STRATUM

Escaping Switch Vendor Lock-in through Open Interfaces and Software

Brian O'Connor, ONF
Devjit Gopalpur*, Google

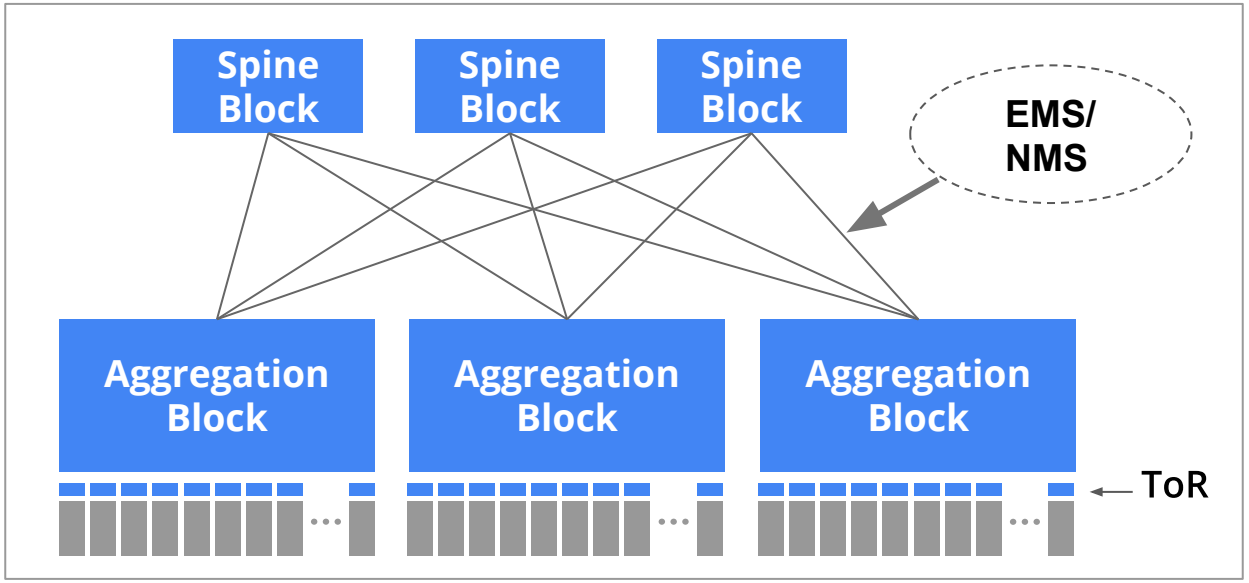
ONS North America - April 3, 2019

*On behalf of many at Google (Alireza Ghaffarkhah, Waqar Mohsin, Shashank Neelam, Jim Wanderer, Lorenzo Vicisano, Amin Vahdat, …)

Single Vendor Networking Makes Life Easy

That is, ***until you want to change things***. A classic example of vendor lock-in.

Explicitly, it may be difficult/impossible to replace individual blocks with other vendor's components.



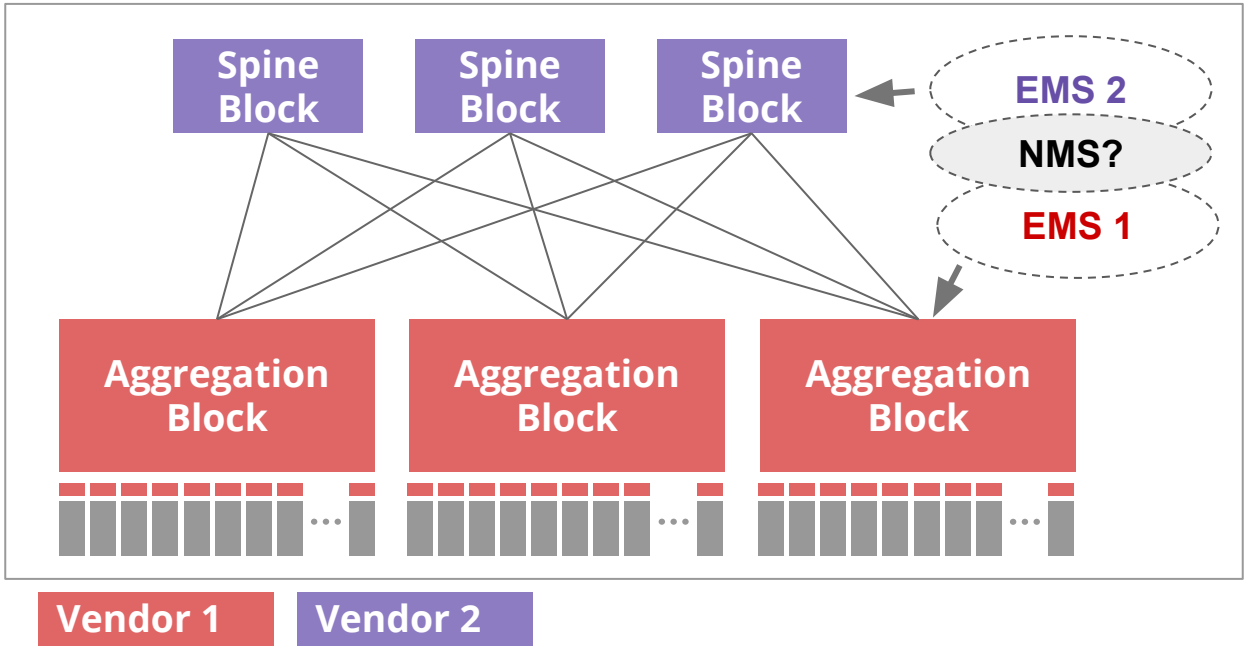
Implicitly, anything you build on this solution is inherently tied to your vendor's interfaces and models, amplifying the lock-in.

Multi-Vendor Introduces Complexity

But, allows for **performance**, **feature**, and **cost optimization**, while maintaining vendor choice.

Vendor-specific EMS understands a device's idiosyncrasies

- Protocols
- Models
- Pipeline



Which vendor's NMS do you use? *Vendor 1 or 2? Both, and add a higher level orchestration layer?* Either way, anything built on this solution is still locked-in.

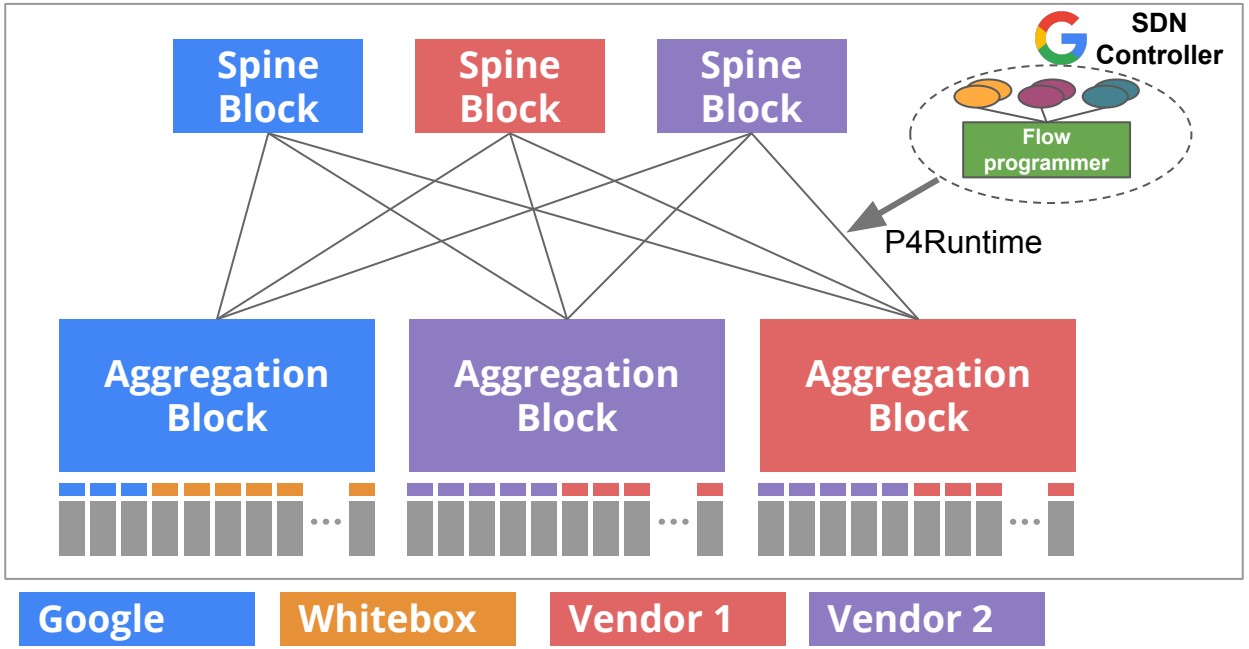
Standard Solutions to Multi-Vendor SDN

1. Least Common Denominator Interfaces (e.g. **`iptables`**, SAI)
   - Makes extensibility without recompiling the whole stack difficult
   - Harder to exploit unique hardware capabilities (e.g. programmability)
2. Underspecified Interfaces (e.g. OpenFlow, Flow Objectives)
   - More easily extensible (with some vendor-specific nuances)
   - In practice, upper layers of the stack need to be written to specific targets
3. Single Vendor Solution (e.g. vendor-specific SDKs/APIs)
   - Easy to exploit hardware's capabilities
   - Solutions usually not portable or reusable (i.e. locked-in)
   - Typically, the only approach that performs reliably and scales

Solution: Single Vendor Networking? … we have a problem!

Google's Approach to Multi-Vendor SDN

- Heterogeneous network
- Single consistent API
  - P4Runtime
  - OpenConfig
- Exploit unique HW capabilities (without changing the interfaces)
- Leverage commercial technology / vendors
  - Networking Vendors
  - ODMs
  - In-house / OEMs



Requirements for Multi-Vendor SDN

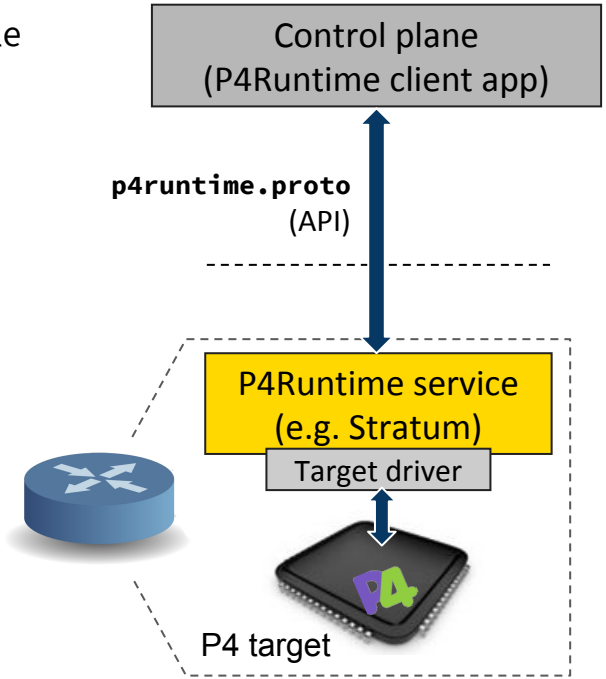
- Support for **vendor-neutral** control applications
  - Control plane is written once, compiled for multiple backends, i.e. hardware.
  - Contract provides extensibility. New use cases and network roles do not require modification of APIs or switch software.
- Support for **programmable hardware**
  - Even more flexibility - backend faithfully mimics software intent.
  - Pushes hardware abstraction up the stack.
  - Uniform runtime interface for heterogeneous silicon as well as network intent.
- Support for a **uniform network model**
  - Vendor-agnostic model of topology.
  - Simplifies operability of a multi-vendor network.

… which also provides …

- Enhanced deployment **velocity** at **scale**
  - Introduction of new functionality, hardware, etc. using common workflows.
  - Incremental support for new equipment.
  - Rapid prototyping by operators and vendors using a well-defined contract.
- Simplified **migration** of services
  - From traditional devices to programmable devices.
  - Between heterogeneous device blocks.
- Unified device **management**
  - Operators use common tools to deploy, configure, monitor and troubleshoot devices from multiple vendors.

Control interface: P4Runtime

- API for runtime control of switches
  - Designed around **PSA reference architecture**
  - Extended to **Fixed Pipeline Model (FPM)** i.e., non-programmable switches
- gRPC/protobuf-based API definition
  - Automatically generate client/server code for many languages
- Program-independent
  - P4 program defines the network function of a device
  - API doesn't change with the P4 program
- Dynamic reconfigurability
  - Push new P4 program at run time
  - Re-configure switch pipeline without modifying switch software

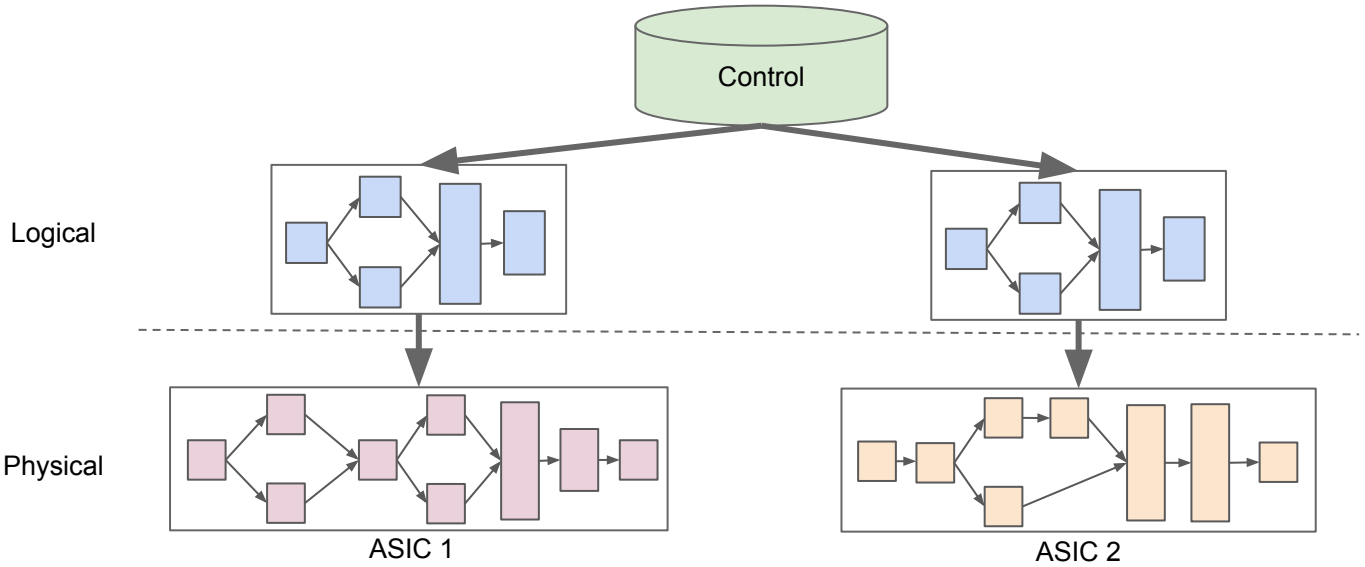


RC of version 1.0 available: https://p4.org/p4-spec/ (p4.org API WG)

Slide courtesy P4.org

Role of P4

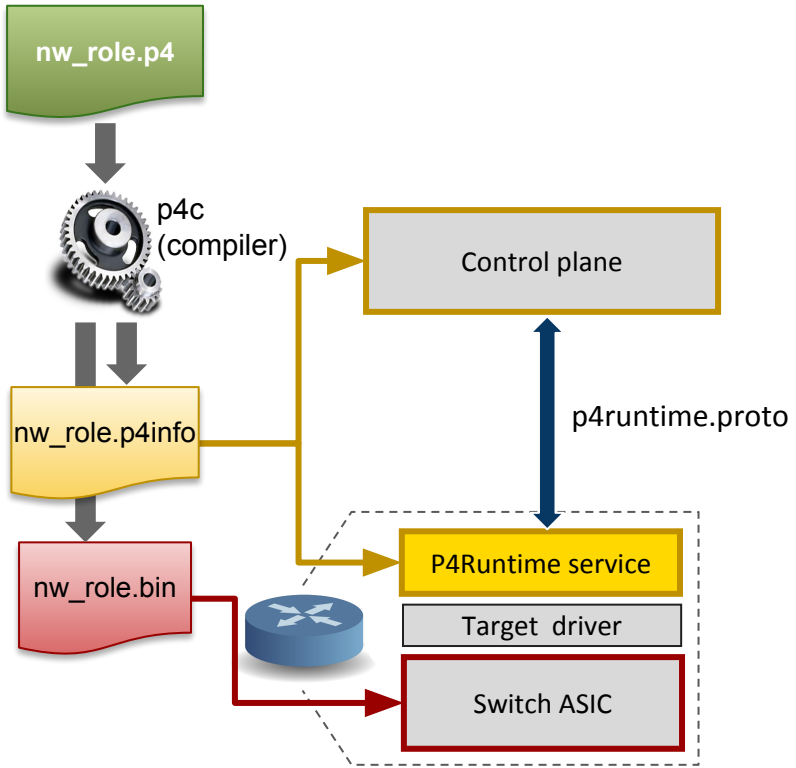
- Provides **formal definition** of the data plane pipeline tailored to a specific role
  - Describes protocol headers, tables, actions, counters, etc.
- Useful for fixed-pipeline/traditional ASICs as well as programmable chips
- Enables **portability**



P4 compiler workflow

P4 compiler generates 2 outputs:

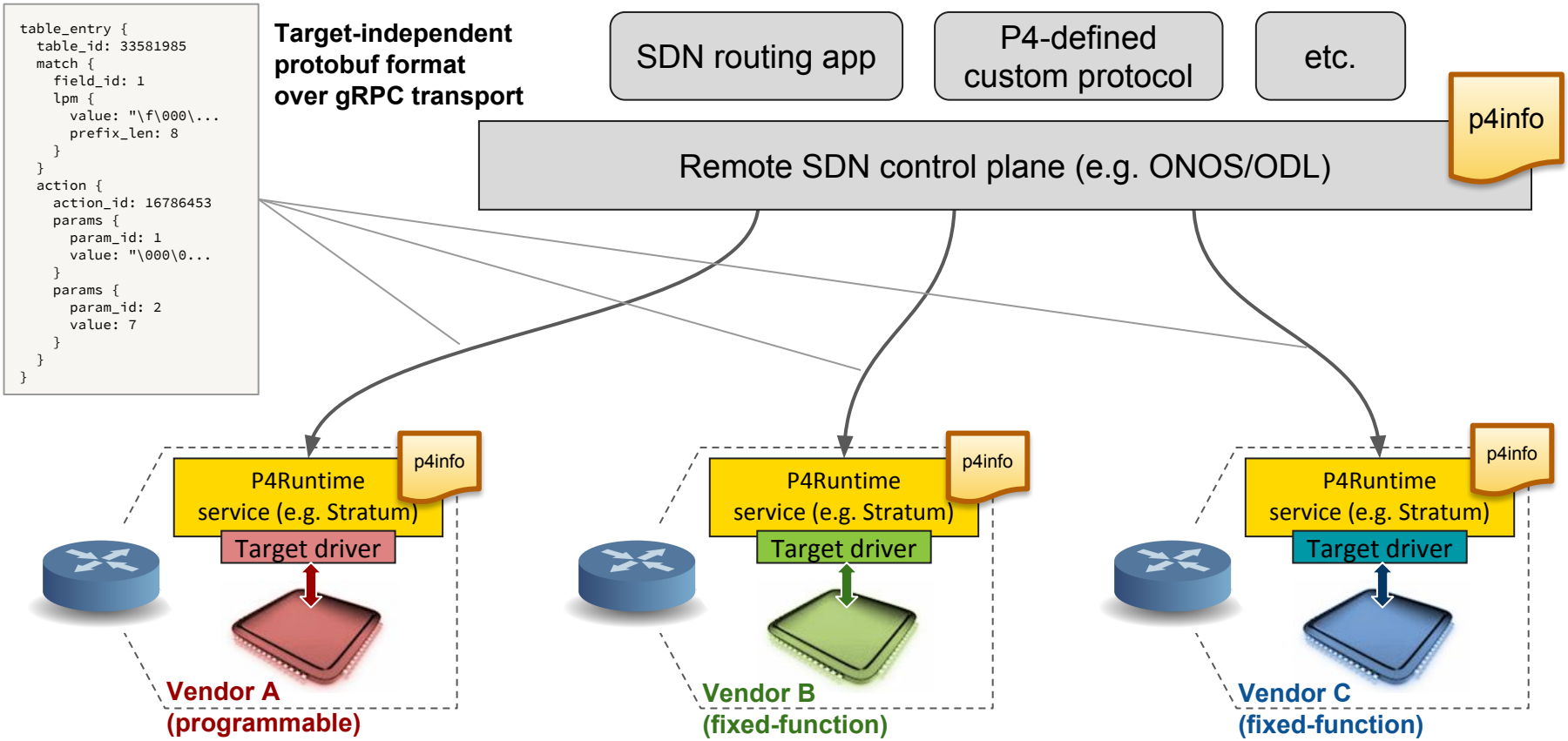
1. Target-specific binaries
   - Used to realize switch pipeline
     (e.g. binary config for ASIC, bitstream for FPGA, etc.)
2. P4Info file
   - Describes “schema” of pipeline for runtime control
     - Captures P4 program attributes such as tables, actions, parameters, etc.
   - Protobuf-based format
   - 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

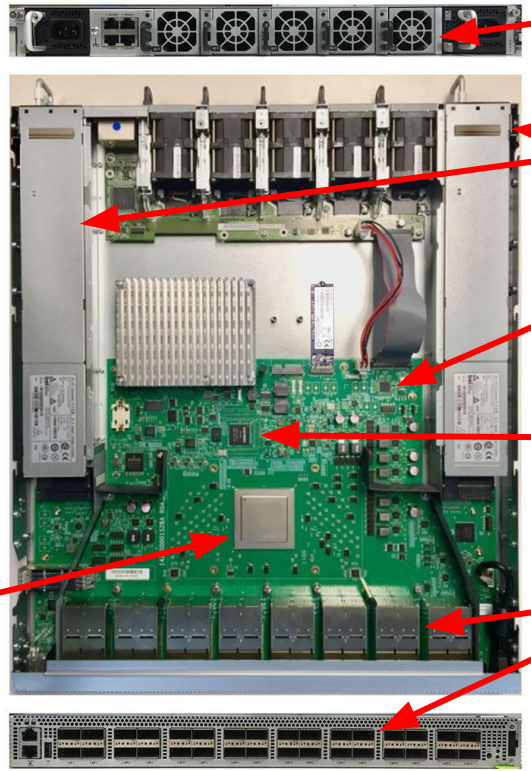
Silicon-independent remote control



Slide courtesy P4.org

OAM Interfaces: gNMI and gNOI

- gNMI for:
  - Configuration
  - Monitoring
  - Telemetry
- gNOI for Operations



Fan Speed

Power supplies

Monitor Sensors
e.g. temperature

Software Deployment and Upgrade

Port State and Mapping
LED Control

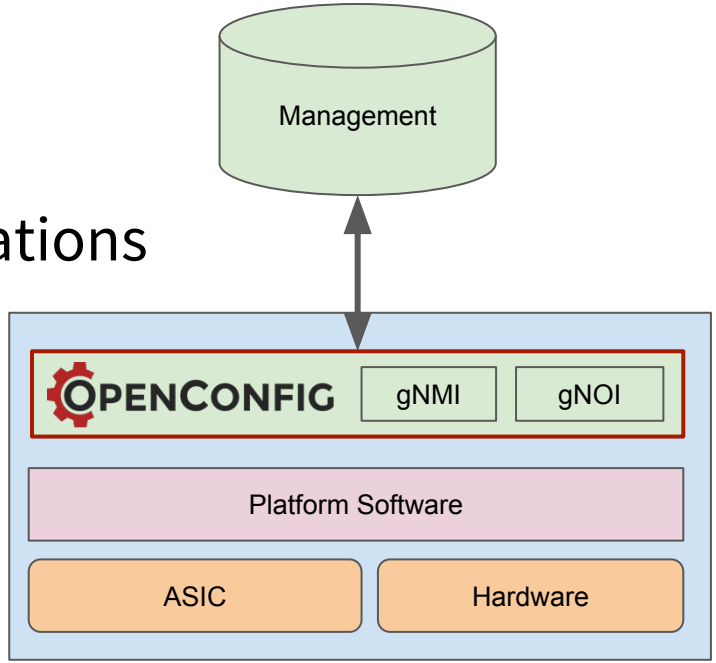
… and the list goes on.

Switch Chip Configuration
QoS Queues and Scheduling
Serialization / Deserialization
Port Channelization

Management Network

Enhanced Configuration

- Configuration and Management
- Declarative configuration
- Streaming telemetry
- Model-driven management and operations
  - gNMI - network management interface
  - gNOI - network operations interface
- Vendor-neutral data models



Augmenting a model


```
module: openconfig-interfaces
  +--rw interfaces
     +--rw interface* [name]
        +--rw config
        |  +--rw name?             string
        |  +--rw type              identityref
        |  +--rw mtu?              uint16
        |  +--rw loopback-mode?    boolean
        |  +--rw description?      string
        |  +--rw enabled?          boolean
        +--ro state
        |  +--ro name?             string
        |  +--ro type              identityref
        |  +--ro mtu?              uint16
        |  +--ro loopback-mode?    boolean
        |  +--ro description?      string
        |  +--ro enabled?          boolean
        |  +--ro ifindex?          uint32
        |  +--ro admin-status      enumeration
        |  +--ro oper-status       enumeration
        |  +--ro last-change?      oc-types:timeticks64
        |  +--ro logical?          boolean
        |  +--ro counters
        |     +--ro in-octets?               oc-yang:counter64
        |     +--ro in-pkts?                 oc-yang:counter64
        |  ...
```

```
augment "/oc-if:interfaces/oc-if:interface/oc-if:config" {
     leaf forwarding-viable {
            type boolean;
            default true;
     }
}
```

```
+--rw forwarding-viable?    boolean
```

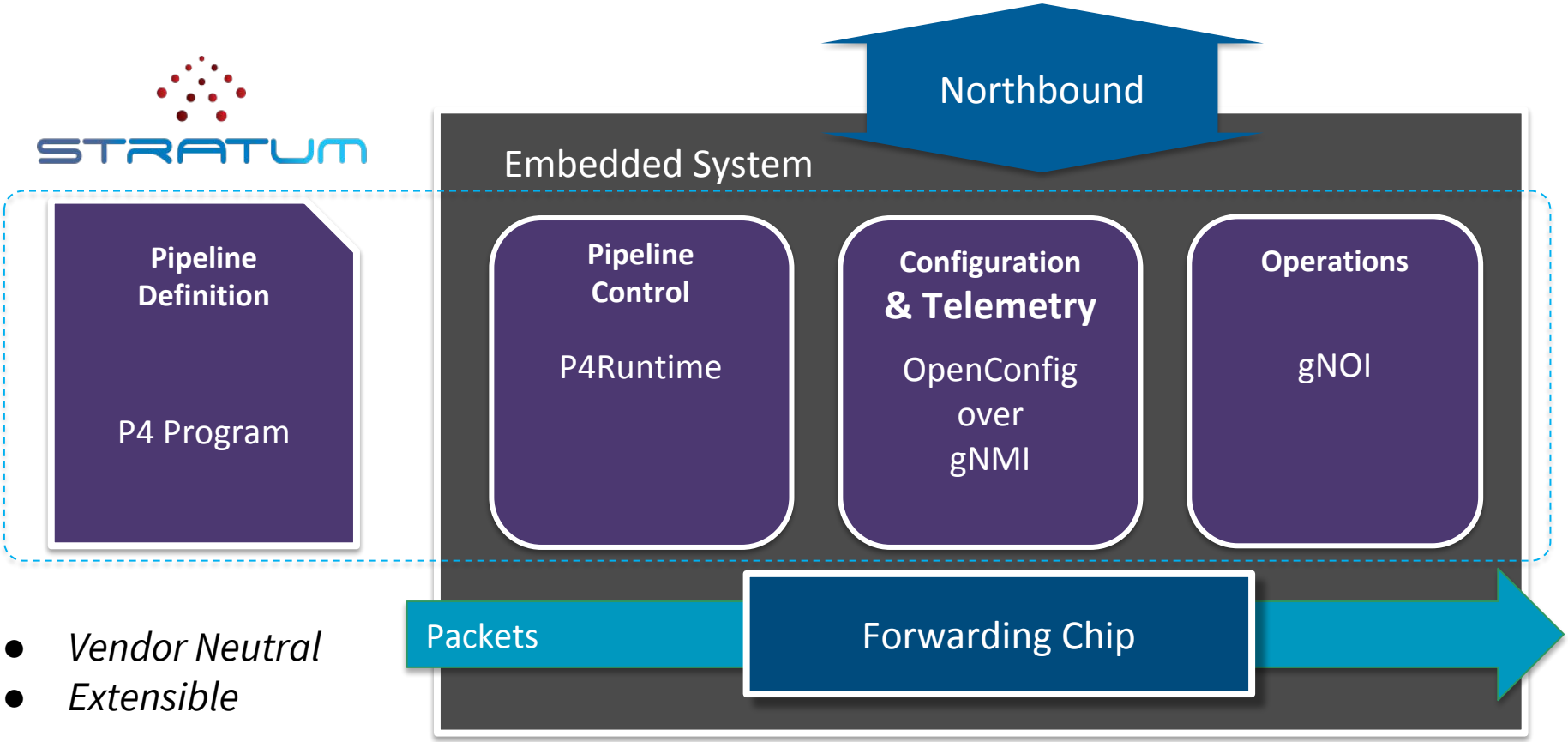
Models are easy to augment, use, and test.

Compile and re-generate topology.

gNOI micro-services

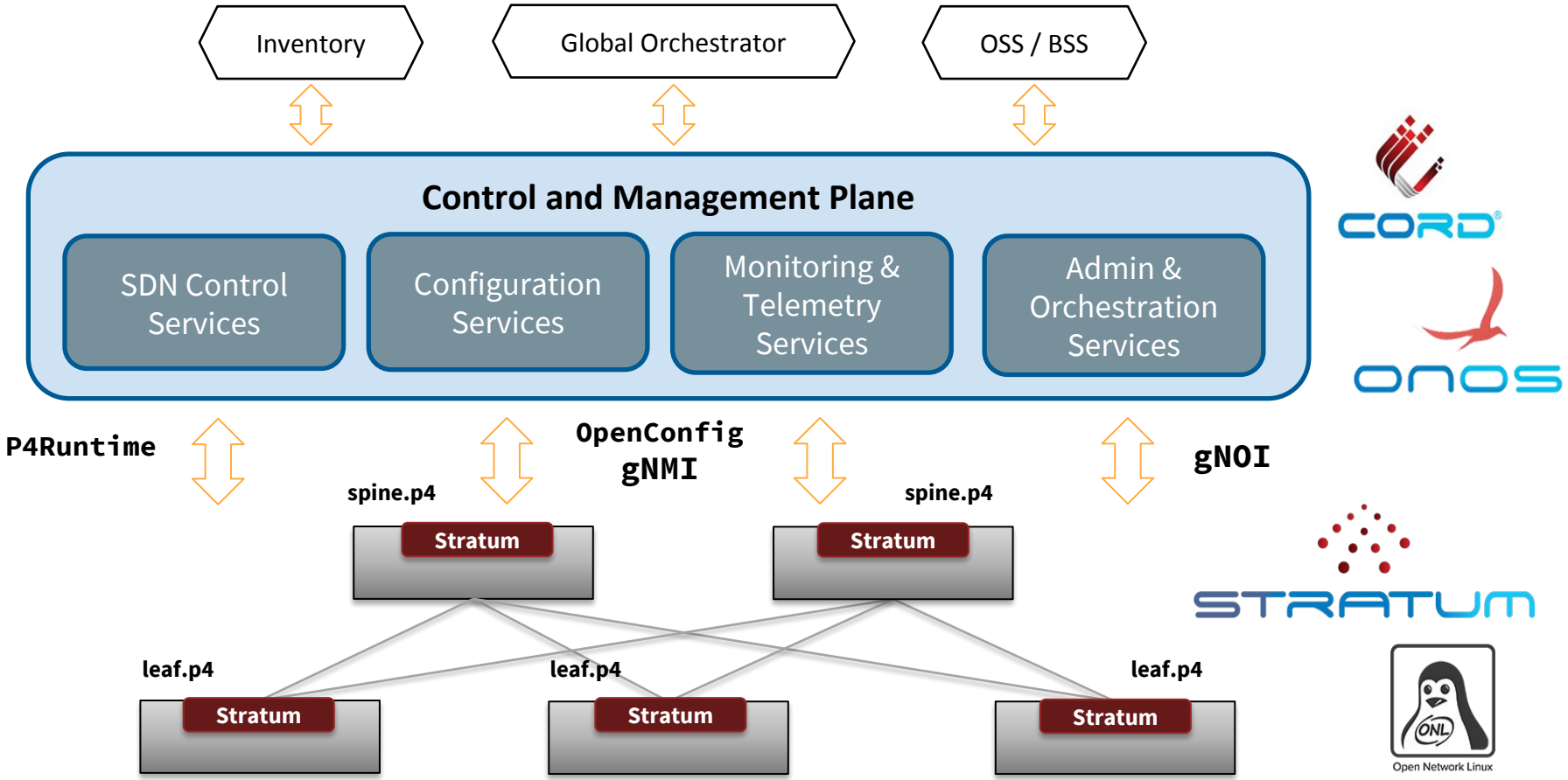
- **`AdminService`**
  - reboot, time, ping, set package…
- **`CertificateManagementService`**
  - rotate, install, revoke certificate
- **`DiagService`**
  - BERT, Burn-in
- **`FileService`**
  - File management

Next Generation SDN Interfaces



- *Vendor Neutral*
- *Extensible*

Next Generation SDN picture



Providing an Implementation: Stratum

Open Interfaces and Models are necessary, **but not sufficient**, for multi-vender interoperability.

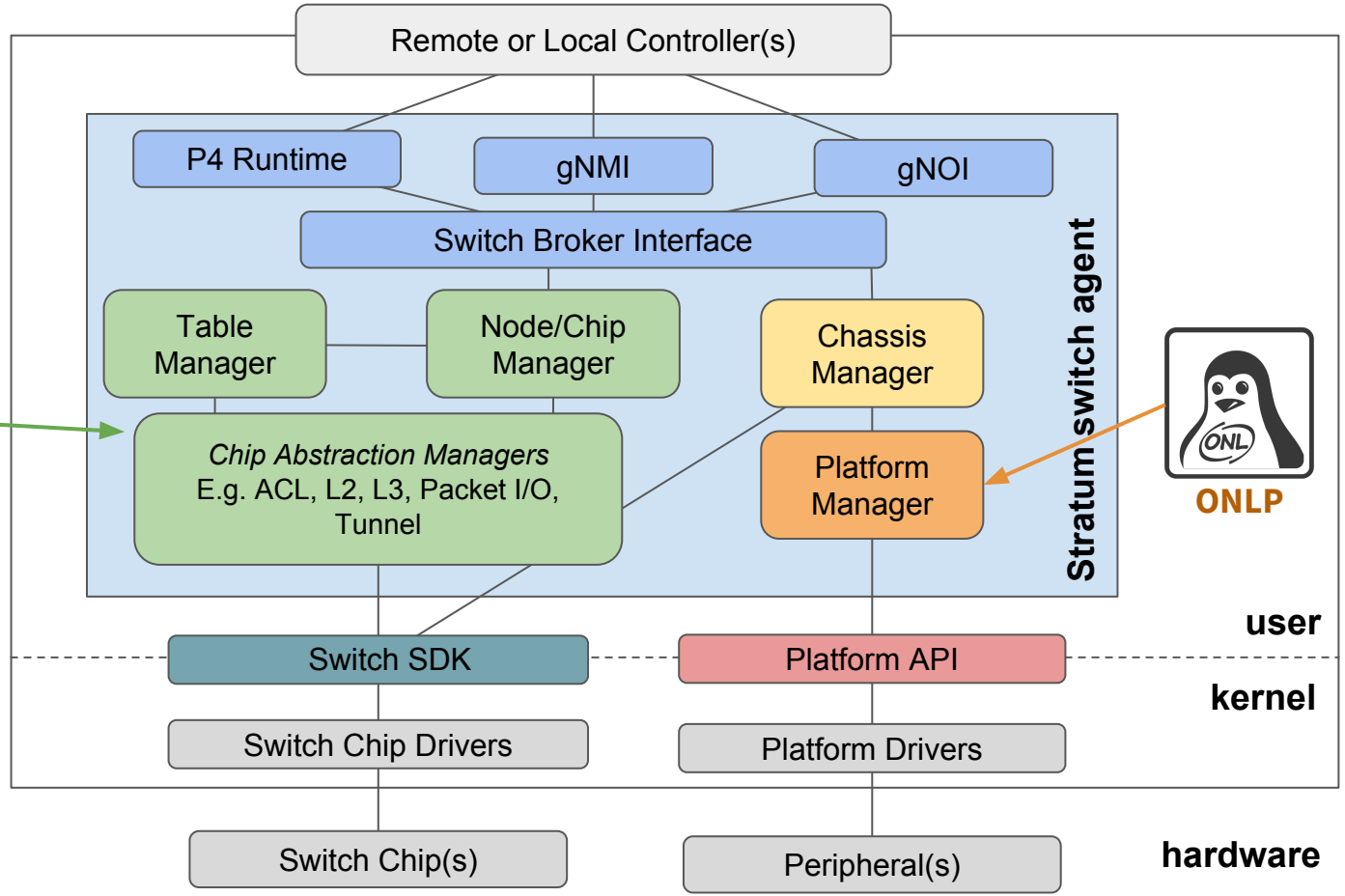
Interfaces are **defined by running code**, so providing an open source implementation helps solidify the interfaces and models. This is not a standards exercise.

If the open source is a fully production ready distribution (ready to run and deploy these interfaces), we can **avoid bugs in different vendor implementations** and improve time to market.

Stratum Design Principles

1. Chip, Platform, and Dataplane independent interfaces
    - Interfaces and architecture are agnostic to Chip, Chassis, Peripheral, Kernel, and P4 Program
2. Generic and common APIs for local and remote control and configuration
    - Enables running control plane on or off the box
3. Lightweight
    - User space, minimal dependencies, easy to deploy, minimal system requirements, no built-in control plane functionality (e.g. BGP)
4. Reusability and extensibility
    - Common interfaces and leverageable reference implementations (“external” switch models like OpenConfig’s, and “internal” component interfaces like Chassis Manager)
    - Flexibility to extend to accommodate chip or platform value-added functionality
    - Favor 3rd party community work when appropriate (ONLP for peripherals)

Stratum High-level Architectural Components



PI and **fpm**-based implementations

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

Stratum Implementation Details

- Implements **P4Runtime**, **gNMI**, and **gNOI** services
- Controlled locally or remotely using **gRPC**
- Written in **C++11**
- Runs as a **Linux** process in user space
- Can be distributed with **ONL**
- Built using **Bazel**

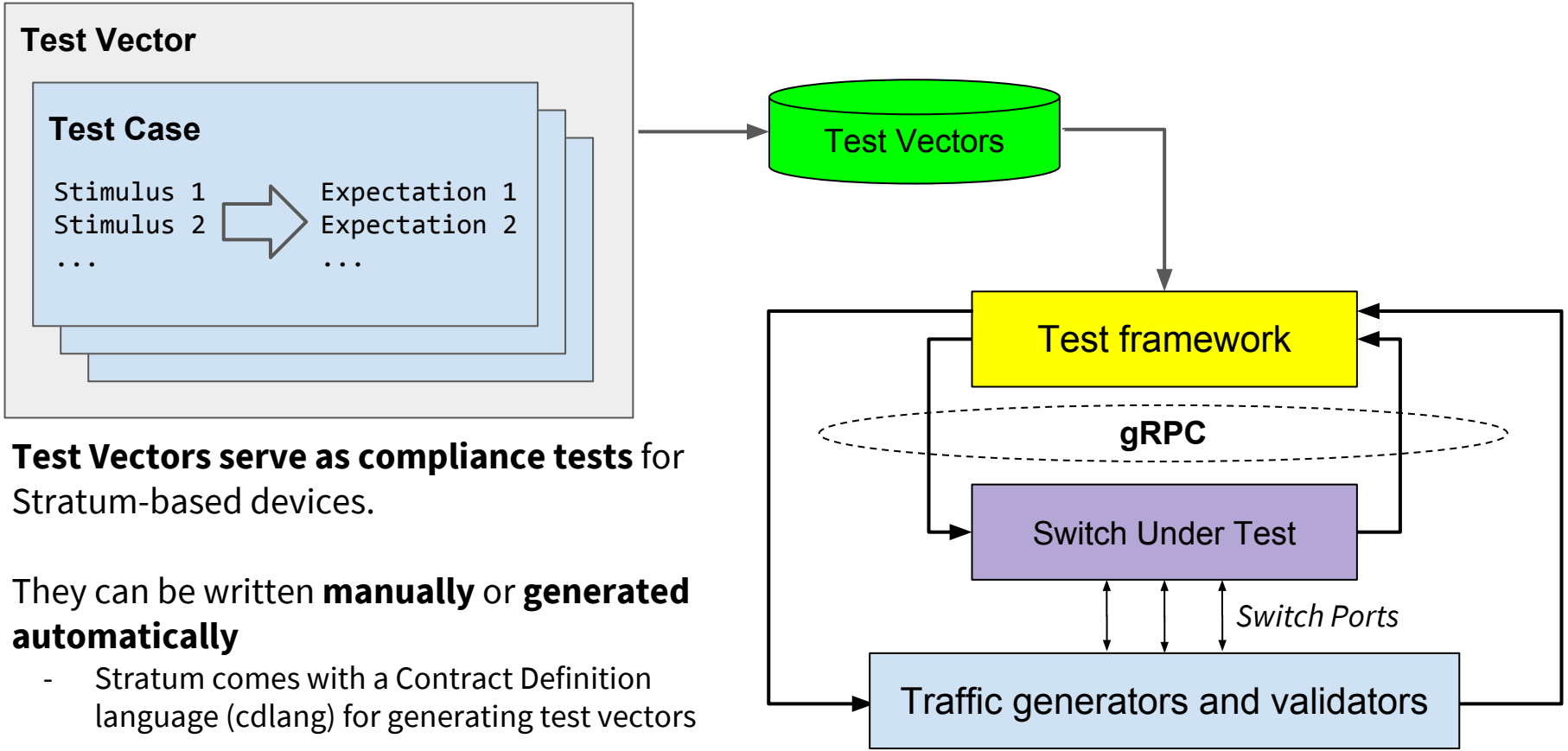
Comprehensive Test Framework

Is an open-source implementation enough for interop?
How to we prevent implementation discrepancies?

There will be other implementations, and they need to be qualified. We also need to make sure that vendor-specific pieces are implemented as expected.

Solution: Provide a **vendor-agnostic, “black box” test framework** for any target that complies with Stratum open APIs (P4Runtime, gNMI, gNOI) along with a **repository of tests.**

Writing Test Vectors

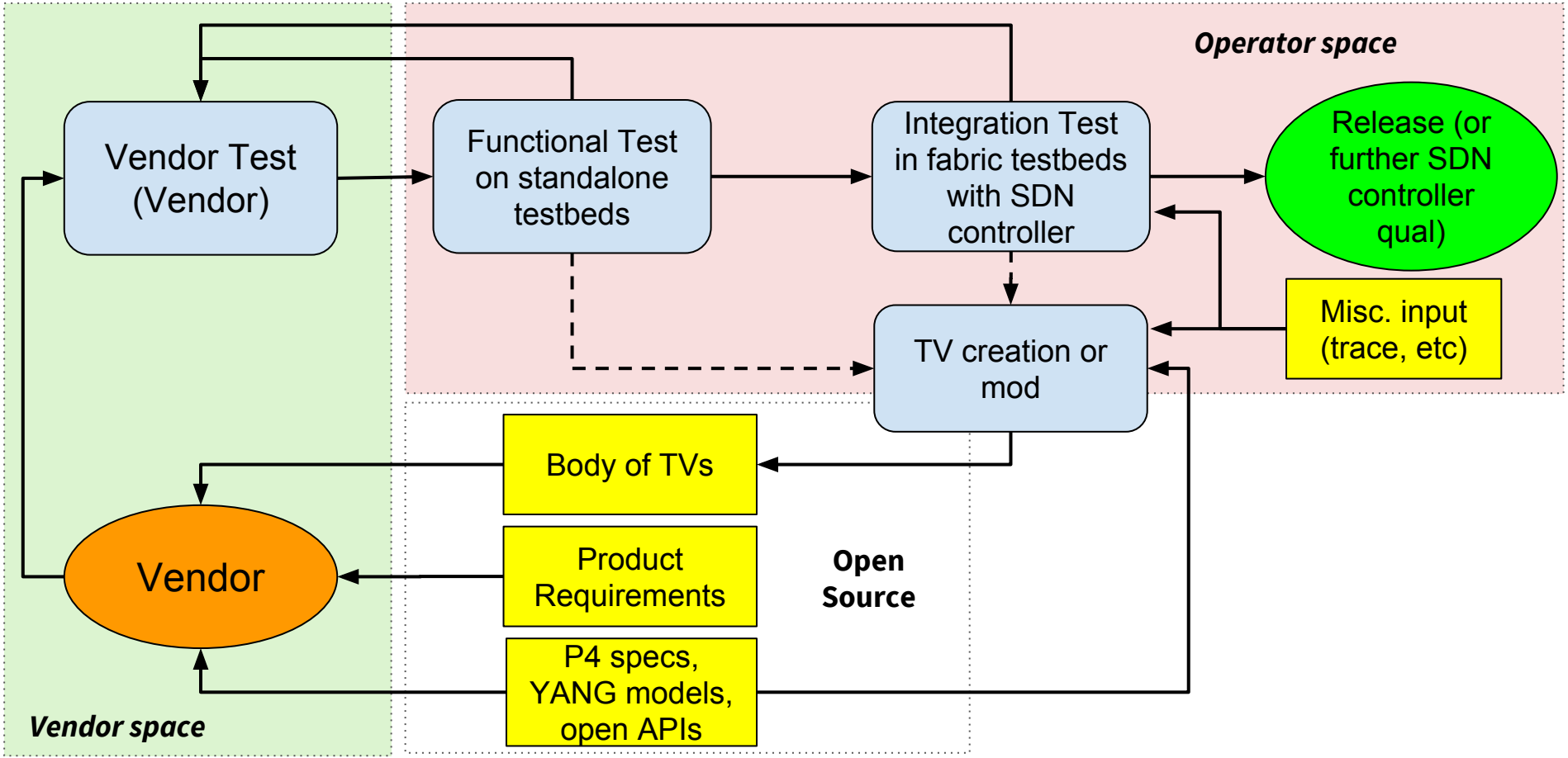


Test Vectors serve as compliance tests for Stratum-based devices.

They can be written **manually** or **generated automatically**

- Stratum comes with a Contract Definition language (cdlang) for generating test vectors

Black Box Qualification



Slide courtesy of Google

Multi-Vendor SDN: Keys to Success

1. Open, vendor-neutral interfaces, models, and pipelines
    - OpenConfig models, P4 programs
    - Interfaces: P4Runtime, gNMI, gNOI
2. Open, vendor-agnostic reference implementation
    - Stratum
3. Open, extensive conformance test framework
    - Test Vectors Framework

Stratum Community

Google

Tencent 腾讯

China unicom 中国联通

kpn

NTT

Türk Telekom

big switch networks

BISDN

Extreme
Connect Beyond the Network

NoviFlow

OSI
One Source Integration

PLVISION
Connectivity of Tomorrow

Ruijie NETWORKS

BAREFOOT NETWORKS

BROADCOM

CISCO

Mellanox TECHNOLOGIES

vmware

XILINX

DATACOM

DELTA

Edge-corE NETWORKS

Inventec

JUNIPER NETWORKS

QCT

STORDIS

ONF
Open Networking Foundation

CORD

ONL
Open Network Linux

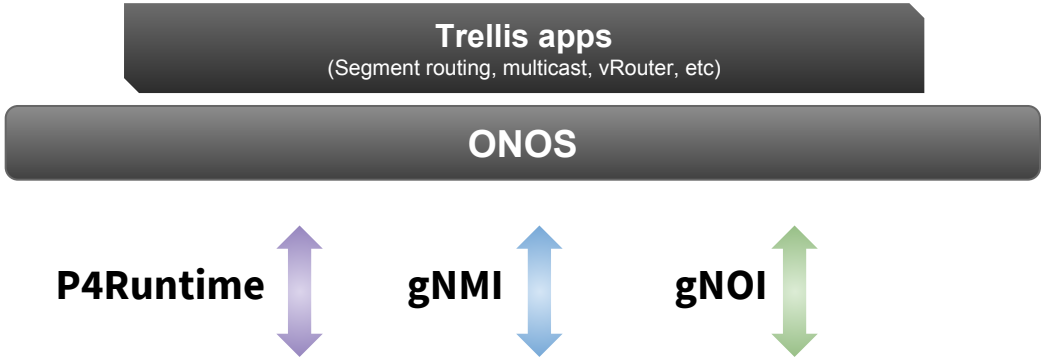
ONOS

OvS
Open vSwitch

P4

SDKLT

Visit the ONF Booth to see Stratum in action!



BAREFOOT NETWORKS

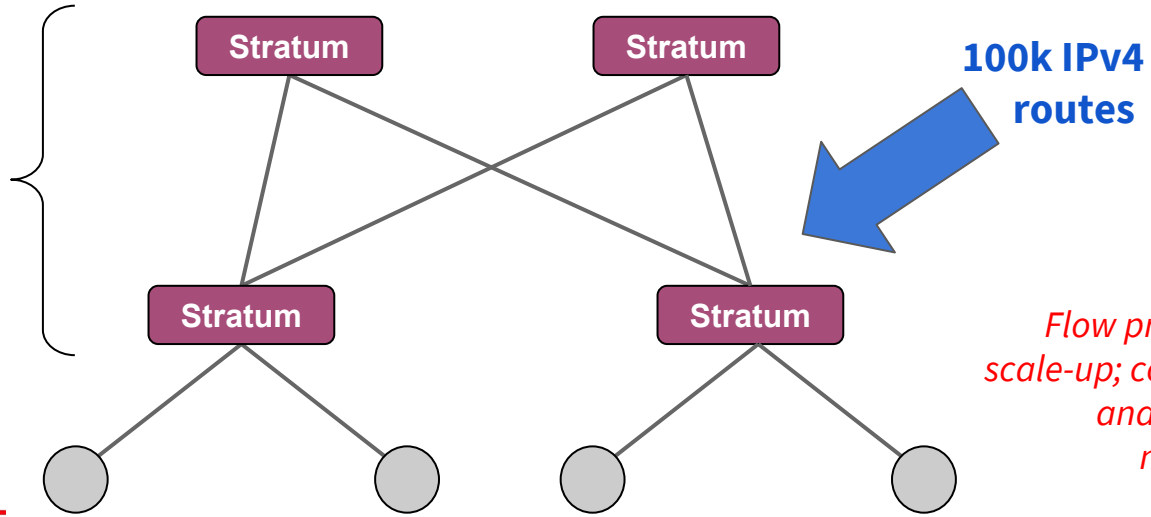
Multi-vendor white-box switches

Edge-Core, Inventec, Delta

DELTA

Edge-corE NETWORKS

Inventec



Flow programming scale-up; configuration and operations; multi-vendor hardware

Getting involved

https://www.opennetworking.org/stratum/

Contribute to the Interfaces and reference P4 programs

- Interfaces and Models: P4Runtime, gNMI, gNOI, and the OpenConfig models
- P4 programs: Fabric.p4, Flex SAI, etc.

Become a Stratum Member

- If you are an employee of a member company, reach out to us for how to get early access

Join the Public Mailing List

- Periodic updates on Stratum's progress.