Open Networking Foundation

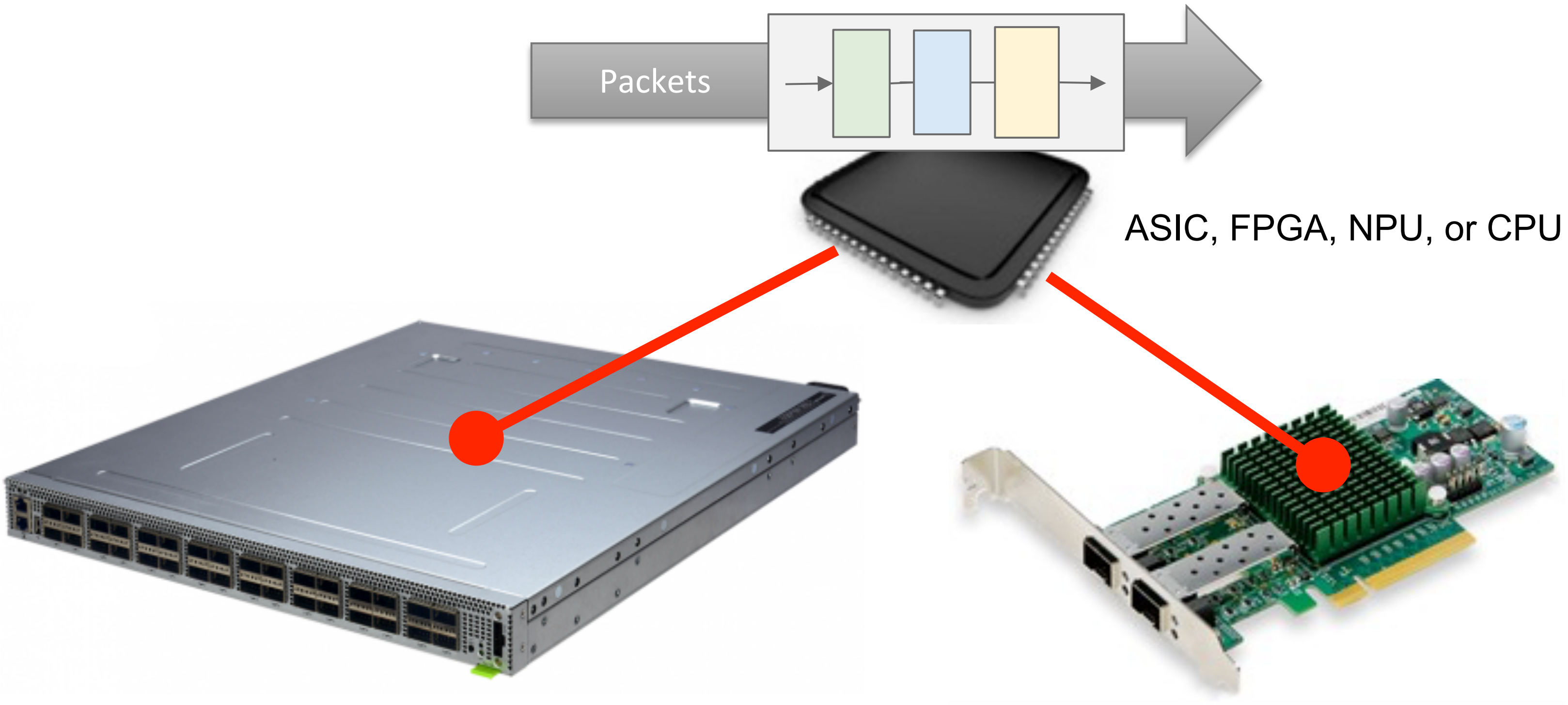
ONOS Support for P4

Carmelo Cascone
MTS, ONF

December 6, 2018

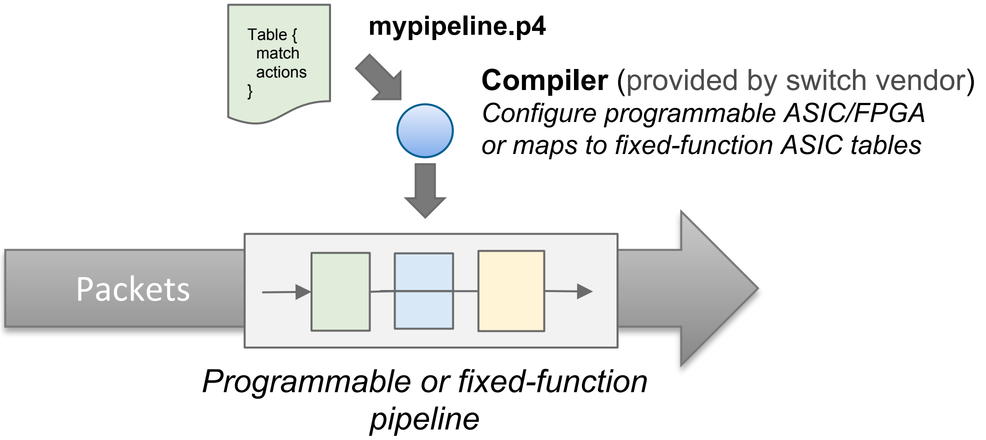
Pipelines

Pipeline of match-action tables



P4 - The pipeline programing language

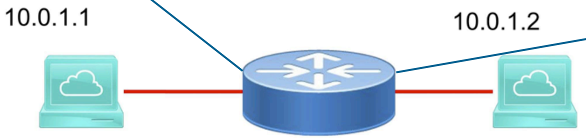
- **Domain-specific language to formally define a forwarding pipeline**
  - Describe protocol headers to parse, lookup tables, actions, counters, etc.
  - Can describe fast pipelines (e.g ASIC, FPGA) as well as a slower ones (e.g. SW switch)
- **Good for programmable switches, as well as fixed-function ones**
  - Defines "contract" between the control plane and data plane for runtime control



Runtime control

- Data plane program (P4)
  - Defines the match-action tables
  - Performs the lookup
  - Executes the chosen action
- Control plane (runtime)
  - Populates table entries with specific information
  - Based on configuration, automatic discovery, protocol calculations

```
action ipv4_forward(bit<48> dst_addr, bit<9> port) {
  ethernet.dst_addr = dst_addr;
  standard_metadata.egress_spec = port;
  ipv4.ttl = ipv4.ttl - 1;
}
table ipv4_routing_table {
 key = {
        ipv4.dst_addr : LPM; // longest-prefix match
 }
 actions = {
        ipv4_forward();
        drop();
  }
}
```

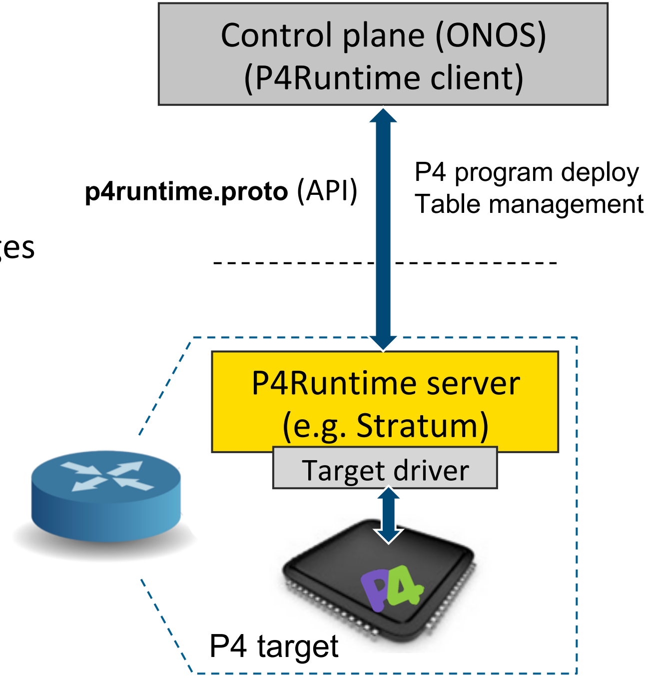


Control plane populates table entries

Key	Action	Action Data
10.0.1.1/32	ipv4_forward	dstAddr=00:00:00:00:01:01 port=1
10.0.1.2/32	drop	
*`	NoAction	

P4Runtime - Runtime API for P4-defined switches

- **In other words, manage P4-defined tables**
- **Community-developed (p4.org API WG)**
  - RC4 of version 1.0 available: https://p4.org/p4-spec/
- **gRPC/protobuf-based API definition**
  - Automatically generate client/server code for many languages
- **P4 program-independent**
  - API doesn't change with the P4 program
  - Independent of the specific protocols or actions
- **Enables field-reconfigurability**
  - Ability to push new P4 program, i.e. re-configure the switch pipeline, without recompiling the switch software stack

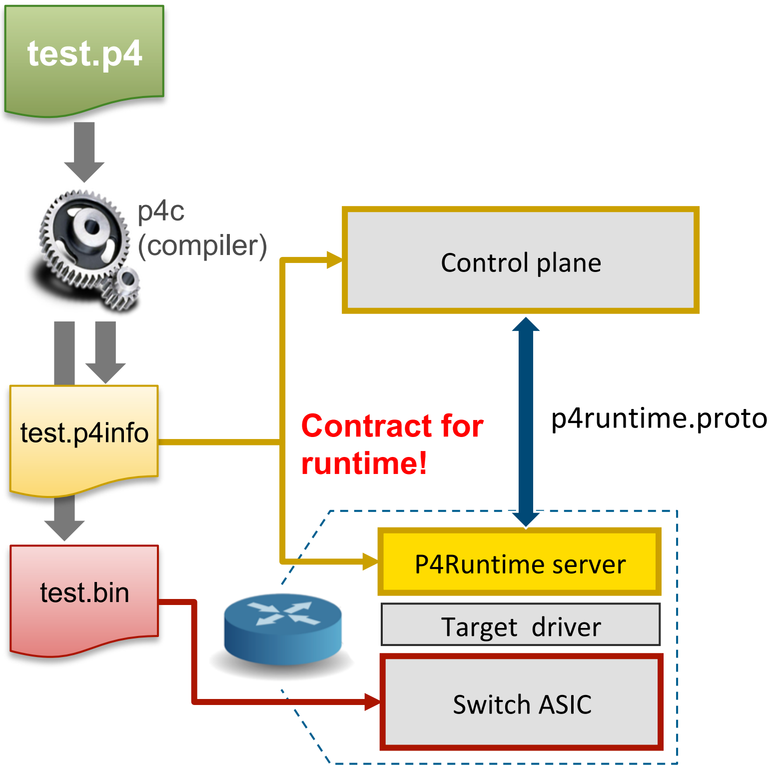


Slide courtesy P4.org

P4 compiler workflow

P4 compiler generates 2 outputs:

1. **Target-specific binaries**
   - Used to realize switch pipeline
     (e.g. binary config for ASIC, bitstream for FPGA, etc.)
2. **P4Info file**
   - Describes “schema” of pipeline for runtime control
     - Describe tables, actions, parameters, etc.
   - Protobuf-based format
   - Target-independent compiler output
     - Same P4Info for SW switch, ASIC, etc.



Full P4Info protobuf specification:
https://github.com/p4lang/p4runtime/blob/master/proto/p4/config/v1/p4info.proto

P4 and P4Runtime support in ONOS

P4 on ONOS: design goals

1. **Allow ONOS users to bring their own P4 program**
2. **Allow existing apps to control *any* P4-defined pipeline, without changing the app**
   - e.g. re-use Trellis apps
3. **Allow apps to control custom/new protocols as defined in the P4 program**
   - e.g. P4-offloaded S/PGW or BNG control plane

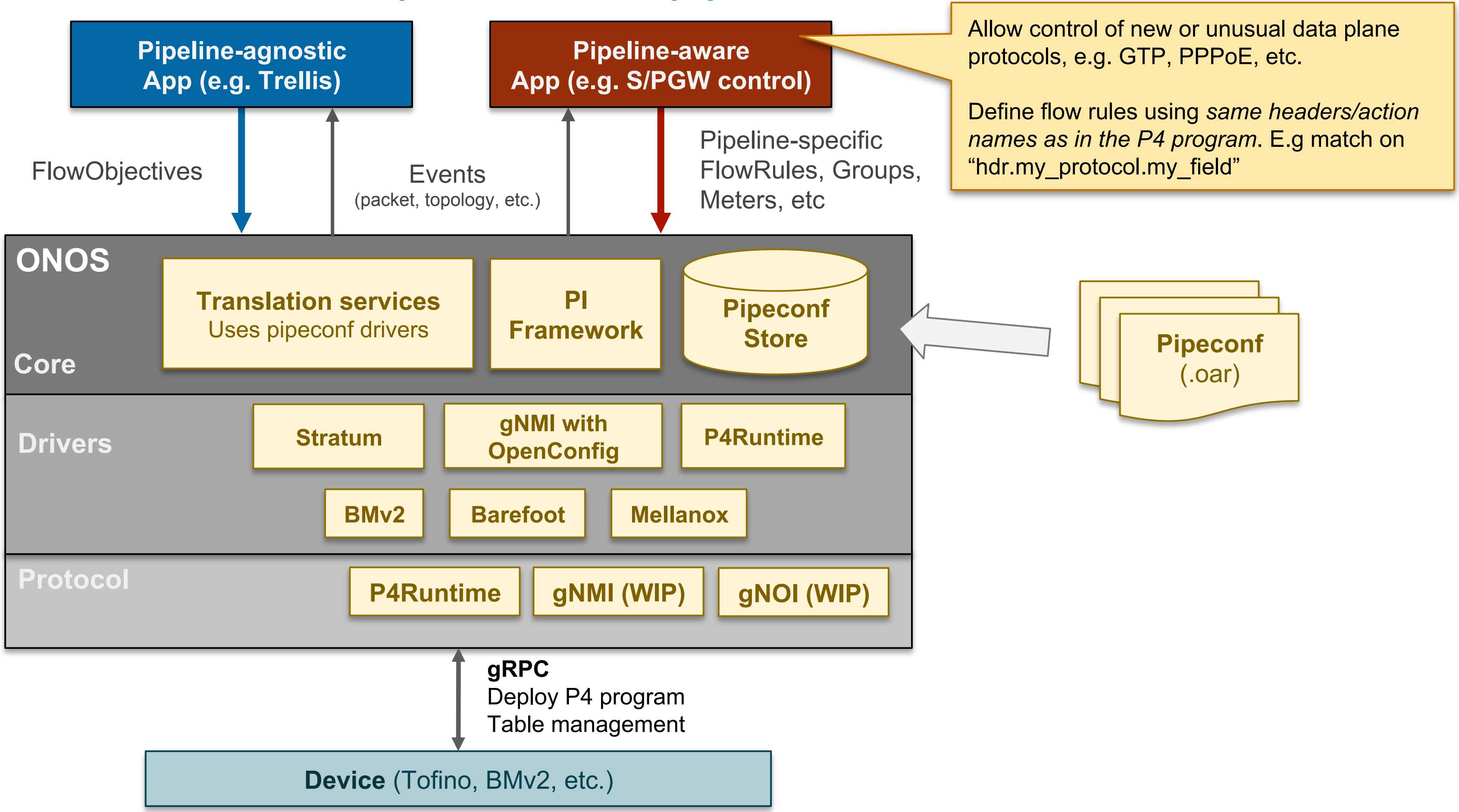
“Pipeconf” - Bring your own pipeline!

- **Package together everything necessary to let ONOS understand, control, and deploy an arbitrary pipeline**
- **Provided to ONOS as an app**
  - Can use .oar format for distribution

pipeconf.oar

1. **Pipeline model**
   - Description of the pipeline understood by ONOS
   - Automatically derived from P4Info
2. **Target-specific binaries to deploy pipeline to device**
   - E.g. BMv2 JSON, Tofino binary, FPGA bitstream, etc.
3. **Pipeline-specific driver behaviors**
   - E.g. “Pipeliner” implementation: logic to map FlowObjectives to P4 pipeline

Pipeconf support in ONOS



PI framework (@beta)

- PI = (data plane) protocol-independent
- Model: abstraction derived from P4Info
- Runtime: abstraction derived from P4Runtime
- Service: to operate on PI-capable devices

onos/core/api/.../pi/model

DefaultPiPipeconf.java
PiActionId.java
PiActionModel.java
PiActionParamId.java
PiActionParamModel.java
PiActionProfileId.java
PiActionProfileModel.java
PiControlMetadataId.java
PiControlMetadataModel.java
PiCounterId.java
PiCounterModel.java
PiCounterType.java
PiData.java
PiMatchFieldId.java
PiMatchFieldModel.java
PiMatchType.java
PiMeterId.java
PiMeterModel.java
PiMeterType.java
...

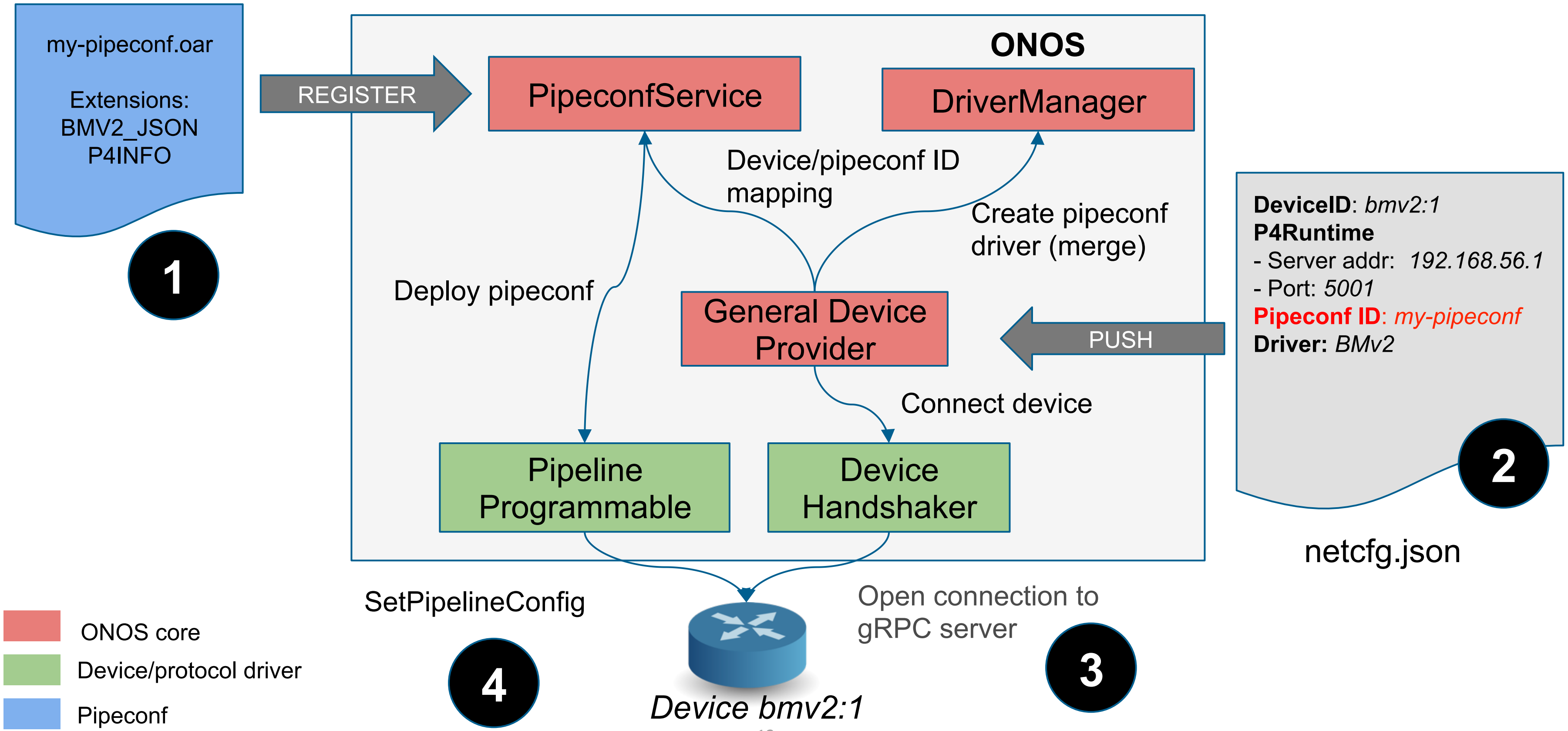
onos/core/api/.../pi/runtime

PiAction.java
PiActionGroup.java
PiActionGroupHandle.java
PiActionGroupId.java
PiActionGroupMember.java
PiActionGroupMemberHandle.java
PiActionGroupMemberId.java
PiActionParam.java
PiControlMetadata.java
PiCounterCell.java
PiCounterCellData.java
PiCounterCellId.java
PiEntity.java
PiEntityType.java
PiExactFieldMatch.java
PiFieldMatch.java
PiGroupKey.java
PiHandle.java
PiLpmFieldMatch.java
...

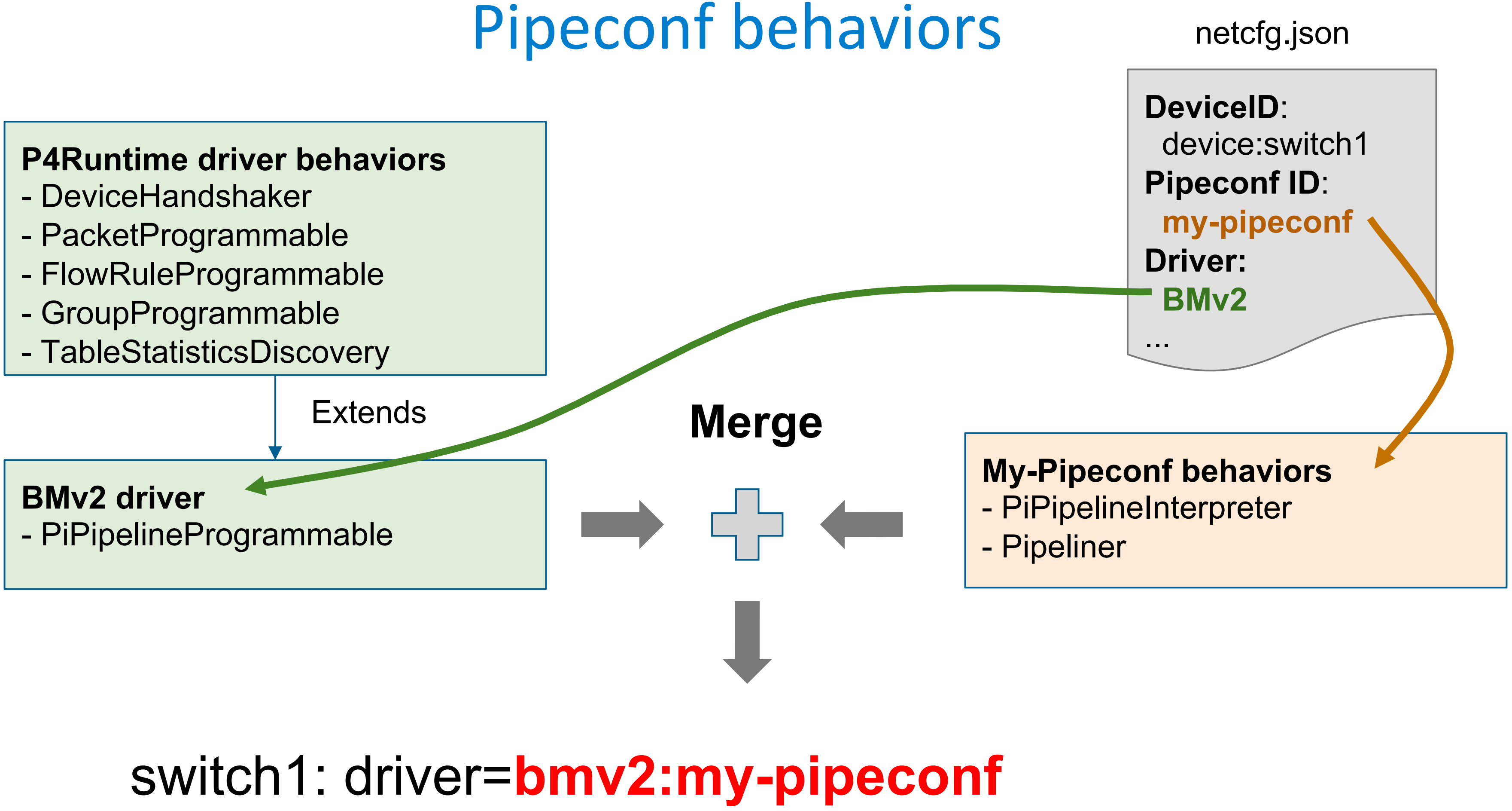
onos/core/api/.../pi/service

PiFlowRuleTranslationStore.java
PiFlowRuleTranslator.java
PiGroupTranslationStore.java
PiGroupTranslator.java
PiMeterTranslationStore.java
PiMeterTranslator.java
PiMulticastGroupTranslationStore.java
PiMulticastGroupTranslator.java
PiPipeconfConfig.java
PiPipeconfDeviceMappingEvent.java
PiPipeconfMappingStore.java
PiPipeconfMappingStoreDelegate.java
PiPipeconfService.java
PiPipeconfWatchdogEvent.java
PiPipeconfWatchdogListener.java
PiPipeconfWatchdogService.java
PiTranslatable.java
PiTranslatedEntity.java
PiTranslationEvent.java
...

Device discovery and pipeconf deploy

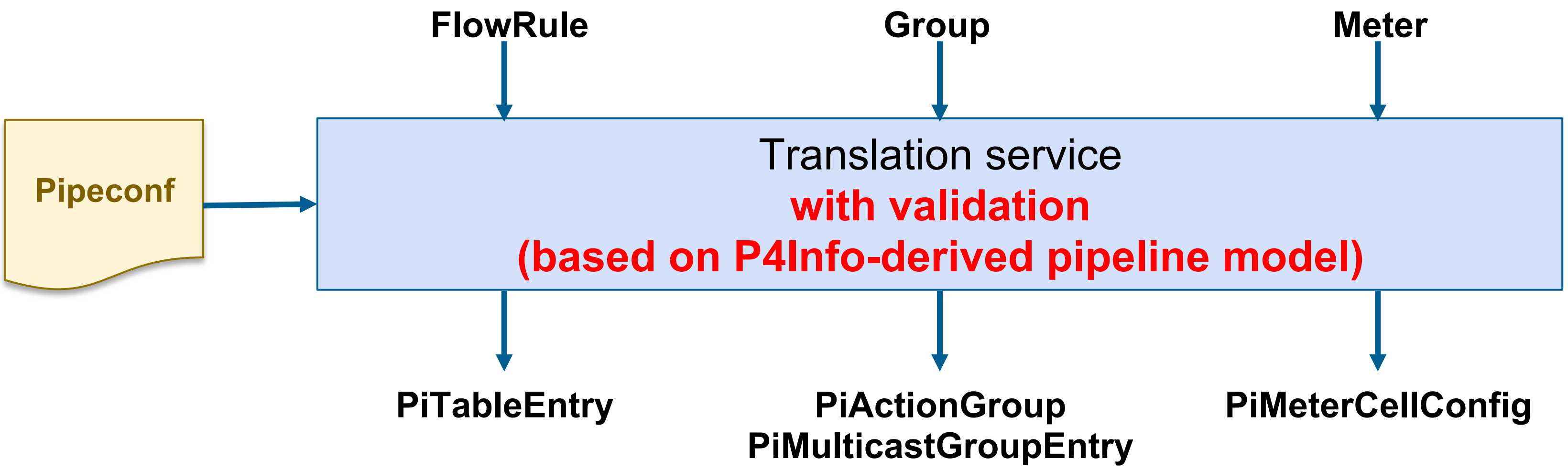


Pipeconf behaviors



PiTranslationService

- Core service, independent of P4/P4Runtime
  - Uses PI framework model and runtime classes
- Translate pipeline-specific entities from protocol-dependent representations to PI ones
  - E.g. OpenFlow-like headers/criteria and actions to P4-specific ones



Flow operations

Pipeconf-based 3 phase translation:

1. Flow Objective → Flow Rule

- Maps 1 flow objective to many flow rules

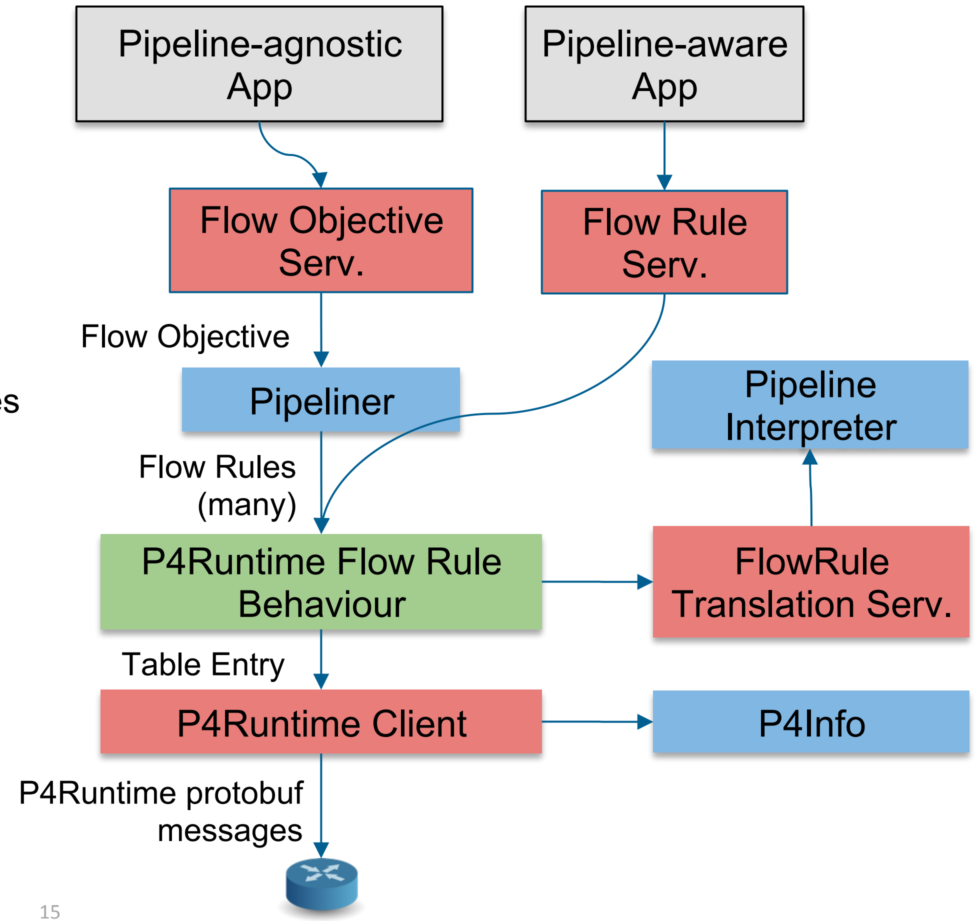
2. Flow Rule → Table entry

- Maps standard headers/actions to P4-defined ones
  E.g. ETH_DST→"hdr.ethernet.dst_addr"

3. Table Entry → P4Runtime message

- Maps P4 names to P4Info numeric IDs

ONOS Core
Device/protocol driver
Pipeconf



Pipeline interpreter (driver behavior)

- Necessary to provide a mapping from OpenFlow-derived ONOS headers/ actions to P4 program-specific entities
- Example: flow rule mapping
  - Match
    - 1:1 mapping between ONOS known headers and P4 header names
    - E.g. ETH_DST → "ethernet.dst_addr" (name defined in P4 program)
  - Action
    - ONOS defines standard actions as in OpenFlow (output, set field, etc)
    - Problem: P4 allows only one action per table entry, ONOS many (as in OpenFlow)
    - E.g. header rewrite + output: 2 actions in ONOS, 1 action with 2 parameters in P4
    - How to map many actions to one? Need interpretation logic (i.e. Java code)!

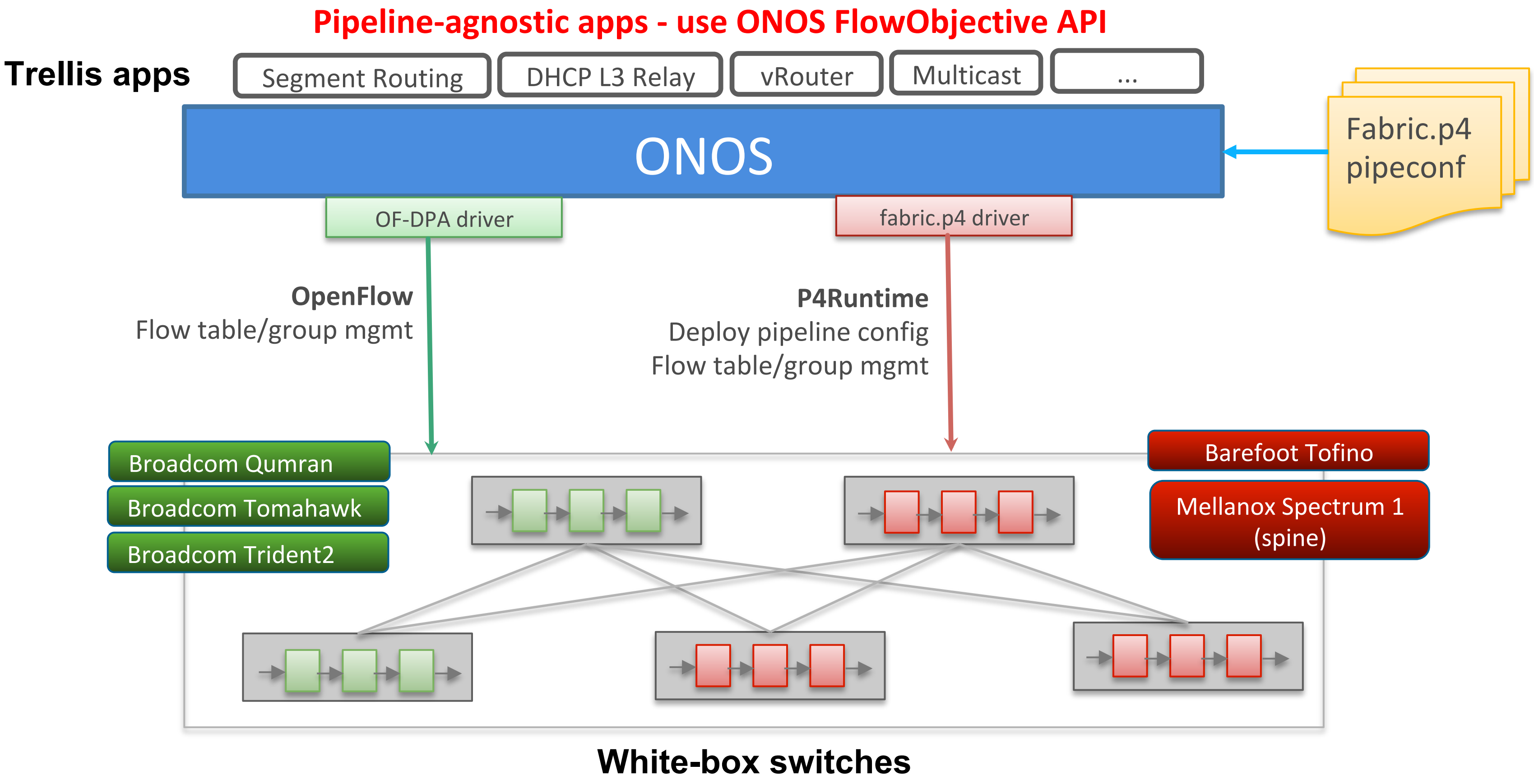
P4Runtime support in ONOS 1.14

P4Runtime control entity	**ONOS API**
Table entry	Flow Rule Service, Flow Objective Service Intent Service
Packet-in/out	Packet Service
Action profile group/members, PRE multicast groups	Group Service
Meter	Meter Service (indirect meters only)
Counters	Flow Rule Service (direct counters) P4Runtime Client (indirect counters)
Pipeline Config	Pipeconf

Unsupported features - community help needed!
Parser value sets, registers, digests, clone sessions

Use case 1: Trellis

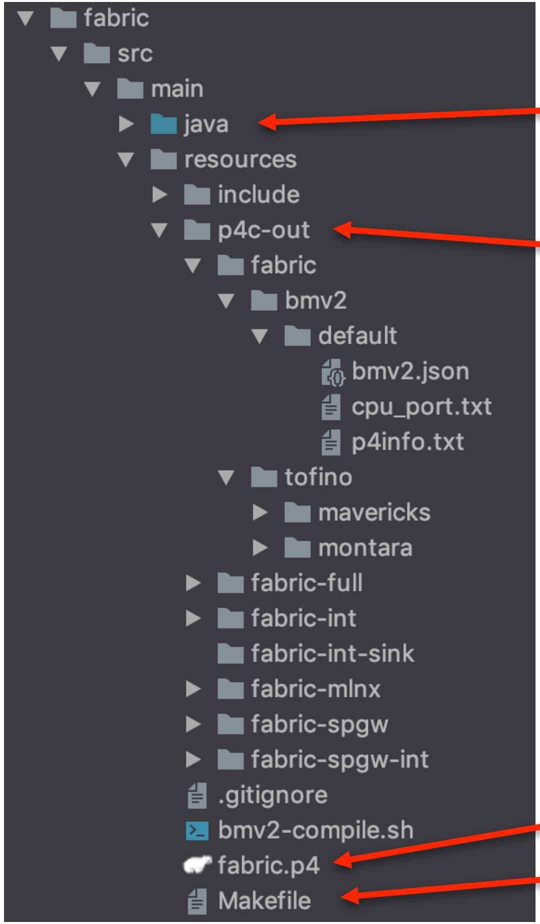
Trellis & P4


Pipeline-agnostic apps - use ONOS FlowObjective API
Trellis apps
Segment Routing
DHCP L3 Relay
vRouter
Multicast
...
ONOS
Fabric.p4 pipeconf
OF-DPA driver
fabric.p4 driver
OpenFlow
Flow table/group mgmt
P4Runtime
Deploy pipeline config
Flow table/group mgmt
Broadcom Qumran
Broadcom Tomahawk
Broadcom Trident2
Barefoot Tofino
Mellanox Spectrum 1 (spine)
White-box switches

Fabric.p4

- **P4 implementation of the Trellis underlay reference pipeline**
  - Inspired by Broadcom OF-DPA pipeline
  - Tailored to Trellis needs (fewer tables, easier to control)
  - Work in progress (missing support for IPv6)
- **Works with both programmable and fixed-function chips**
  - Logical simplified pipeline of standard L2/L3/MPLS features
  - Any switch pipeline that can be mapped to fabric.p4 can be used with Trellis
- **Extensible open-source implementation**
  - github.com/opennetworkinglab/onos/.../fabric.p4

Fabric pipeconf



onos/pipeline/fabric

Pipeliner and interpreter behaviors impl

P4 compiler outputs, organized per profile, target, platform

- Only BMv2 outputs are shipped with ONOS
- In the future, generate BMv2 JSON and P4Info during ONOS build

Example registered pipeconf IDs:

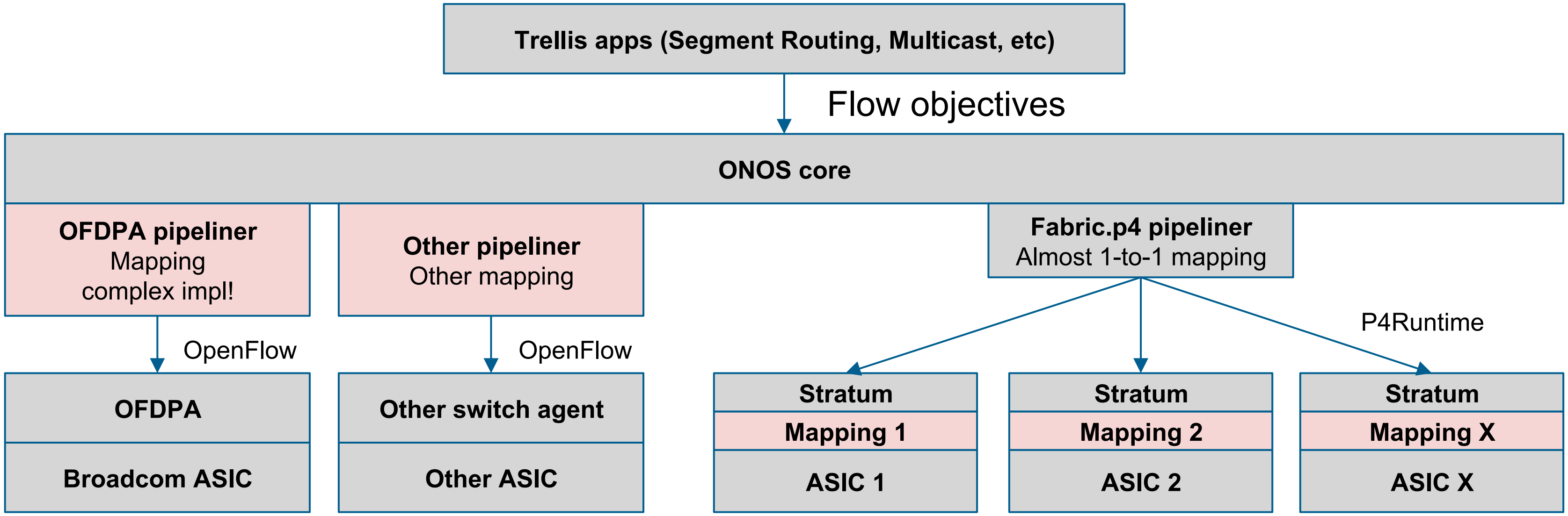
org.onosproject.pipelines.fabric
org.onosproject.pipelines.fabric-spgw
org.onosproject.pipelines.fabric.mavericks (Tofino x65 ports)
org.onosproject.pipelines.fabric-spgw.montara (Tofino x32 ports)
org.onosproject.pipelines.fabric-mlnx (Mellanox Spectrum 1)
etc.

Top level P4 file

Makefile with profile flags (e.g. make fabric-spgw)

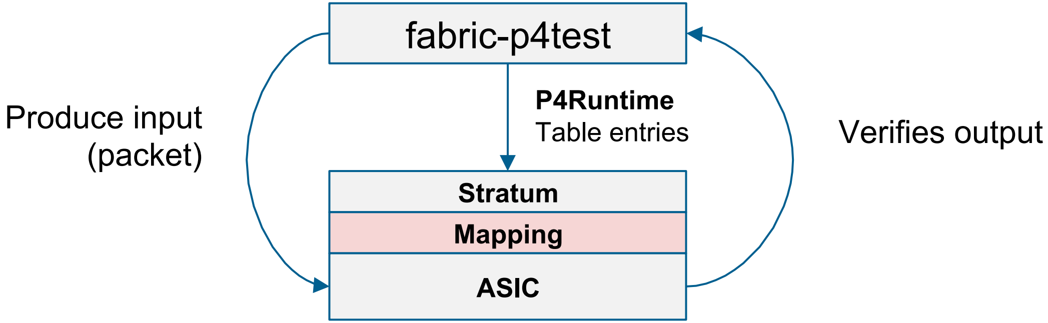
"Easier" silicon independence

- **Mapping FlowObjective is hard**
  - Underspecified/ambiguous pipeline abstraction
- **Any switch ASIC that can be mapped to fabric.p4 can be used with Trellis**
  - Both programmable and fixed function
- **Mapping effort is left to P4 compilers or ASIC vendors (manual), not ONOS drivers**
  - Fabric.p4 pipeliner (driver) unchanged

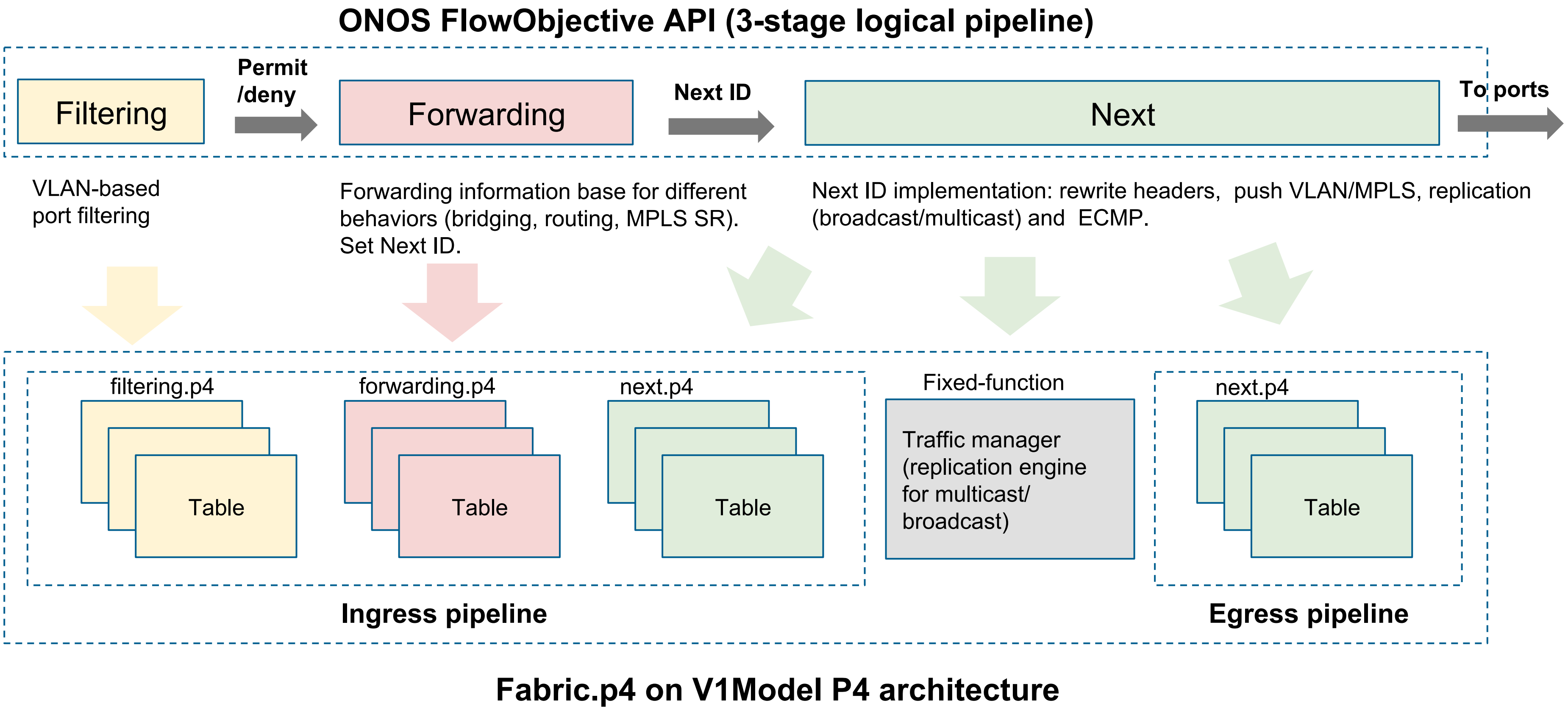


Fabric-p4test - Data plane unit testing

- https://github.com/opennetworkinglab/fabric-p4test
- Test cases for different forwarding behaviors
  - VLAN port trunking, bridging, routing, multicast, ECMP, MPLS SR, etc.
- Based on Packet Test Framework (PTF)
  - Similar to OFTest, without OpenFlow
- Test fabric.p4 implementation with BMv2 (reference software switch)
- Test ASIC mapping
  - Barefoot Tofino, Mellanox Spectrum (WIP)



Fabric.p4 design rationale



OF-DPA vs fabric.p4 Pipeliner

fabric.p4

```
$ cd onos/pipelines/fabric/.../pipeliner
$ wc -l *.java
  106 AbstractObjectiveTranslator.java
  284 FabricPipeliner.java
   58 FabricPipelinerException.java
  237 FilteringObjectiveTranslator.java
  252 ForwardingFunctionType.java
   43 ForwardingFunctionTypeCommons.java
  284 ForwardingObjectiveTranslator.java
  498 NextObjectiveTranslator.java
  209 ObjectiveTranslation.java
   20 package-info.java
 1991 total
```

OF-DPA

```
$ cd onos/drivers/.../pipeline/ofdpa/
$ wc -l Ofdpa*.java
  1985 Ofdpa2GroupHandler.java
  1933 Ofdpa2Pipeline.java
   514 Ofdpa3GroupHandler.java
   913 Ofdpa3Pipeline.java
    49 Ofdpa3QmxPipeline.java
   772 OfdpaGroupHandlerUtility.java
  6166 total
```

x3 more LOCs

Use case-based ASIC resource tuning

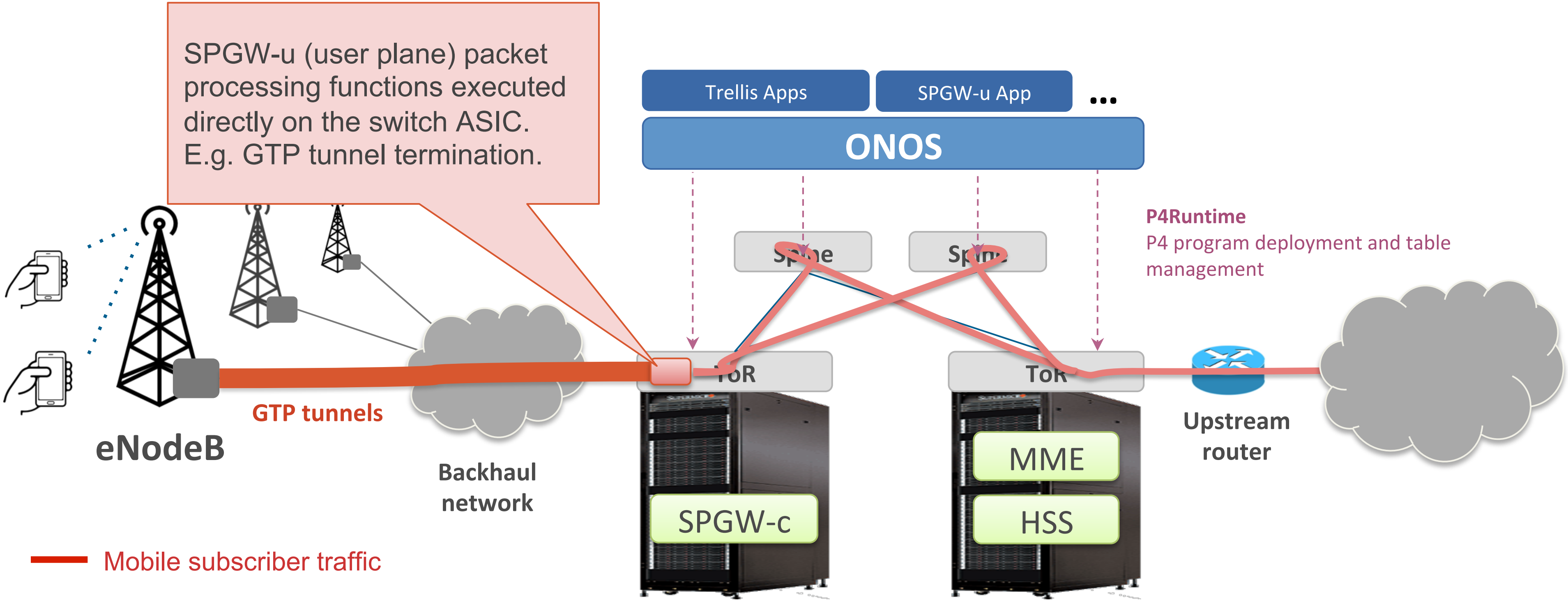
- **NFV fabric for access:**
  - Small bridging table, bigger routing table (e.g. 100x more table entries)
- **SEBA: 2 modes of operation**
  - Double-VLAN cross-connect (between OLT and BNG)
    - No routing, most memory goes to VLAN table
  - Double-VLAN termination (fabric is BNG, pop VLAN and route)
    - Same size VLAN and routing table (20k in realistic deployment)

```
table routing_v4 {
  key = {
     hdr.ipv4.dst_addr: lpm;
  }
  actions = {
     set_next_id_routing_v4;
     nop_routing_v4;
  }
  counters = routing_v4_counter;
  size = 1500000;
}
```

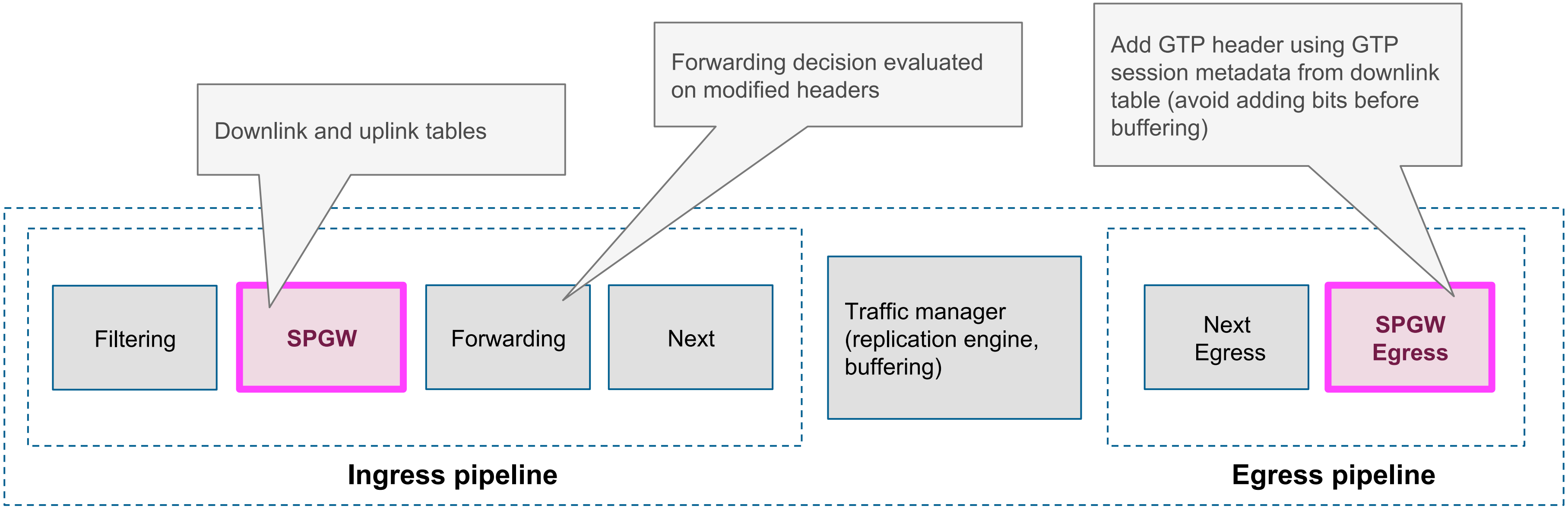
```
table routing_v4 {
  key = {
     hdr.ipv4.dst_addr: lpm;
  }
  actions = {
     set_next_id_routing_v4;
     nop_routing_v4;
  }
  counters = routing_v4_counter;
  size = 20000;
}
```

Use case 2: VNF offloading

M-CORD with offloaded SPGW-u VNF

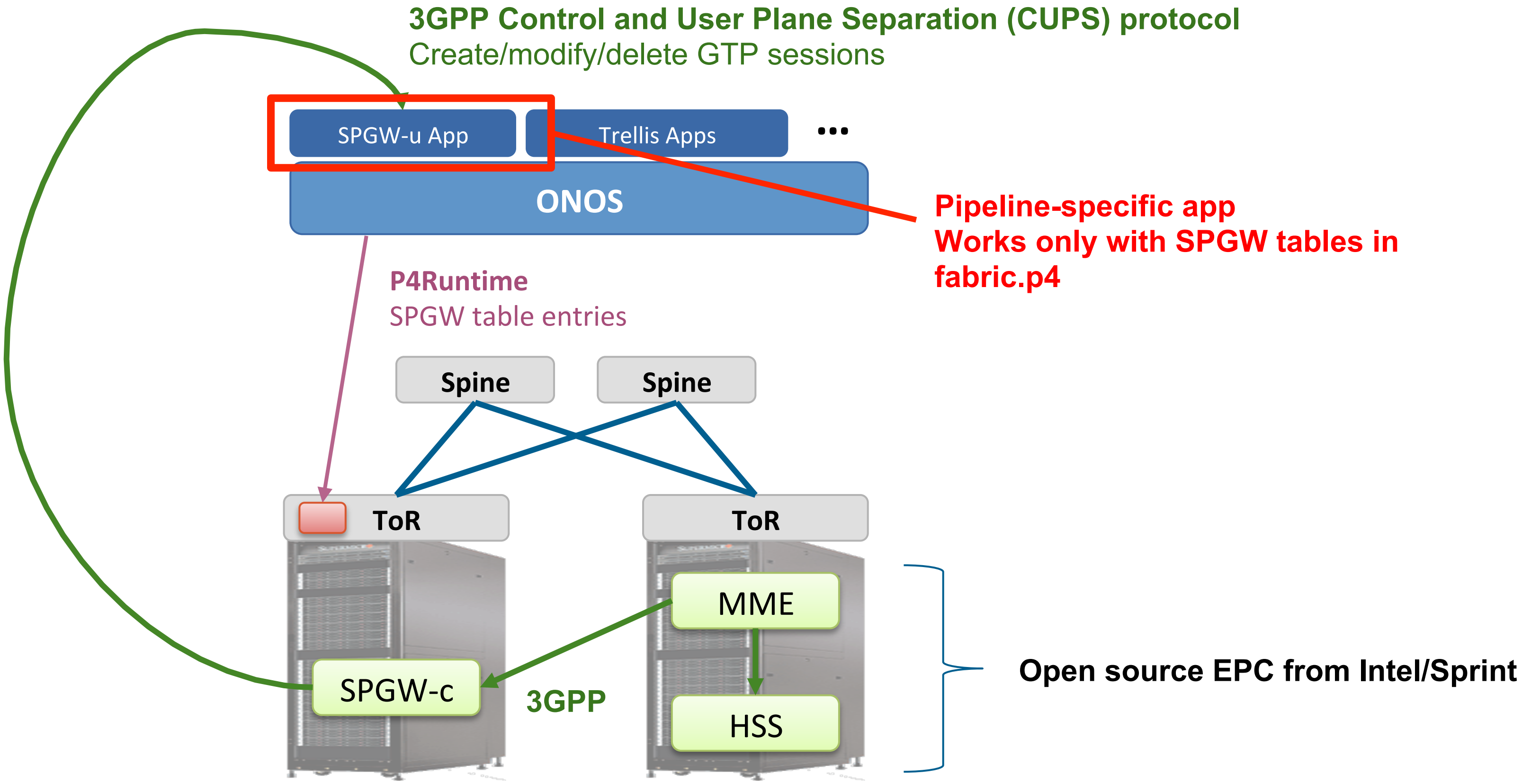


SPGW-u integration with fabric.p4



Fabric.p4 on V1Model P4 architecture

SPGW-u ONOS app



ONOS+P4 workflow recap

- **Write P4 program and compile it**
  - Obtain P4Info and target-specific binaries to deploy on device
- **Create pipeconf**
  - Implement pipeline-specific driver behaviours (Java):
    - Pipeliner (optional - if you need FlowObjective mapping)
    - Pipeline Interpreter (to map ONOS known headers/actions to P4 program ones)
    - Other driver behaviors that depend on pipeline
- **Use existing pipeline-agnostic apps (e.g. Trellis)**
  - Apps that program the network using FlowObjectives
- **Write new pipeline-aware apps (e.g. S/PGW)**
  - Apps can use same string names of tables, headers, and actions as in the P4 program

Thanks!