P4 on Fixed-Function Switches with Stratum

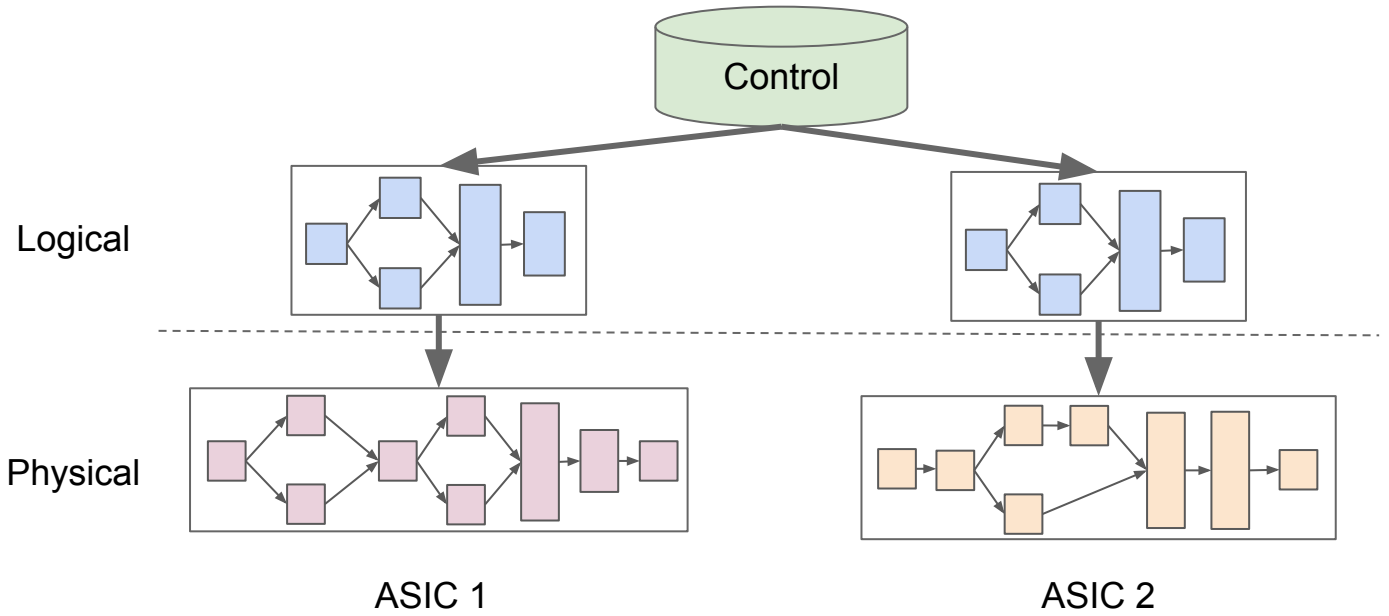
Maximilian Pudelko
ONF

Why are you here? / Outline

- You are interested in what it means to compile P4 programs for fixed-pipeline switches
  - How does it work and where are the limitations?
- You are a vendor and want to know how to port platforms to Stratum
  - What steps have to be done and how much effort is it?
- You are an operator and want to know how to use Stratum on fixed-function switches
  - What's possible with Stratum today?

Role of P4

- Provide clear pipeline definition using P4 tailored to role
- Useful for fixed-function/traditional ASICs as well as programmable chips
- Enables portability control plane apps



Slide adapted from Google

Benefits of P4 on Fixed-Function Switches

P4 program as an unambiguous *contract* describing the complete network behaviour in *machine-readable* format.

Benefits:

- Simplification of the control plane
- Easier, better and automated switch testing and validation
- Optionality of targets

What about SAI?

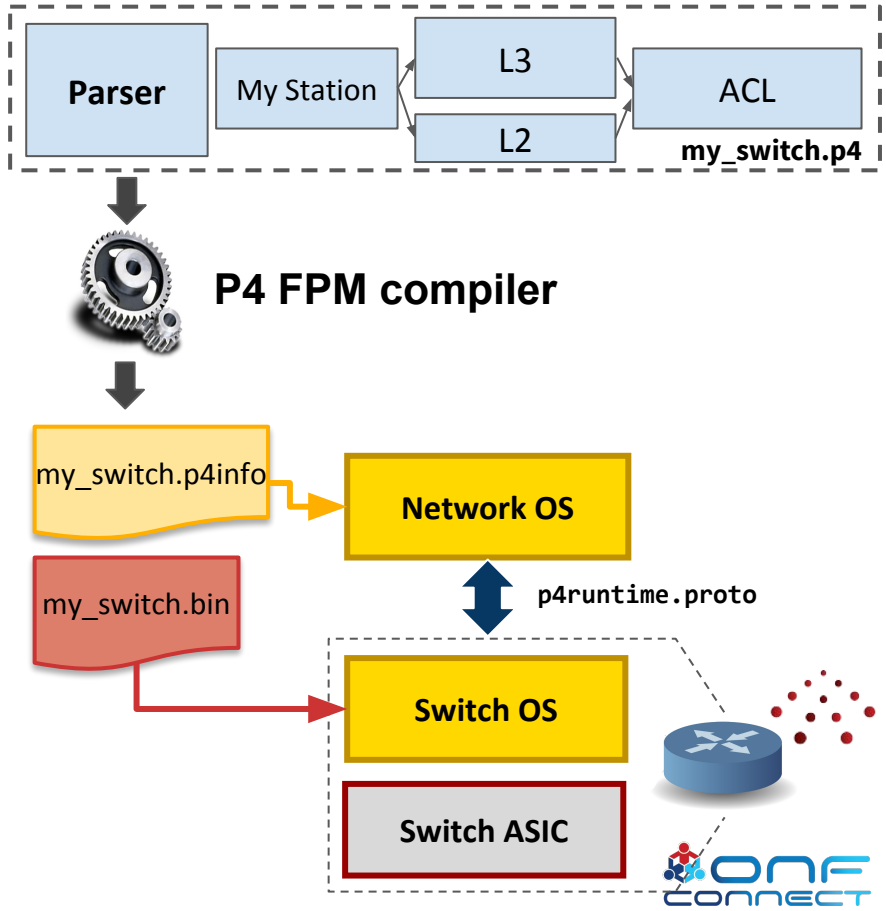
- SAI pipeline semantics are in English
  - Prone to errors and interpretation
  - Manual test generation
- Fixed API that is tightly coupled to pipeline definition
- Universal APIs become more complicated with more features
- Stratum provides better upgrade path
  - Just limited by the underlying SDK
  - New features can be mapped easily and are backwards compatible

Enabling P4 on Broadcom XGS

Enabling P4 on Broadcom XGS

Bringing P4 to FPM has two sides:

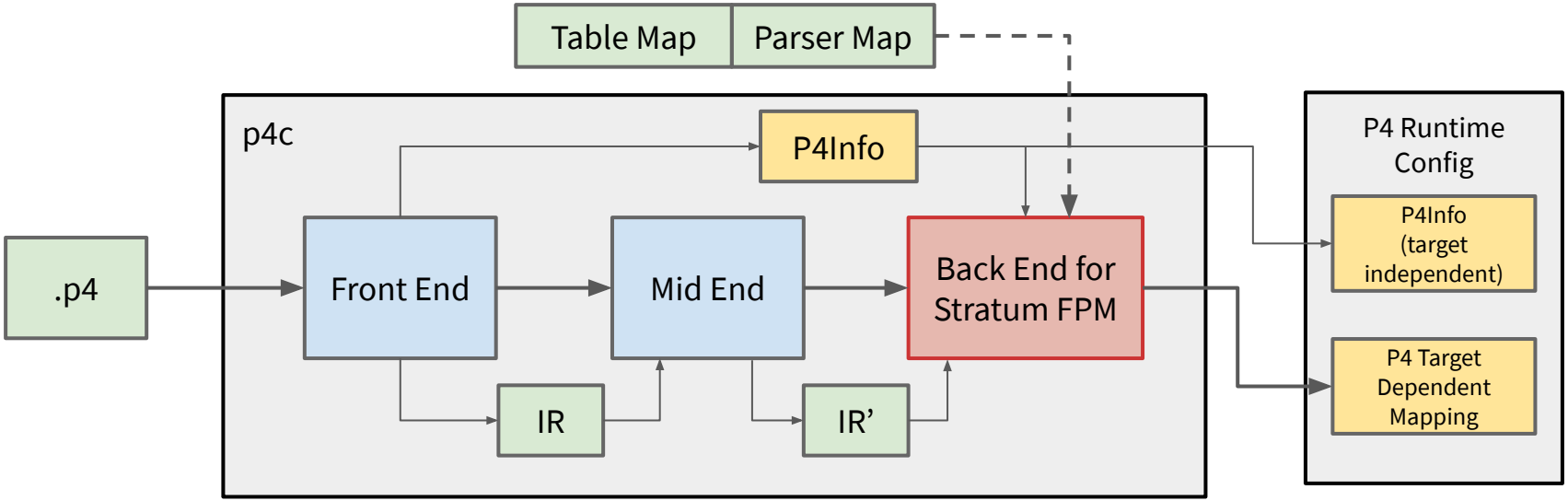
- Map arbitrary P4 code to fixed-function ASIC
  - New compiler needed
- Handle P4RT requests at runtime
  - Layers of resource managers
  - Wrapper around SDK



Enabling P4 on Broadcom XGS

Compiler

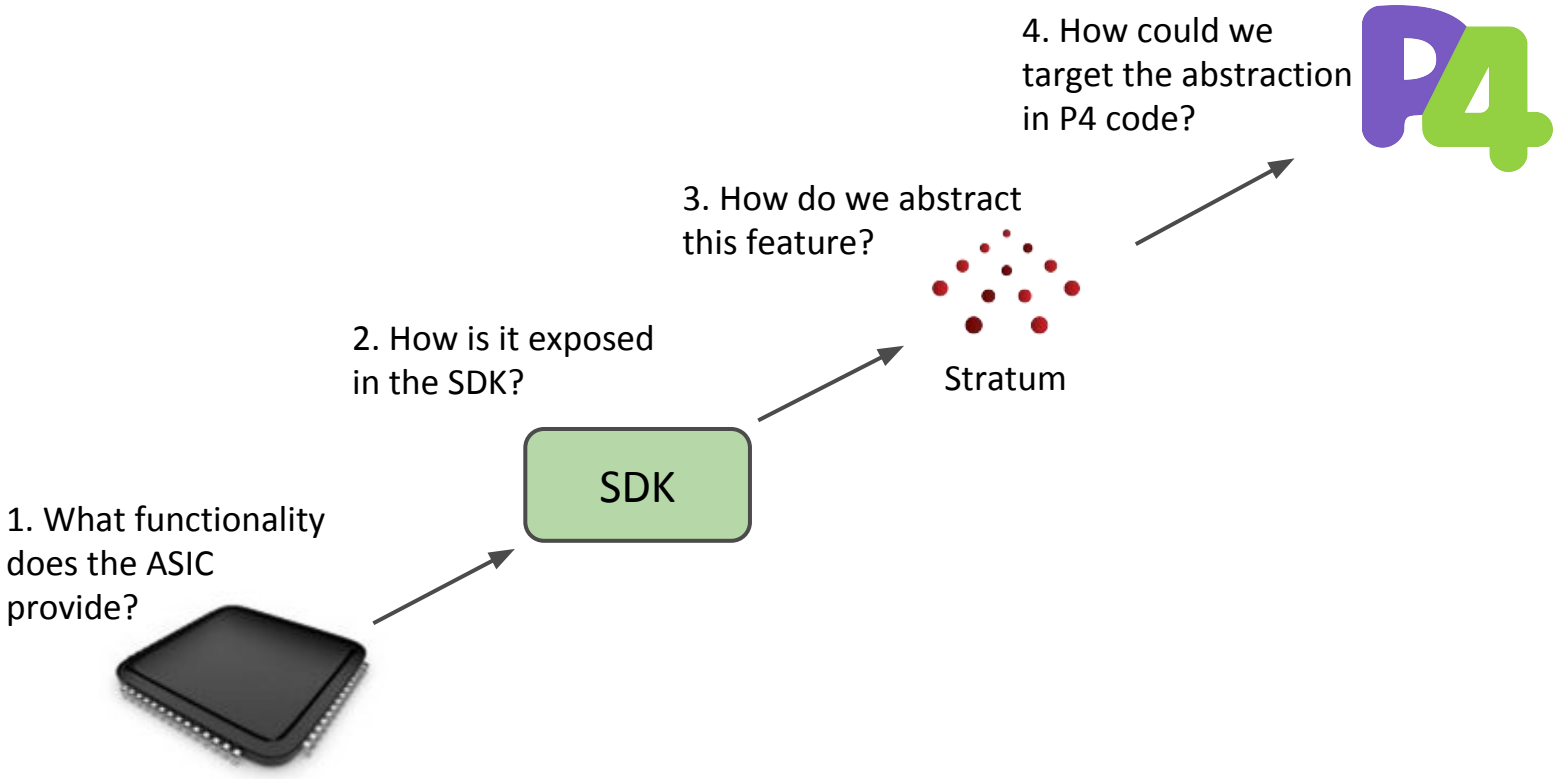
P4C FPM Compiler



- Front and mid end from open source P4 compiler
- Backend takes IR and mapping files
- Creates target-dependent mapping (pipeline configuration)
- Passes through target-independent configuration (p4info)

Fixed Function Feature Mapping

From the View of a Compiler (Writer)



Enabling P4 on Broadcom XGS

L3 forwarding as the ASIC/SDKLT understands it

L3_IPV4_UC_ROUTE_VRF		
Key	**Type**	**Mandatory?**
IPv4	IPV4_T	Yes
IPv4 Mask	IPV4_T	Yes
VRF ID	VRF_T	Yes
ECMP ID	ECMP_T	
NHOP ID	NHOP_T	
ECMP_NHOP	BOOL_T	

ECMP		
Key	**Type**	**Mandatory?**
ECMP ID	ECMP_T	Yes
NHOP IDs	NHOP_T[1024]	
Num Paths	UINT16_T	

L3_UC_NHOP		
Key	**Type**	**Mandatory?**
NHOP ID	NHOP_T	Yes
MODPORT	PORT_T	
VLAN ID	VLAN_T	
MAC DA	MAC_T	
L3 EIF ID	INTF_T	

L3_EIF		
Key	**Type**	**Mandatory?**
L3 EIF ID	INTF_T	Yes
MAC SA	MAC_T	
VLAN ID	VLAN_T	

- Table based approach
- INSERT/LOOKUP/DELETE
- Each row is an entry
- Be careful to keep them in sync!

Enabling P4 on Broadcom XGS

L3 forwarding as we expose it

L3_IPV4_UC_ROUTE_VRF		
Key	**Type**	**Mandatory?**
IPv4	IPV4_T	Yes
IPv4 Mask	IPV4_T	Yes
VRF ID	VRF_T	Yes
ECMP ID	ECMP_T	
NHOP ID	NHOP_T	
ECMP_NHOP	BOOL_T	

```
@switchstack("pipeline_stage: L3_LPM")
table l3_fwd_table {
    key = {
                                        : lpm;
                                        : exact;
```

demo_table_map.pb.txt

```
table_addenda_map {
  key: "ingress.l3_fwd.l3_fwd_table"
  value {
    type: P4_TABLE_L3_IP
  }
}
```

```
                                    on_profile;
```

ECMP		
Key	**Type**	**Mandatory?**
ECMP ID	ECMP_T	Yes
NHOP IDs	NHOP_T[1024]	
Num Paths	UINT16_T	

```
@max_group_size("1024")
action_selector (
        HashAlgorithm.crc16, 32w1024, 32w14
) ecmp_action_profile;
```

Enabling P4 on Broadcom XGS

L3 forwarding as we expose it

L3_EIF		
Key	**Type**	**Mandatory?**
L3 EIF ID	INTF_T	Yes
MAC SA	MAC_T	
VLAN ID	VLAN_T	

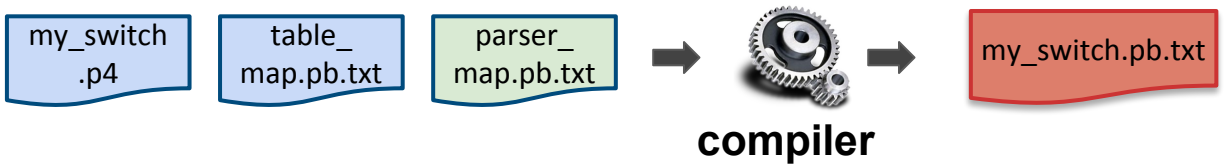
L3_UC_NHOP		
Key	**Type**	**Mandatory?**
NHOP ID	NHOP_T	Yes
MODPORT	PORT_T	
VLAN ID	VLAN_T	
MAC DA	MAC_T	
L3 EIF ID	INTF_T	

```
action set_nexthop (
        PortNum port,
        EthernetAddress smac,
        EthernetAddress dmac,
        bit<12> dst_vlan) {
    standard_metadata.egress_spec = port;
    local_metadata.dst_vlan = dst_vlan;
    hdr.ethernet.src_addr = smac;
    hdr.ethernet.dst_addr = dmac;
    hdr.ipv4_base.ttl = hdr.ipv4_base.ttl - 1;
}
```

- Table entries are created automatically
- IDs are maintained by Stratum
- Unnecessary details are hidden

Enabling P4 on Broadcom XGS

Compiler Output



- Table descriptors
- Field descriptors
- Action descriptors
- P4 control blocks

```
table_map {
  key: "ingress.l3_fwd.l3_fwd_table"
  value {
    table_descriptor {
      type: P4_TABLE_L3_IP
      pipeline_stage: L3_LPM
    }
  }
}
```

```
table_map {
  key: "hdr.ipv4_base.dst_addr"
  value {
    field_descriptor {
      type: P4_FIELD_TYPE_IPV4_DST
      valid_conversions {
        match_type: LPM
        conversion: P4_CONVERT_TO_U32_AND_MASK
      }
      bit_offset: 128
      bit_width: 32
      header_type: P4_HEADER_IPV4
    }
  }
```

This is not a binary/bitstream passed to the ASIC!

Enabling P4 on Broadcom XGS

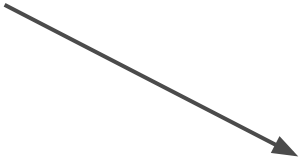
Runtime

Fixed Function Feature Mapping

Inside the Statum Runtime

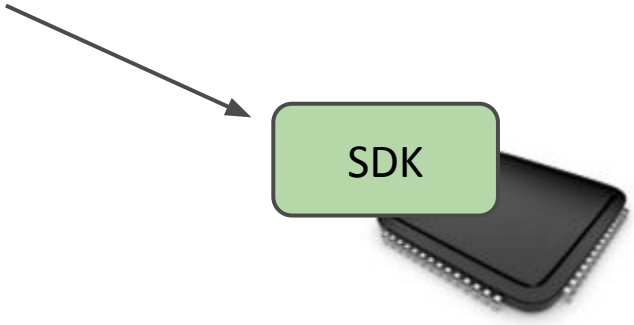
Runtime

Flow request from control plane



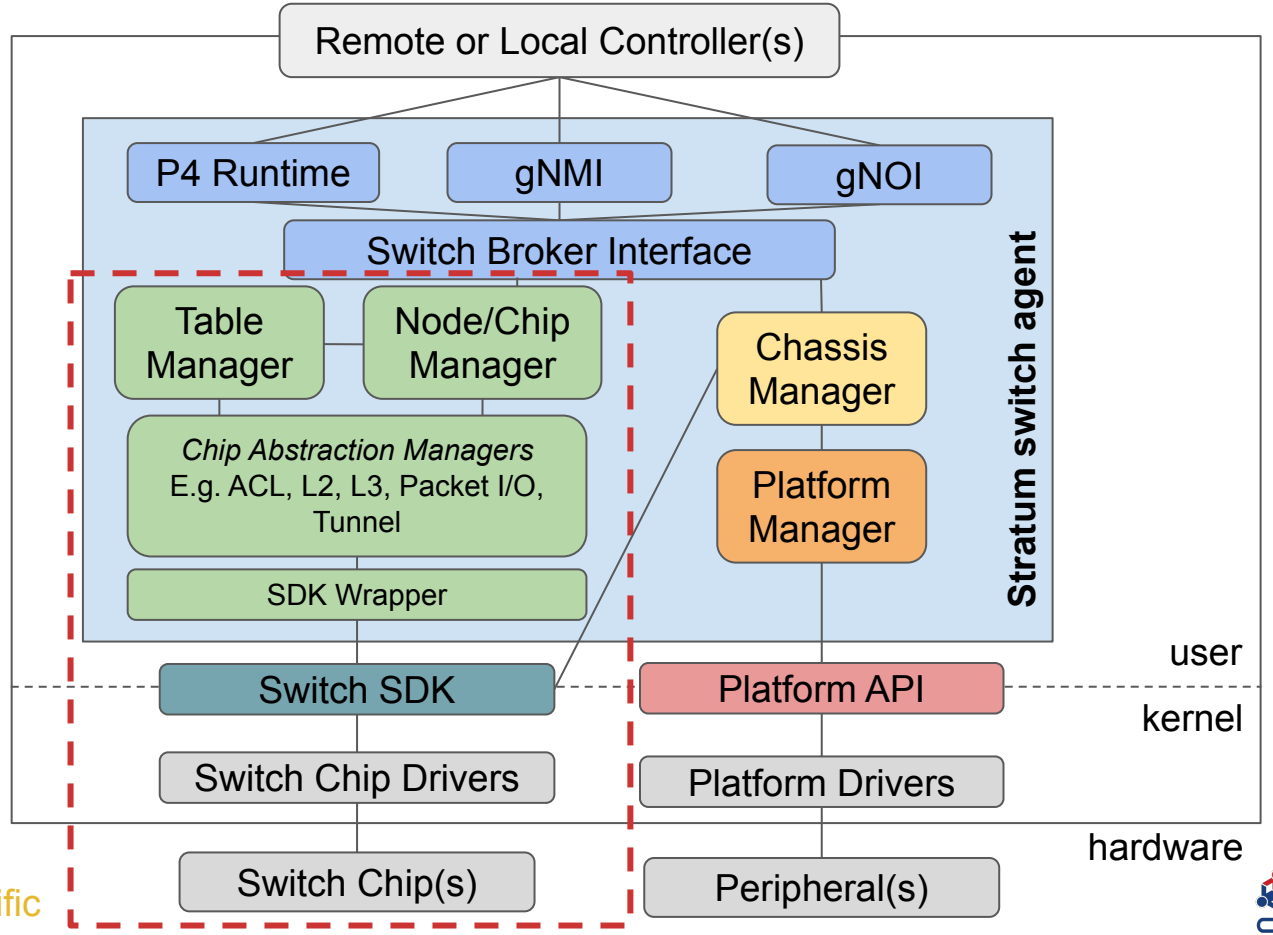
Stratum

How do we recognize a request and handle it correctly?



Make appropriate SDK calls

Enabling P4 on BCM Tomahawk



TOFINO programmable

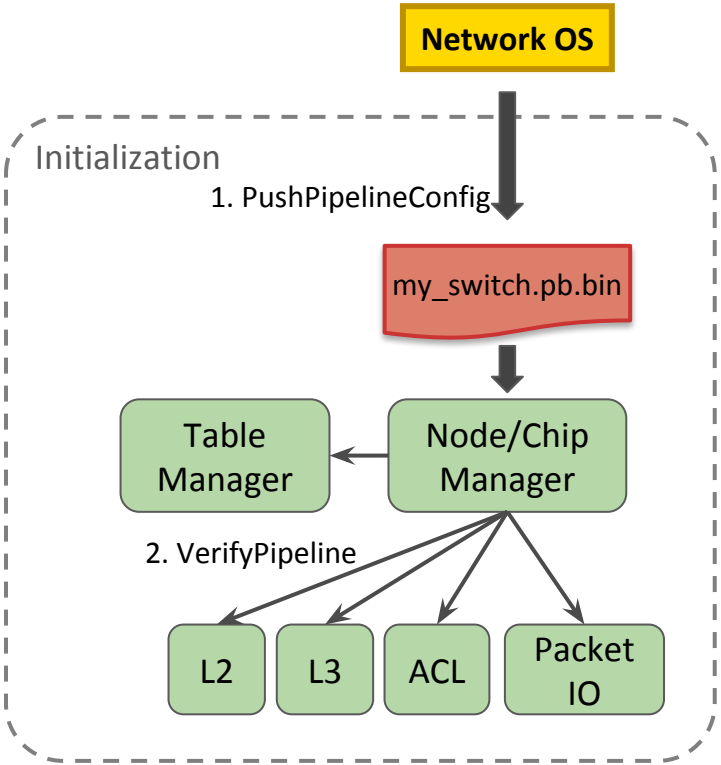
BROADCOM fixed

Shared (HW agnostic)
Chip specific
Platform specific
Chip and Platform specific

ONF CONNECT

Enabling P4 on BCM Tomahawk

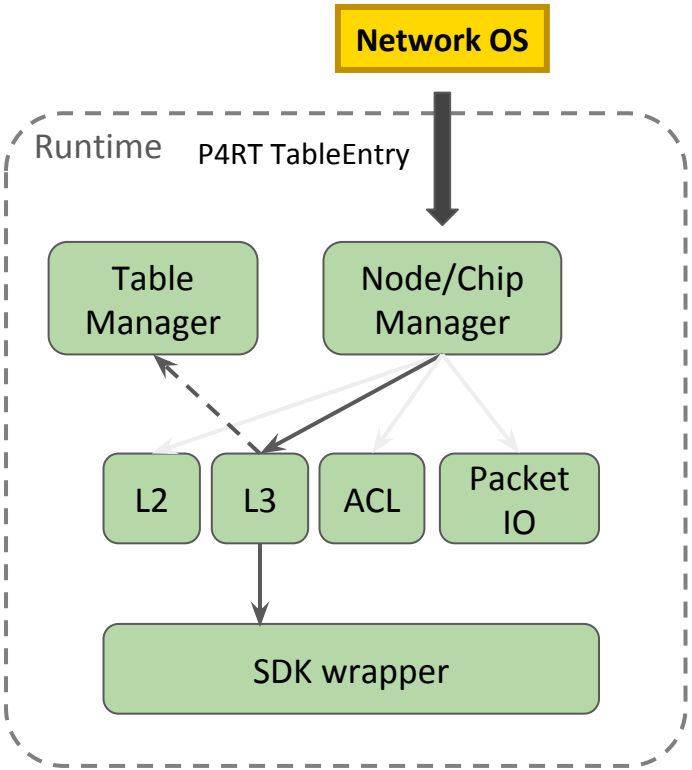
Runtime Mapping



- NOS pushes compiler output
- Node manager distributes it to feature managers
- “Is this pipeline mapping acceptable?“
- Feature managers perform necessary setup, resource allocation

Enabling P4 on BCM Tomahawk

Runtime Mapping




```
lt L3_IPV4_UC_ROUTE_VRF insert
        VRF_ID=0
        IPV4=10.2.0.0
        IPV4_MASK=255.255.0.0
        ECMP_ID=1
        ECMP_NHOP=true
```

- NOS wants to insert new L3 flow
- Node translate P4RT request to BCM flow with mapping from compiler
- Hands over to feature manager
- Feature manager validates flow and maps parameters
- Realization through SDK wrapper
- Inform Table manager about new flow

Enabling P4 on Broadcom XGS

Conclusion

Enabling P4 on Broadcom XGS

Caveats

Key differences to Tofino or bmv2:

- Pipeline defined by switch chip and Stratum's abstraction of it, P4 code only describes it
- Tables and action fields are given by the ASIC
- Fixed headers and parser (no custom protocols)
- Some implicit behavior has to be accounted for (e.g. TTL dec.)
- Your annotations help the compiler to map headers and tables

Enabling P4 on Broadcom XGS

Chip SDKs

Statum on SDKLT:

- Complete open source stack today!
- Free to use and modify
- Support limited to Tomahawk chips
- Available features: L2, L3, ECMP, VLAN, ACL, PacketIO, Port counters, ...
- Missing features: VXLAN, L2 mcast, MPLS
- Future chip support possible

Enabling P4 on Broadcom XGS

Getting Started

- Try our demo: https://github.com/opennetworkinglab/stratum-onos-demo
- Compiler available as part of Stratum source tree: stratum/p4c_backends/fpm/
- Docker container: https://hub.docker.com/r/opennetworking/p4c

```
p4c-fpm --p4c_fe_options="-I /usr/share/p4c/p4include demo.p4" \
        --p4_info_file=build/fpm/p4info.txt \
        --p4_pipeline_config_text_file=build/fpm/pipeline_config.txt \
        --p4_pipeline_config_binary_file=build/fpm/pipeline_config.bin \
        --p4c_annotation_map_files=demo_table_map.pb.txt,demo_field_map.pb.txt \
        --target_parser_map_file=standard_parser_map.pb.txt \
        --slice_map_file=sliced_field_map.pb.txt
```

Next Steps

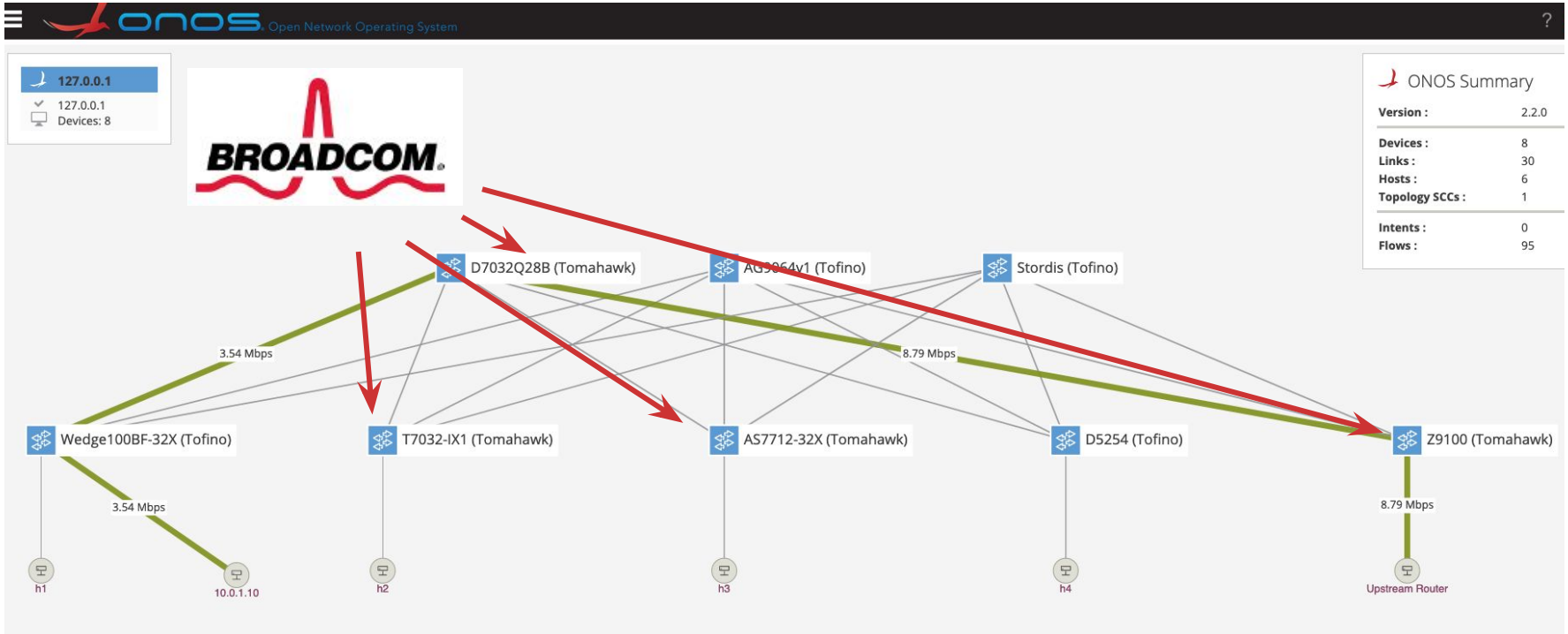
Stratum Roadmap

- More features to support Trellis (Double VLAN, MPLS)
- More ASICs from Broadcom XGS family
- Performance testing

Community Opportunities

- Porting FPM and Stratum runtime to SAI or other ASICs
- Support for other networking features (VXLAN)

P4 live on FPM: Visit our Demo!



- 5x3 Leaf-Spine topology
- Two chipsets
- 6 Vendors

- 4 Hosts
- Upstream router
- Free WiFi!

Thank You

Contribute to Stratum today:
https://github.com/opennetworkinglab/stratum

Demo source code:
https://github.com/opennetworkinglab/stratum-onos-demo

Note pad

What is is the main message?

Progress update? Developer tutorial for FPM P4? Dev tutorial to develop stratum_bcm?

- P4 is for more than just Tofino
- Control plane op: how do I use this?
- Vendor: How to I support stratum on my chip? How do I add new features (extend bcm_sdk_wrapper)?
- SAI people: We can reuse some of your efforts

One slide of P4 pitch

Benefits of P4 on a fixed function ASIC

- Clear semantic and simplified model for device roles, “just what you need”
- Easier and better testing
- Optionality of targets

“Why is this better than SAI?”

SAI semantics are in english

API Tightly coupled to pipeline definitions

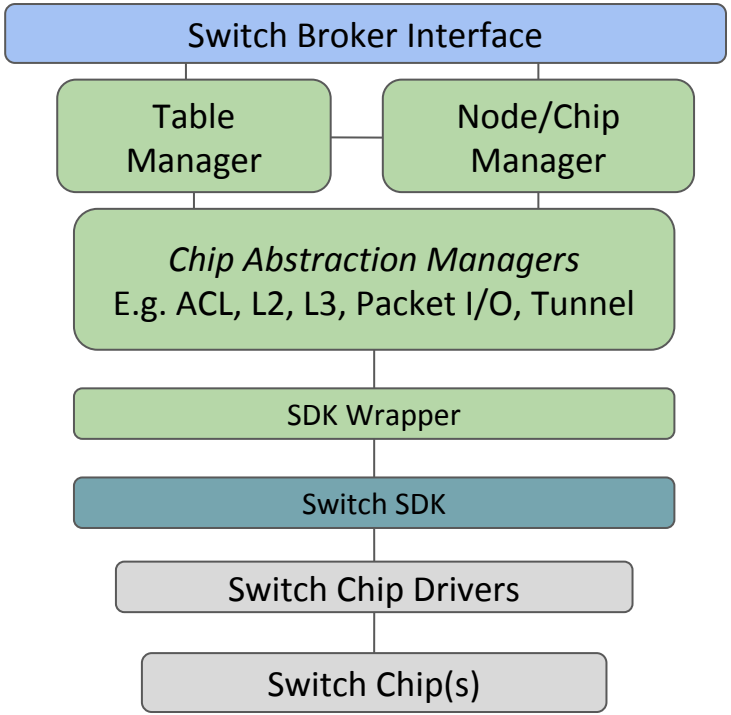
=> No/harder automated test generation

Test cases and targets need to be manually written

Stratum + SAI?

From a vendors perspective:

- How do we get Stratum support?
- How to we leverage the spent effort creating SAI support?
- Duplicate work necessary?



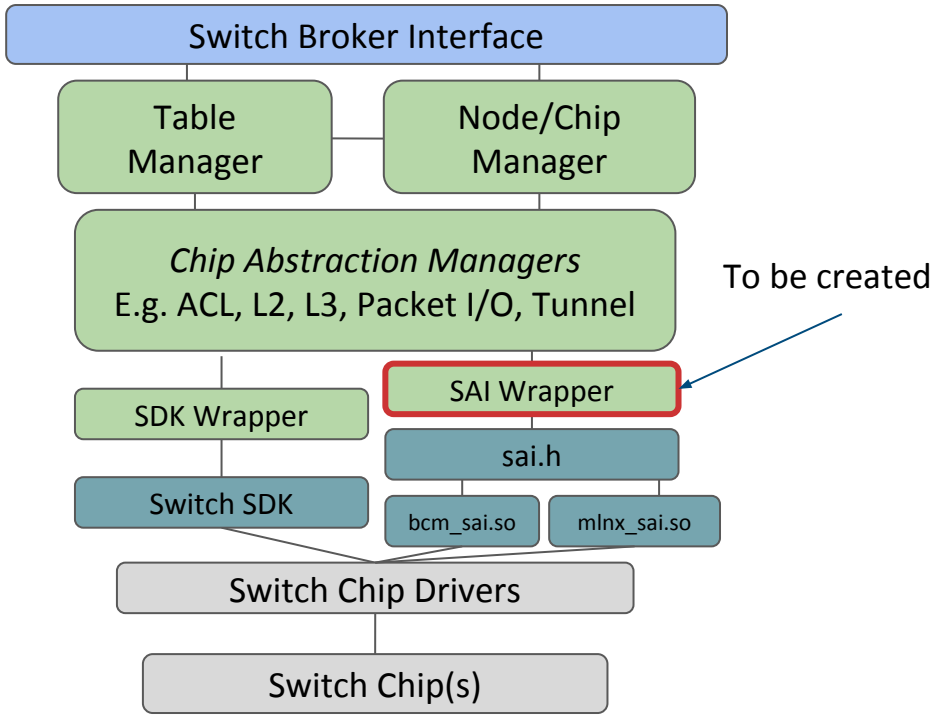
Stratum + SAI?

From a vendors perspective:

- How do we get Stratum support?
- How to we leverage the spent effort creating SAI support?
- Duplicate work necessary?

No!

- SAI like abstraction with SDK wrapper layer already in place
- SAI wrapper as an alternative to SDK wrapper, with SAI as backend



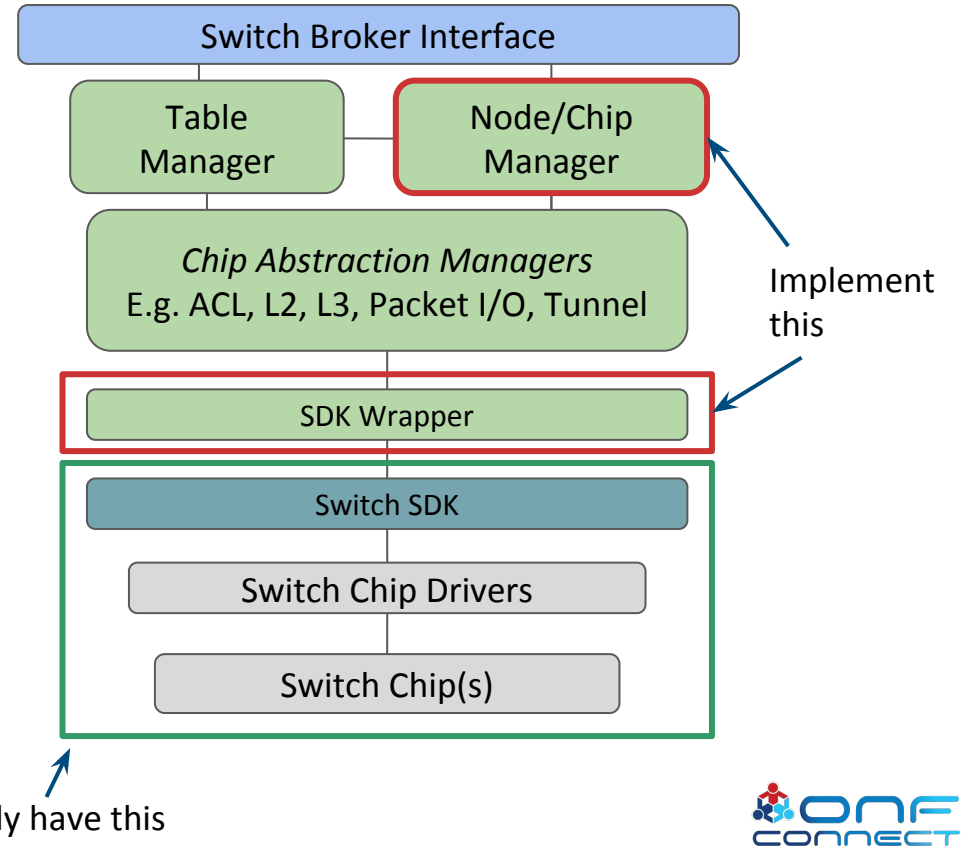
Porting Stratum to other SDKs/Vendors

High level steps:

- ASIC/SDK init code
- Map P4 code to flow entries (common representation)
  - Reuse or add compiler annotations
- Translate flow entries at runtime to SDK calls

Best case:

- Our L2/L3/... abstractions match your SDK structure
- Just provide your SDK wrapper code implementing the interface
- Get P4RT, gNOI, gNMI for free
- ONLPv2 ready



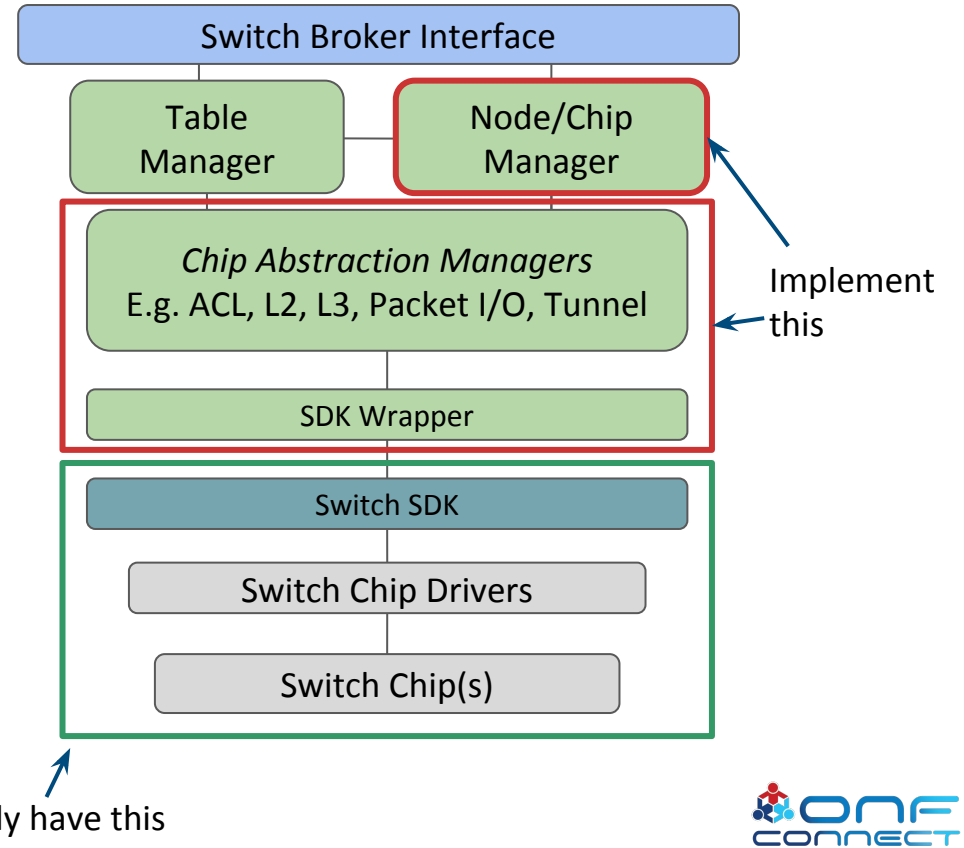
Porting Stratum to other SDKs/Vendors

High level steps:

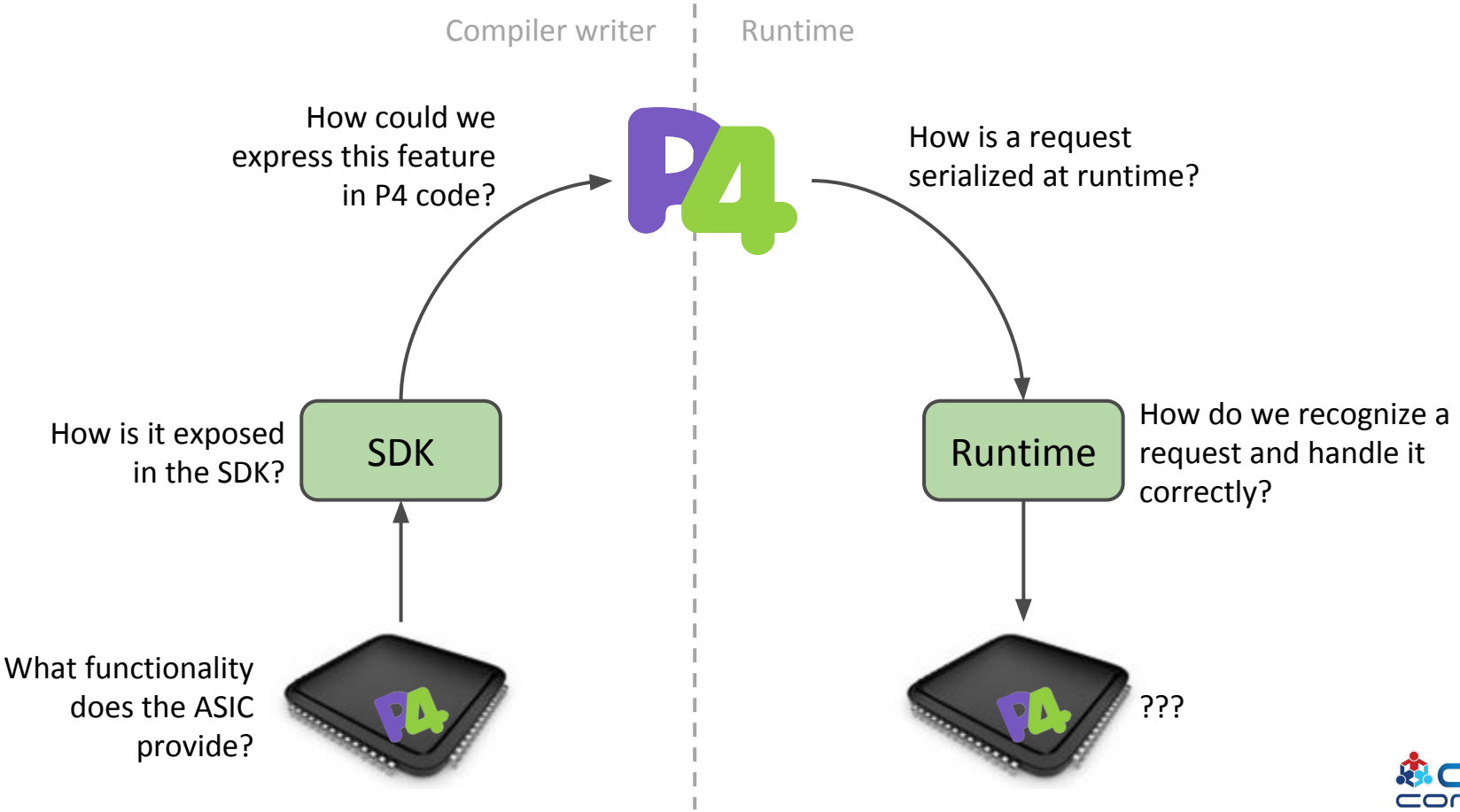
- ASIC/SDK init code
- Map P4 code to flow entries (common representation)
  - Reuse or add compiler annotations
- Translate flow entries at runtime to SDK calls

Slightly worse case:

- SDK is too different from abstractions
- Bring your own abstractions
- Still get P4RT, gNOI, gNMI for free
- ONLPv2 ready



Fixed Function Feature Mapping



Community Contributions related to stratum_bcm

- Initial seed code and architecture
- BcmNode and SDK wrapper for SDK6
- FPM (fixed pipeline mapping) P4 compiler backend
- Extensive code review and architectural guidance

- SDKLT wrapper co-development and testing
- Code porting to open source libraries, Bazel, P4Runtime v1.0, gNMI latest
- Support for deployment using Docker
- Bug fixes for Stratum, SDKLT, ONLPv2

- ONLPv2 support for their platforms
- Platform configuration files
- Debugging sessions

- SDK wrapper for SDKLT