Stratum
Enabling P4 in the Data Plane

Brian O'Connor
brian@opennetworking.org

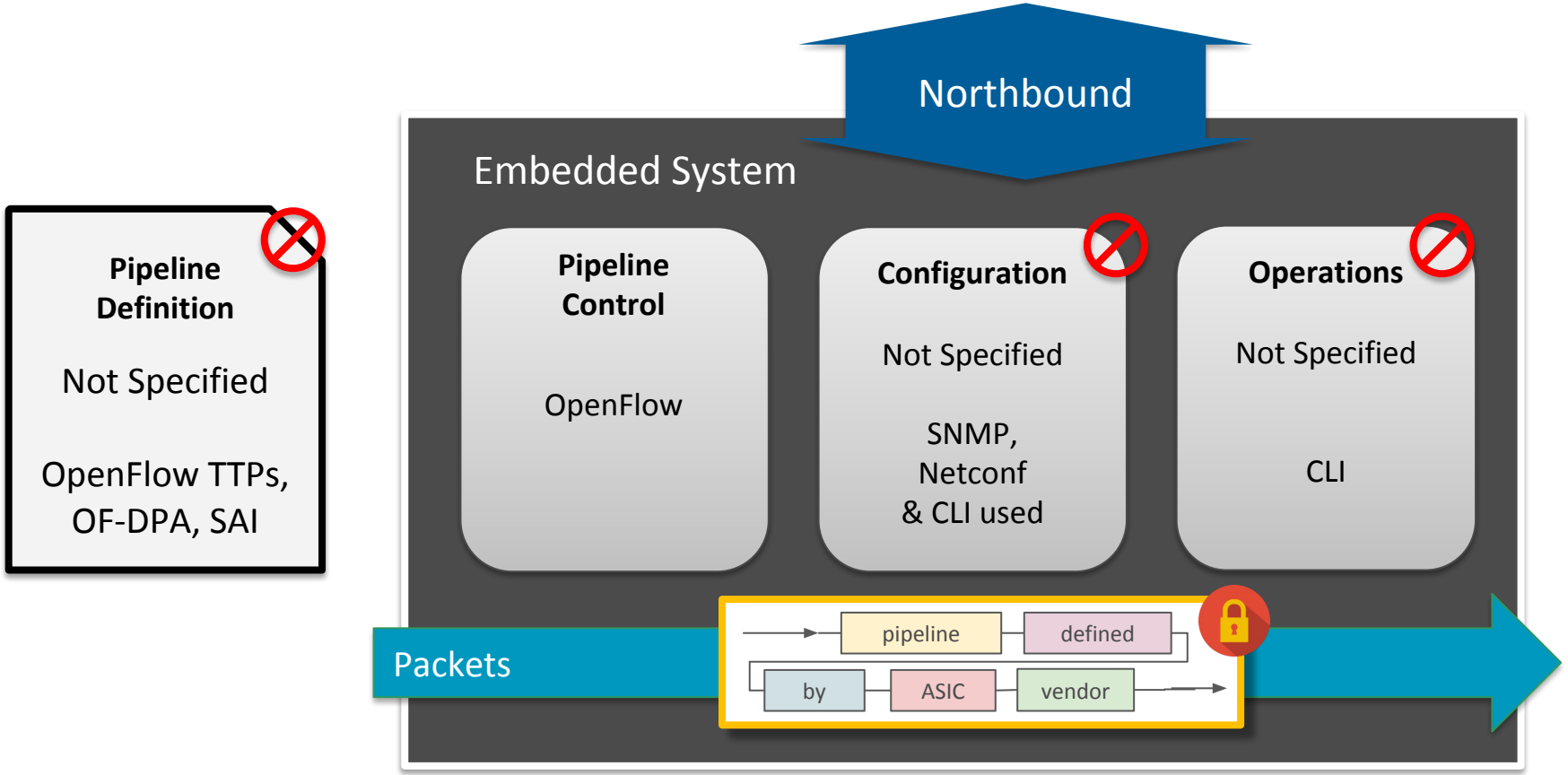
Challenges

Wanted:

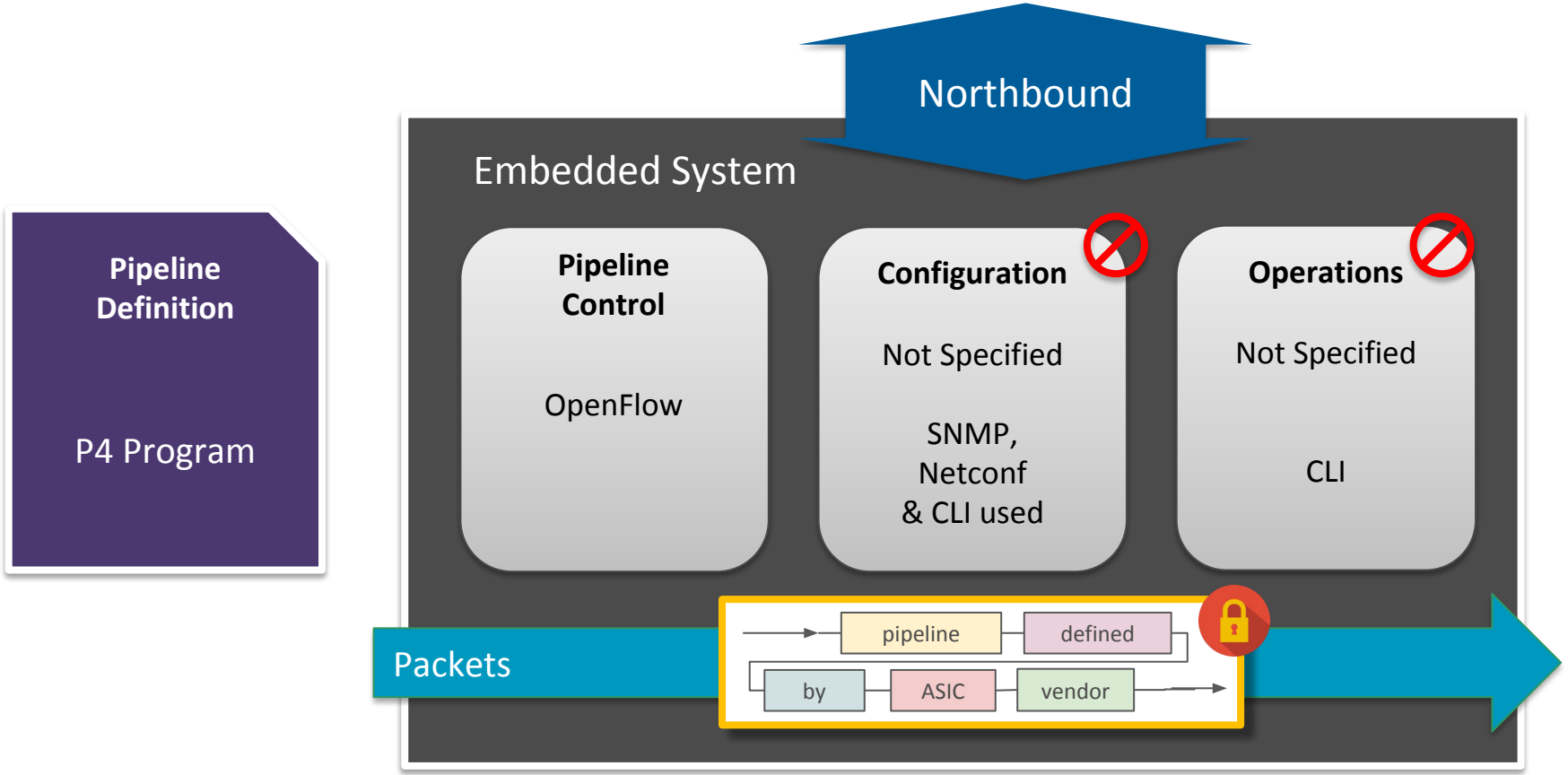
A lightweight, production quality switch agent with a consistent set of interfaces across a wide variety of fixed function and programmable hardware.

- Overcome limitations and gaps with existing control and management protocols
- Handle incremental migration across multiple axes
  - E.g. fixed-function to programmable switching chips, traditional network to SDN
- Enable new generation of programmable devices

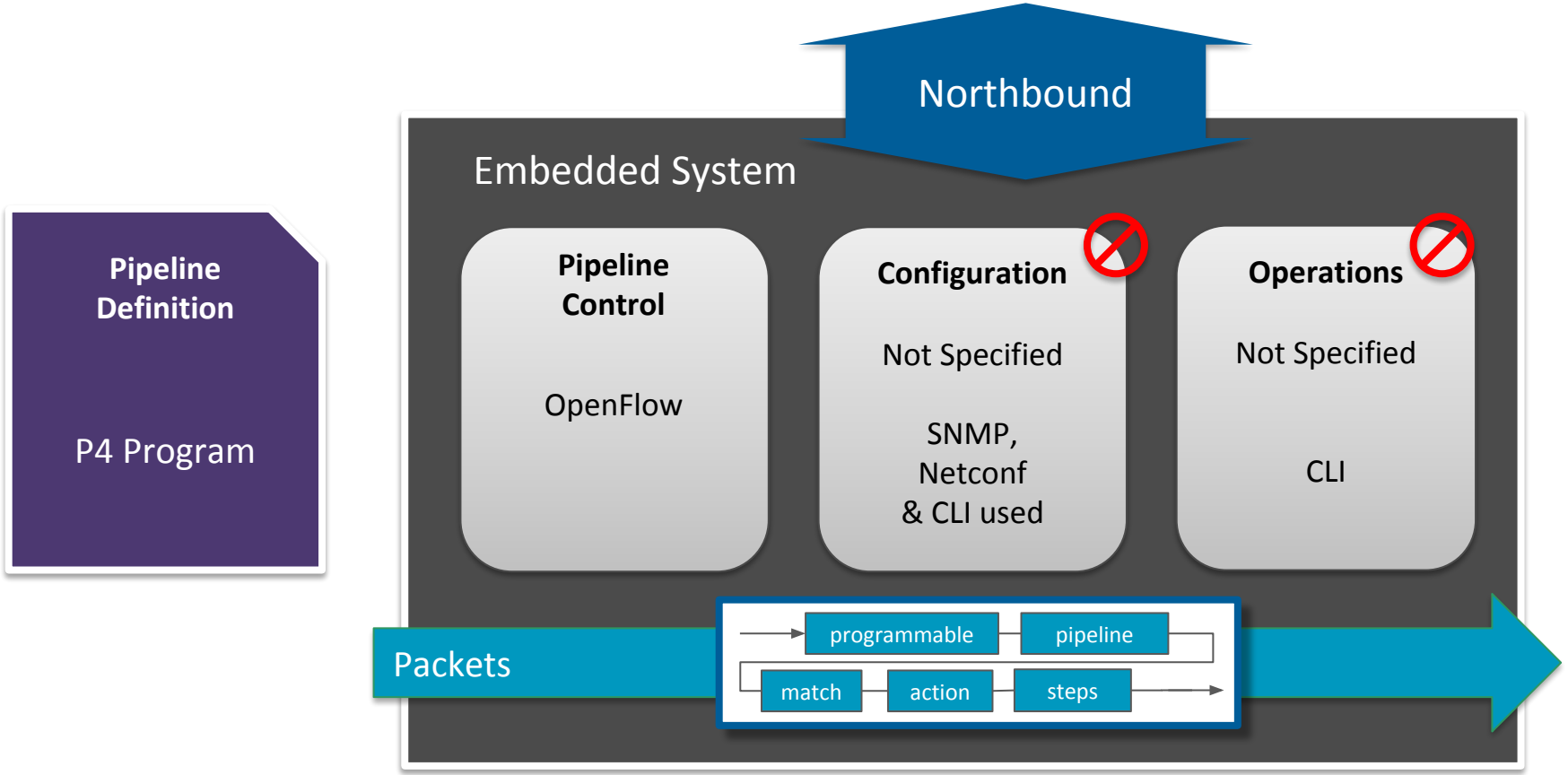
Current SDN Interfaces



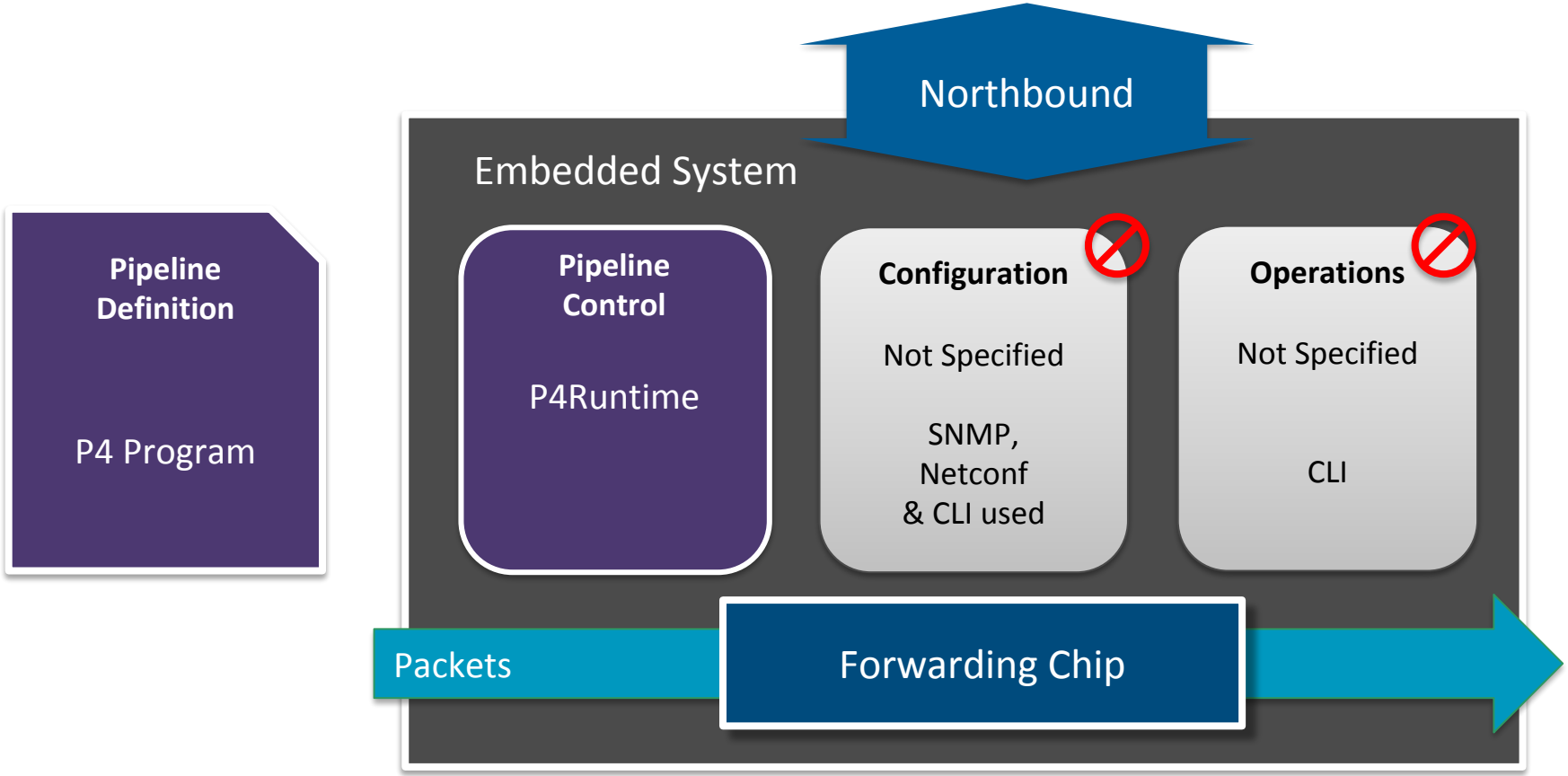
Pipeline Definition Interface



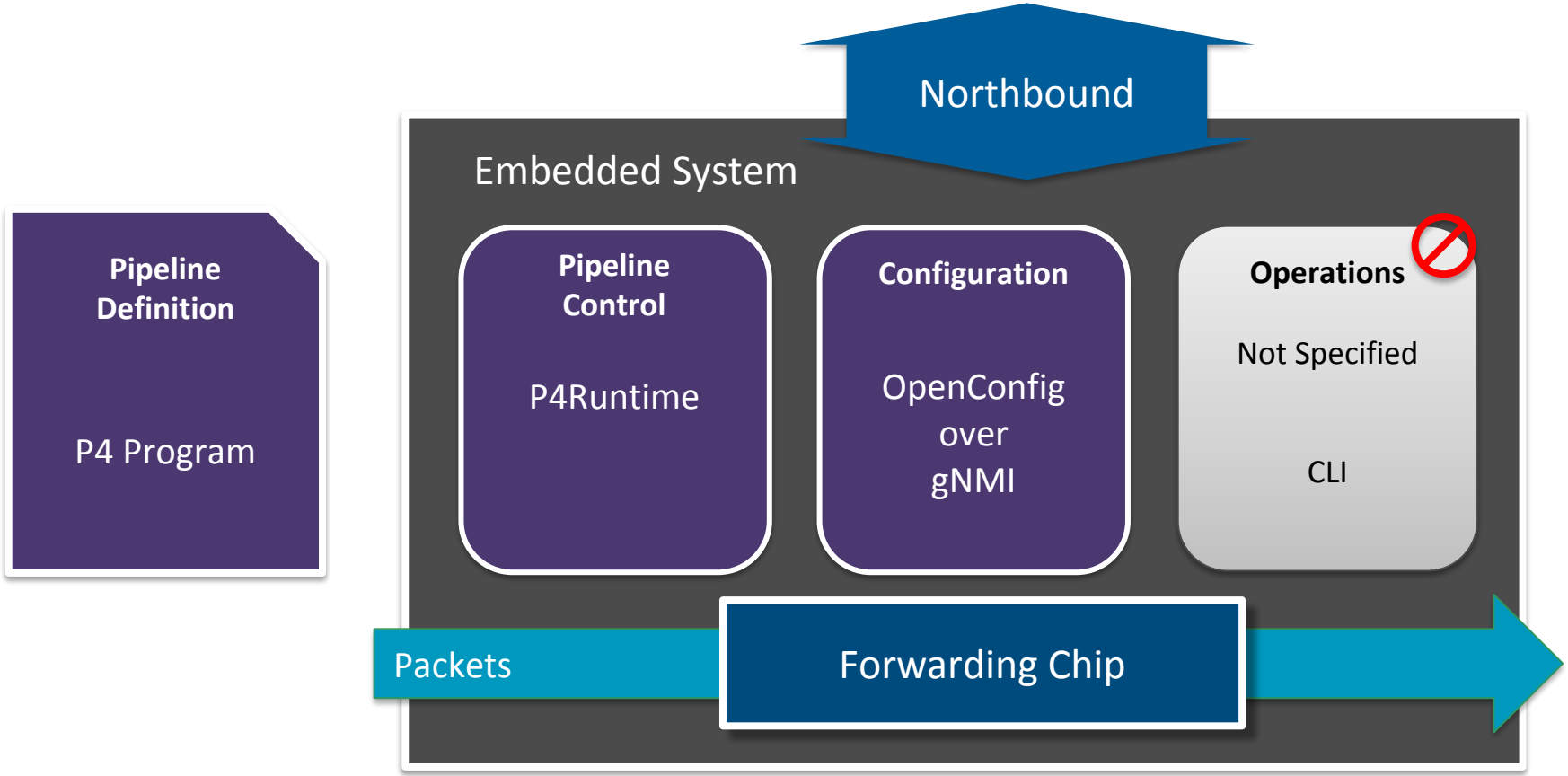
P4 works for Fixed and Programmable Pipelines



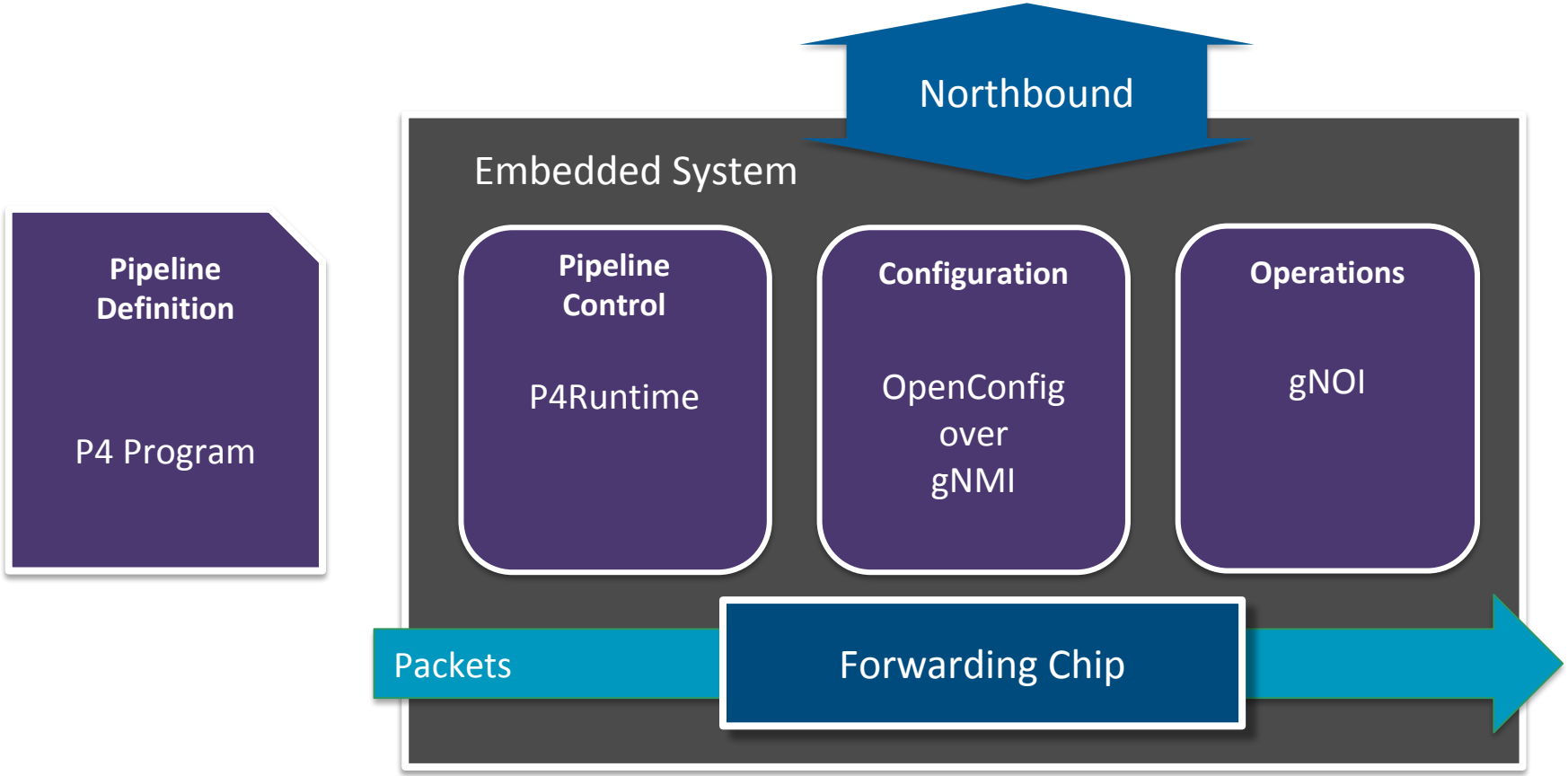
Pipeline Control Interface



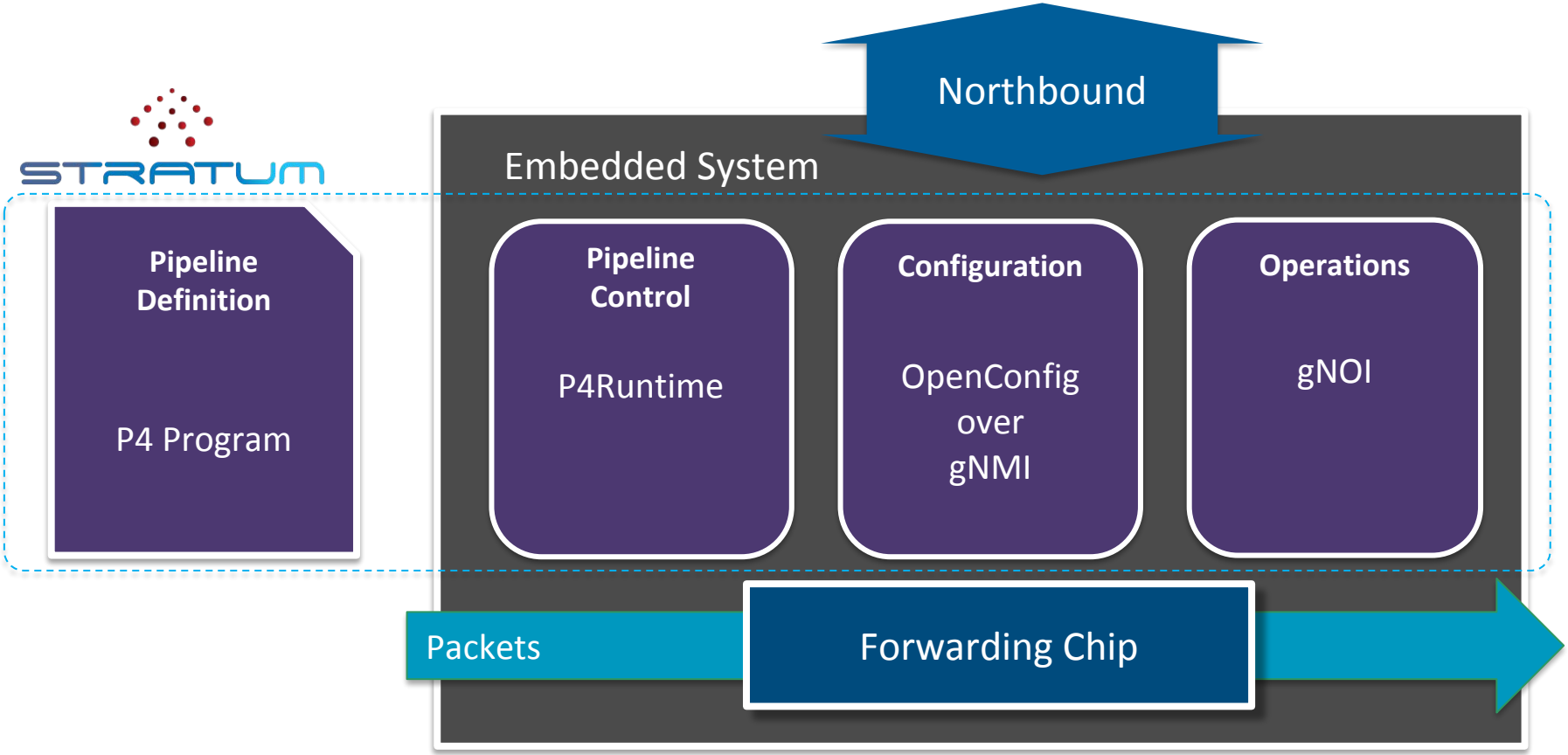
Configuration and Monitoring Interface



Operations Interface



Lightweight and Production-ready Implementation



Stratum Interfaces

External interfaces are drawn from community working groups (i.e. P4 Language Consortium, OpenConfig working group, and gRPC)

- Seamlessly support remote and local interaction
- Not tied to a particular programming language or kernel

Which Models and Programs are supported?

P4 program

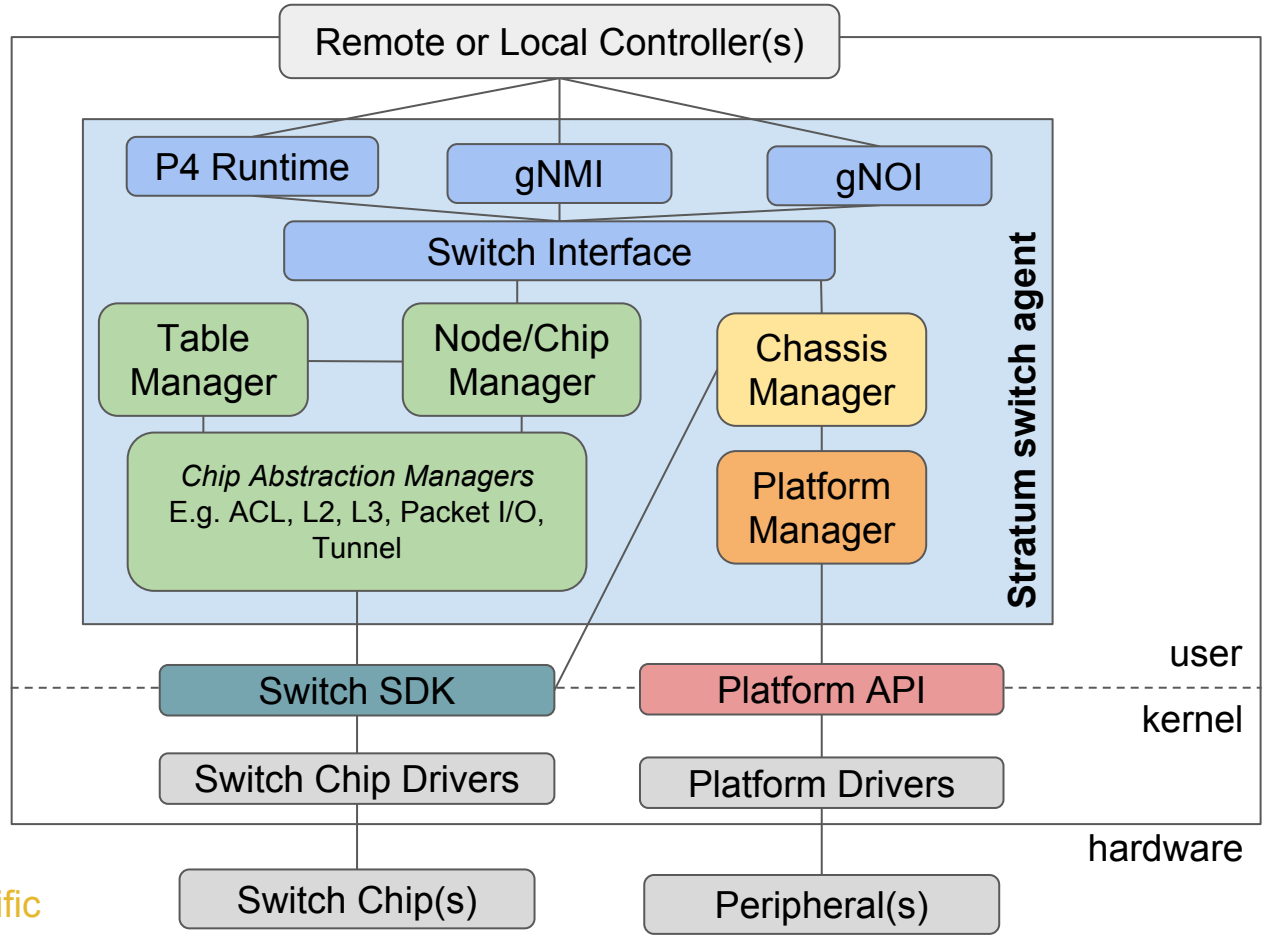
- Stratum is not tied to specific P4 programs
- Depends on what the target (and compiler) will support
- Examples:
  - ONOS fabric.p4
  - SAI.p4

YANG models

- Initial support for a subset of **OpenConfig**
  - *interfaces*, *lacp*, *platform*, *qos*, *vlan*, *alarm*
- Options for other models
  - Augmentations and extensions to OpenConfig
  - In the future, maybe OpenROADM for optical devices

Operations (gNOI) - Initial support for *cert*, *file*, *interface*, *layer2*, *system*

Switch Agent Architectural Components



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

Switch Common Interfaces

- P4Runtime (service definition, documentation)
  - gRPC-based data plane control protocol that is chip-, pipeline-, and packet header-agnostic
  - Message payloads derived from a P4 program using program-dependent P4Info instance to build messages
  - Enables a local or remote entity to load the pipeline/program, arbitrate mastership, read and write forwarding table entries, counters, and other chip features, as well as send and receive packets
- gNMI (service definition, models)
  - gRPC-based service to modify configuration and stream telemetry information
  - Messages payload is modelled in Yang, and Stratum prefers OpenConfig models
  - Config that gNMI deals with tend to be long-lived (i.e. persistent across device reboots), but mutable
- gNOI (service definitions)
  - gRPC-based collection of micro-services for runtime management, for example:
    - Device reboots, pushing/rotating SSL keys/certs, BERT [bit error rate testing on a link/port], ping testing
    - Ephemeral state management (clearing L2 neighbor discovery/spanning tree, resetting a BGP neighbor session)

Service Definitions

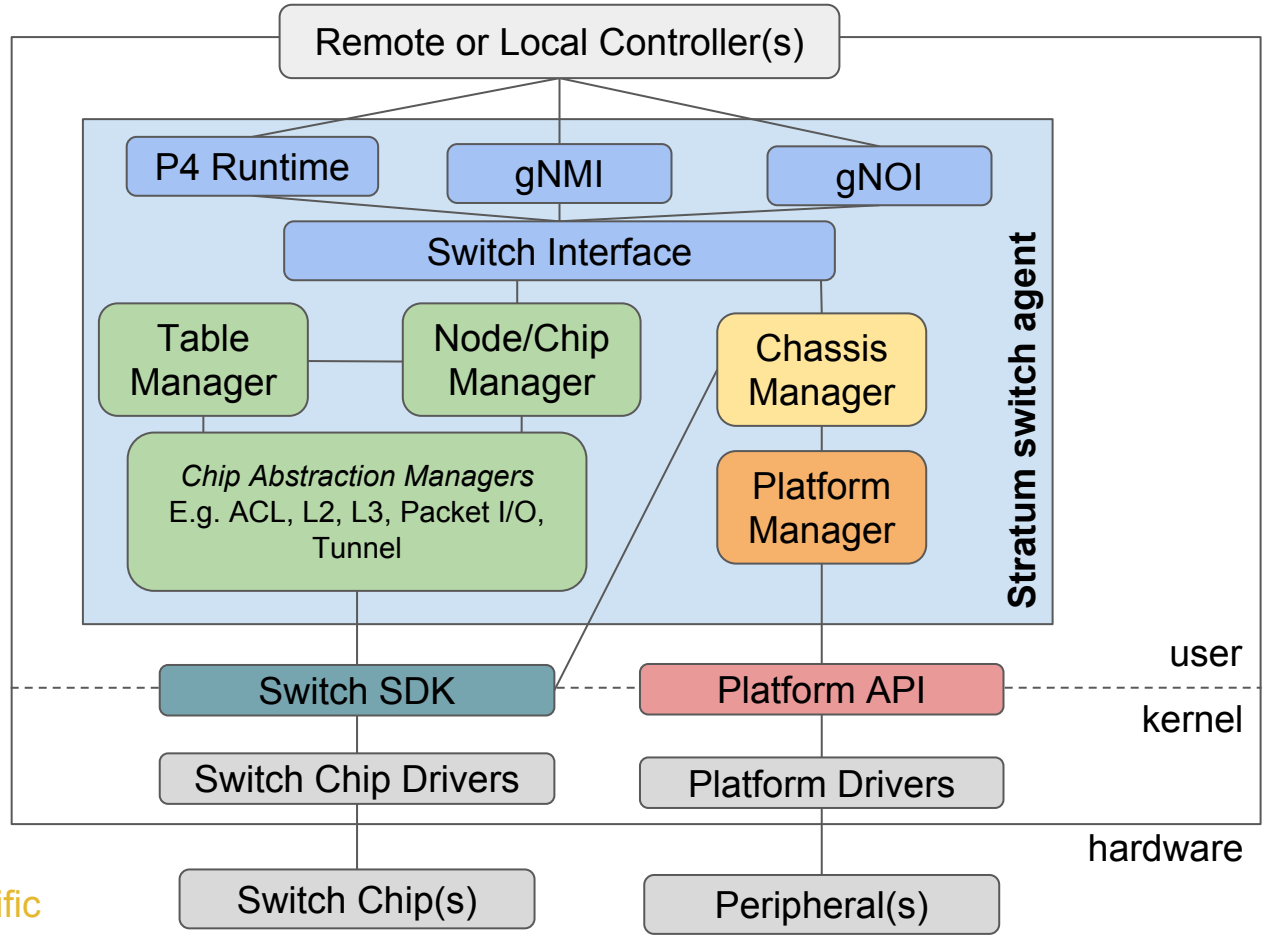
```
service P4Runtime {
  // Update one or more P4 entities on the target.
  rpc Write(WriteRequest) returns (WriteResponse) {
  }
  // Read one or more P4 entities from the target.
  rpc Read(ReadRequest) returns (stream ReadResponse) {
  }

  // Sets the P4 fowarding-pipeline config.
  rpc SetForwardingPipelineConfig(SetForwardingPipelineConfigRequest)
      returns (SetForwardingPipelineConfigResponse) {
  }
  // Gets the current P4 fowarding-pipeline config.
  rpc GetForwardingPipelineConfig(GetForwardingPipelineConfigRequest)
      returns (GetForwardingPipelineConfigResponse) {
  }

  // Represents the bidirectional stream between the controller and the
  // switch (initiated by the controller), and is managed for the following
  // purposes: - connection initiation through master arbitration -
  // indicating switch session liveness: the session is live when switch
  // sends
  //   a positive master arbitration update to the controller, and is
  //   considered dead when either the stream breaks or the switch sends a
  //   negative update for master arbitration
  // - the controller sending/receiving packets to/from the switch
  rpc StreamChannel(stream StreamMessageRequest)
      returns (stream StreamMessageResponse) {
  }
}
```

```
service gNMI {
  // Capabilities allows the client to retrieve the set of capabilities that
  // is supported by the target. This allows the target to validate the
  // service version that is implemented and retrieve the set of models that
  // the target supports. The models can then be specified in subsequent RPCs
  // to restrict the set of data that is utilized.
  // Reference: gNMI Specification Section 3.2
  rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);
  // Retrieve a snapshot of data from the target. A Get RPC requests that the
  // target snapshots a subset of the data tree as specified by the paths
  // included in the message and serializes this to be returned to the
  // client using the specified encoding.
  // Reference: gNMI Specification Section 3.3
  rpc Get(GetRequest) returns (GetResponse);
  // Set allows the client to modify the state of data on the target. The
  // paths to modified along with the new values that the client wishes
  // to set the value to.
  // Reference: gNMI Specification Section 3.4
  rpc Set(SetRequest) returns (SetResponse);
  // Subscribe allows a client to request the target to send it values
  // of particular paths within the data tree. These values may be streamed
  // at a particular cadence (STREAM), sent one off on a long-lived channel
  // (POLL), or sent as a one-off retrieval (ONCE).
  // Reference: gNMI Specification Section 3.5
  rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

Switch Agent Architectural Components

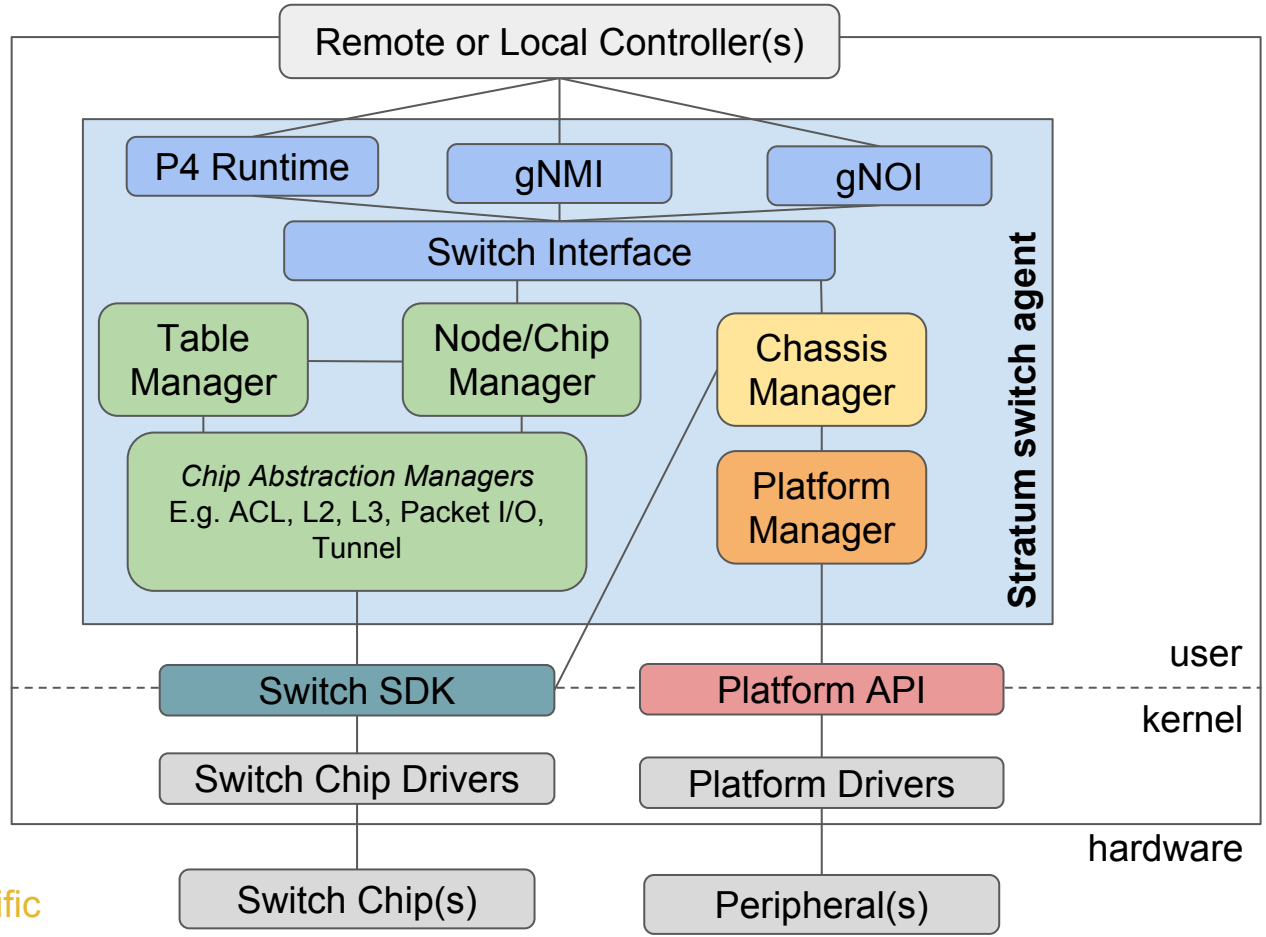


Shared (HW agnostic)
Chip specific
Platform specific
Chip and Platform specific

Forwarding chip-specific Interfaces

- Node / Chip Manager
  - Upon initialization, one instance is registered with `SwitchInterface` per chip
  - Works in conjunction with the Table Manager and Chip Abstraction Managers to provide access to switch chip functionality
- Table Manager
  - Responsible for translating P4 Runtime calls to chip specific entities
- Chip Abstraction Managers
  - Abstracts switch functionality such as L2, L3, ACL, Packet I/O
  - Each of these managers define their own interface, which can potentially be reused across implementations
- Chassis Manager
  - Currently, responsible for chip initialization and dataplane port mapping

Switch Agent Architectural Components



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

Platform-specific Interfaces

- Chassis Manager
  - Provides configuration and access to telemetry from ports and peripherals
  - Responsible for mapping logical (e.g. SDN) ports, physical (e.g. slot, port, channel), and chip specific (e.g. dataplane) ports
- Platform Manager
  - Only class that interacts with platform components
  - Currently, ONLP is proposed as the platform API, so a common implementation of this manager could be used across any ONLP compliant boxes

Implementation and Deployment

- Both remote controllers and local control planes/agents connect through common gRPC interfaces
- A distribution is realized by defining a target, or collection of Node and Chassis Managers, and building a target specific binary
- A P4 Program is pushed to each device after device initialization and is not part of the distribution
  - Relies on a target specific P4 compiler to produce binaries
- A Distribution deployed as a user-space process
- The agent process's lifecycle is managed by a process manager

Design Principles

1. Chip, Platform, and Dataplane independent interfaces

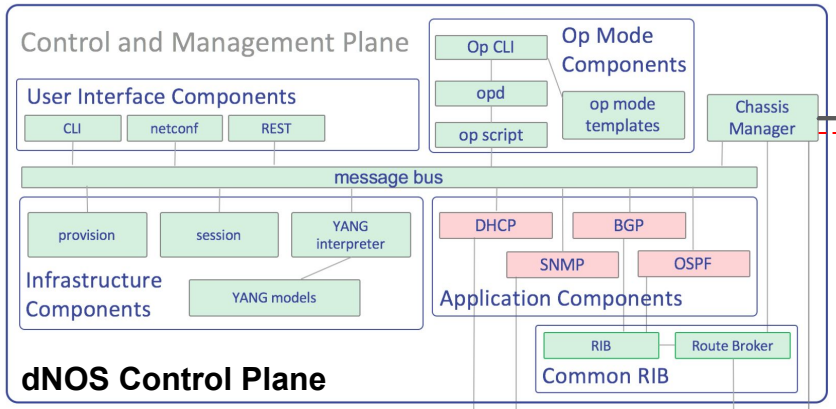
2. Generic and common APIs for local and remote control and configuration

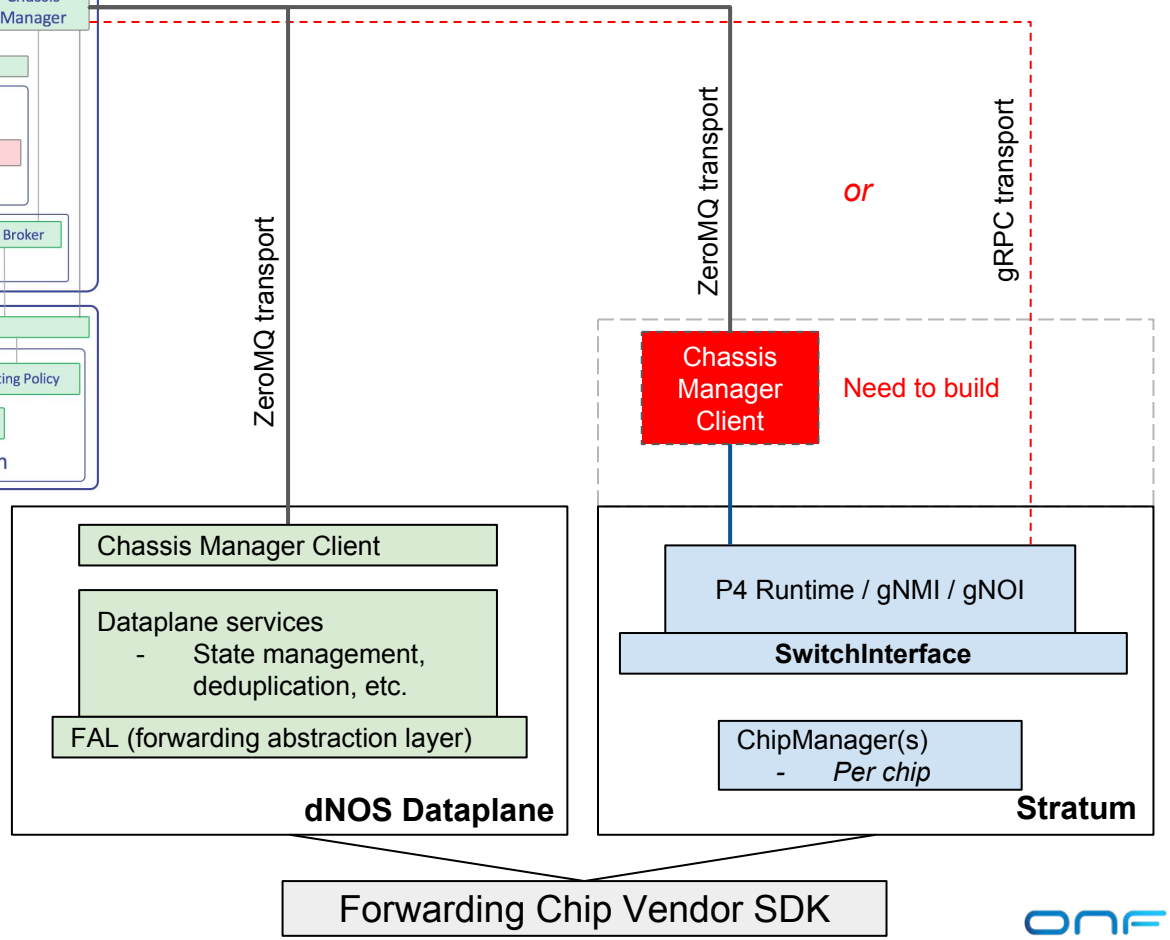
3. Lightweight
   - User space, minimal dependencies, easy to deploy, minimal system requirements, no built-in control plane functionality (e.g. BGP)

4. Reusability and extensibility
   - Common interfaces both internally and externally
   - Flexibility to extend to accommodate chip or platform value-added functionality
   - Favor 3rd party community work when appropriate (e.g. ONLP for peripherals)

Integration and Network Transformation

dNOS + Stratum





Stratum as a parallel dataplane for dNOS

Chassis Manager serves as `netlink`/Stratum mapper

- Can implement gRPC services directly
- Or, can build client that runs on dataplane to map ZeroMQ to gRPC services

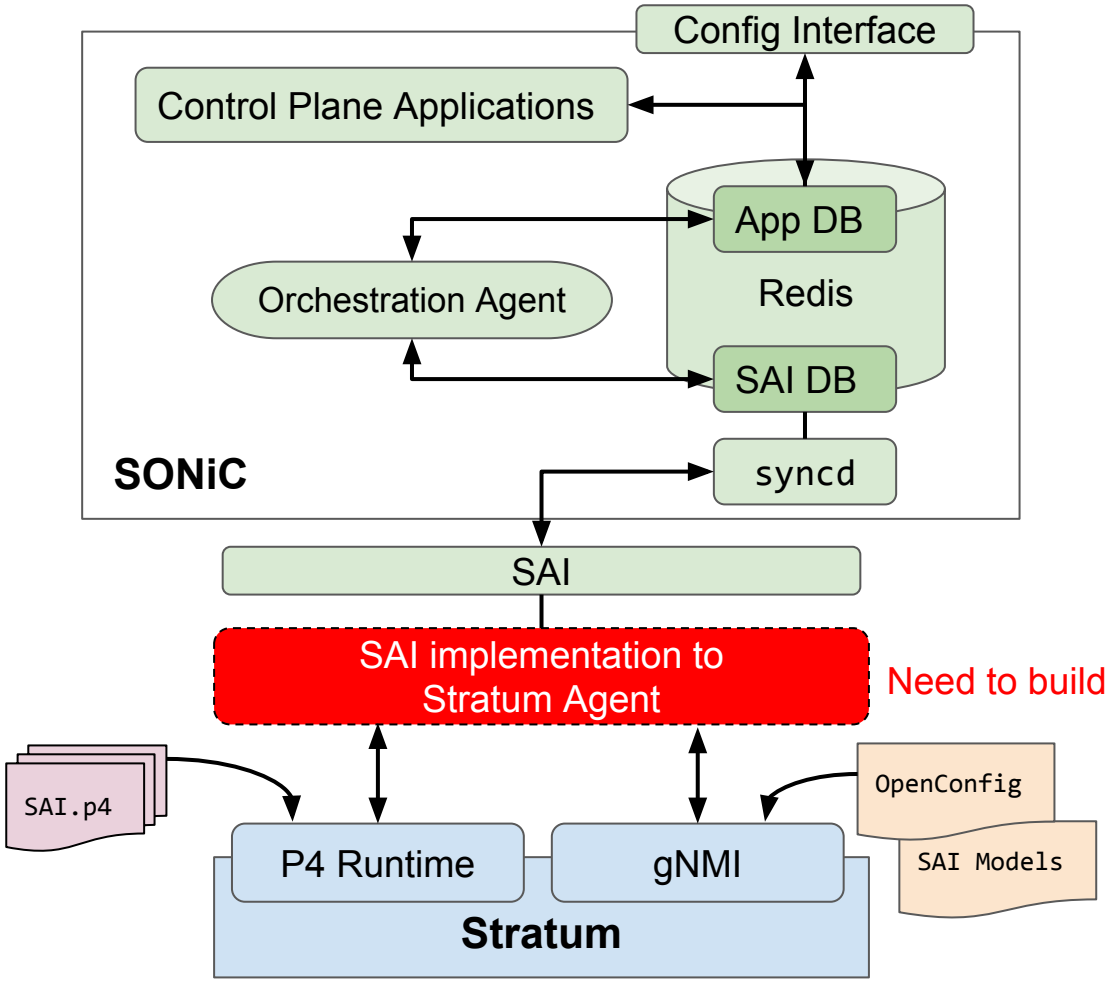
SONiC + Stratum

SONiC uses Stratum as implementation of SAI

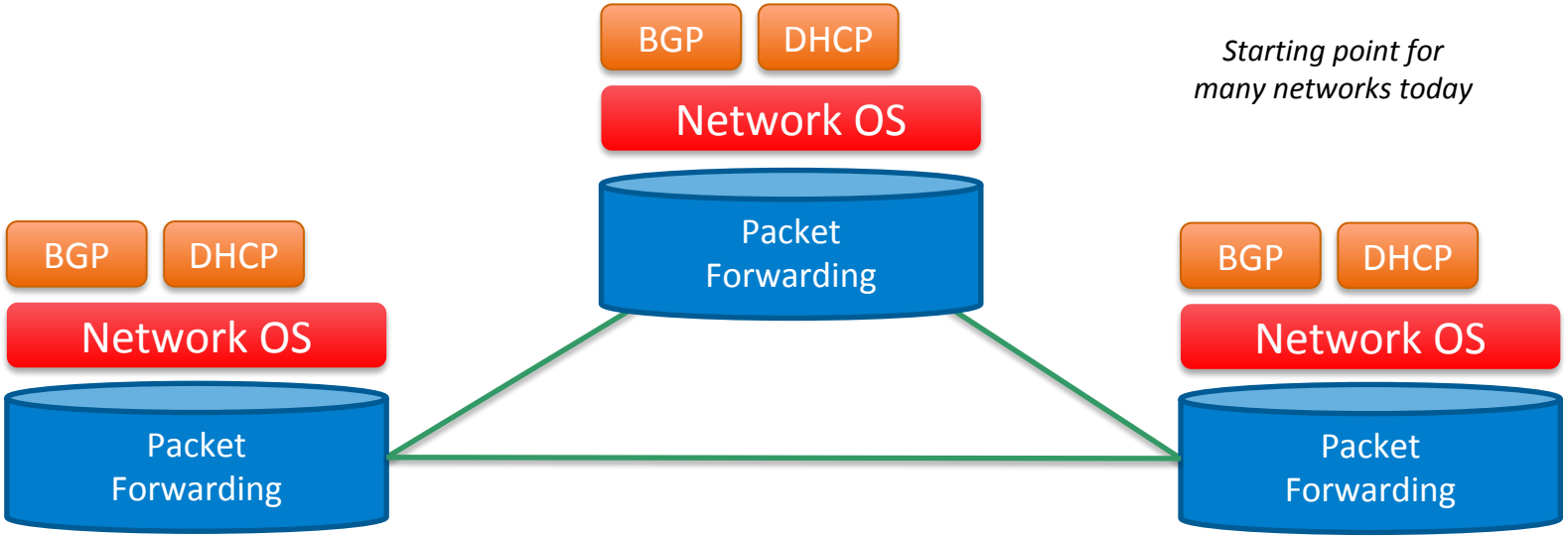
SAI.p4, OpenConfig and maybe additional SAI models used as dataplane contract

Customer specific extensions can be exposed to control plane apps via SAI DB or P4 Runtime interface

Mellanox has a Thrift-based adapter to BMv2; useful for reference for P4 Runtime/gNMI

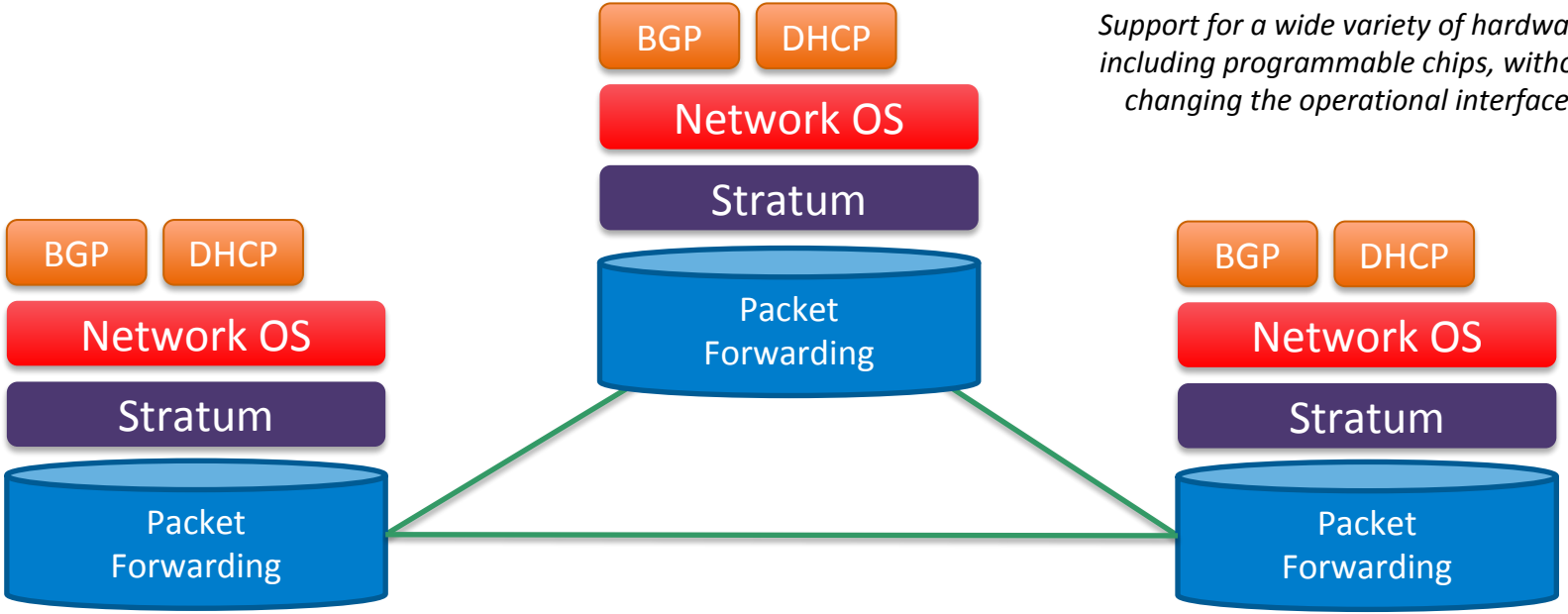


Use Case: Network Transformation with Stratum



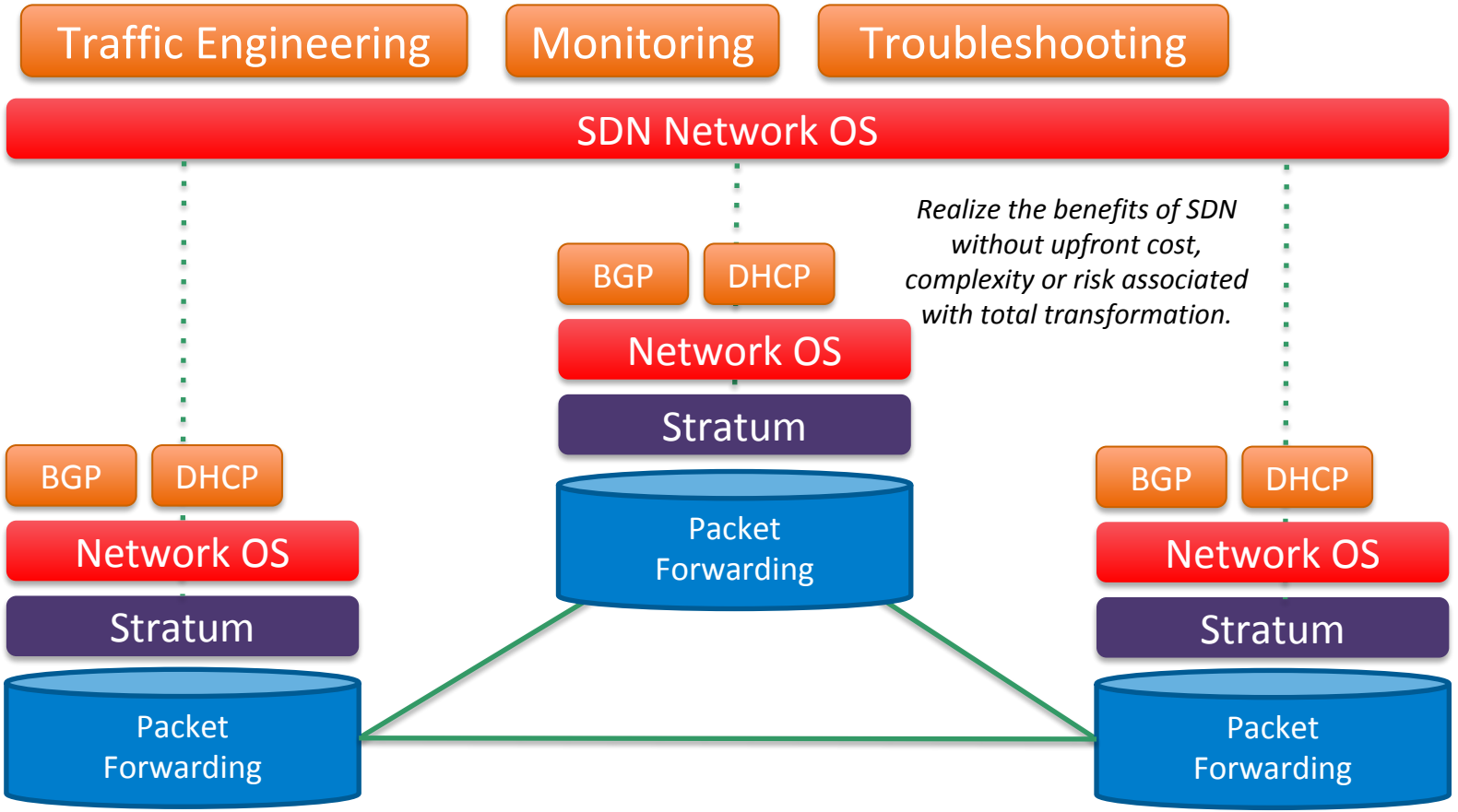
Starting point for many networks today

Upgrade to a Stratum-powered NOS

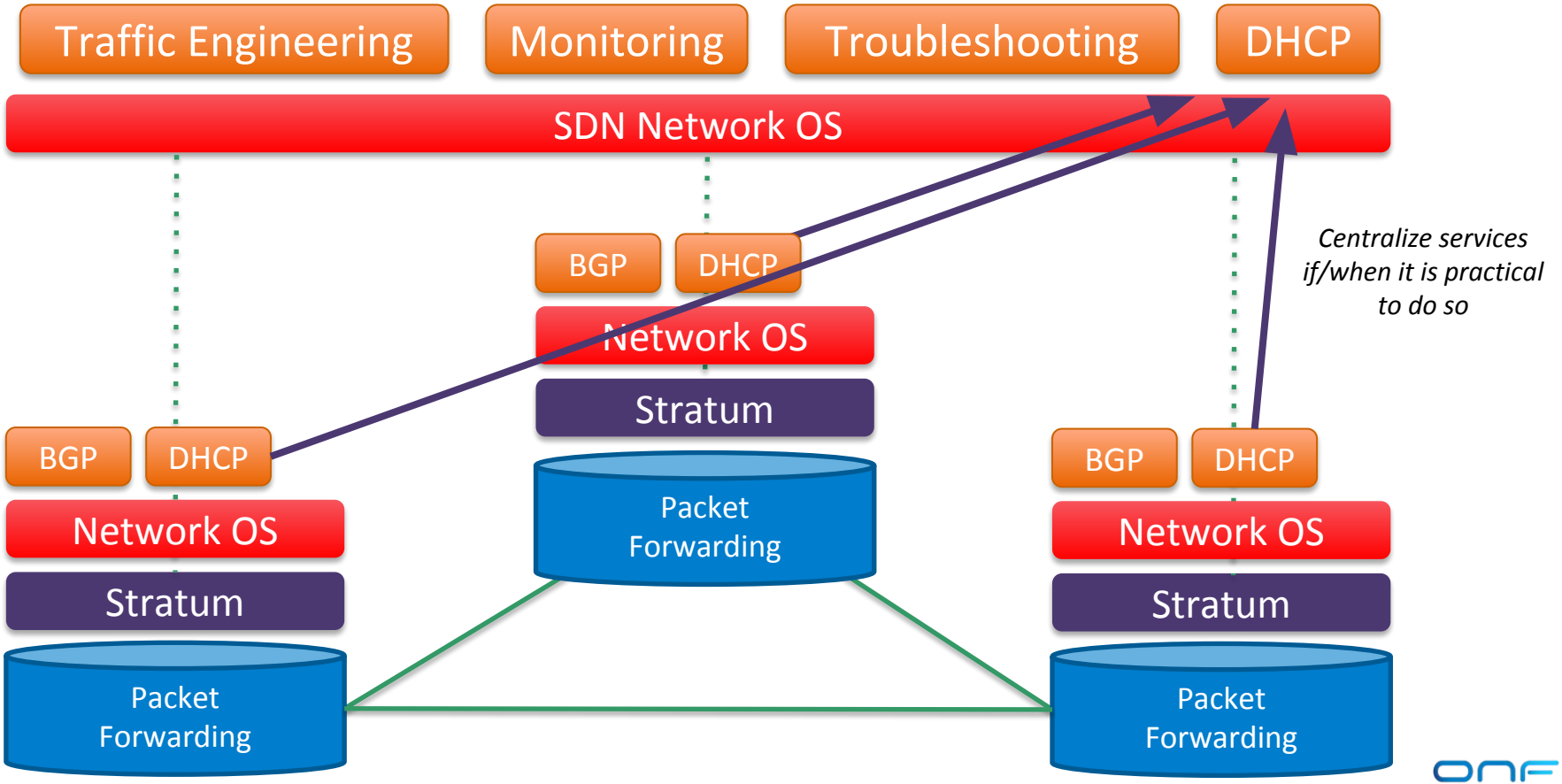


Support for a wide variety of hardware, including programmable chips, without changing the operational interface.

Add an SDN OS and new services



Migrate existing services



Cloud Providers

Telecom Operators

Networking Vendors

White Box ODM Vendors

Silicon Vendors

Other Open Source Projects

Stratum Development Timeline

Stratum Community Launch
with 22 partners

Seed code from Google available to pioneering members (estimate)

Pioneer work days
- Reference Platform Support (HW & SW)
- Development Infrastructure (Build, CI, etc.)

Field Trials, Production Deployments
on cloud and telco networks

Open Source Launch
with forwarding chip and platform support for every vendor member

Community Development

Codebase GA for Stratum Members
- Expanded platform support
- Feature development
- Hackathons

Stratum Summary

- Common interfaces for control, configuration, monitoring and telemetry
- Minimal design for high performance local or remote control and management
- Incremental migration paths enables incremental value-add
  (e.g. SDN, programmable hardware)
- Broad switching chip and platform support underway
- Production-root implementation designed to scale

https://stratumproject.org/

To become project member, or to join the announcement mailing list:
https://wiki.opennetworking.org/display/COM/Stratum+Wiki+Home+Page