ONOS
Open Network Operating System

Architecture, Abstractions & Performance

ONOS Day
September 15th, 2015

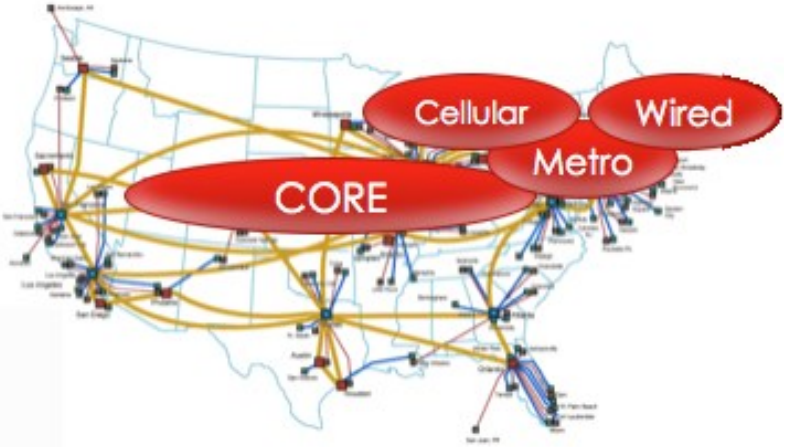
Thomas Vachuska & Ali Al-Shabibi

Architectural Tenets

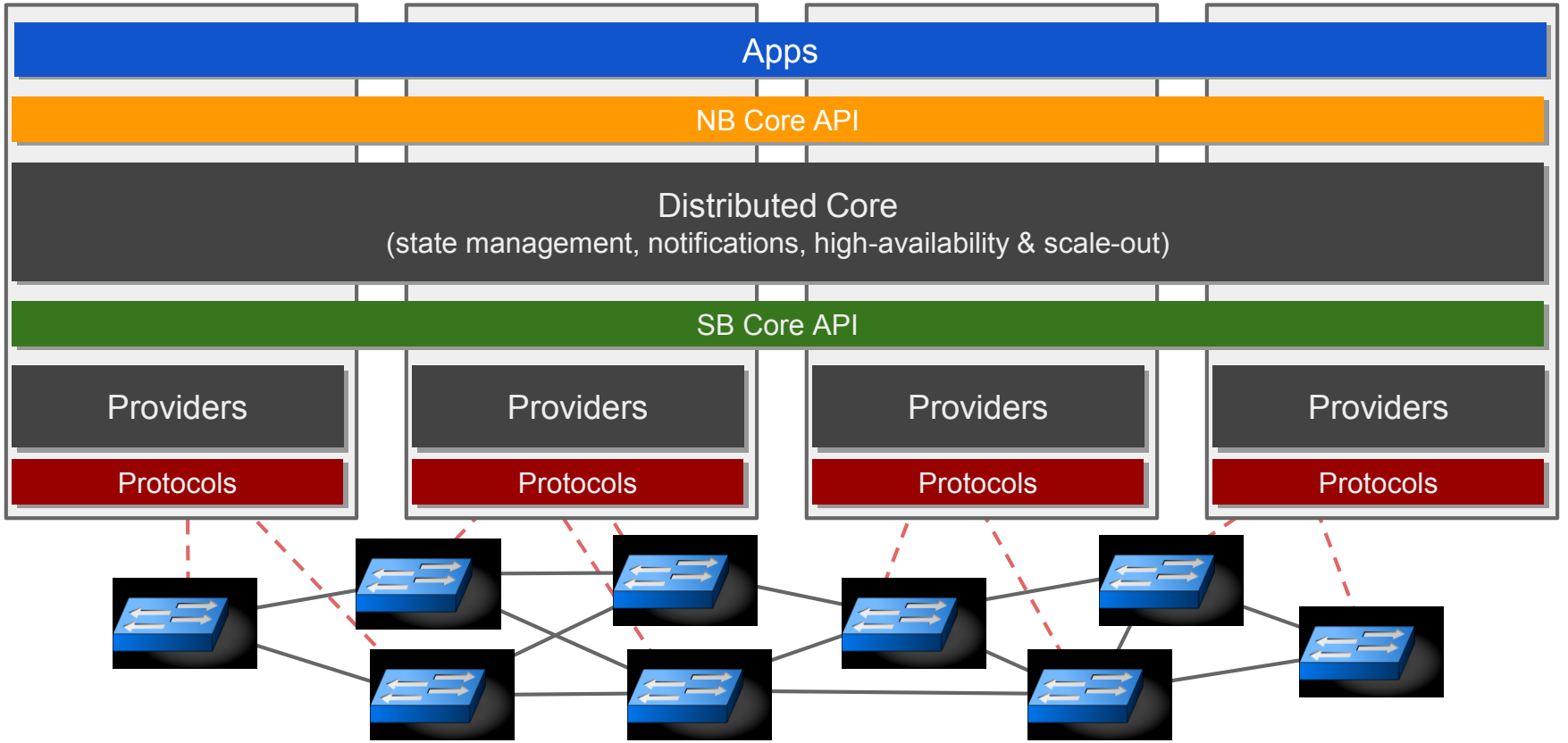
- High-availability, scalability and performance
  - required to sustain demands of service provider & enterprise networks
- Strong abstractions and simplicity
  - required for development of apps and solutions
- Protocol and device behaviour independence
  - avoid contouring and deformation due to protocol specifics
- Separation of concerns and modularity
  - allow tailoring and customization without speciating the code-base

Service Provider Networks

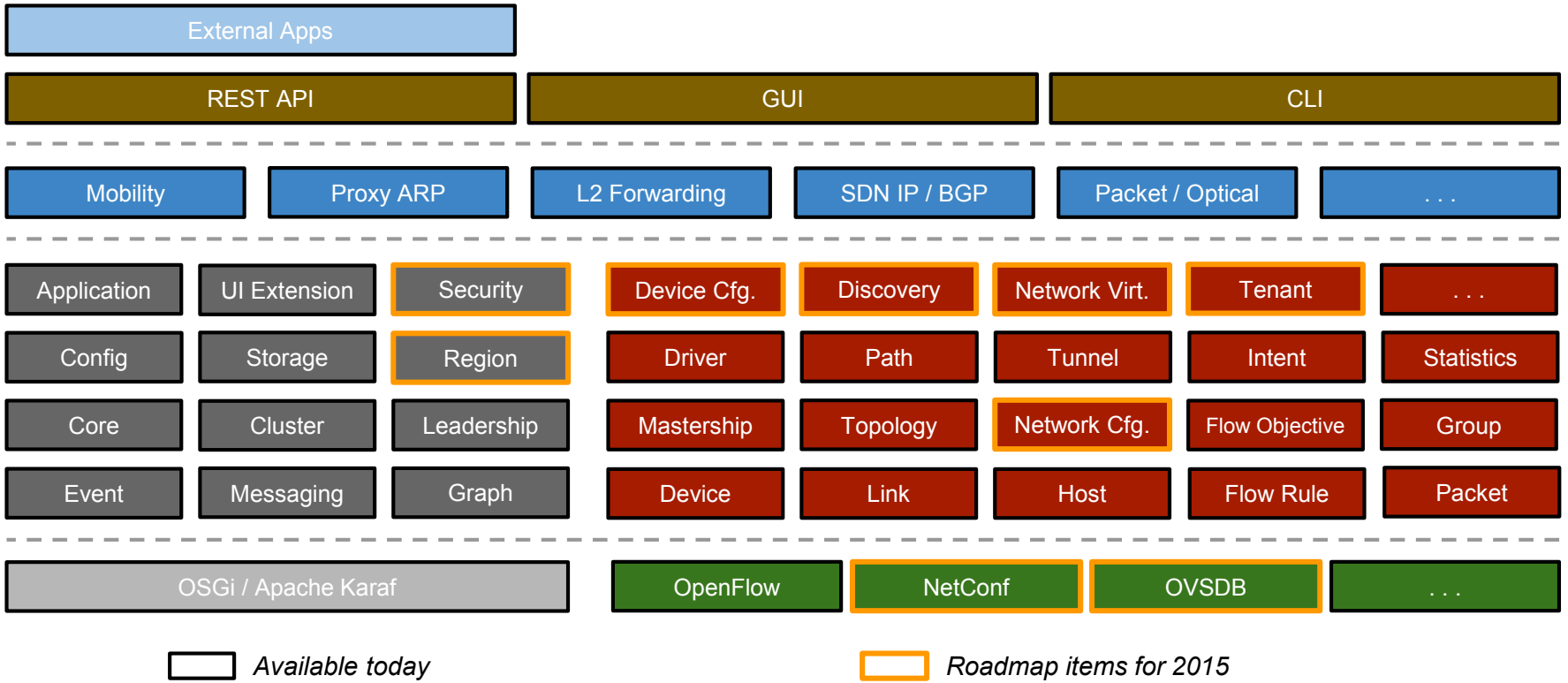
- WAN core backbone
  - Multi-Protocol Label Switching (MPLS) with Traffic Engineering (TE)
  - *200-500 routers, 5-10K ports*
- Metro Networks
  - Metro cores for access networks
  - *10-50K routers, 2-3M ports*
- Cellular Access Networks
  - LTE for a metro area
  - *20-100K devices, 100K-100M ports*
- Wired access / aggregation
  - Access network for homes; DSL/Cable
  - *10-50K devices, 100K-1M ports*



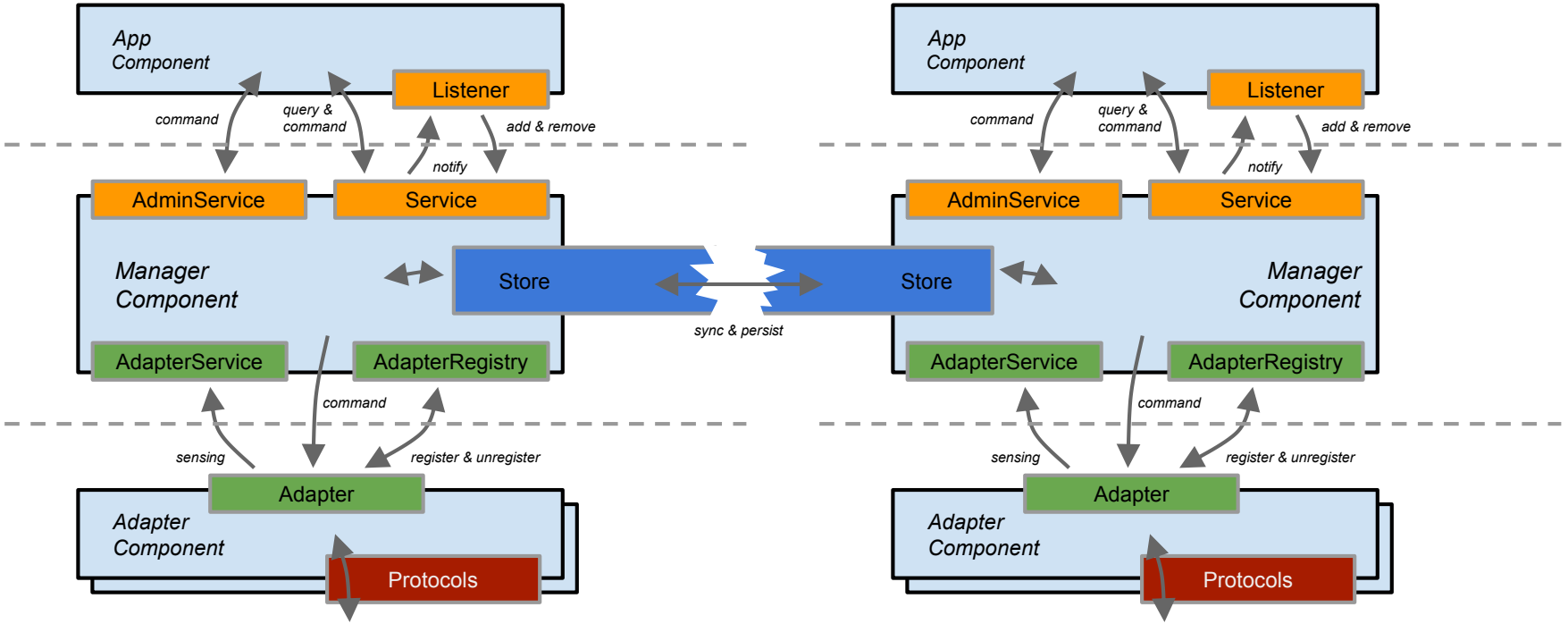
ONOS Distributed Architecture



ONOS Subsystems - Today & 2015

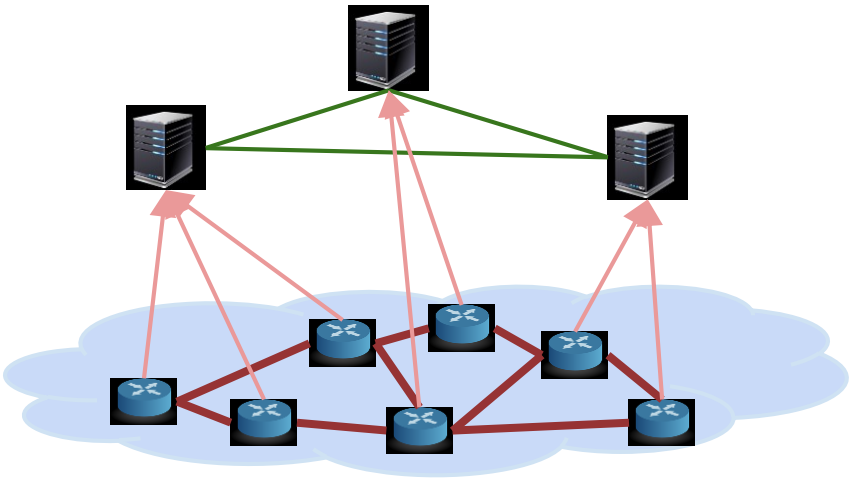


ONOS Core Subsystem Structure



ONOS Core

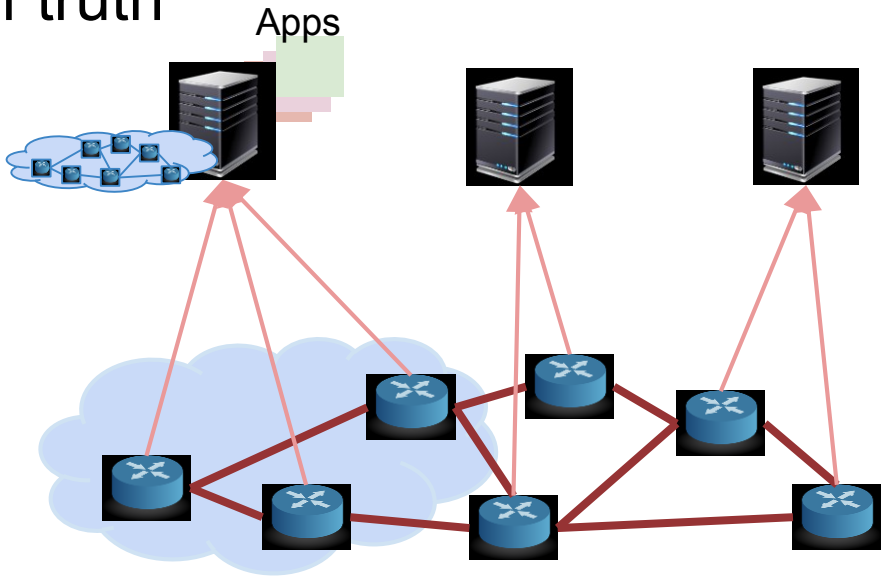
Control Plane State



- Topology
- Flows
- Intents
- Switch to controller mapping
- Resource allocations
- Network Configuration
- And a plethora of application generated state

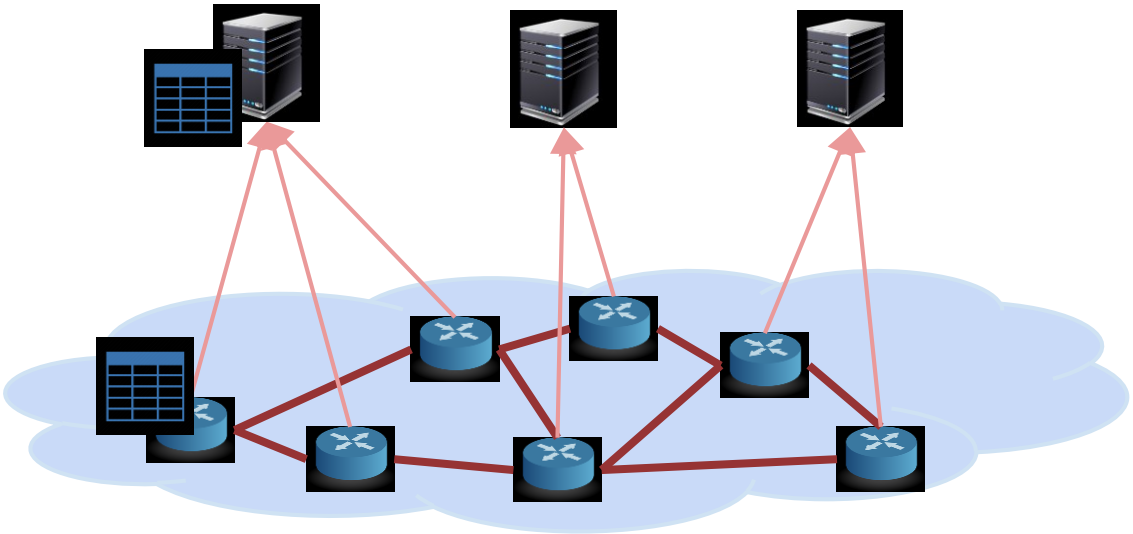
Topology

- Observed network state
- Each controller directly observes a subset of network
- Applications access Global Network View in its entirety
- Data plane is source of truth



Flows

- Data plane forwarding rules
- Naturally partitioned by forwarding element (switch)
- Control plane is the source of truth

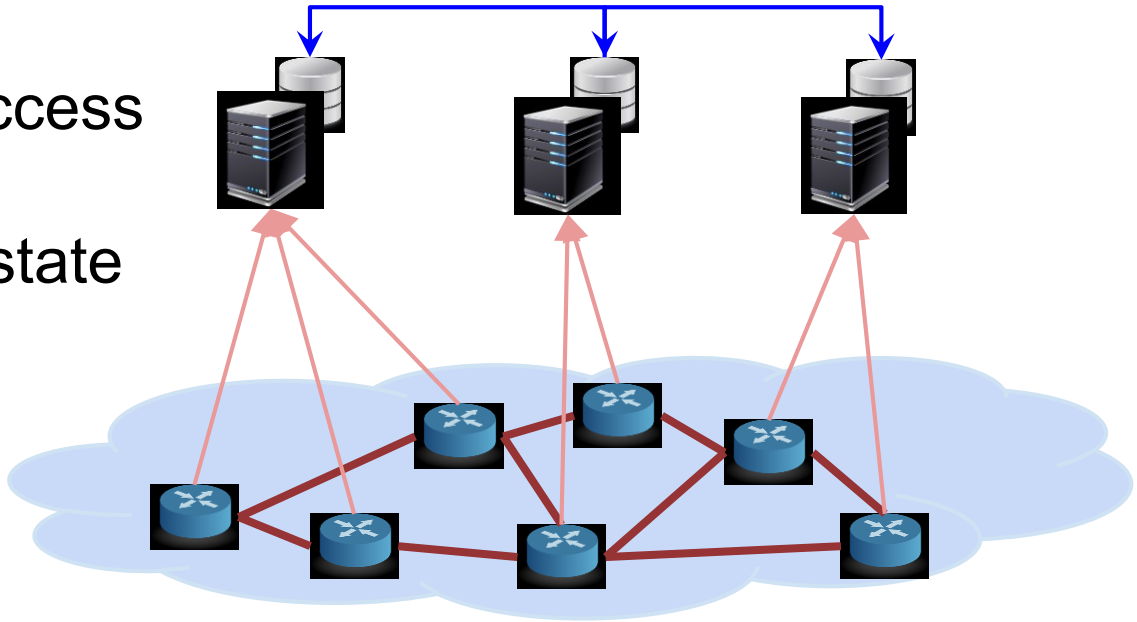


State and Properties

State	Properties
Network Topology	Eventually consistent, low latency access
Flow Rules, Flow Stats	Eventually consistent, shardable, soft state
Switch - Controller mapping Distributed Locks	Strongly consistent, slow changing
Application Intents Resource Allocations	Strongly consistent, durable Immutable

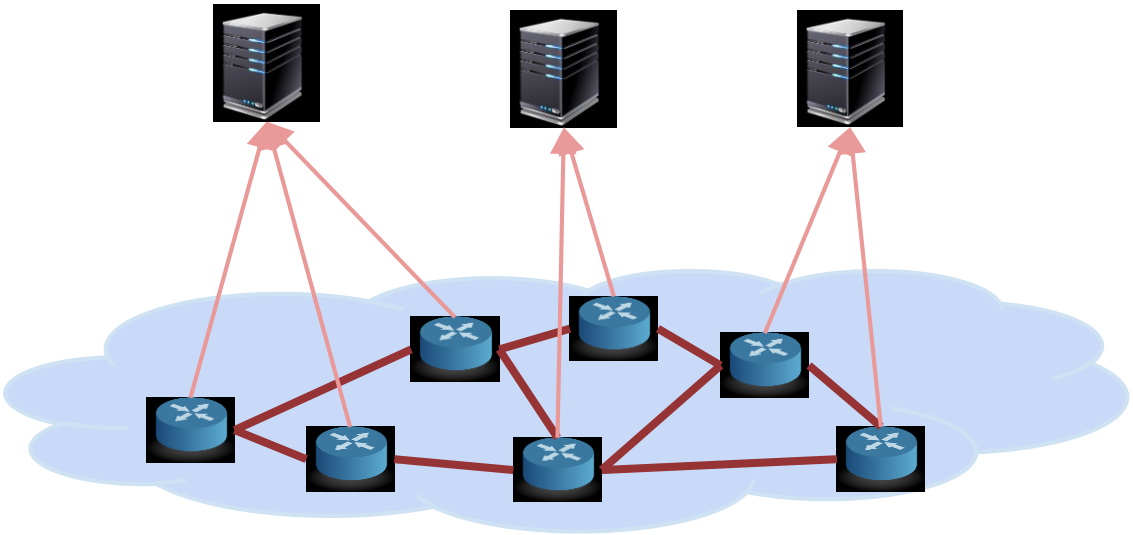
Drawback of a single (Logically) Central Datastore

- Tuned for either high availability or strong consistency
- Fails to exploit the access patterns and locality constraints of some state



ONOS: Polyglot State Management

- There is no one solution to rule them all
- A core set of state management primitives



ONOS Core Summary

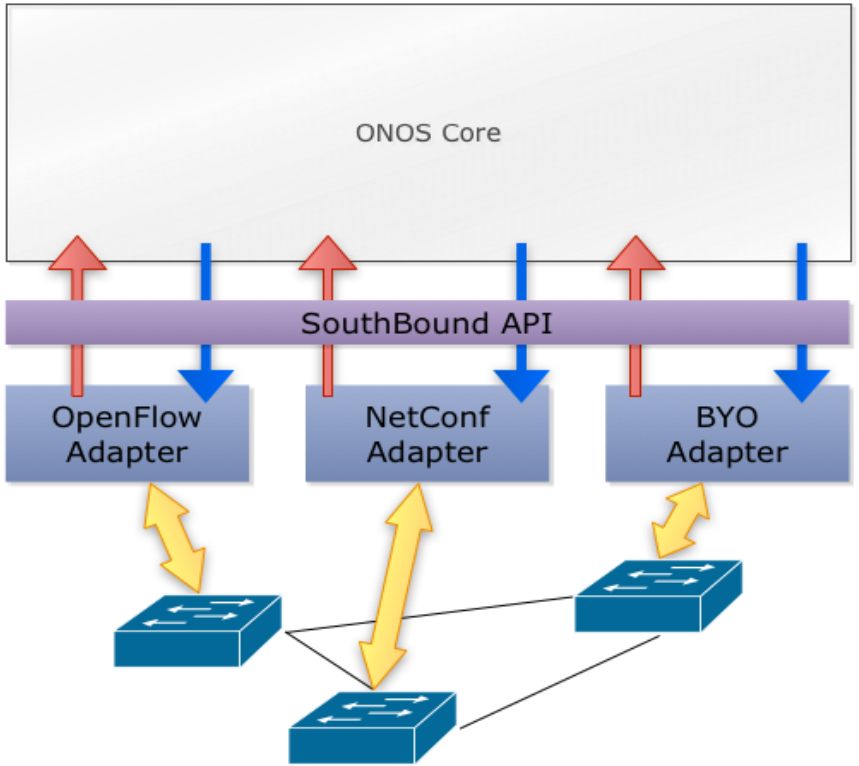
- All the distributed notions used to implement our solutions are available to you:

  - ✓ Transactional Map
  - ✓ Distributed Set
  - ✓ Atomic Counter
  - ✓ Leadership Service
  - ✓ EventuallyConsistentMap<K, V, T extends Timestamp>
  - ✓ ConsistentMap<K, V>

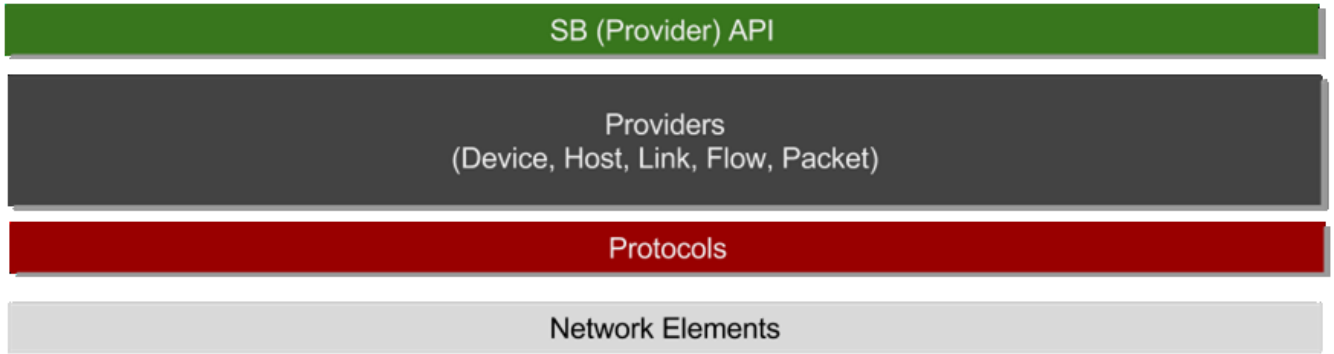
ONOS Southbound

ONOS Southbound

- ONOS supports multiple southbound protocols, enabling a transition to true SDN.
- Adapters provide descriptions of dataplane elements to the core - core utilizes this information.
- Adapters hide protocol complexity from ONOS.

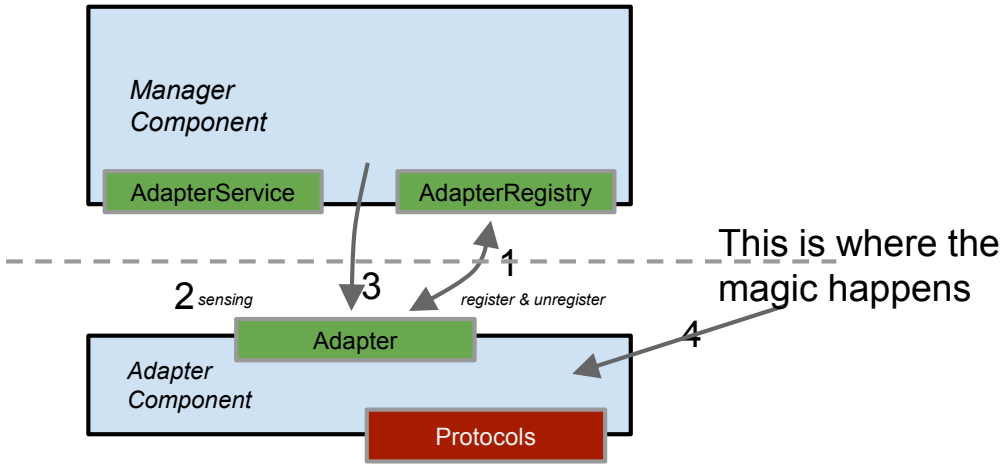


Area of focus



- Attempt to be as generic as possible
- Enable partners/contributors to submit their own device/protocol specific providers
- Providers should be stateless; state may be maintained for optimization but should not be relied upon

Adapter Pattern



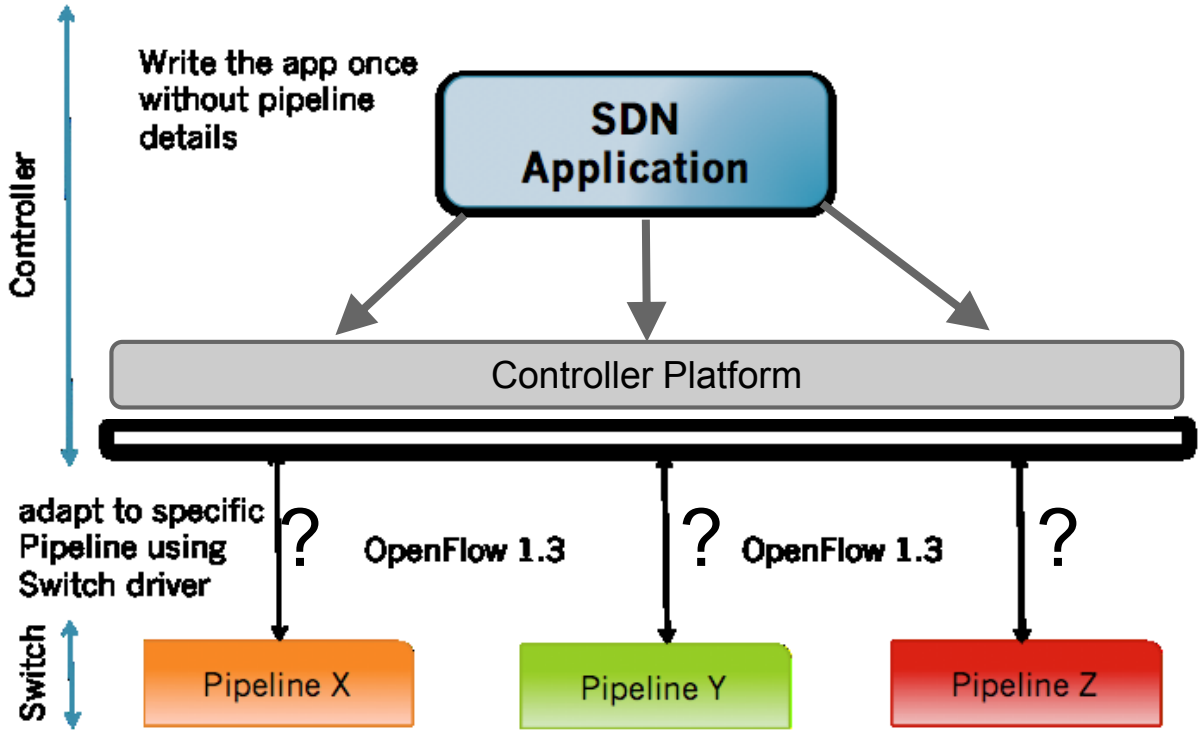
1. Adapter registers with core
   a. Core returns a AdapterService bound to the Adapter
2. Adapter uses AdapterService to notify core of new events (device connected, pktin) via Descriptions
3. Core can use Adapter to issue commands to elements under Adapter control
4. Eventually, the adapter unregisters itself; core will invalidate the AdapterService

Descriptions

- Serve as scratch pads to pass information to core
- Descriptions are immutable and extremely short lived
- Descriptions contains URI for the object they are describing
- URI also encode the Adapter the device is linked to

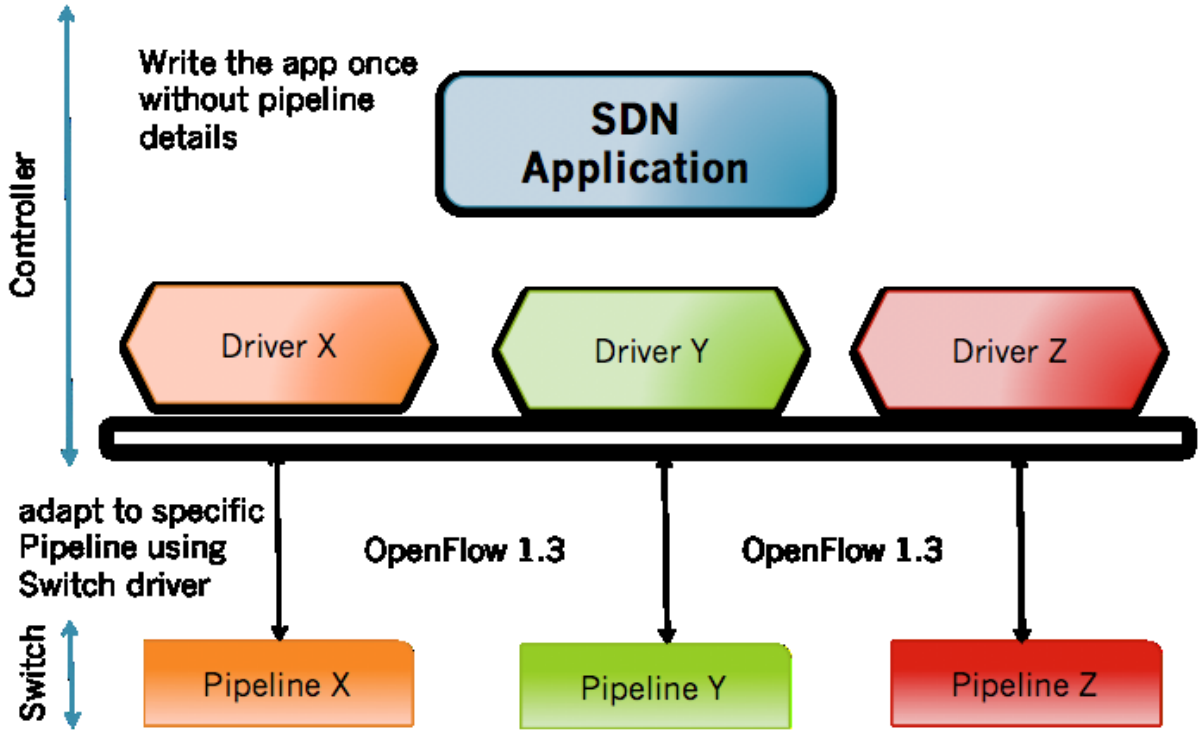
Flow Objective Subsystem

- **Problem:** Applications today must be pipeline aware, effectively making them applicable to specific HW.



Flow Objective Abstraction

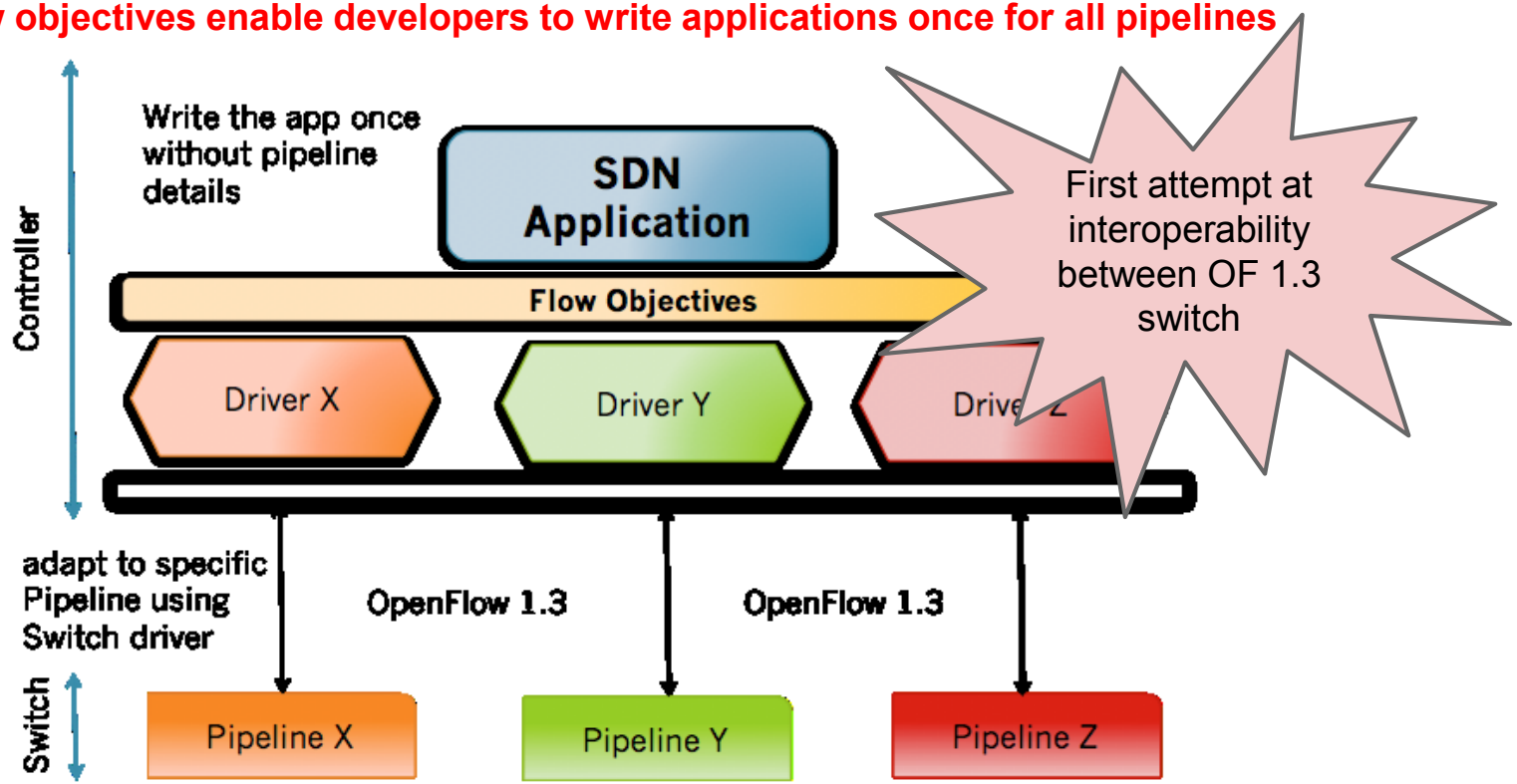
- **Problem:** Applications currently must be pipeline aware, effectively making applicable on specific HW.



Flow Objective Abstraction

- **Problem:** Applications currently must be pipeline aware, effectively making applicable on specific HW.

Flow objectives enable developers to write applications once for all pipelines



Flow Objective Summary

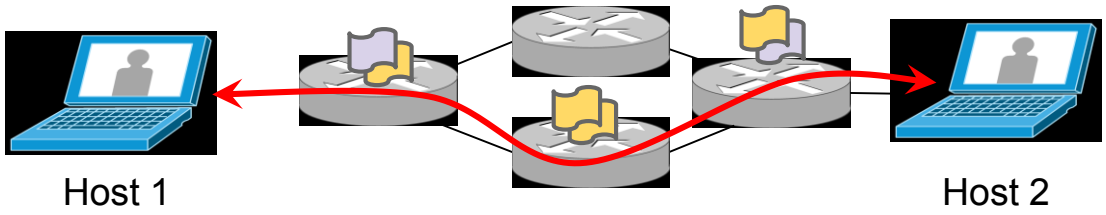
- *Flow Objective Service:* **Abstraction** for applications to be **pipeline unaware** while **benefiting** from scalable, multi-table architectures
- Aims to make it **simple** to write apps
- First attempt at achieving **interoperability** between OF 1.3 implementations

ONOS Northbound

Building Network Applications

Objective: Connect Host 1 and Host 2

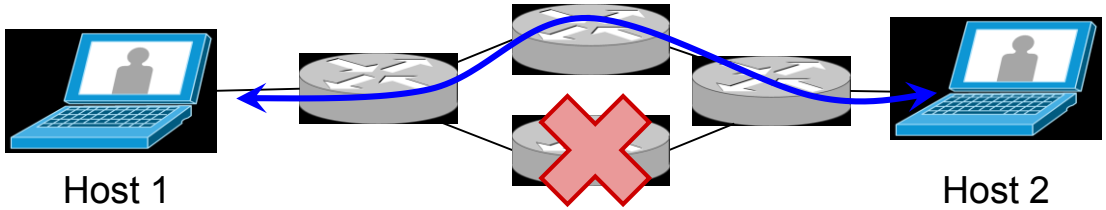
1. Read/discover the topology
2. Compute a path
3. Build flow objectives for each device
4. Install rules in consistent way



Building Network Applications

What can go wrong?

- Missing / rejected / dropped rules
  - Monitor devices connections
  - Send barriers between rule updates
  - Poll flow state
- Topology changes
  - Listen to switch, port, link and host events
  - Compute new path that leverage or remove old flows

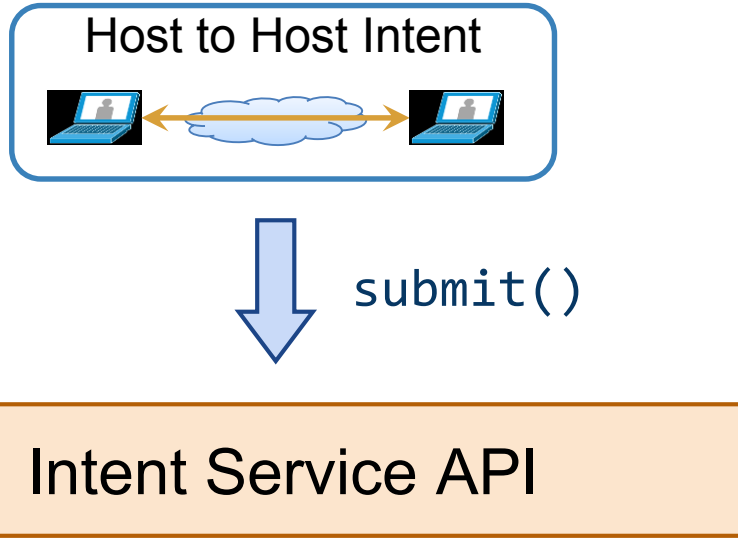


Intent Framework

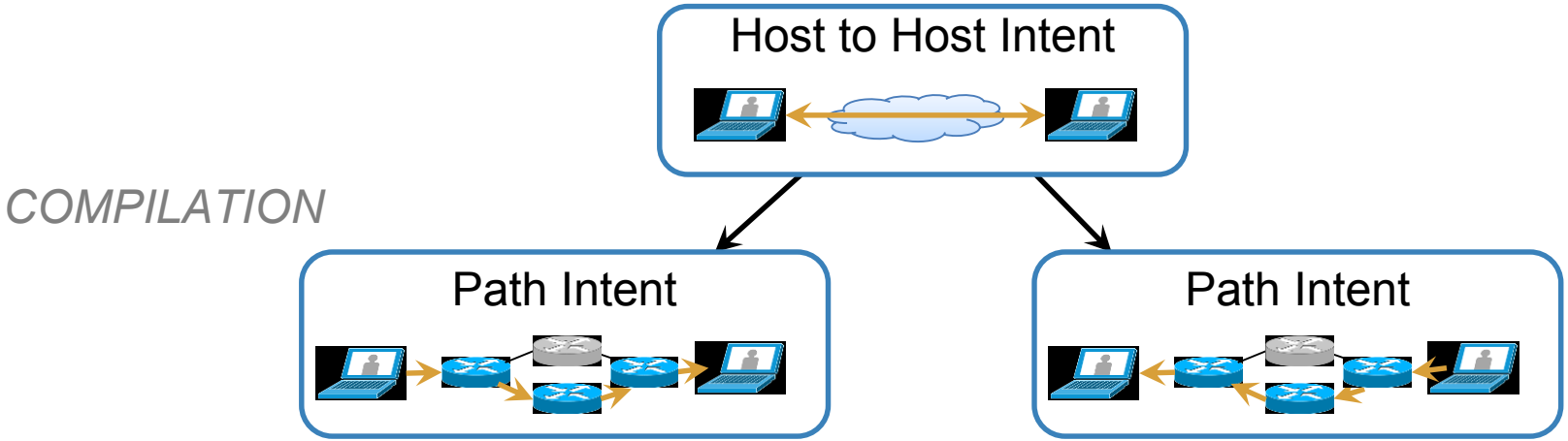
- Provides **high-level**, **network-centric** interface that focuses on ***what*** should be done rather than ***how*** it is specifically programmed
- Abstracts unnecessary network complexity from applications
- Maintains requested semantics as network changes
- High availability, scalability and high performance

Intent Example

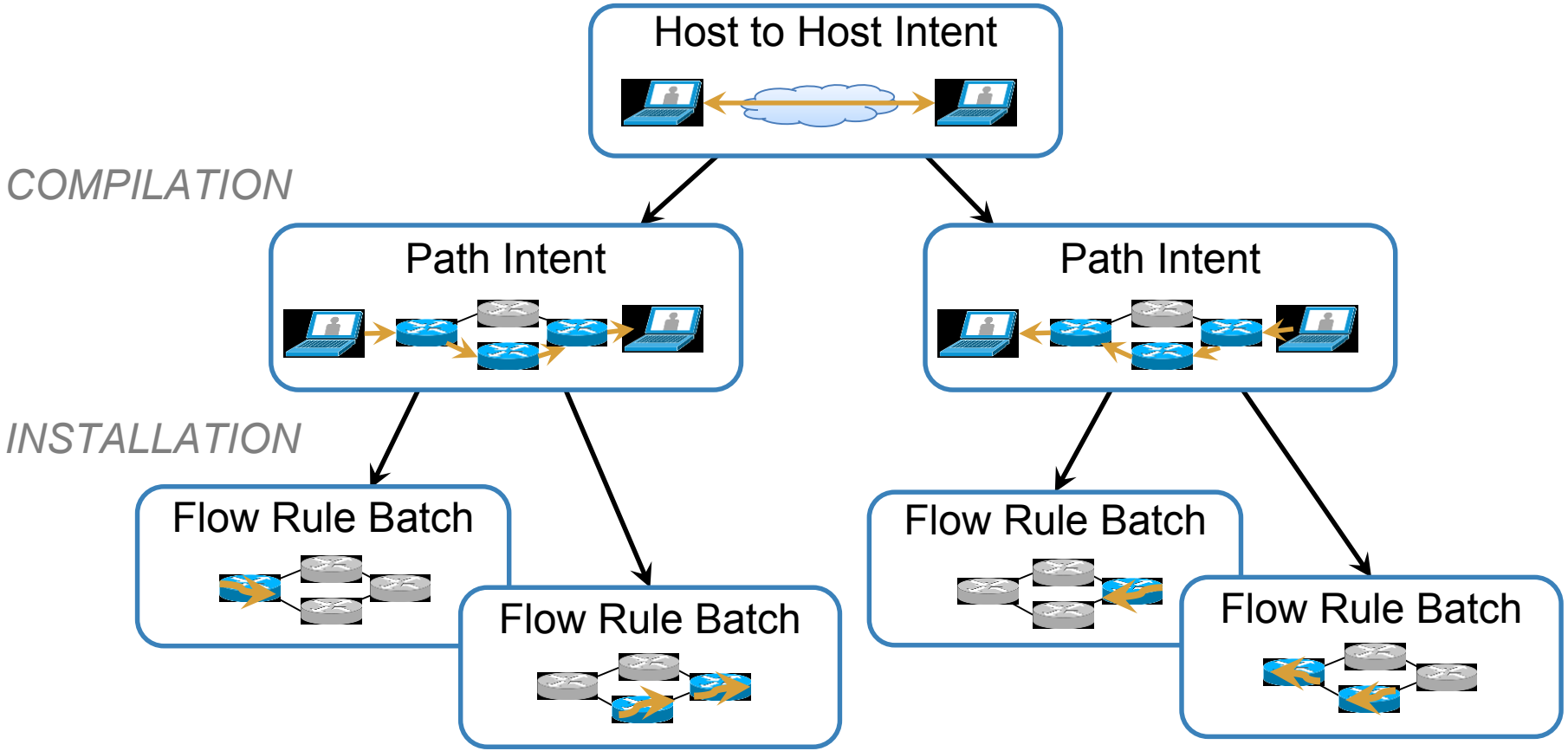
Intent Example



Intent Example



Intent Example



Intent Framework Summary

- Intents are a ***network-centric programming abstraction*** that **reduce application complexity.**
- Intents provide **device-agnostic behavior** with **persistency** and **high performance** across network failures.
- Intent framework has moved from prototype to **production** deployments.

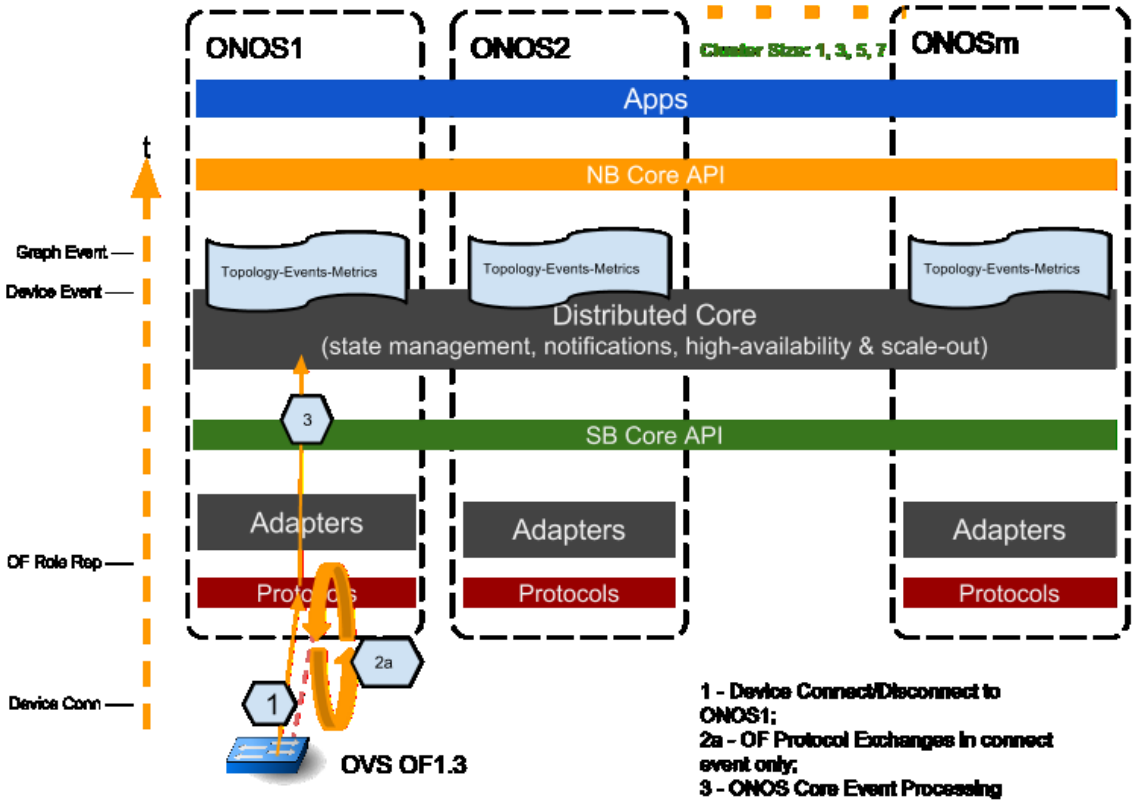
ONOS Performance

Control Plane Performance

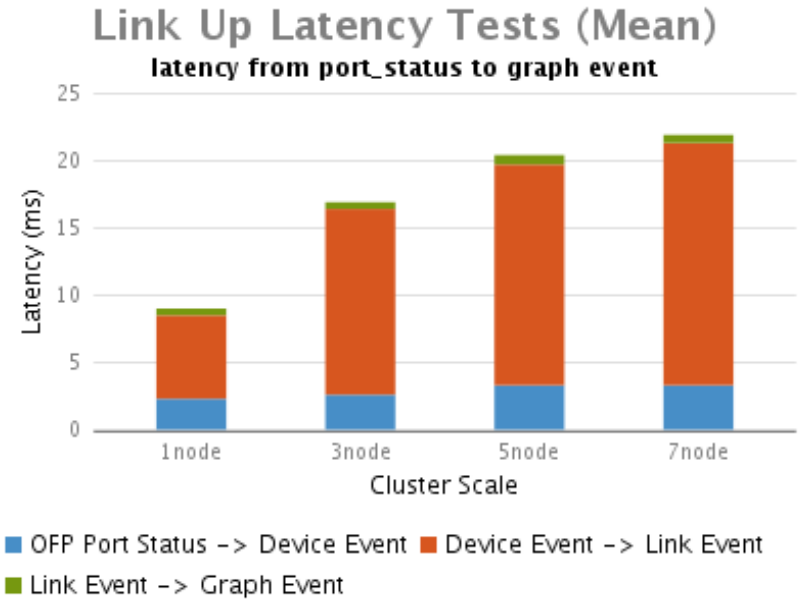
- Throughput of proactive provisioning actions
  - path flow provisioning
  - global optimization or rebalancing of existing path flows
- Latency of responses to topology changes
  - path repair in wake of link or device failures
- Throughput of distributing and aggregating state
  - batching, caching, parallelism
  - dependency reduction
- Controller vs. device responsibilities
  - defer to devices to do what they do best, e.g. low-latency reactivity, backup paths

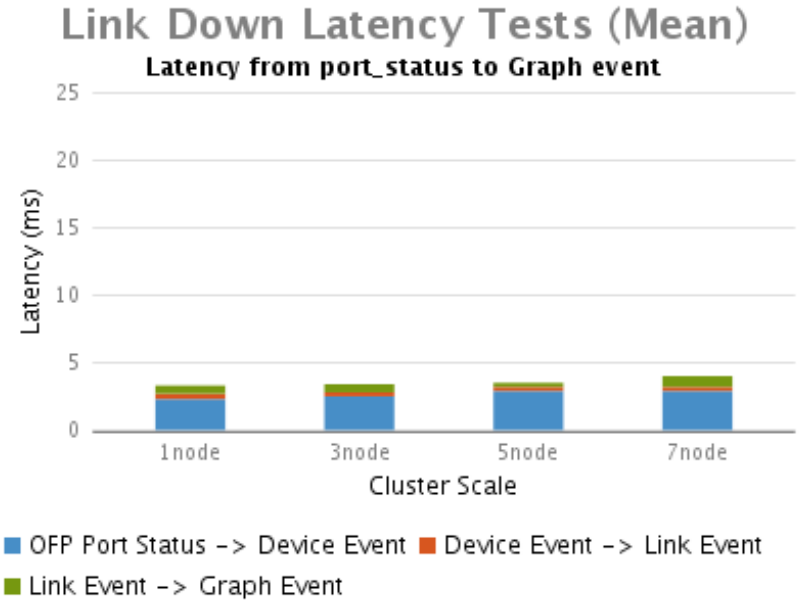
Performance Metrics

- Device & link sensing latency
  - measure how fast can controller react to environment changes, such as switch or port down to rebuild the network graph and notify apps
- Flow rule operations throughput
  - measure how many flow rule operations can be issued against the controller and characterize relationship of throughput with cluster size
- Intent operations throughput
  - measure how many intent operations can be issued against controller cluster and characterize relationship of throughput with cluster size
- Intent operations latency
  - measure how fast can the controller react to environment changes and reprovision intents on the data-plane and characterize scalability


ONOS1
ONOS2
Cluster Size: 1, 3, 5, 7
ONOSm
Apps
NB Core API
t
Graph Event
Device Event
Topology-Events-Metrics
Topology-Events-Metrics
Topology-Events-Metrics
Distributed Core
(state management, notifications, high-availability & scale-out)
3
SB Core API
Adapters
Adapters
Adapters
OF Role Rep
Protocols
Protocols
Protocols
2a
1
Device Conn
OVS OF1.3
1 - Device Connect/Disconnect to ONOS1;
2a - OF Protocol Exchanges in connect event only;
3 - ONOS Core Event Processing

Link Up/Down Latency

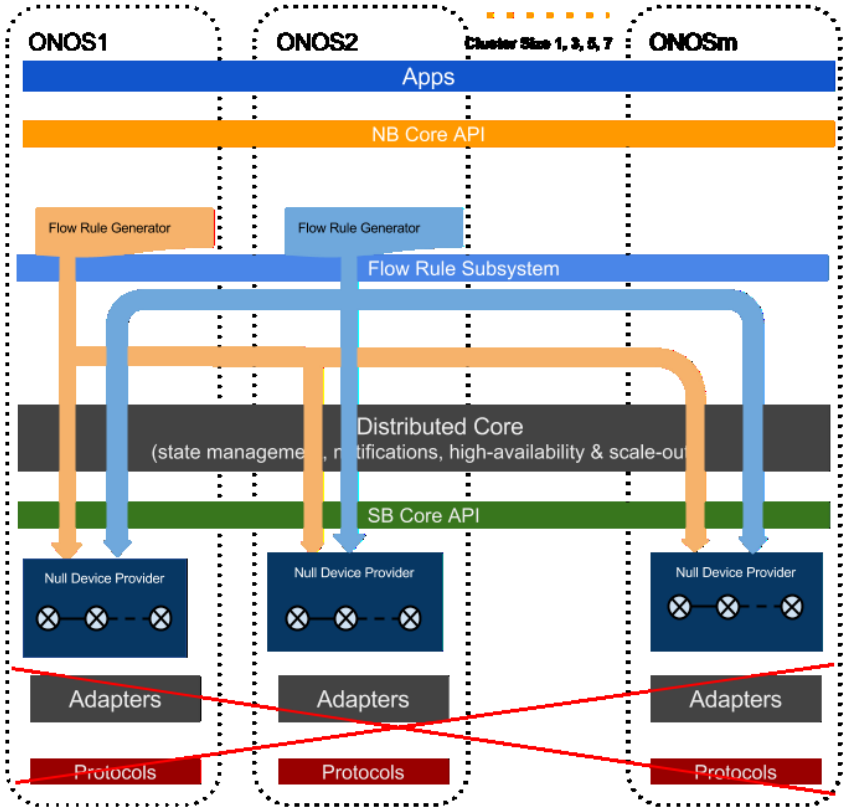




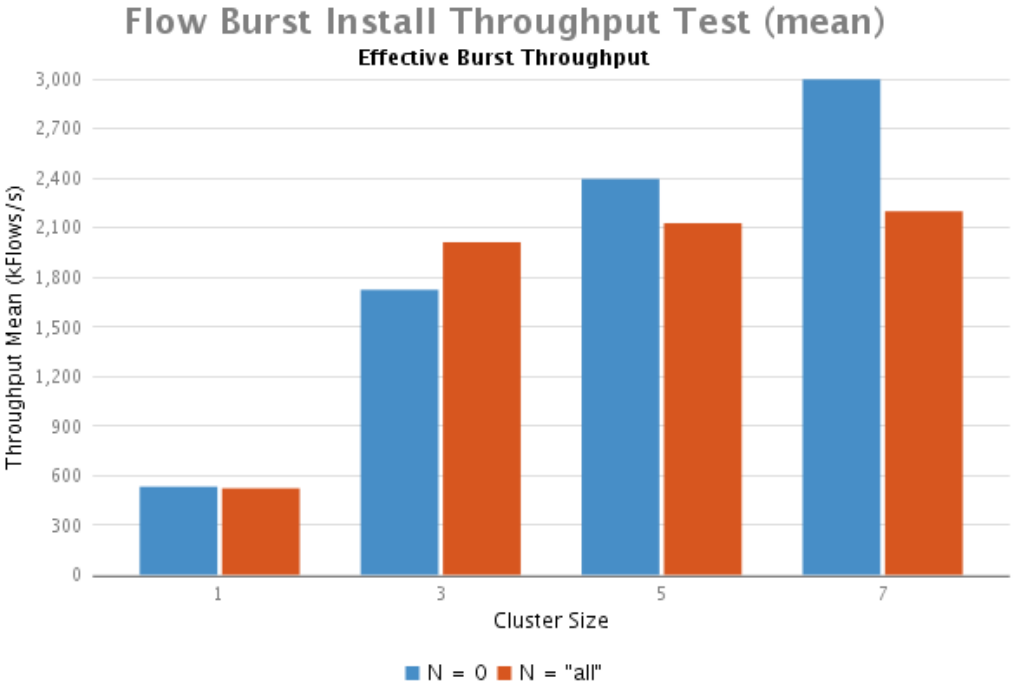
- Since we use LLDP & BDDP to discover links, it takes longer to discover a link coming up than going down

- Port down event trigger immediate teardown of the link.

Flow Rule Operations Throughput

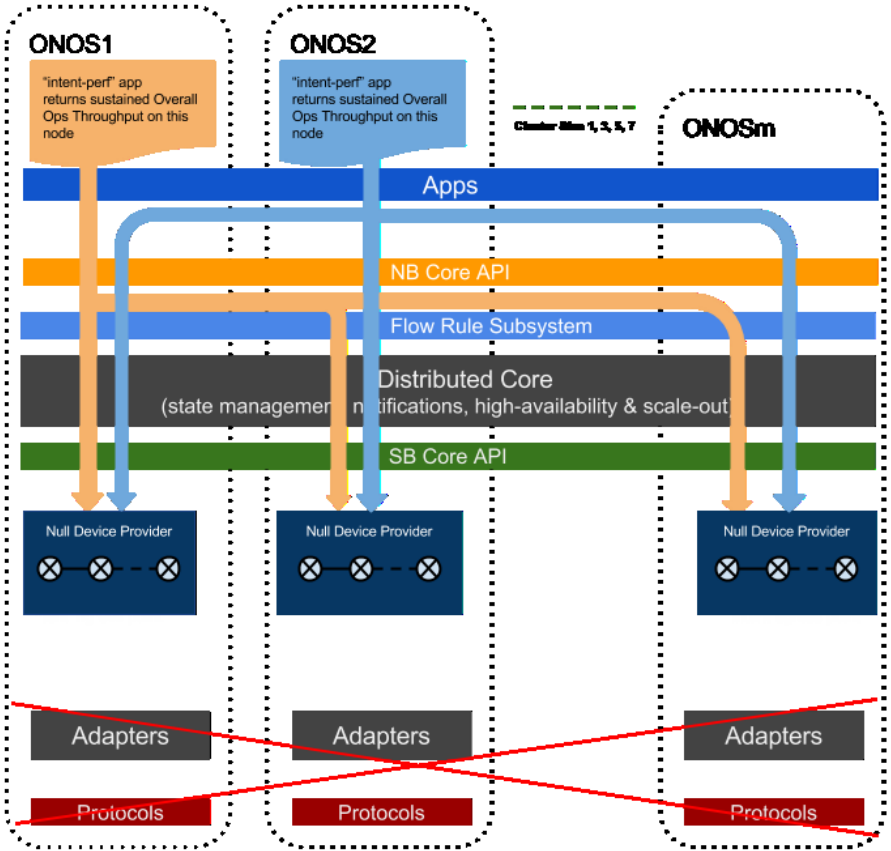


Flow Throughput results

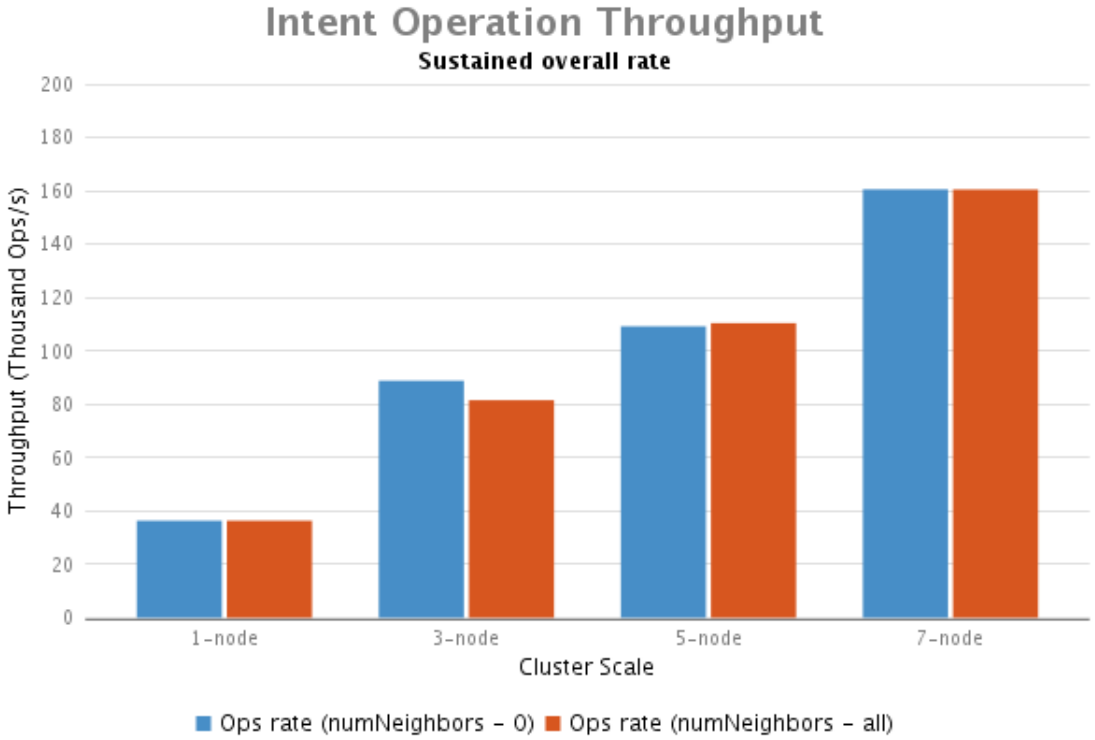


- Single instance can install over 500K flows per second
- ONOS can handle 3M local and 2M non local flow installations
- With 1-3 ONOS instances, the flow setup rate remains constant no matter how many neighbours are involved
- With more than 3 instances injecting load the flow performance drops off due to extra coordination requires.

Intent Throughput Experiment

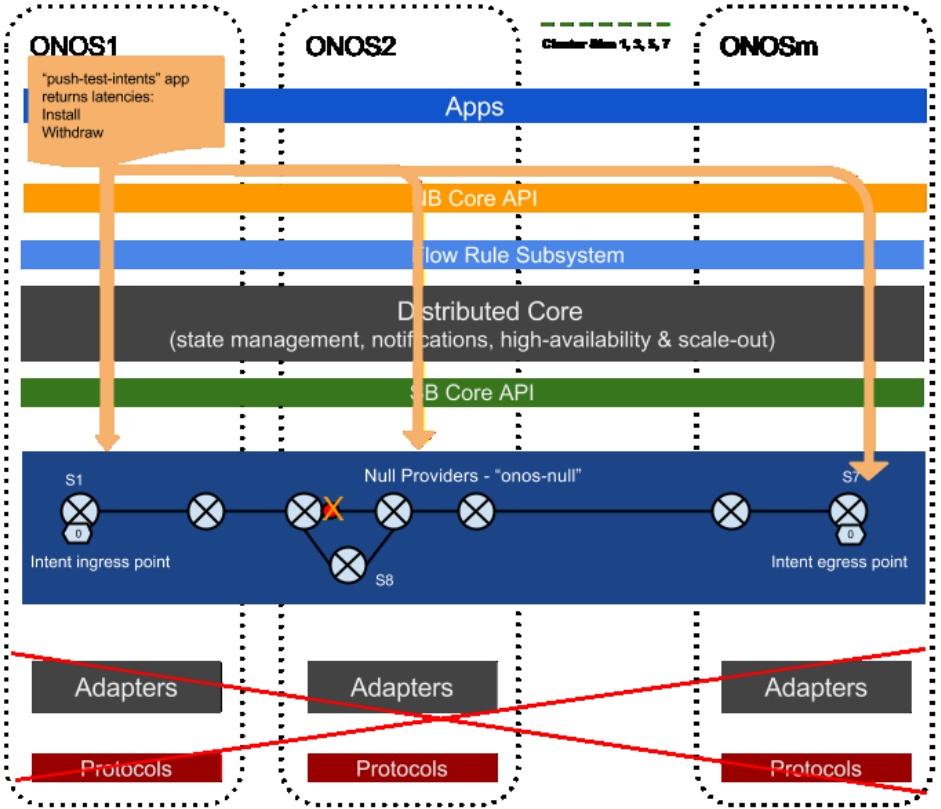


Intent Throughput Results

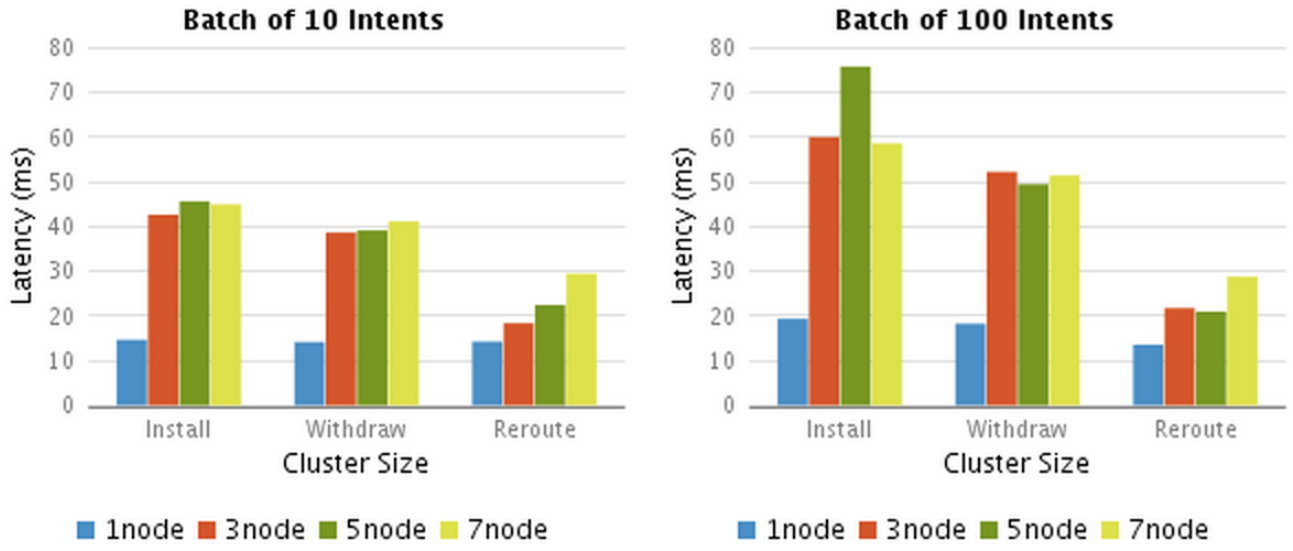


- Processing clearly scales as cluster size increases

Intent Latency Experiment



Intent Latency Results



- Less than 100ms to install or withdraw a batch of intents
- Less than 50ms to process and react to network events
  - Slightly faster because intent objects are already replicated

Key Takeaways

- Lack of high performance, scalable, highly available SDN control plane and solutions are key barriers to SDN adoption in service provider networks
- ONOS addresses this challenge with logically centralized but distributed architecture that provides performance, scale, HA together
- Comprehensive set of metrics and measurements for ONOS Blackbird release are published on the wiki @ http://bit.ly/blackbird-eval

Summary

ONOS continues to evolve to:

- operate ever more efficiently and reliably
- support larger scale networks
- further simplify the programming model for applications
- broaden the set of supported devices and protocols
- simplify SDN deployments

Checkout our tutorials at:
wiki.onosproject.org

Software Defined Transformation of Service Provider Networks

Join the journey @ onosproject.org