug



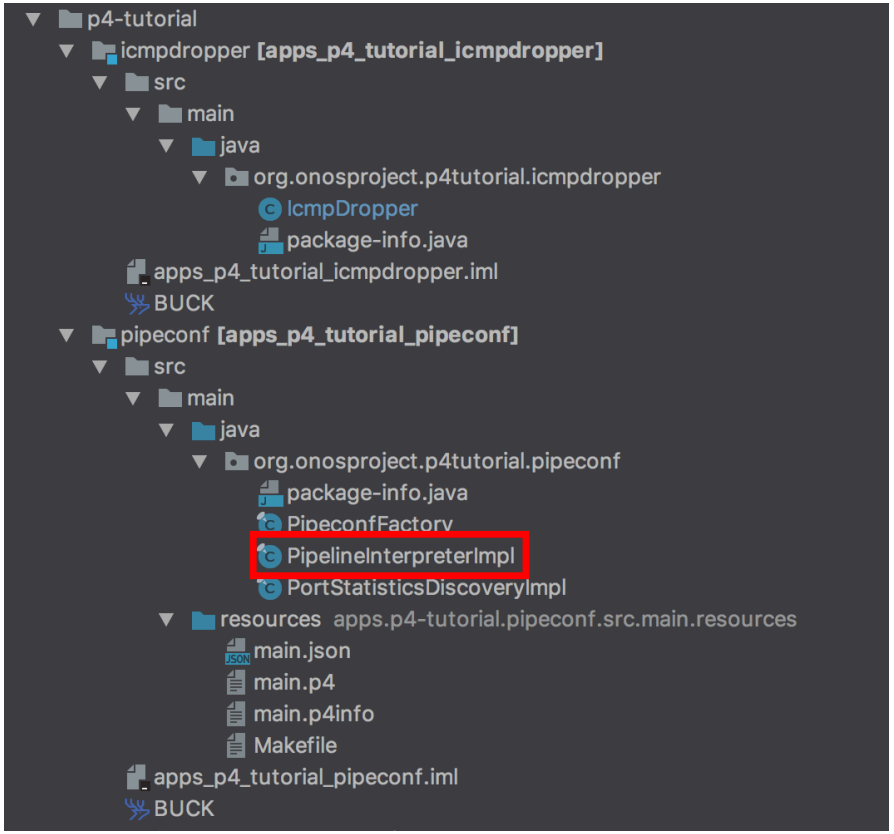
# PipeconfFactory.java

- Class that defines the pipeconf
- 3 driver behaviours
  - Interpreter (mandatory)
  - Pipeliner (default single table)
  - PortStatisticsDiscovery (used to read port counters)
- 2 extensions
  - P4Info
  - BMv2 JSON
- Class activated by OSGi runtime
  - @component
  - Pipeconf registered at component activation



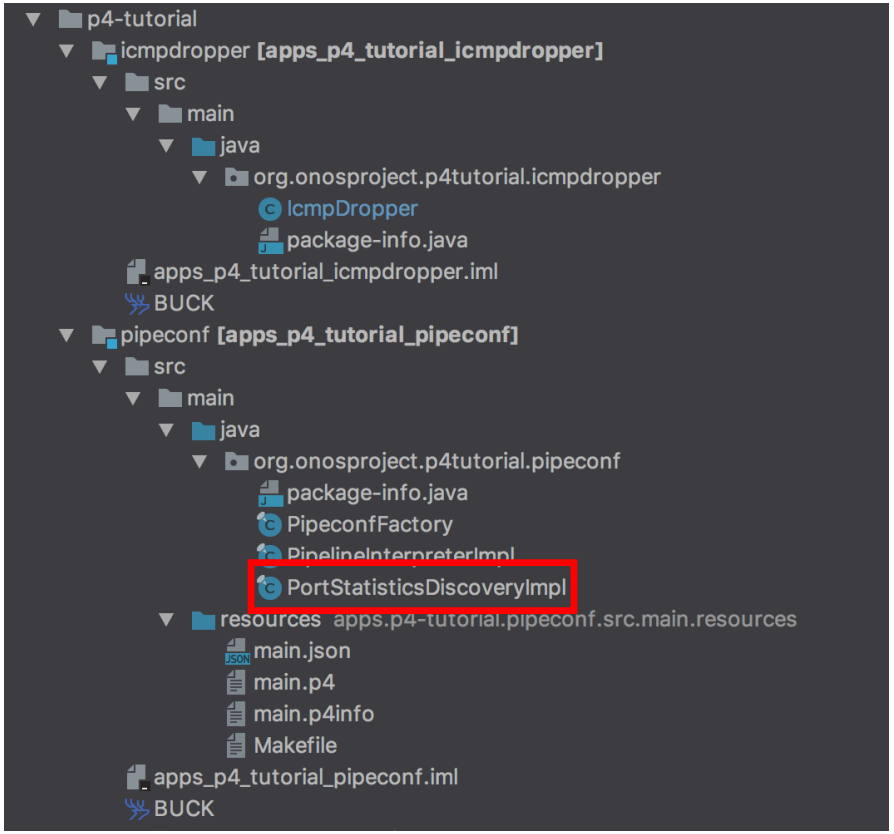
# PipelineInterpreterImpl.java

- Criterion mapping
  - IN_PORT, ETH_SRC, ETH_DST, ETH_TYPE
- Treatment mapping
  - No instructions = drop
  - OUTPUT
    - Switch physical ports
    - Controller
    - Other logical ports are not supported (e.g. FLOOD)
- Table ID mapping
  - 0 = table0
  - 1 = ip_proto_filter_table
- Packet I/O mapping



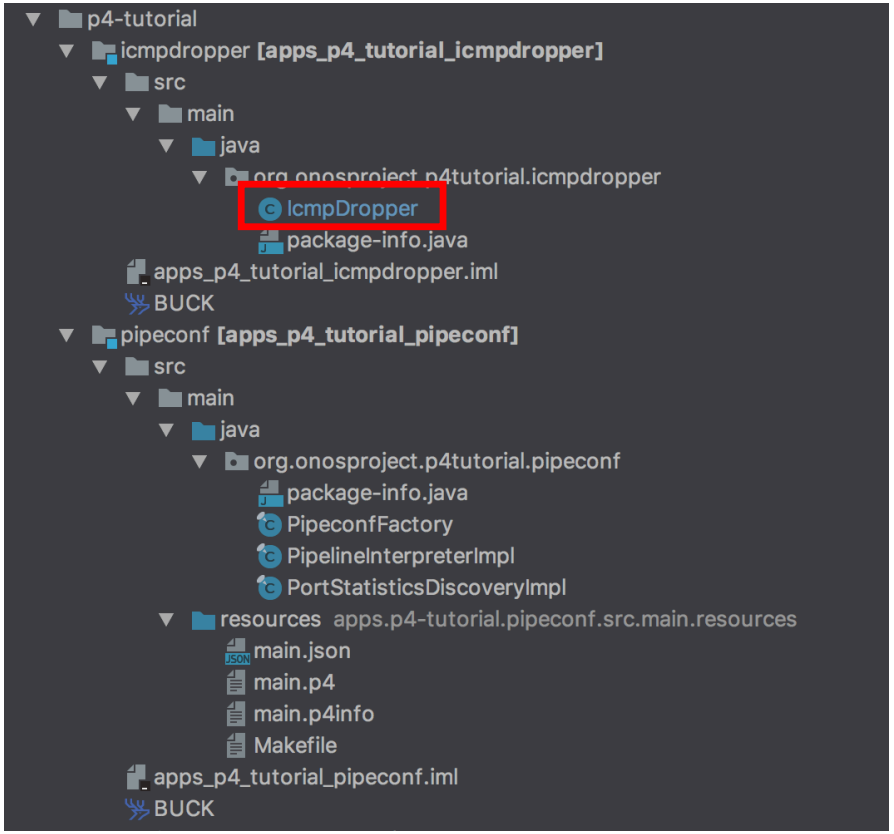
# PortStatisticsDiscoveryImpl.java

- Use P4RuntimeClient to read port counters from device
- Specific to main.p4 program
  - Port counter names defined there
    - igr_port_counter (ingress)
    - egr_port_counter (egress)

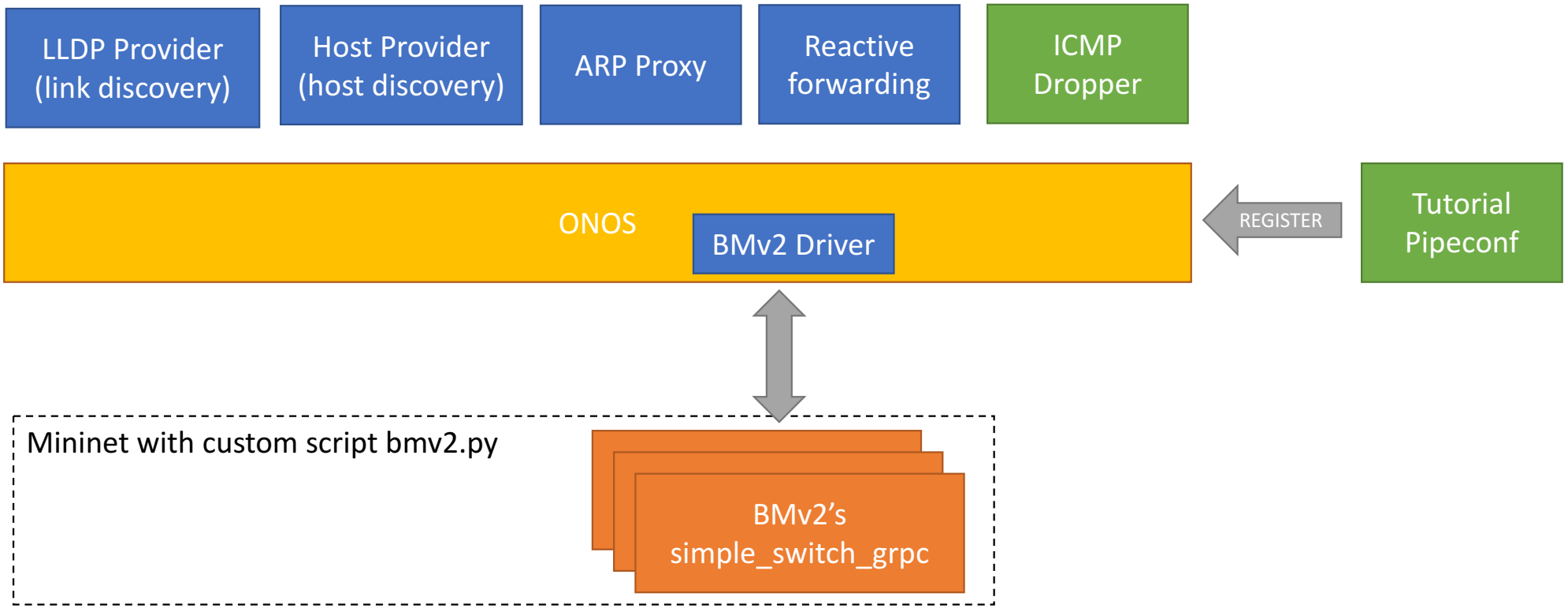


# IcmpDropper.java

- Example applications
- Registers a device listener
- Installs 1 flow rule for each new device
  - Drop ICMP packets
- Flow rule uses PI Criterion/Instruction
  - Header field name and action name same as in main.p4



# Environment



# LLDP Provider

- App ID: org.onosproject.lldpprovider
- Provides means to discover network links by injecting LLDP packets in the network
  - Install packet request (flow rule) on each device
    - Match: ETH_TYPE = LLDP
    - Instructions: OUTPUT(CONTROLLER)
  - Periodically issues LLDP packets via packet-out for each switch port

# ARP Proxy

- App ID: org.onosproject.proxyarp
- Intercepts ARP requests sent by hosts via packet-in, generates ARP replies. I.e. ARP packets are not forwarded on the network, but are handled exclusively by the controller
  - Match: ETH_TYPE = ARP
  - Instructions: OUTPUT(CONTROLLER)

# Host location provider

- App ID: org.onosproject.hostprovider
- Learns location of hosts by passively intercepting ARP packets
  - Mapping: ETH/IP addr ⟷ Switch-ID:Port-ID
- No rules installed, listens for packet requests installed by other applications (e.g. ARP proxy)

# Reactive forwarding

- App ID: org.onosproject.~~hostprovider~~ **fwd**
- Intercepts IP packets for which there are no matching flow rules on the switch
- If the location of the destination host is known, installs the necessary flow rule to forward subsequent packets
- Otherwise, packet is flooded on all ports of the switch where it was received via packet-out
- Flow rules
  - IP packet intercept
    - Match: ETH_TYPE = IPv4, Instruction: OUTPUT(CONTROLLER)
  - Forwarding
    - Match: IN_PORT, ETH_SRC, ETH_DST, Instruction: OUTPUT(port)

# Mininet script: bmv2.py

- Provided with ONOS
- Executes BMv2's simple_switch_grpc inside Mininet
- Automatically generates and push to ONOS the netcfg.json
  - Files located under /tmp
- Example command to use bmv2.py

```
sudo -E mn --custom $BMV2_MN_PY \
  --switch onosbmv2,loglevel=debug,pipeconfId=p4-tutorial-pipeconf
  --controller remote,ip=<IP of your ONOS instance>
```

`$BMV2_MN_PY` env variable that points to the location of the bmv2.py script

# 2 Exercises

- Look at the instructions here: http://bit.ly/onos-p4-tutorial
- Exercise 1
  - Run Mininet with BMv2, program devices with example pipeconf and use standard ONOS applications (pipeline-agnostic) to provide basic forwarding capabilities
- Exercise 2
  - Load ICMP dropper application

# Pointers

- P4 Brigade
  - https://wiki.onosproject.org/x/2oS9
- Subscribe to the mailing list
- Will share there slides and tutorial instructions