Tutorial:

P4 and P4Runtime Technical Introduction and Use Cases for Service Providers

Carmelo Cascone
Open Networking Foundation

Open Networking Summit 2018, September 27 2018

Outline

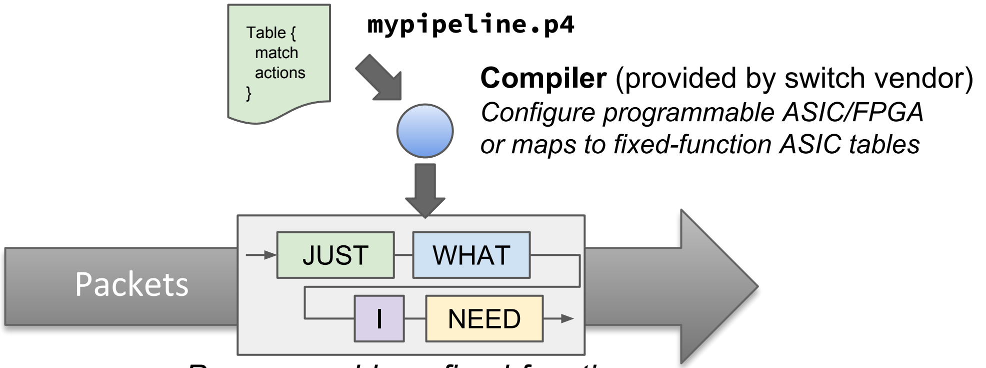
- **P4**
- **P4Runtime**
- **ONOS support for P4/P4Runtime**
- **Use cases**
  - Vendor/silicon-independent fabric
  - VNF offloading

P4

The Data Plane Programming Language

P4 - The Data Plane Programming Language

- **Domain-specific language to formally define the data plane pipeline behavior**
  - Describe protocol headers, 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



Programmable or fixed-function data plane pipeline

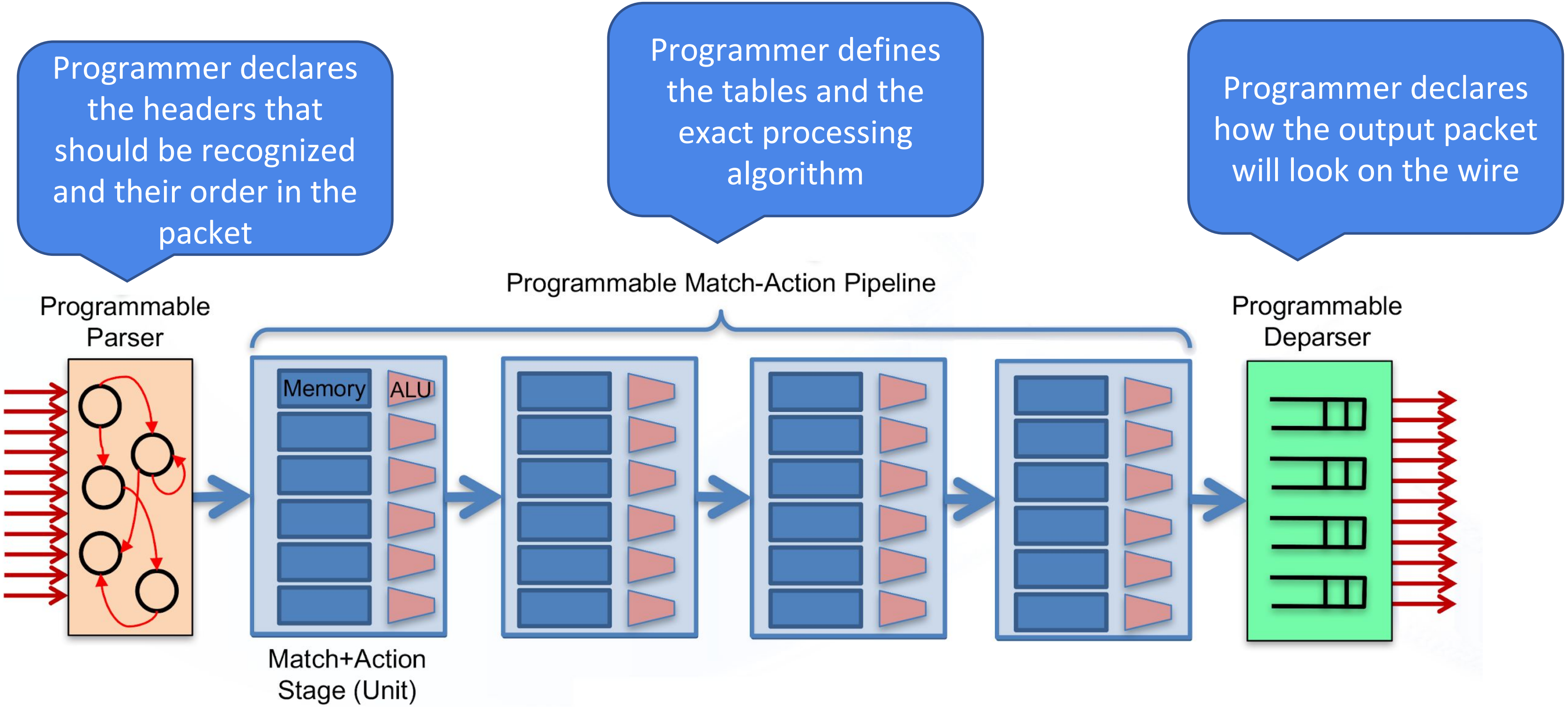
Evolution of the language

- **$P4_{14}$**
  - Original version of the language
  - Assumed specific device capabilities
  - Good only for a subset of programmable switch/targets

- **$P4_{16}$**
  - More mature and stable language definition
  - Does not assume device capabilities, which instead are defined by target manufacturer via external libraries/architecture definition
  - Good for many targets, e.g. switches and NICS, programmable or fixed-function
  - Focus of this tutorial

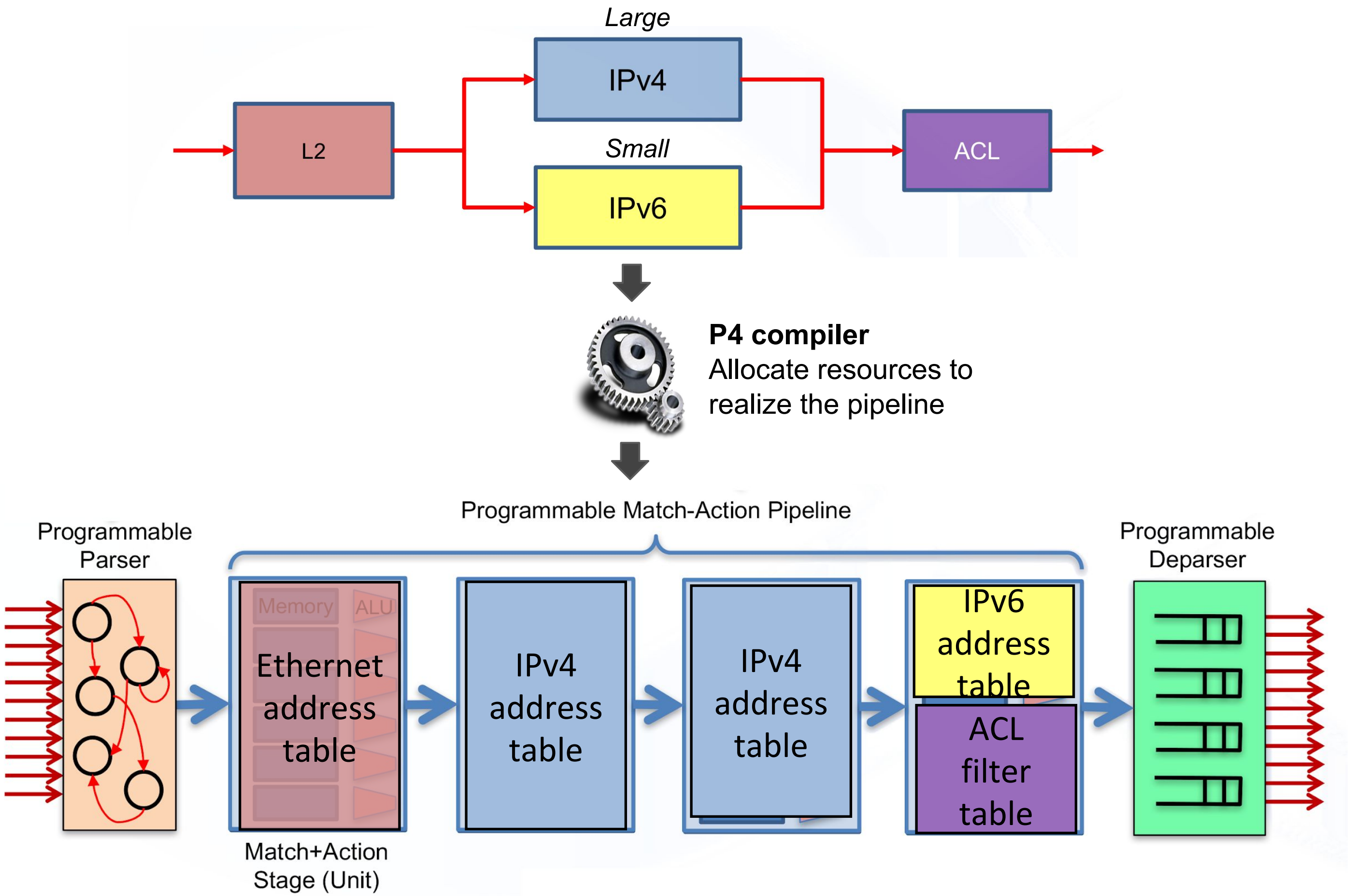
PISA: Protocol-Independent Switch Architecture

Abstract machine model of programmable switch architecture



Slide courtesy P4.org

Mapping a simple logical pipeline on PISA

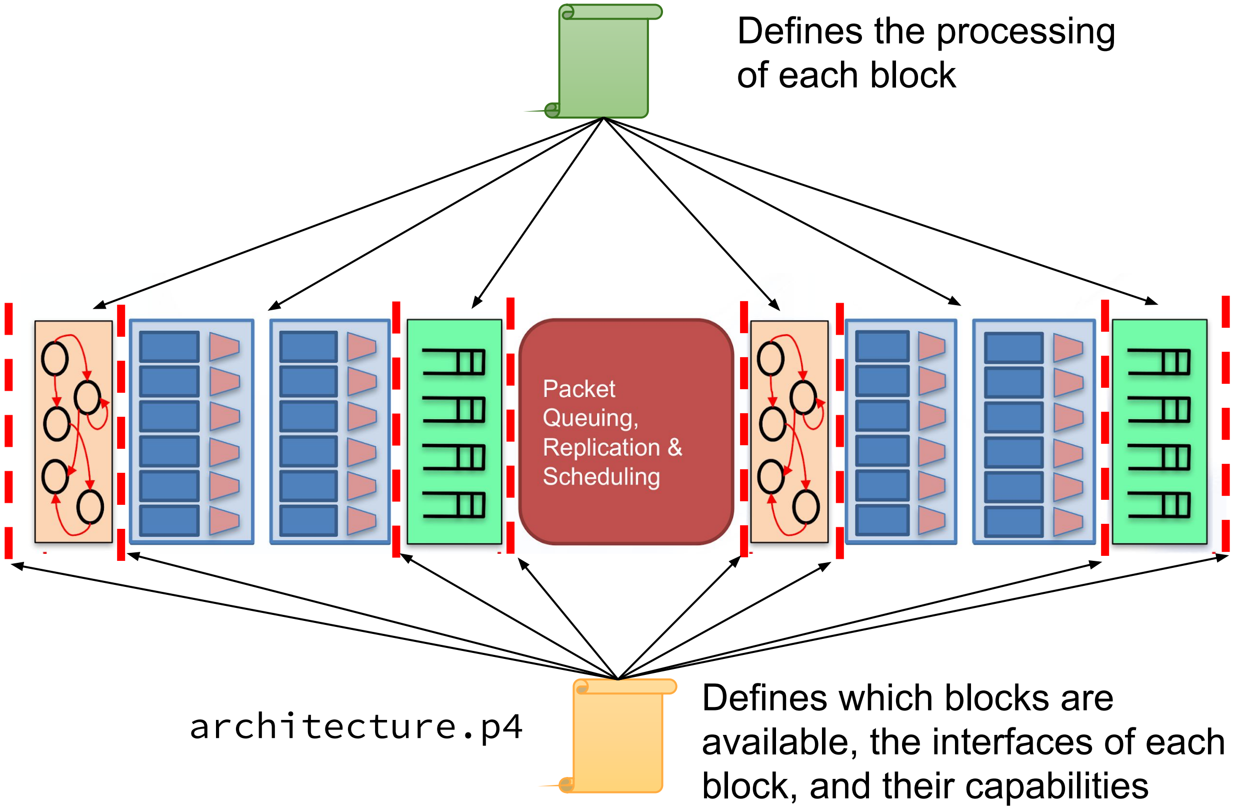


Slide courtesy P4.org

P4 programs and architectures

my_program.p4

Defines the processing of each block

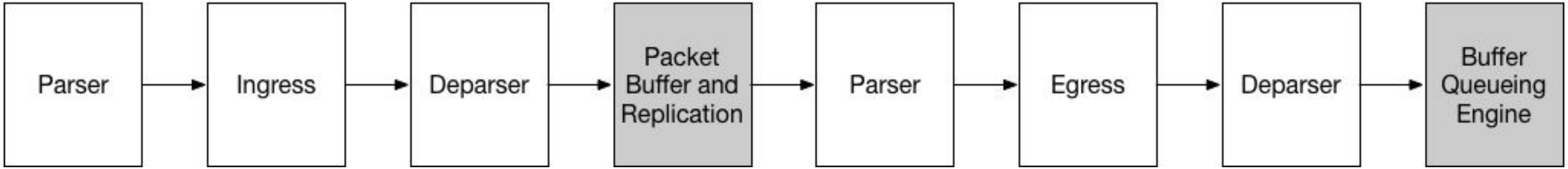


architecture.p4

Defines which blocks are available, the interfaces of each block, and their capabilities

PSA - Portable Switch Architecture

- **Community-developed architecture**
  - https://github.com/p4lang/p4-spec/tree/master/p4-16/psa
- **Describes common capabilities of a network switch**
  - Which process and forward packets across multiple interface ports
- **6 programmable P4 blocks + 2 fixed-function blocks**
- **Defines capabilities beyond match+action tables**
  - Counters, meters, stateful registers, hash functions, etc.



P4 program template (V1Model architecture)

```
#include <core.p4>
#include <v1model.p4>
/* HEADERS */
struct metadata { ... }
struct headers {
  ethernet_t    ethernet;
  ipv4_t        ipv4;
}
/* PARSER */
parser MyParser(packet_in packet,
         out headers hdr,
         inout metadata meta,
         inout standard_metadata_t smeta) {
  ...
}
/* CHECKSUM VERIFICATION */
control MyVerifyChecksum(in headers hdr,
                         inout metadata meta) {
   ...
}
/* INGRESS PROCESSING */
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
  ...
}
```

```
/* EGRESS PROCESSING */
control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t std_meta) {
  ...
}
/* CHECKSUM UPDATE */
control MyComputeChecksum(inout headers hdr,
                          inout metadata meta) {
  ...
}
/* DEPARSER */
control MyDeparser(inout headers hdr,
                   inout metadata meta) {
  ...
}
/* SWITCH */
V1Switch(
  MyParser(),
  MyVerifyChecksum(),
  MyIngress(),
  MyEgress(),
  MyComputeChecksum(),
  MyDeparser()
) main;
```

Slide courtesy P4.org

P4 program example: simple_router.p4

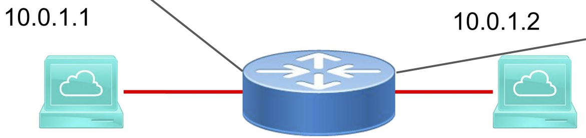

```
header ethernet_t {
  bit<48> dst_addr;
  bit<48> src_addr;
  bit<16> eth_type;
}

header ipv4_t {
  bit<4>  version;
  bit<4>  ihl;
  bit<8>  diffserv;
  …
}

parser parser_impl(packet_in pkt, out headers_t hdr) {
  /* Parser state machine to extract header fields */
}
```

```
action set_next_hop(bit<48> dst_addr) {
  ethernet.dst_addr = dst_addr;
  ipv4.ttl = ipv4.ttl - 1;
}
...
table ipv4_routing_table {
  key = {
        ipv4.dst_addr : LPM; // longest-prefix match
  }
  actions = {
        set_next_hop();
        drop();
   }
  size = 4096; // table entries
}
```

Simple router example

- **Data plane (P4) program**
  - Defines the format of the table
    - Match fields, actions, action data (parameters)
  - Performs the lookup
  - Executes the chosen action
- **Control plane**
  - Populates table entries with specific information
    - Based on configuration, automatic discovery, protocol calculations

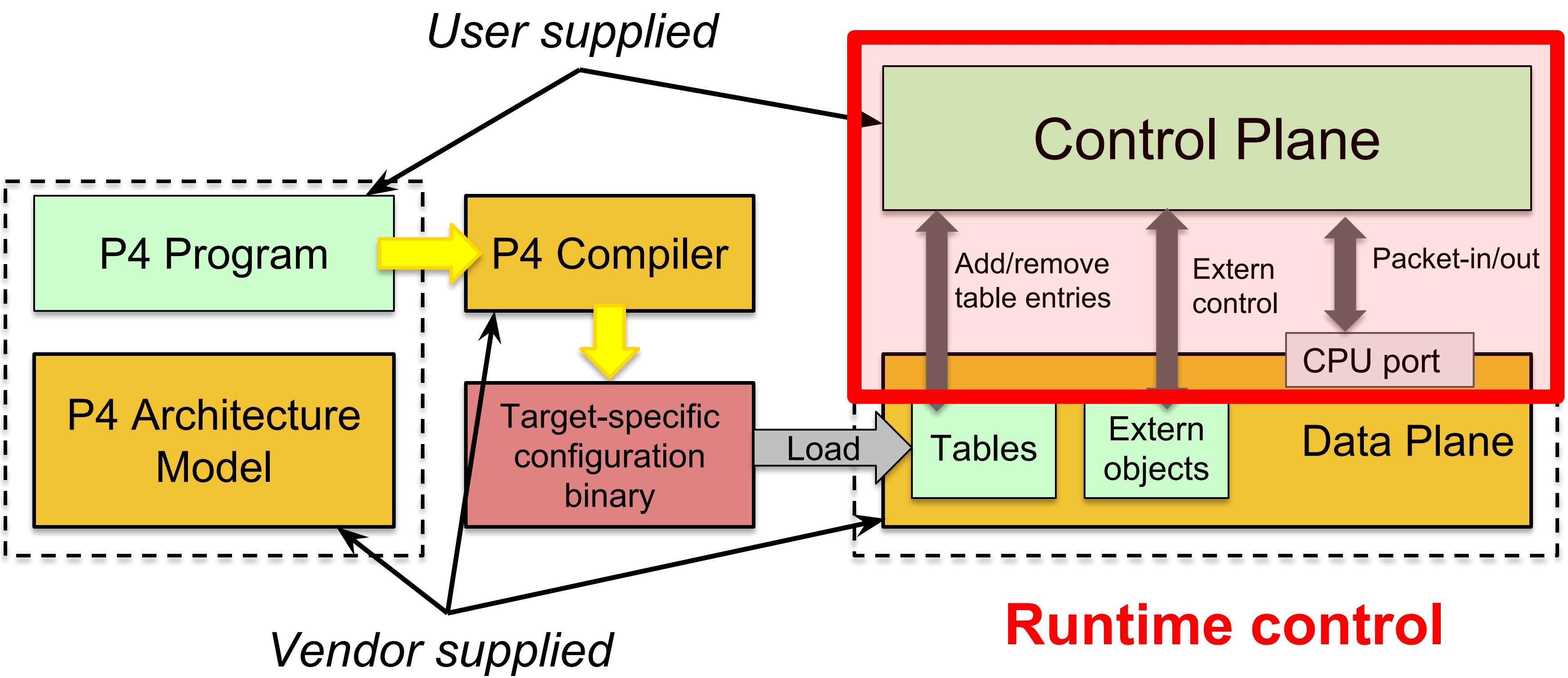
```
action ipv4_forward(bit<48> dst_addr, bit<9> port) {
  ethernet.dst_addr = dst_addr;
  standard_metadata.egress_spec = port;
  ipv4.ttl = ipv4.ttl - 1;
}
table ipv4_routing_table {
  key = {
        ipv4.dst_addr : LPM; // longest-prefix match
  }
  actions = {
        ipv4_forward();
        drop();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



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	

P4 workflow

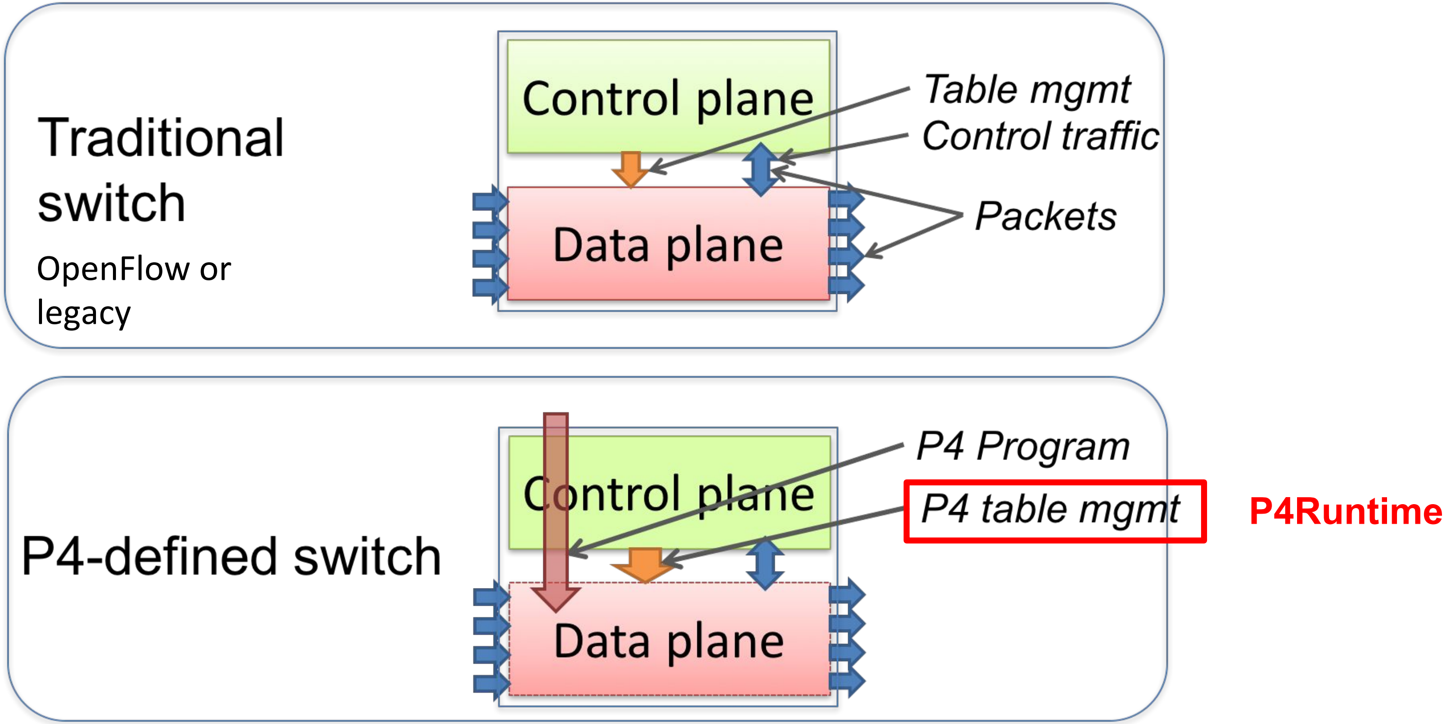


Slide courtesy P4.org

P4Runtime

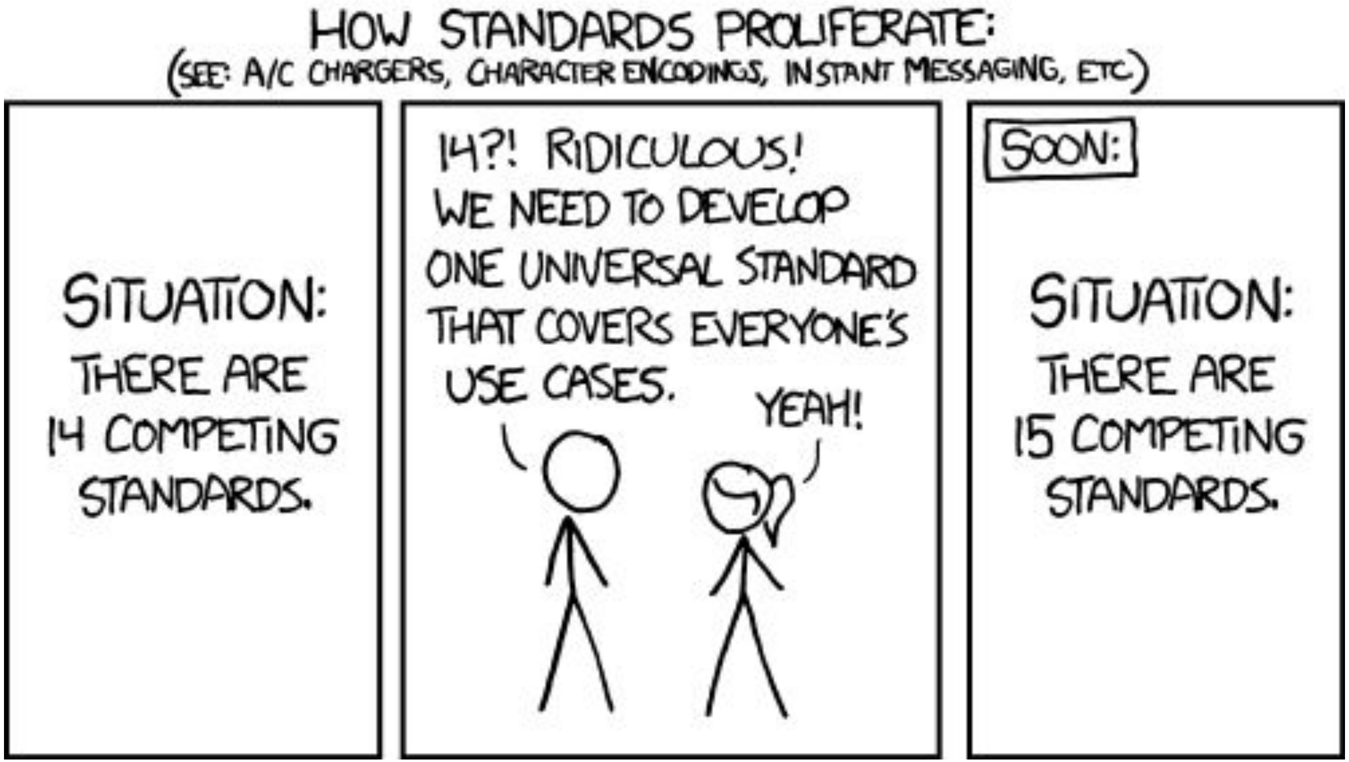
Control protocol for P4-defined data planes

Traditional/OpenFlow vs. P4 paradigm



Slide courtesy P4.org

Do we need yet another data plane control API?



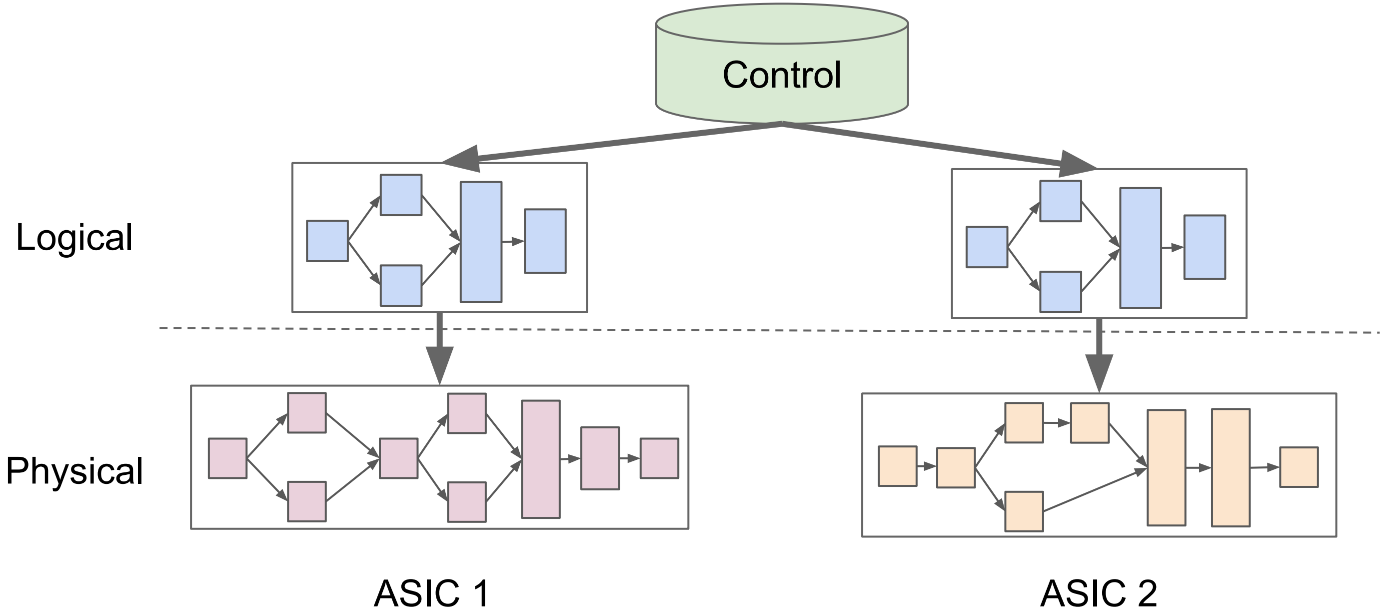
Yes, we need P4Runtime

API	**Target-independent** Same API works with different switches/vendors	**Protocol-independent** Same API allows control of new protocols	**Pipeline-independent** Same API allows control of many pipelines formally specified
OpenFlow	✔	✘ Protocol headers and actions hard-coded in the spec	✘ Pipeline specification is not mandated (TTPs did not solve the problem)
Switch Abstraction Interface (SAI)	✔	✘ Designed for legacy forwarding pipelines (L2/L3/ACL)	✘ Implicit fixed-function pipeline
P4Runtime	✔	✔	✔ (with P4)

Slide from Google

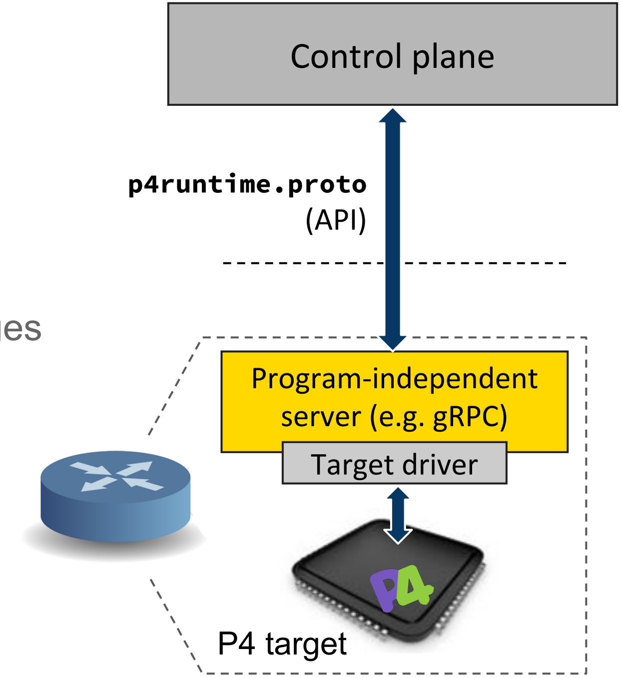
P4 Program as Fixed-Function Chip Abstraction

- P4 program tailored to apps / role - does not describe the hardware
- Switch maps program to fixed-function ASIC
- Enables portability



P4Runtime overview

- **Protocol for runtime control of P4-defined switches**
  - Designed around PSA architecture but can be extended to others
- **Work-in-progress by the p4.org API WG**
  - Initial contribution by Google and Barefoot
  - Draft of version 1.0 available: https://p4.org/p4-spec/
- **Protobuf-based API definition**
  - Automatically generate client/server code for many languages
  - gRPC transport
- **P4 program-independent**
  - API doesn't change with the P4 program
- **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

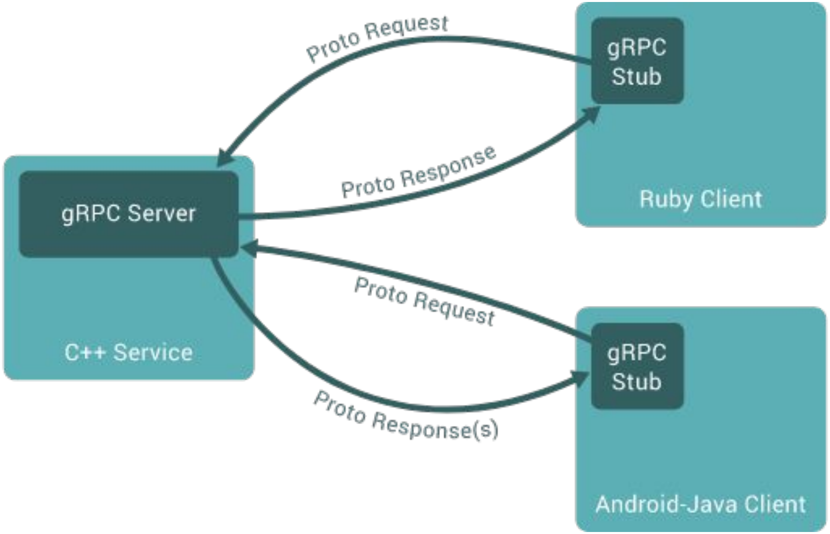
Protocol Buffers (protobuf) Basics

- Language for describing data for serialization in a structured way
- Common binary wire-format
- Language-neutral
  - Code generators for: *Action Script, C, C++, C#, Clojure, Lisp, D, Dart, Erlang, Go, Haskell, Java, Javascript, Lua, Objective C, OCaml, Perl, PHP, Python, Ruby, Rust, Scala, Swift, Visual Basic, ...*
- Platform-neutral
- Extensible and backwards compatible
- Strongly typed

```
syntax = "proto3";

message Person {
  string name = 1;
  int32 id = 2;
  string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    string number = 1;
    PhoneType type = 2;
  }

  repeated PhoneNumber phone = 4;
}
```

gRPC Basics

- Use Protobuf to define service API and messages
- Automatically generate native stubs in:
  - C / C++, C#, Dart, Go, Java, Node.js, PHP, Python, Ruby
- Transport over HTTP/2.0 and TLS
  - Efficient single TCP connection implementation that supports bidirectional streaming



p4runtime.proto (gRPC service)

Enables a local or remote entity to load the pipeline/P4 program, send/receive packets, arbitrate mastership, read and write forwarding table entries, counters, and other P4 entities.

```
service P4Runtime {
  rpc Write(WriteRequest) returns (WriteResponse) {}
  rpc Read(ReadRequest) returns (stream ReadResponse) {}
  rpc SetForwardingPipelineConfig(SetForwardingPipelineConfigRequest)
      returns (SetForwardingPipelineConfigResponse) {}
  rpc GetForwardingPipelineConfig(GetForwardingPipelineConfigRequest)
      returns (GetForwardingPipelineConfigResponse) {}
  rpc StreamChannel(stream StreamMessageRequest)
      returns (stream StreamMessageResponse) {}
}
```

From: https://github.com/p4lang/p4runtime/blob/master/proto/p4/v1/p4runtime.proto

P4Runtime Write Request

```
message WriteRequest {
 uint64 device_id = 1;
 uint64 role_id = 2;
 Uint128 election_id = 3;
 repeated Update updates = 4;
}
```

```
message Update {
 enum Type {
   UNSPECIFIED = 0;
   INSERT = 1;
   MODIFY = 2;
   DELETE = 3;
 }
 Type type = 1;
 Entity entity = 2;
}
```

```
message Entity {
 oneof entity {
   ExternEntry extern_entry = 1;
   TableEntry table_entry = 2;
   ActionProfileMember
          action_profile_member = 3;
   ActionProfileGroup
          action_profile_group = 4;
   MeterEntry meter_entry = 5;
   DirectMeterEntry direct_meter_entry = 6;
   CounterEntry counter_entry = 7;
   DirectCounterEntry direct_counter_entry = 8;
   PacketReplicationEngineEntry
          packet_replication_engine_entry = 9;
   ValueSetEntry value_set_entry = 10;
   RegisterEntry register_entry = 11;
 }
}
```

P4Runtime Table Entry

p4runtime.proto simplified excerpts:

```
message TableEntry {
  uint32 table_id;
  repeated FieldMatch match;
  Action action;
  int32 priority;
  ...
}
```

```
message Action {
  uint32 action_id;
  message Param {
    uint32 param_id;
    bytes value;
  }
  repeated Param params;
}
```

```
message FieldMatch {
  uint32 field_id;
  message Exact {
    bytes value;
  }
  message Ternary {
    bytes value;
    bytes mask;
  }
  message LPM {
    bytes value;
    uint32 prefix_length;
  }
  ...
  oneof field_match_type {
    Exact exact;
    Ternary ternary;
    LPM lpm;
    ...
  }
}
```

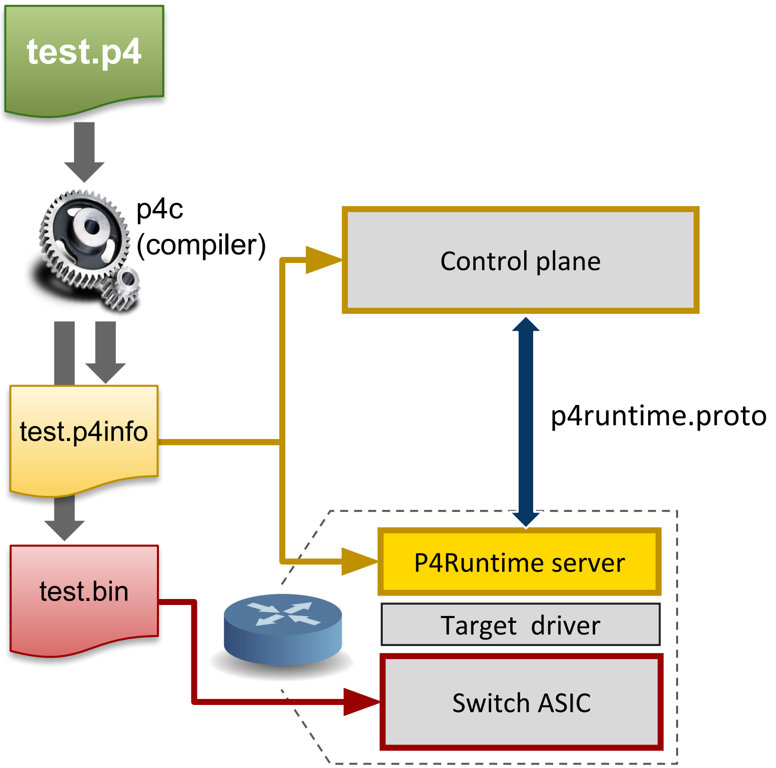
To add a table entry, the control plane needs to know:

- **IDs of P4 entities**
  - Tables, field matches, actions, params, etc.
- **Field matches for the particular table**
  - Match type, bitwidth, etc.
- **Parameters for the particular action**
- **Other P4 program attributes**

P4 compiler workflow

P4 compiler generates 2 files:

1. **Target-specific binaries**
   - Used to configure switch pipeline
     (e.g. binary config for ASIC, bitstream for FPGA, etc.)
2. **P4Info file**
   - Describes "schema" of pipeline for runtime control
   - Captures P4 program attributes
     - 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

Slide courtesy P4.org

P4Info example

basic_router.p4

```
...

action ipv4_forward(bit<48> dstAddr,
                    bit<9> port) {
   /* Action implementation */
}

...

table ipv4_lpm {
   key = {
       hdr.ipv4.dstAddr: lpm;
   }
   actions = {
       ipv4_forward;
       ...
   }
   ...
}
```

P4 compiler

basic_router.p4info

```
actions {
  id: 16786453
  name: "ipv4_forward"
  params {
    id: 1
    name: "dstAddr"
    bitwidth: 48
    ...
    id: 2
    name: "port"
    bitwidth: 9
  }
}
...
tables {
  id: 33581985
  name: "ipv4_lpm"
  match_fields {
    id: 1
    name: "hdr.ipv4.dstAddr"
    bitwidth: 32
    match_type: LPM
  }
  action_ref_id: 16786453
}
```

P4Runtime table entry example

basic_router.p4

```
action ipv4_forward(bit<48> dstAddr,
                    bit<9>  port) {
   /* Action implementation */
}
table ipv4_lpm {
   key = {
       hdr.ipv4.dstAddr: lpm;
   }
   actions = {
       ipv4_forward;
       ...
   }
   ...
}
```

Logical view of table entry

```
hdr.ipv4.dstAddr=10.0.1.1/32
      -> ipv4_forward(00:00:00:00:00:10, 7)
```

Control plane generates

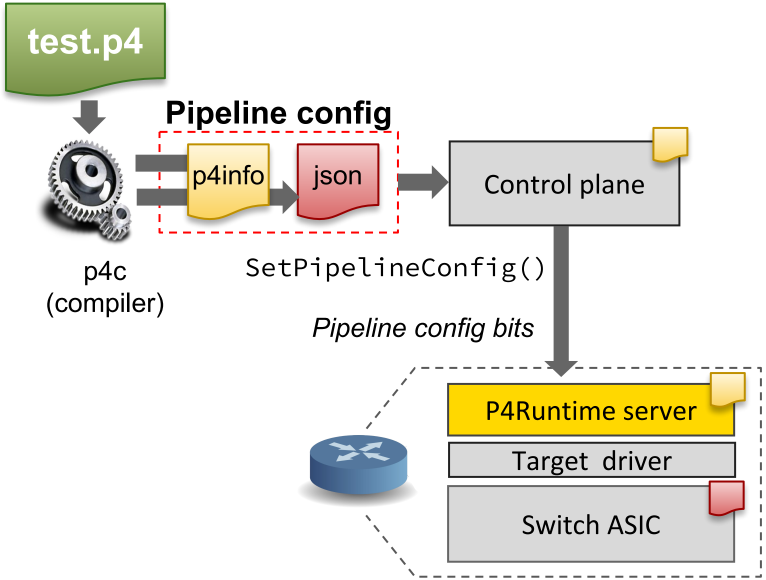
Protobuf message

```
table_entry {
  table_id: 33581985
  match {
    field_id: 1
    lpm {
      value: "\n\000\001\001"
      prefix_len: 32
    }
  }
  action {
    action_id: 16786453
    params {
      param_id: 1
      value: "\000\000\000\000\000\n"
    }
    params {
      param_id: 2
      value: "\000\007"
    }
  }
}
```

Slide courtesy P4.org

P4Runtime SetPipelineConfig

```
message SetForwardingPipelineConfigRequest {
 enum Action {
   UNSPECIFIED = 0;
   VERIFY = 1;
   VERIFY_AND_SAVE = 2;
   VERIFY_AND_COMMIT = 3;
   COMMIT = 4;
   RECONCILE_AND_COMMIT = 5;
 }
 uint64 device_id = 1;
 uint64 role_id = 2;
 Uint128 election_id = 3;
 Action action = 4;
 ForwardingPipelineConfig config = 5;
}
```




```
message ForwardingPipelineConfig {
 config.P4Info p4info = 1;
 // Target-specific P4 configuration.
 bytes p4_device_config = 2;
}
```

Project Stratum - P4Runtime switch agent implementation

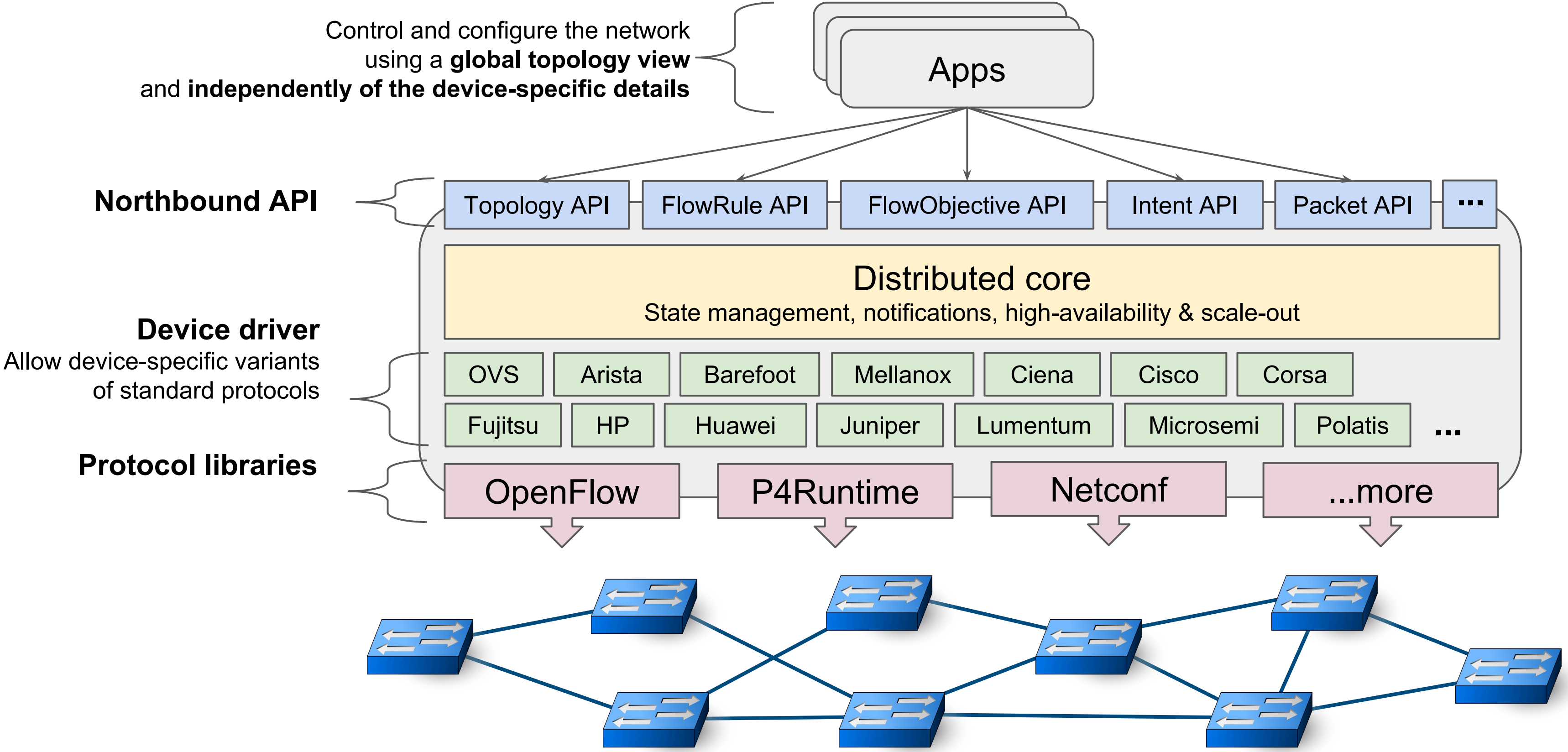
- **Open source, lightweight, production quality thin switch OS**
- **Implements next-gen SDN interfaces**
  - P4Runtime for control
    - Uses P4 as the data pipeline contract across fixed function and programmable hardware
  - gNMI using OpenConfig models for configuration/monitoring/telemetry
  - gNOI for operations
- **Rich community of service and cloud providers, chipset vendors, whitebox and blackbox switch vendors**
  - Google committed to using Stratum in production network at scale

https://stratumproject.org/

STRATUM

ONOS
A control plane for P4/P4Runtime devices

ONOS architecture recap

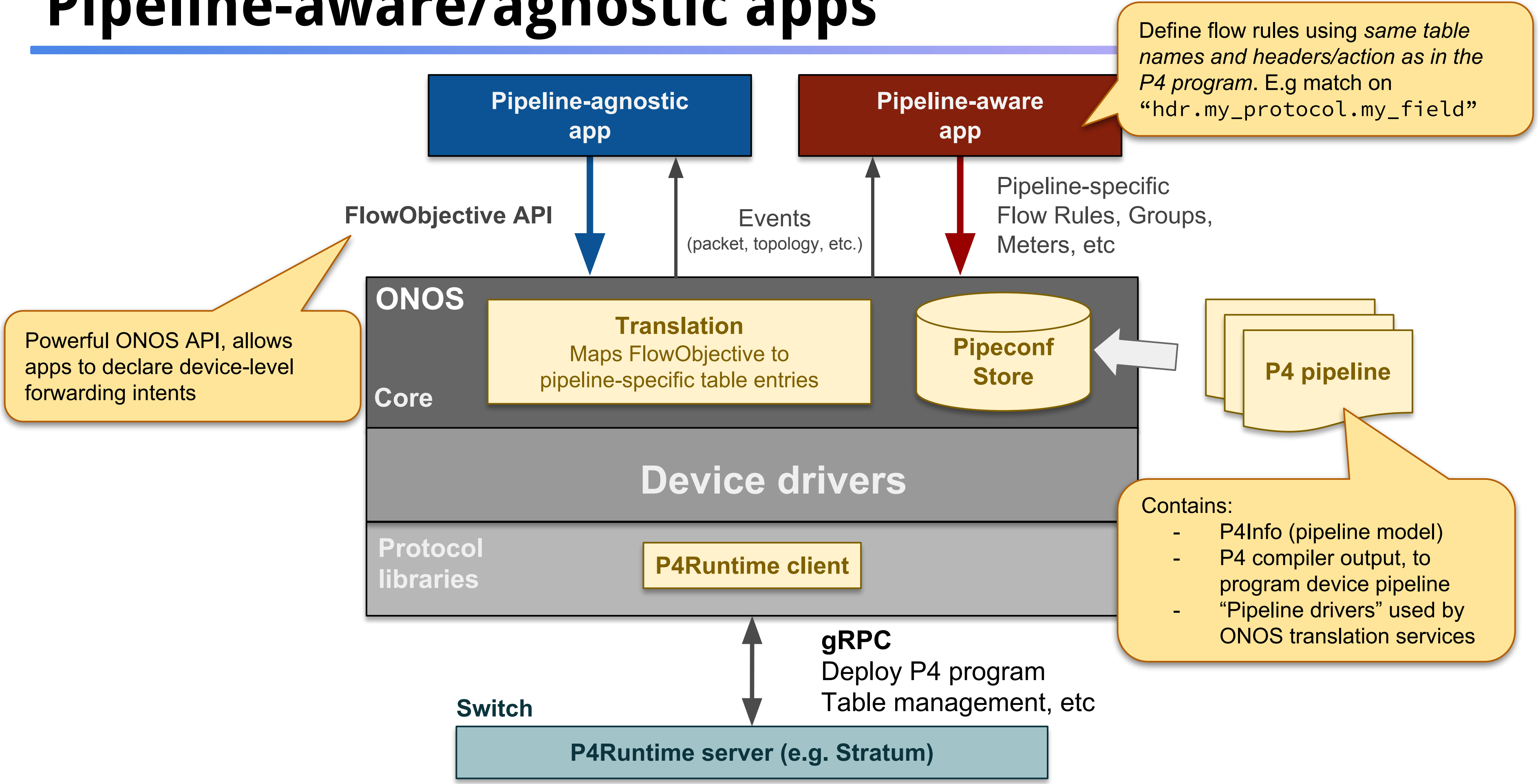


P4 and P4Runtime support in ONOS

Goals:

1. Allow ONOS users to bring their own P4 program
2. Allow apps to control custom/new protocols, as defined in the P4 program
3. Allow *existing* apps to control *any* P4 pipeline without changing the app, i.e. enable app portability accros many P4 pipelines

Pipeline-aware/agnostic apps



P4Runtime support in ONOS 1.14 (Owl)

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

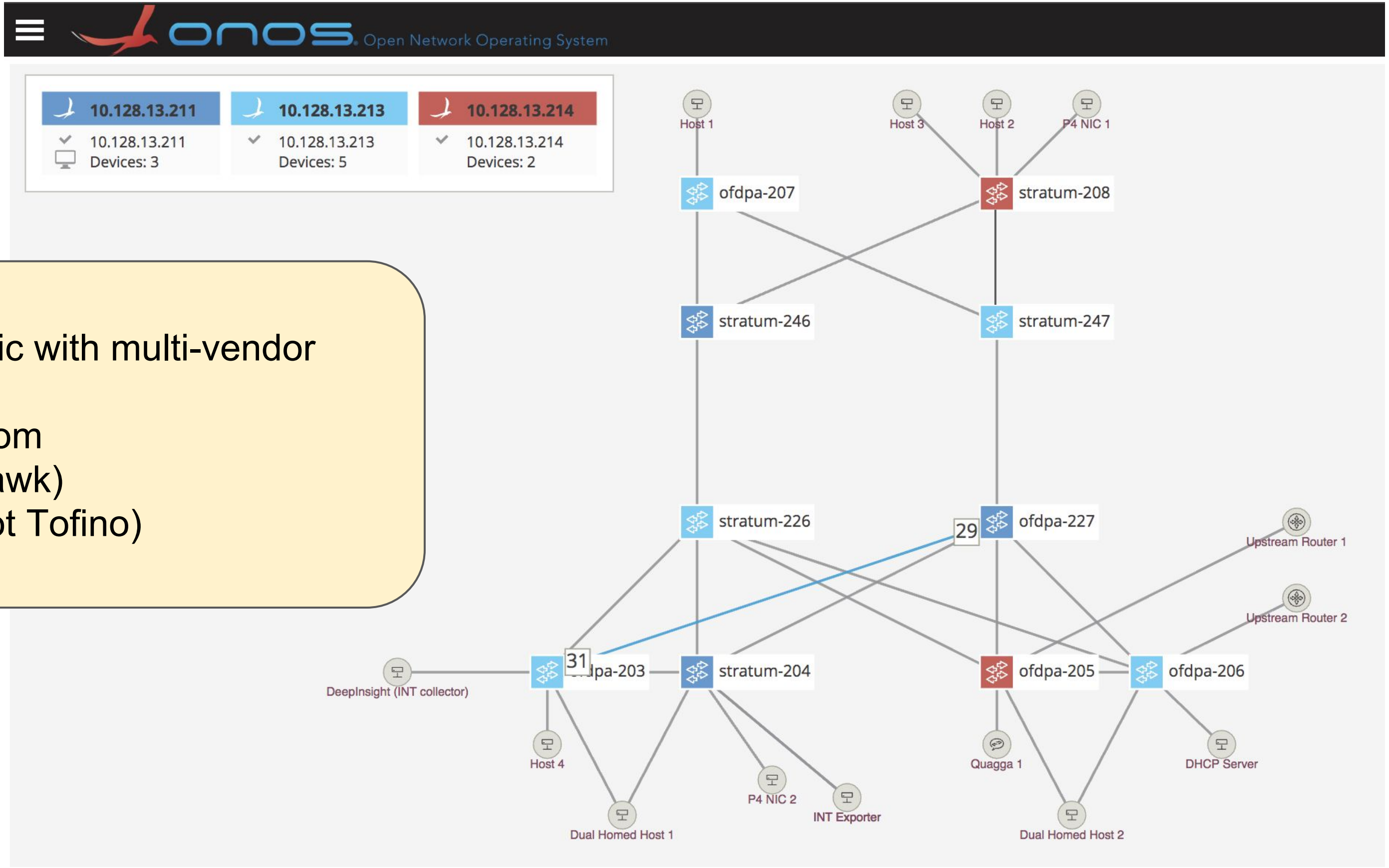
Unsupported features - community help needed!
Parser value sets, registers, digests, clone sessions

Use case 1:
silicon-independent fabric

Trellis – Multi-purpose Leaf-Spine Fabric

- **Prominent example of ONOS application**
  - In production at Comcast
- **Multi-purpose leaf-spine fabric designed for NFV and access/edge applications**
  - Built with white-box switches, open source software, SDN based
- **Extensive feature set**
  - Bridging/VLANs, IPv4/v6 unicast and multicast routing, DHCP-relay, pseudowires, QinQ, vRouter & more
- **Works with OpenFlow and P4/P4Runtime**

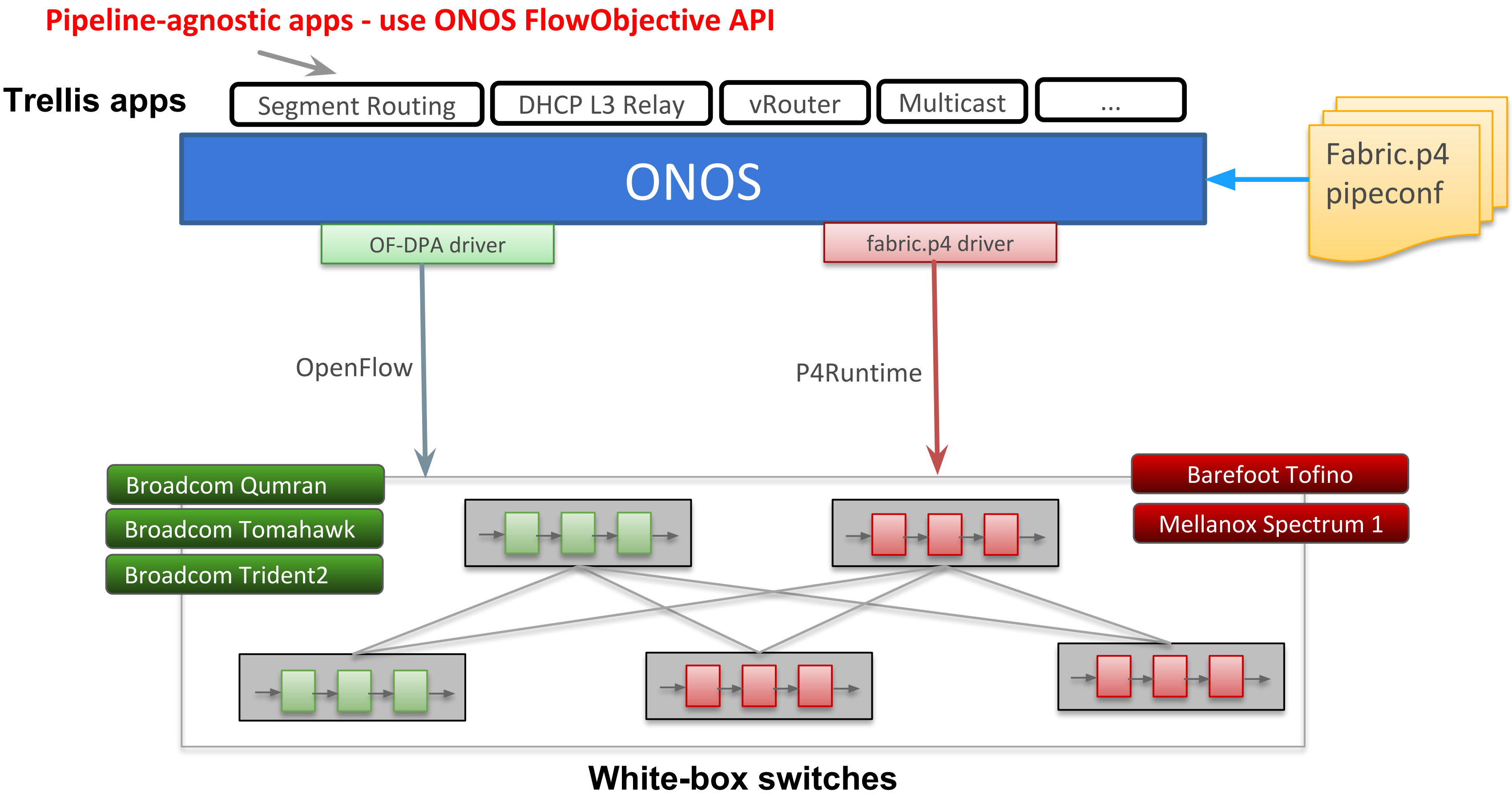
Trellis demo @ Booth 5



2-stage leaf-spine fabric with multi-vendor white-box switches:

- OFDPA (Broadcom Trident2/Tomahawk)
- Stratum (Barefoot Tofino)

Trellis & P4

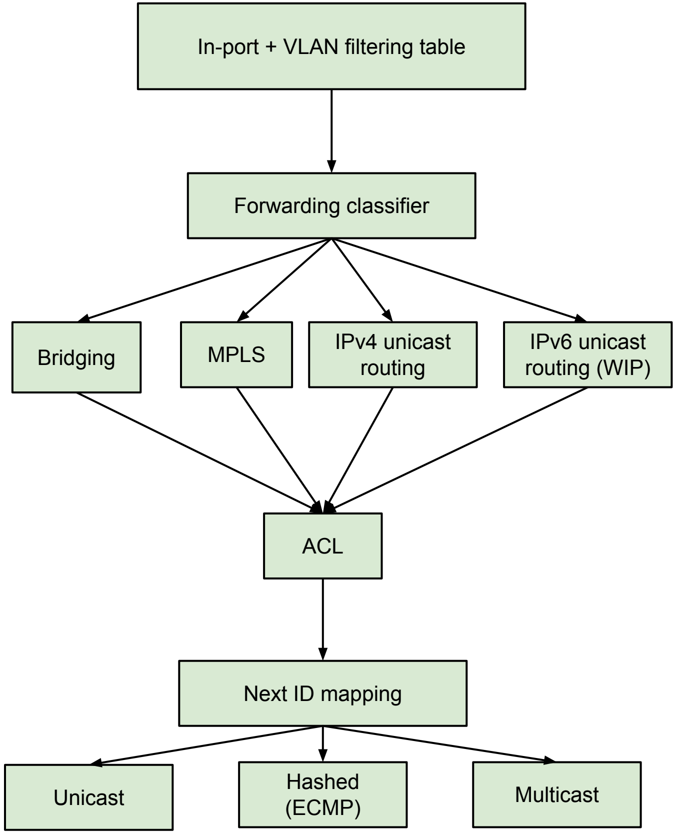


Pipeline-agnostic apps - use ONOS FlowObjective API
Trellis apps
Segment Routing
DHCP L3 Relay
vRouter
Multicast
...
ONOS
Fabric.p4 pipeconf
OF-DPA driver
fabric.p4 driver
OpenFlow
P4Runtime
Broadcom Qumran
Broadcom Tomahawk
Broadcom Trident2
Barefoot Tofino
Mellanox Spectrum 1
White-box switches

fabric.p4

- **P4 implementation of the Trellis reference pipeline**
  - Inspired by Broadcom OF-DPA pipeline
  - Tailored to Trellis needs (fewer tables, easier to control)
  - Work in progress:
    - Missing support for IPv6, double-VLAN termination
- **Bring more heterogeneity in Trellis with P4-capable silicon**
  - Works with both programmable and fixed-function chips (logical pipeline of legacy L2/L3/MPLS features)
  - Any switch pipeline that can be mapped to fabric.p4 can be used with Trellis
- **Extensible open-source implementation**
  - https://github.com/opennetworkinglab/onos/.../fabric.p4

fabric.p4 pipeline

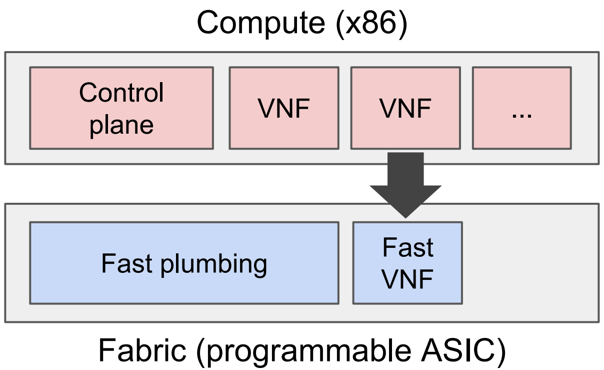


Use case 2:
VNF offloading

VNF offloading

- **Programmable data planes offer great flexibility beyond “plumbing”**

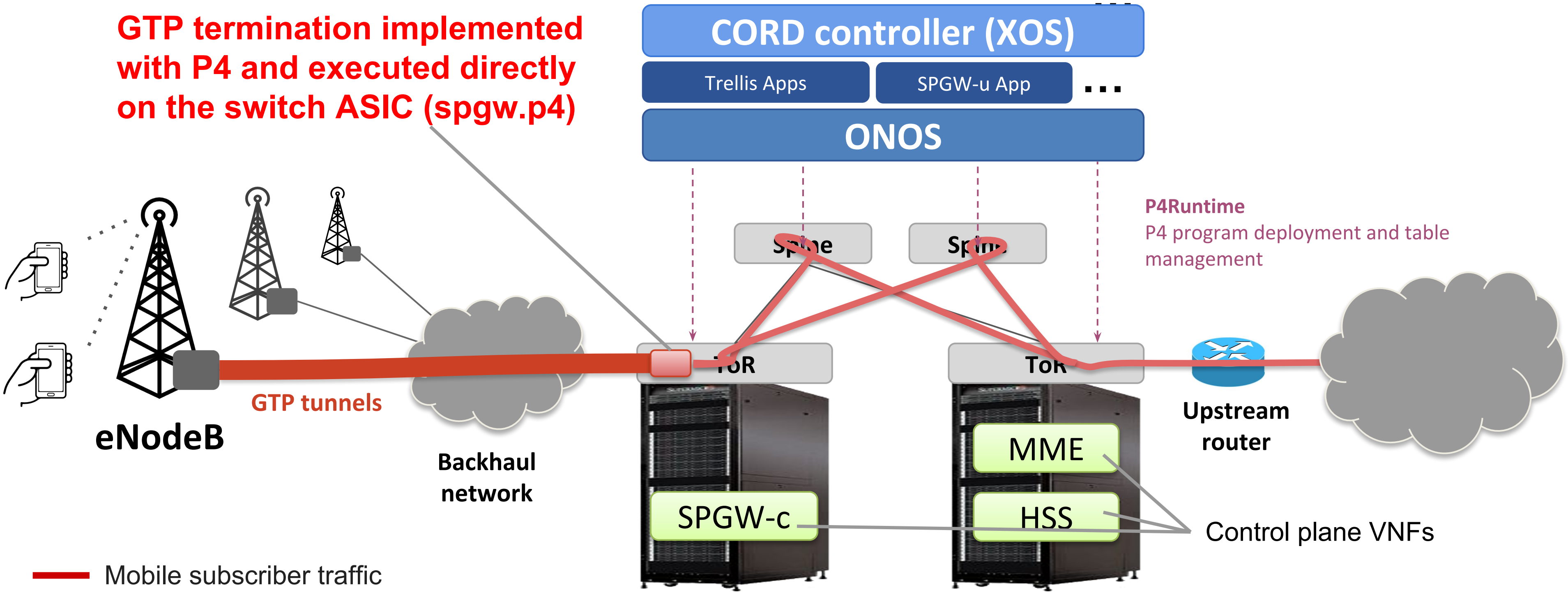
Progr. ASIC capabilities	VNF building blocks
Arbitrary header parsing/deparsing	Domain specific encap/decap (e.g. PPPoE termination, GTP, etc.)
Stateful memories	TCP connection tracking (L4 load balancing, NAT, firewall, etc.)
Computational capabilities	Billing



- **Benefits**
  - **Performance** - VNFs executed at line rate, e.g. O(Tbit/s) for DC switch
  - **Low latency and jitter** - Avoid non-determinism of x86 processing
  - **Power consumption** - Less CPU resources for packet processing, use switch that is there anyways

M-CORD with P4 fastpath

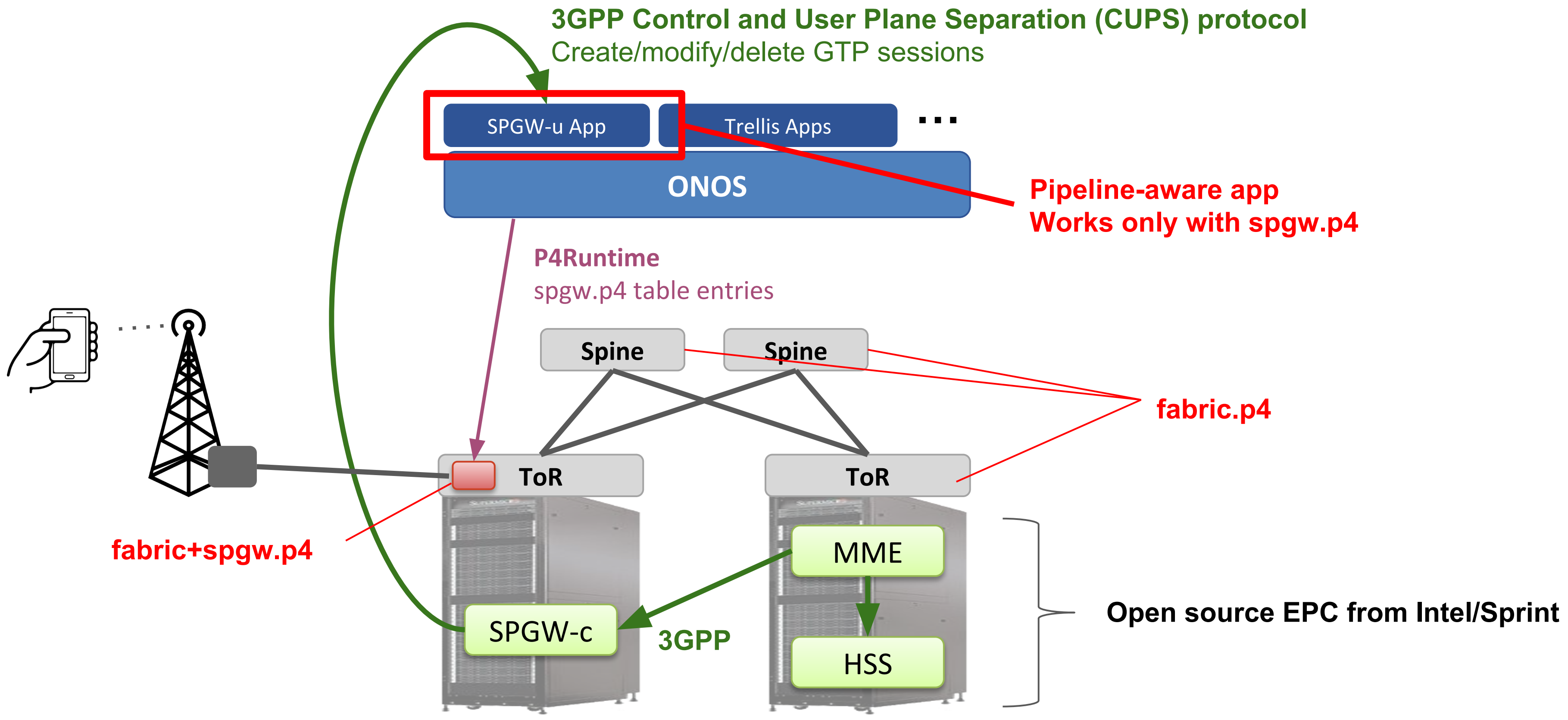
Demo @ MWC & ONS NA '18



spgw.p4

- **PoC P4 implementation of the Serving and Packet Gateway (S/PGW) user plane:**
  - ~300 lines of P4_16 code
  - Integrated with fabric.p4
  - https://github.com/opennetworkinglab/onos/.../spgw.p4
- **Good enough to demonstrate end-to-end connectivity**
  - Support GTP encap/decap, filtering, charging functionalities
- **Missing features (future work - need help)**
  - *QoS, downlink buffering during handovers*

SPGW-u ONOS App



Residential service edge/BNG (se.p4)

- **ONF is working with Deutsche Telekom to open-source a production-grade implementation of a residential service edge/BNG in P4**
- **Enables fast path for residential access**
- **Features:**
  - PPPoE termination
  - Reverse-path filtering (MAC, IPv4/v6)
  - Metering
  - TR-101 double-VLAN termination
  - 2-label MPLS termination
- **Community help needed for integration with Trellis and fabric.p4**

Pointers

- **P4_16 / P4Runtime specifications**
  - https://p4.org/specs/
- **Stratum project**
  - https://stratumproject.org/
- **ONOS**
  - https://wiki.onosproject.org/display/ONOS/Wiki+Home
- **Fabric.p4**
  - https://wiki.onosproject.org/x/wgBkAQ
- **Hands-on tutorial with P4/P4Runtime/ONOS**
  - http://bit.ly/onos-p4-tutorial-slides

ONOS-P4 Brigade - Join the effort!

Learn more - P4 Brigade Wiki:

https://wiki.onosproject.org/display/ONOS/P4+brigade

P4 Brigade mailing list:

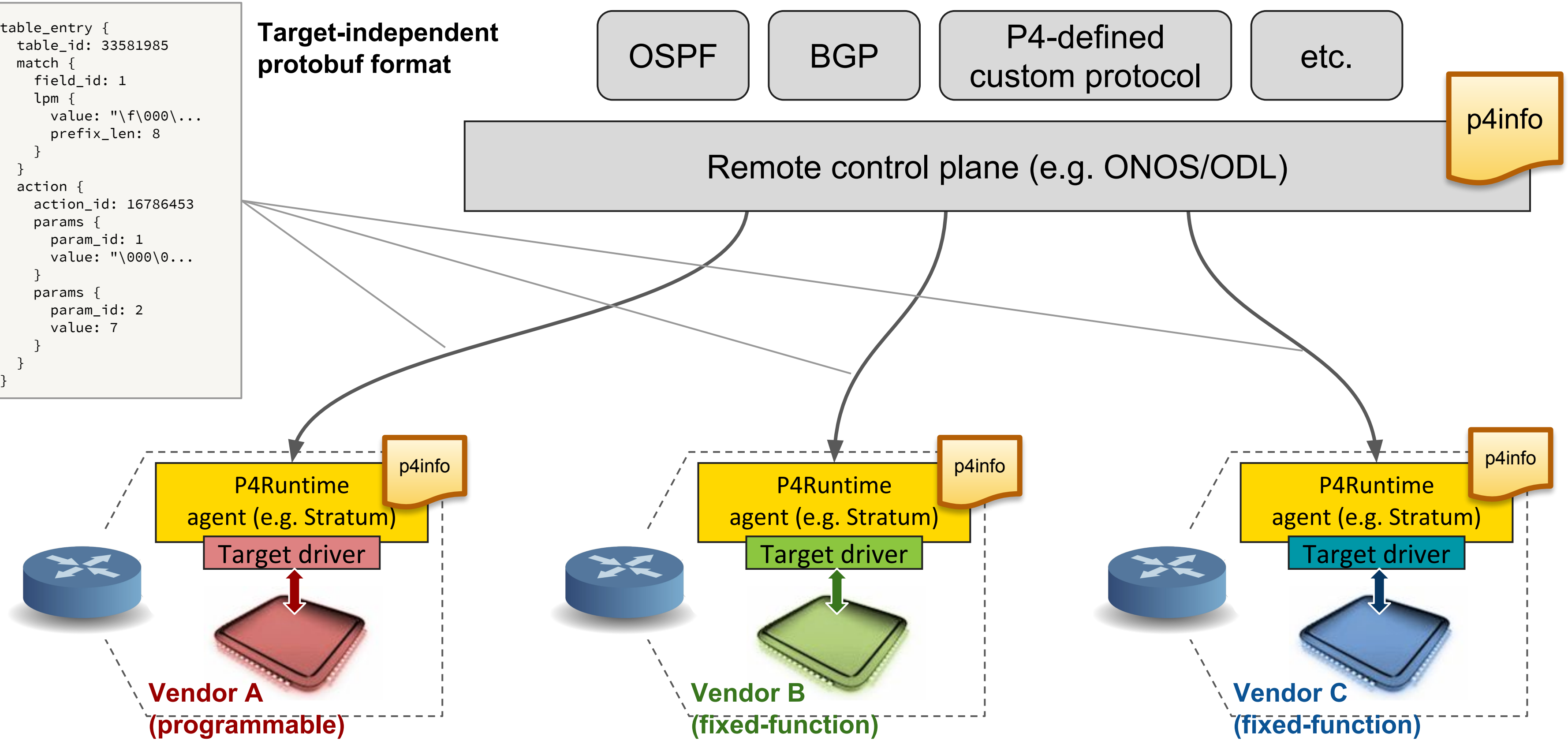
brigade-p4@onosproject.org

Thanks

Silicon-independent remote control

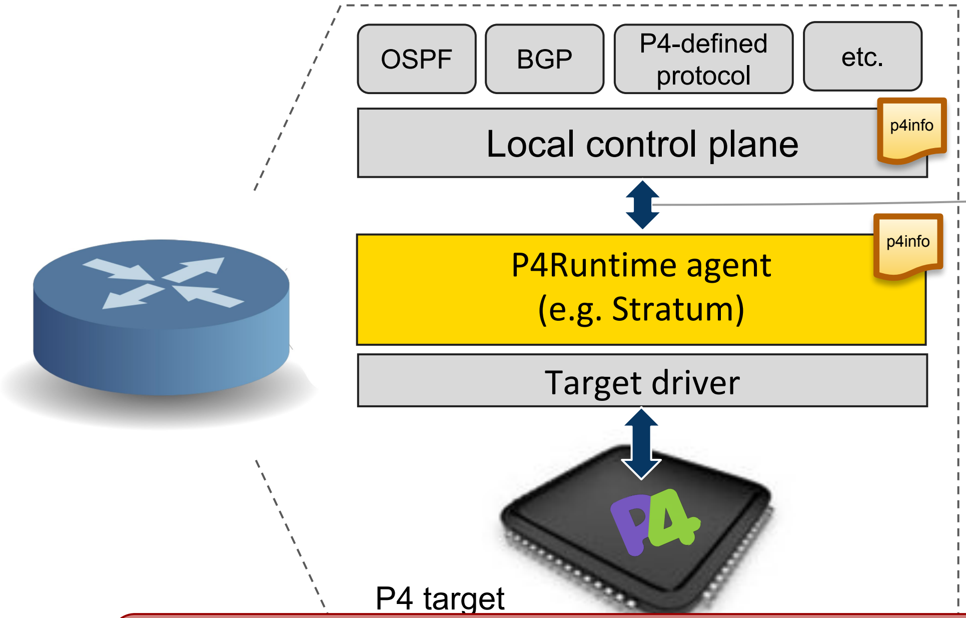
```
table_entry {
  table_id: 33581985
  match {
    field_id: 1
    lpm {
      value: "\f\000\...
      prefix_len: 8
    }
  }
  action {
    action_id: 16786453
    params {
      param_id: 1
      value: "\000\0...
    }
    params {
      param_id: 2
      value: 7
    }
  }
}
```

Target-independent protobuf format



Slide courtesy P4.org

Portability of local control plane



Same target-independent protobuf format

```
table_entry {
  table_id: 33581985
  match {
    field_id: 1
    lpm {
      value: "\f\000\...
      prefix_len: 8
    }
  }
  action {
    action_id: 16786453
    params {
      param_id: 1
      value: "\000\0...
    }
    params {
      param_id: 2
      value: 7
    }
  }
}
```

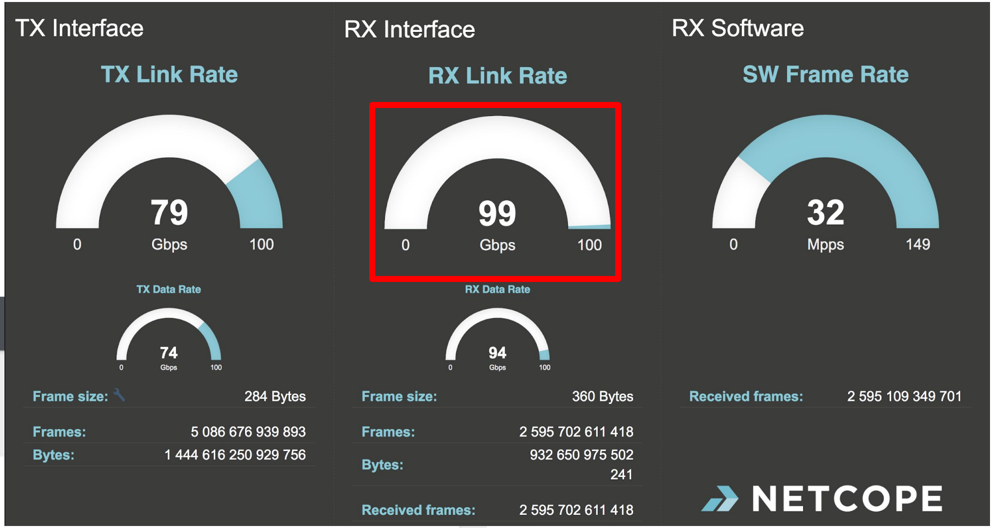
The P4 Runtime API can be used equally well by a remote or local control plane

Slide courtesy P4.org

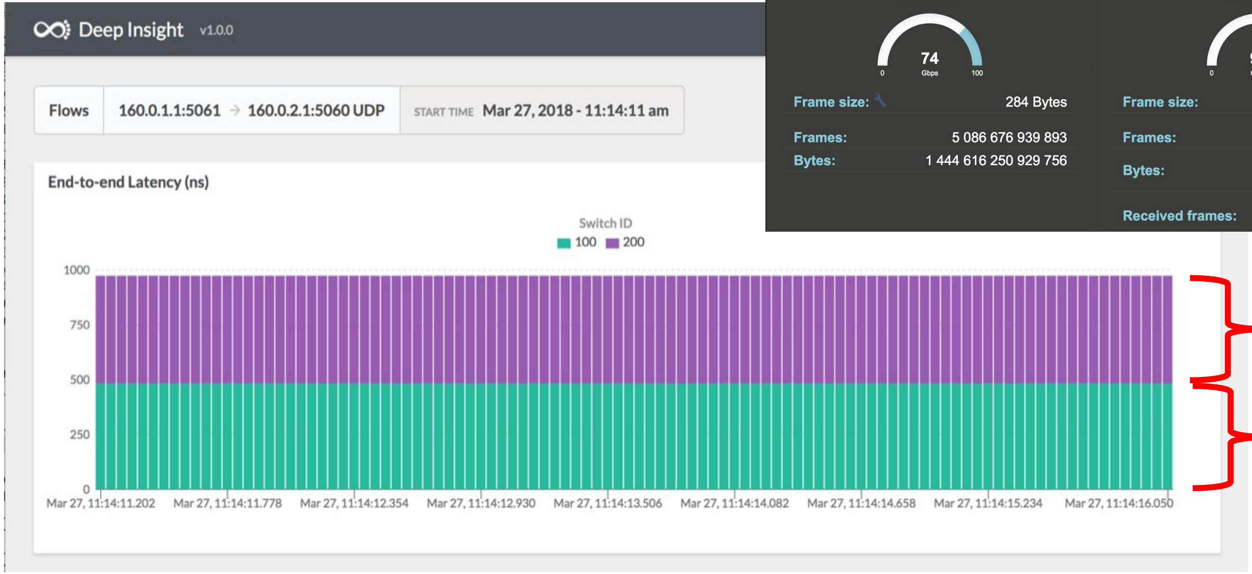
Performance demo @ ONS NA 2018

Overhead due to GTP and INT headers (when processing small packets)

Throughput



Inband network telemetry (INT) used to measure GTP processing latency



~490ns to perform GTP encap plus forwarding (ToR 1)

~480ns to perform forwarding (ToR 2)