The Open Network Operating System

Carmelo Cascone, Andrea Campanella, Andrea Biancini

Politecnico di Milano, Università degli studi di Milano & ON.Lab, Reti S.p.a.

CommTech Talks, DEIB, Politecnico di Milano

October 25, 2016

#ONOSProject

ONOS Open Network Operating System

Outline

- Why do we need a network OS?
  - Motivating the need for Software-Defined Networking
- ONOS overview
  - Architecture
  - APIs
  - Applications
- Demo
- Deployments and use cases
- Community & how to get involved

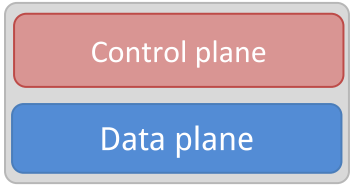
What is ONOS?

Open Network Operating System (ONOS) is an open source Software-Defined Network (SDN) operating system...

What is SDN? Why do we need a network OS?

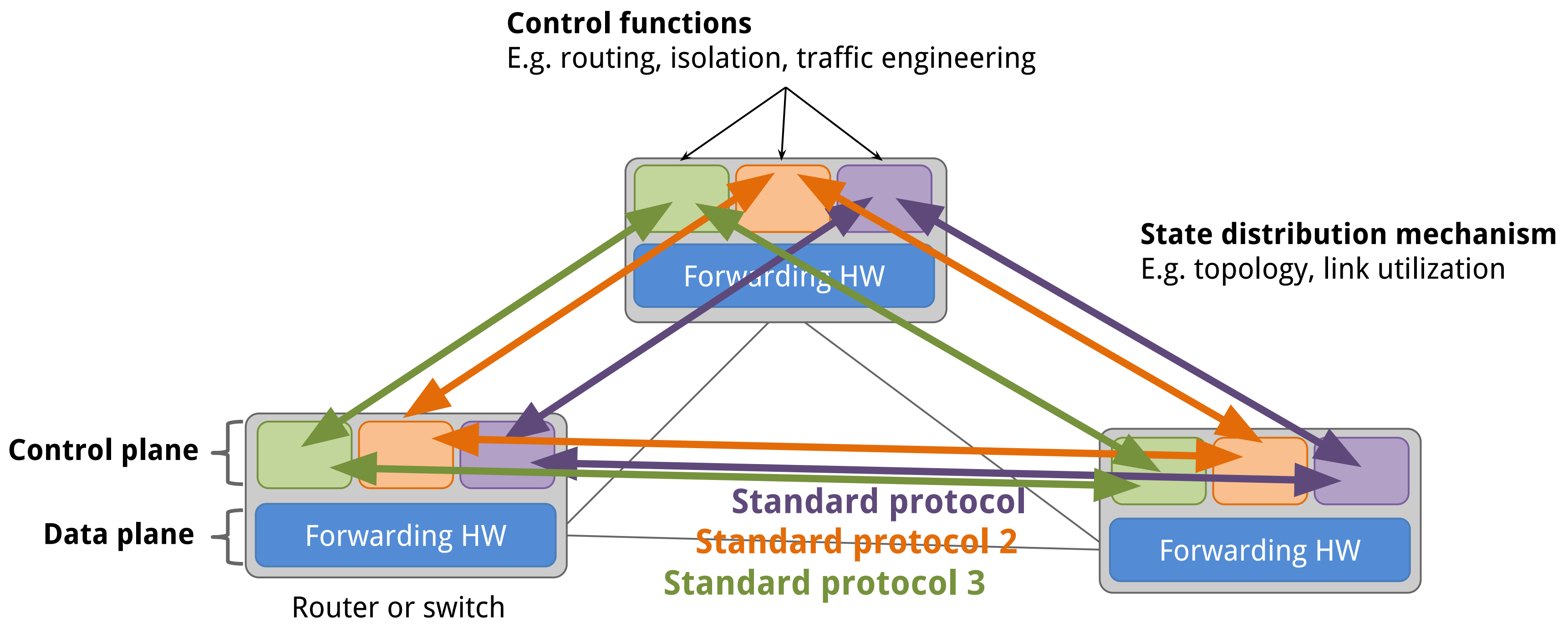
Basic network abstractions

- **Data plane**
  - Basic packet forwarding functionality
    - Forward, filter, buffer, mark, rate-limit, and measure packets
  - Usually implemented in hardware
  - Uses only local information
    - *f(pkt header, input port) ➔ output port or drop*
  - Usually abstracted with tables
    - E.g. routing tables, switching tables, ACLs, etc.
- **Control plane** **This talk & ONOS**
  - Compute the configuration of each physical device
    - E.g routing, isolation, traffic engineering
  - Usually implemented in software
  - Based on global information
    - E.g. *f(net topology graph, weights) ➔ routing table*



Router or switch

Traditional networking paradigm



Designing control functions

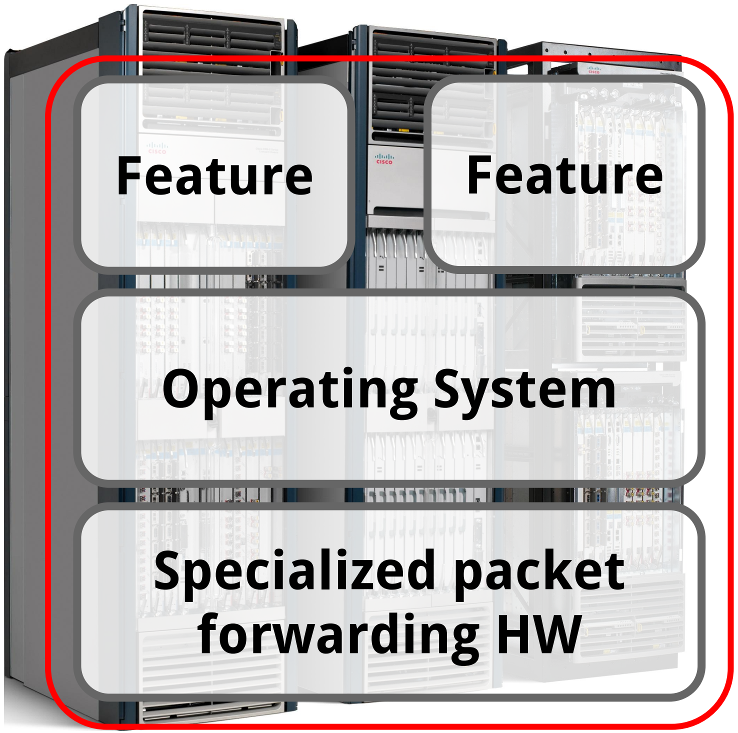
E.g. to define a new routing protocol

Given a network of arbitrary topology and size...

1. **Design a distributed algorithm**
   - Each device has the same topology view, is aware of link failures...
2. **Handle communication errors**
   - Network is unreliable: packets dropped, arrive out of sync...
3. **Define a communication protocol**
4. **Wait for standardization**
5. **Wait for vendors to adopt the standard**

It takes years... What if there's a bug?

Closed market (until 2008)



Little ability for small players and researchers to implement or try new features.

Same vendor, closed platform

Software-Defined Networking (2008)

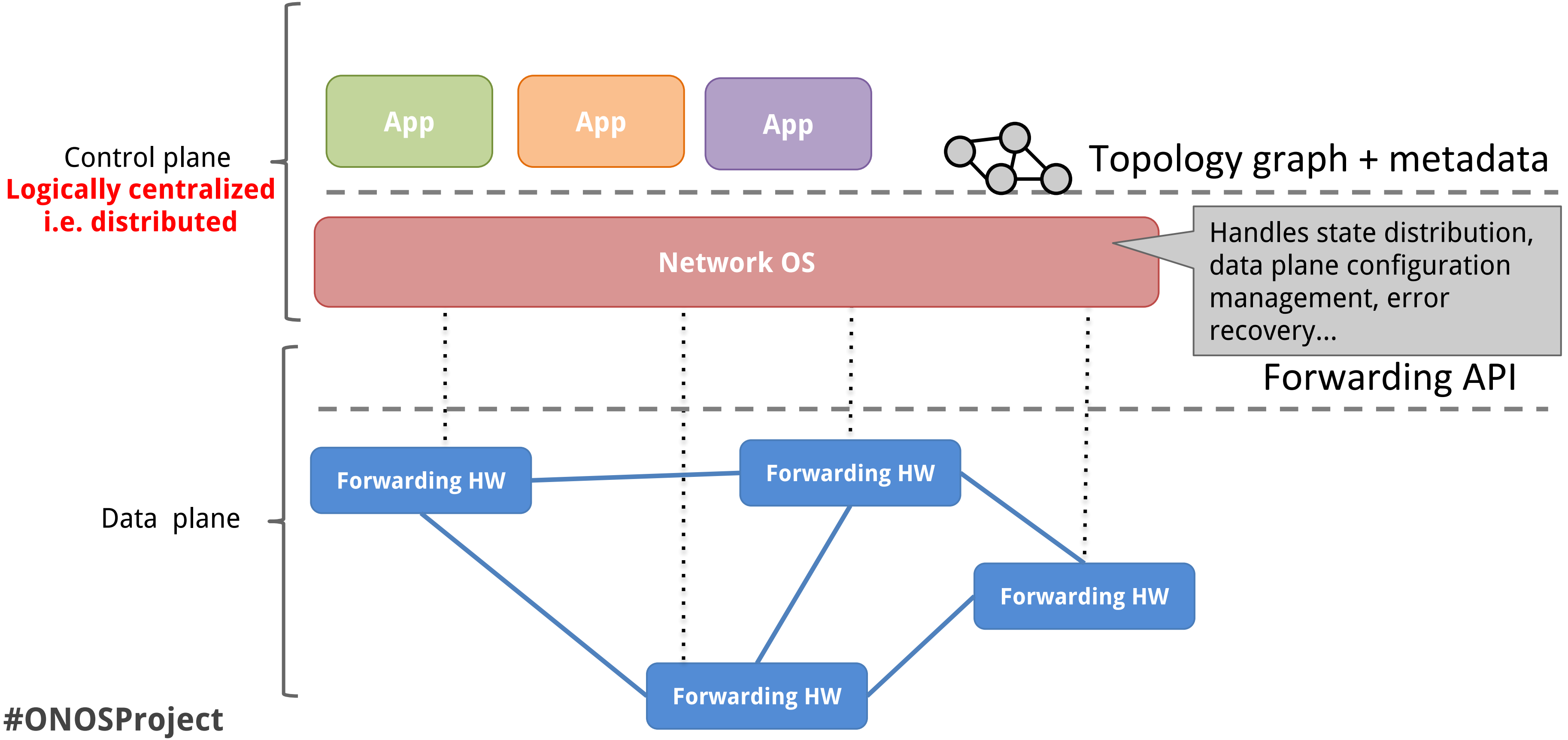
What is all about?
The “Scott Shenker view”:

- Define software abstractions that can be reused when building control plane functions
  - State distribution abstraction
    - Solve the problem once, for every function
  - Forwarding abstraction
    - Control the data plane in a vendor-independent manner

How?

- Separation and centralization of the control plane

SDN Architecture



#ONOSProject

Designing control functions with SDN

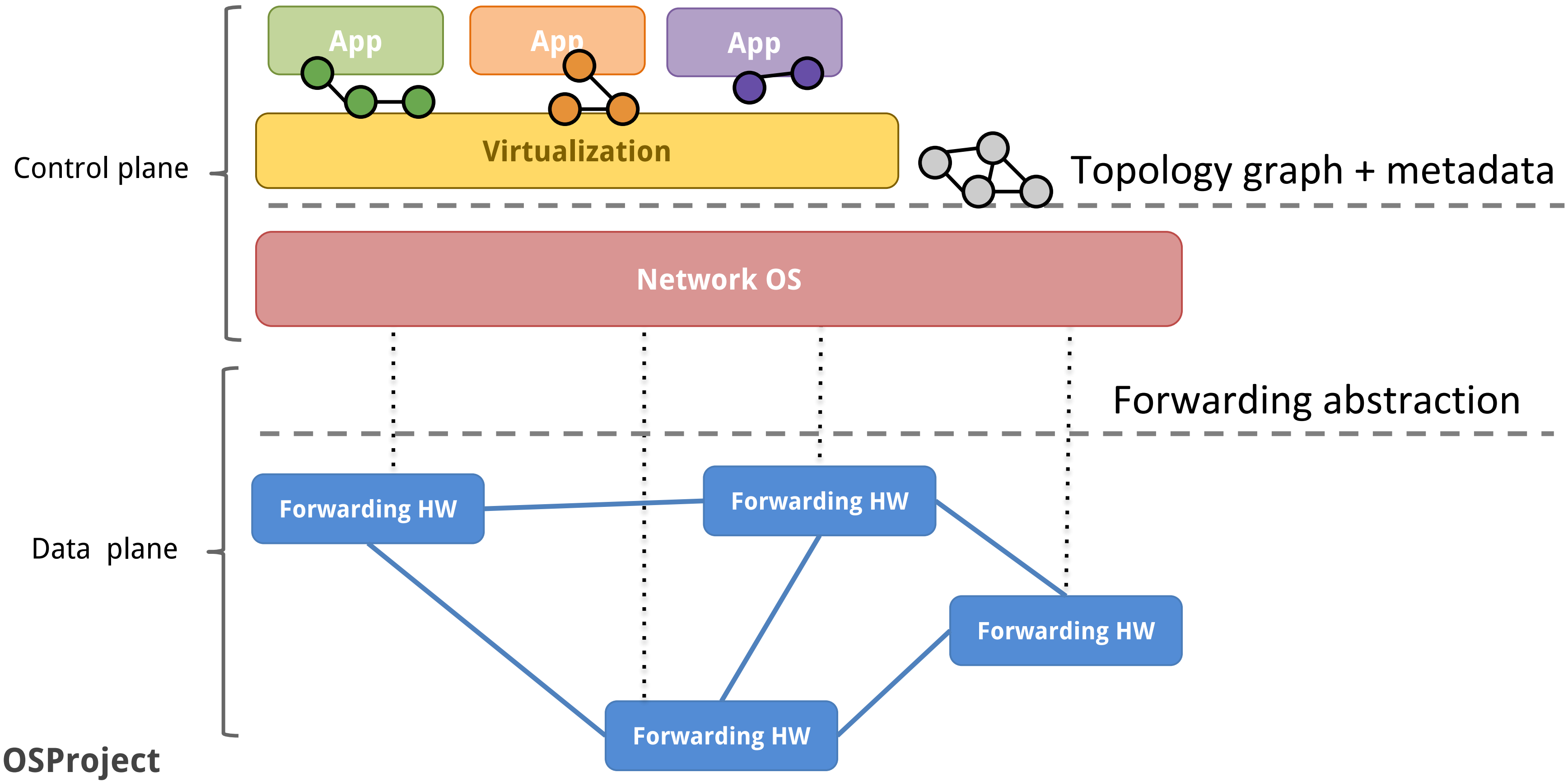
E.g. to define a new routing protocol

Given a network of arbitrary topology and size:

1. **Write an algorithm over a data structure**
   - The topology graph, annotated with metadata
2. **Program it via a software API**
3. **What if there's a bug?**
   - Solve it and push a software update!

SDN enables innovation at the speed of writing and deploying software!

SDN Virtualization



OpenFlow (2008)

- The most prominent SDN forwarding abstraction
  - But not the only one...

1) Install/update rules

2) Packet/port notifications

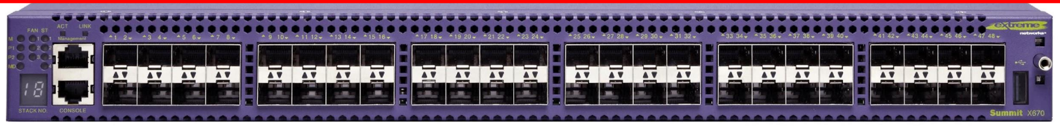
control channel

Network OS

Packet

Match-action table

IP src	IP dest	TCP dest	...	Actions
192.168/16	10/8	any	…	Port 2
192.168/16	any	80	…	Rate limit, Port 13
any	192.168/16	22	…	Drop
any	any	any	…	Send to controller



SDN Ecosystem Today

- Wide adoption in data center networks
  - Google, Facebook, Microsoft, etc.
- Big service providers starting to transition their networks
  - AT&T "Domain 2.0" project, Verizon, Deutsche Telekom, etc.
  - Becoming more software company
- White-box switching market
  - New vendors offer cheap, off-the-shelf OpenFlow HW switches
  - Facebook OCP project open sourced a HW design for a SDN switch
- New players in the "softwarized" networking market
  - VMware offers an SDN virtualization solution called NSX

What is ONOS?

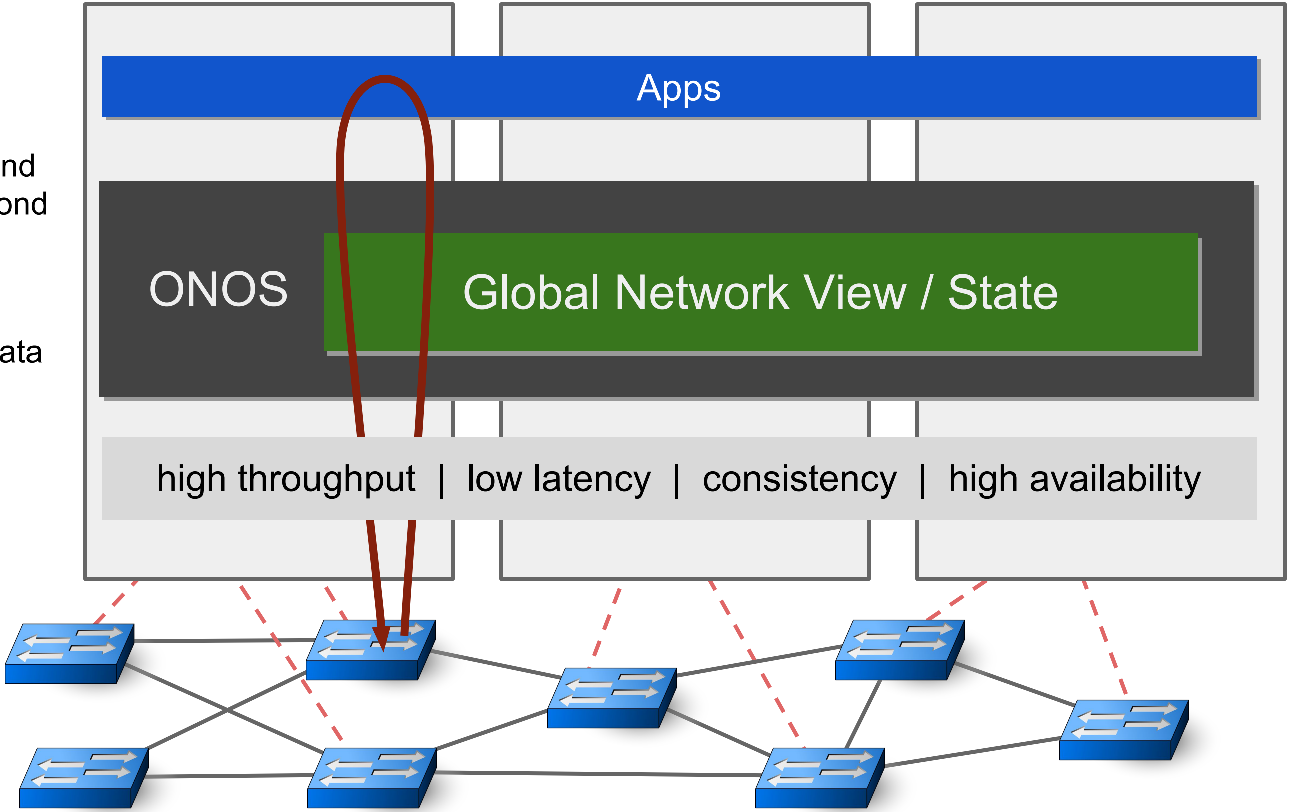
- SDN network OS
- Provides abstractions to make it easy to create apps and service to control a network.
- Designed for scalability, high availability, and performance.
- Focus on service provider networks, but not limited to it

Key Performance Requirements

High Throughput:
~500K-1M paths setups / second
~3-6M network state ops / second

High Volume:
~500GB-1TB of network state data

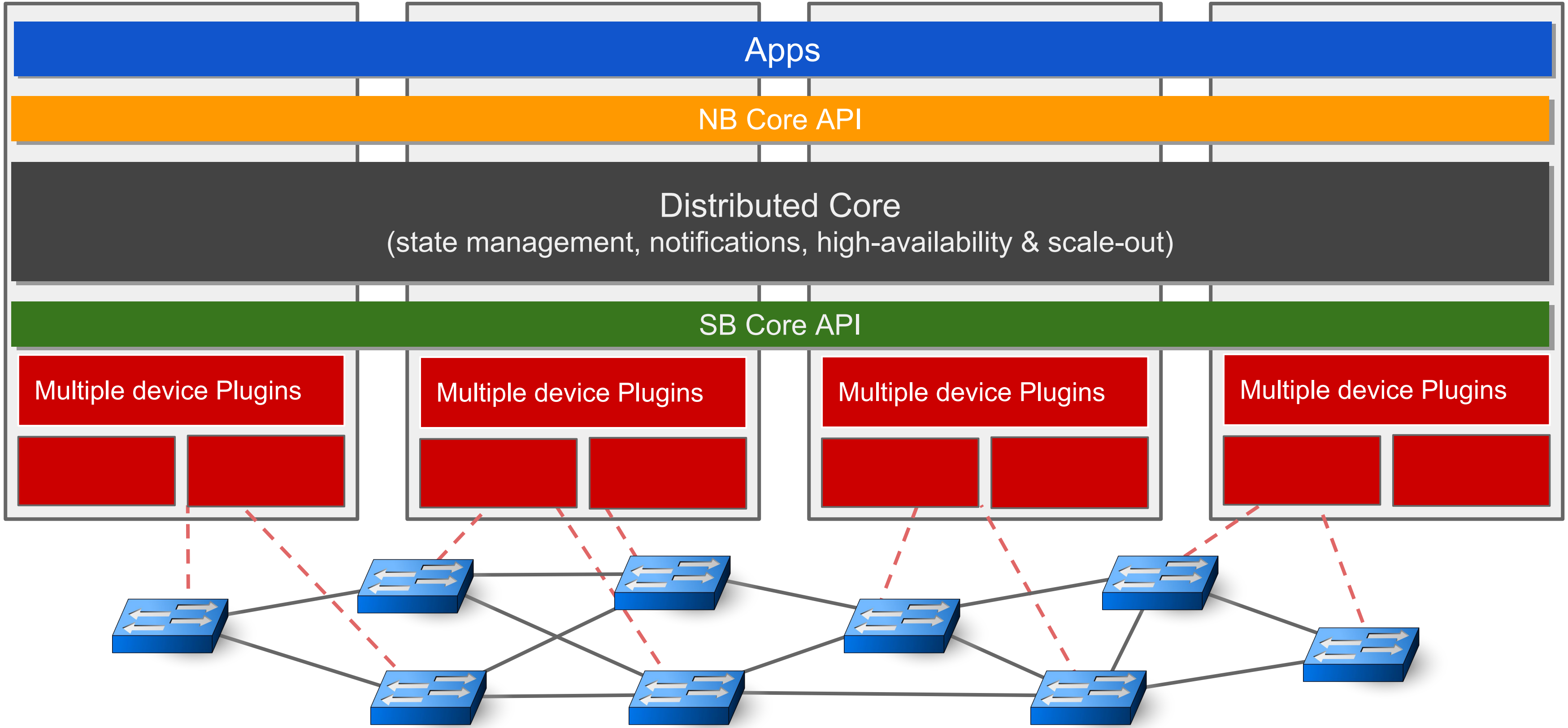
Difficult challenge!



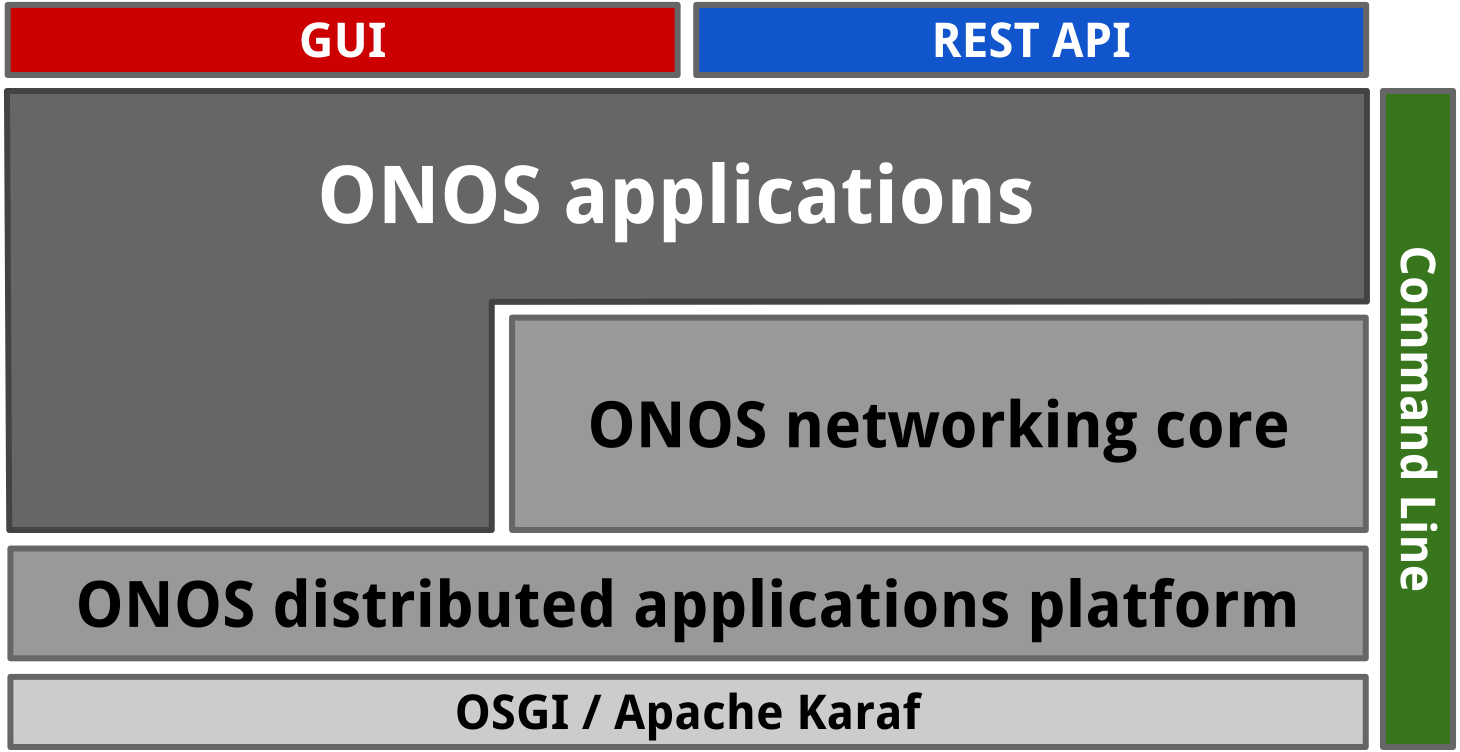
Architectural Tenets

- High-availability, scalability and performance
  - required to sustain demands of service provider & enterprise networks → valid also for datacenters
- 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

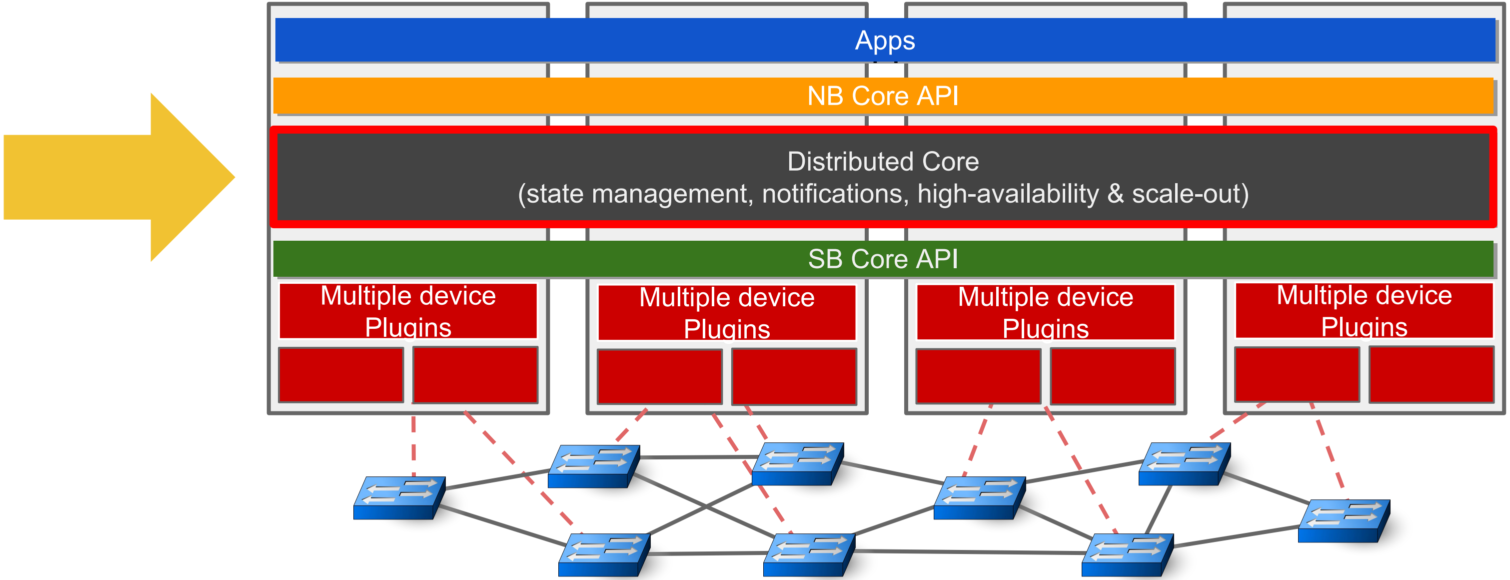
ONOS Architecture



ONOS Interfaces



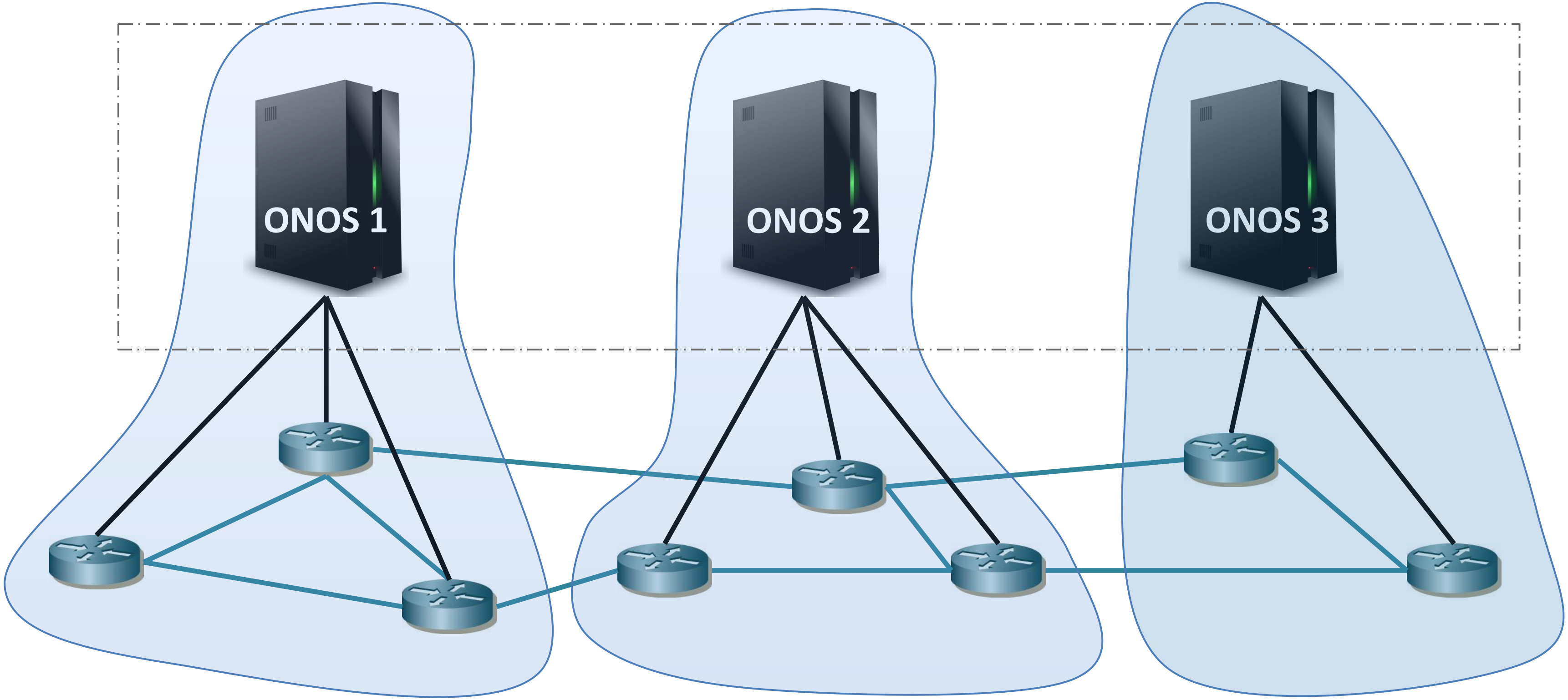
Distributed Core



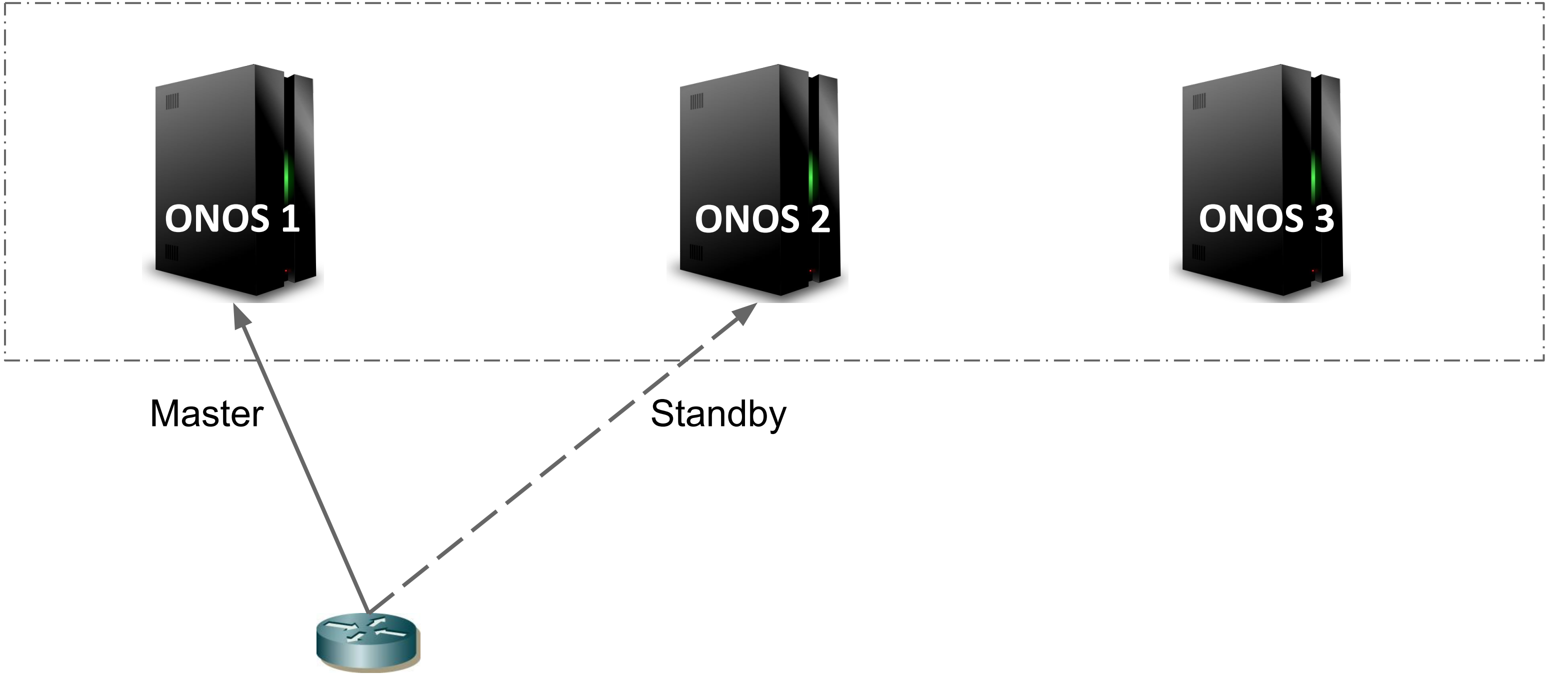
ONOS Distributed Architecture

- **Distributed** → Set up as a cluster of instances
- **Symmetric** → Each instance runs identical software and configuration
- **Fault-tolerant** → Cluster remains operational in the face of node failures
- **Location Transparent** → A client can interact with any instance. The cluster presents the abstraction of a single logical instance
- **Dynamic** → The cluster can be scaled up/down to meet usage demands
- **Raft consensus** → Replicated State Machine

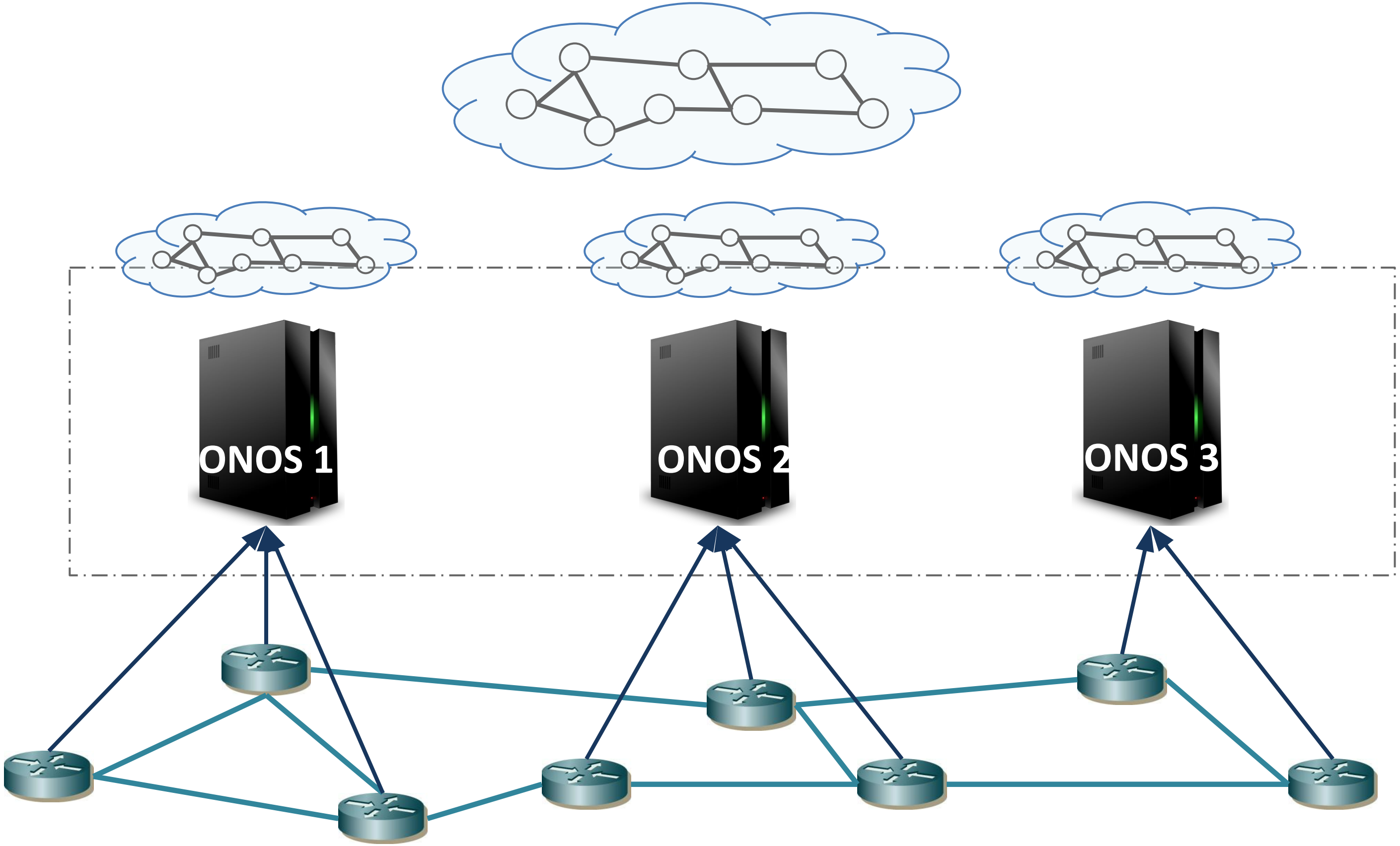
ONOS Cluster



ONOS Cluster



ONOS Cluster



ONOS Distributed Primitives

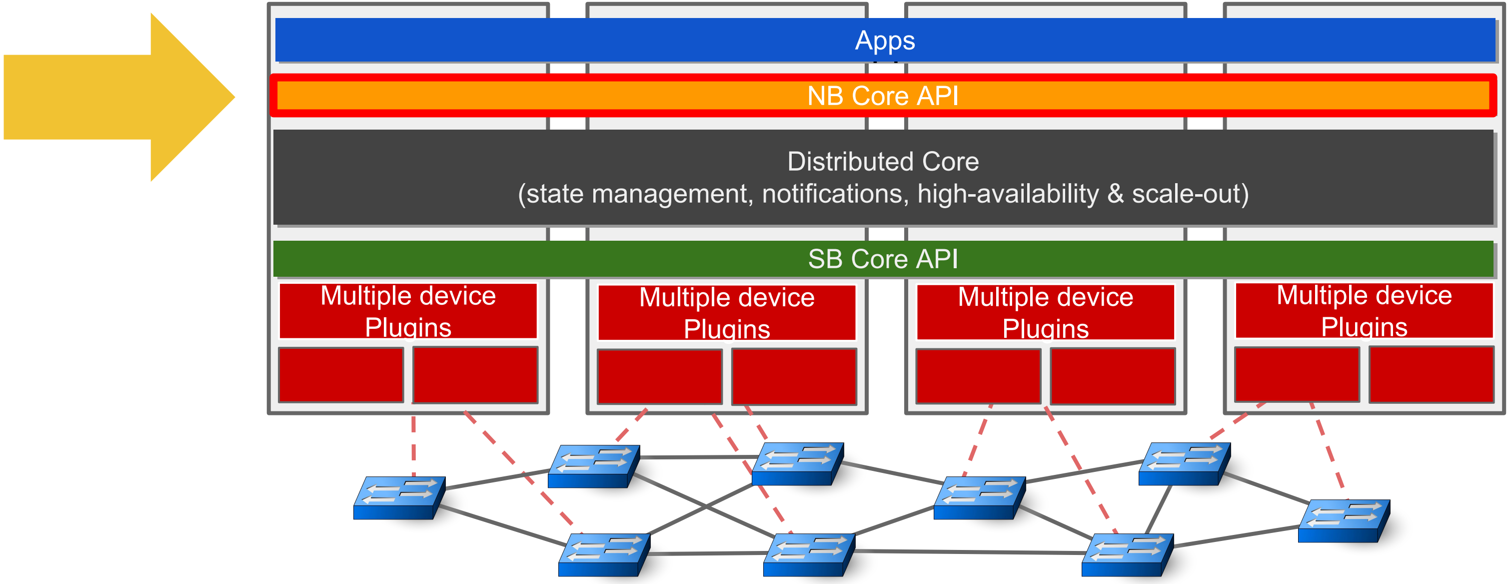
- `EventuallyConsistentMap<K, V>`
  - Map abstraction with eventual consistency guarantee
- `ConsistentMap<K, V>`
  - Map abstraction with strong linearizable consistency
- `LeadershipService`
  - Distributed Locking primitive
- `DistributedQueue<E>`
  - Distributed FIFO queue with long poll support
- `DistributedSet<E>`
  - Distributed collection of unique elements
- `AtomicCounter`
  - Distributed version of Java AtomicLong
- `AtomicValue<V>`
  - Distributed version of Java AtomicReference

State Management in ONOS

- Core platform feature
- Applications can focus on business logic
- ONOS exposes a set of primitives to cater to different use cases
- Primitives span the consistency continuum

- Eventually Consistent
  - Reads are **monotonically consistent**
- Low overhead reads and writes
  - 2-3 ms latency for reacting to network events

Northbound



Apps
NB Core API
Distributed Core
(state management, notifications, high-availability & scale-out)
SB Core API
Multiple device Plugins
Multiple device Plugins
Multiple device Plugins
Multiple device Plugins

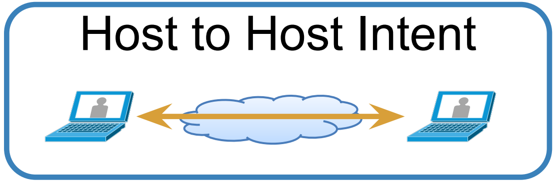
Key Northbound Abstractions

- **Network Graph**
  - Directed, cyclic graph comprising of infrastructure devices, infrastructure links and end-station hosts
- **Flow Objective**
  - Device-centric abstraction for programming data-plane flows in version and vendor-independent manner
- **Intent**
  - Network-centric abstraction for programming data-plane in topology-independent manner

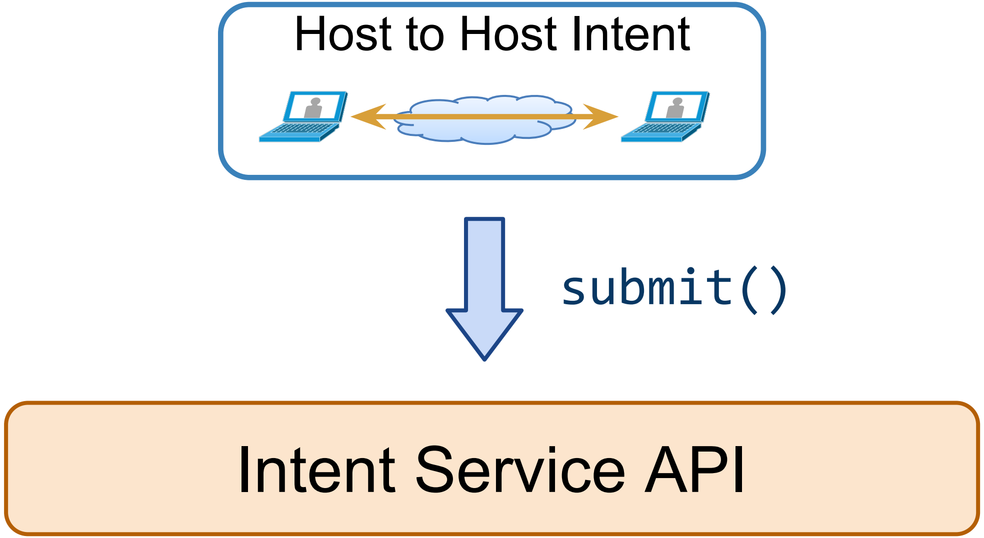
Intent Framework

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

Intent Example

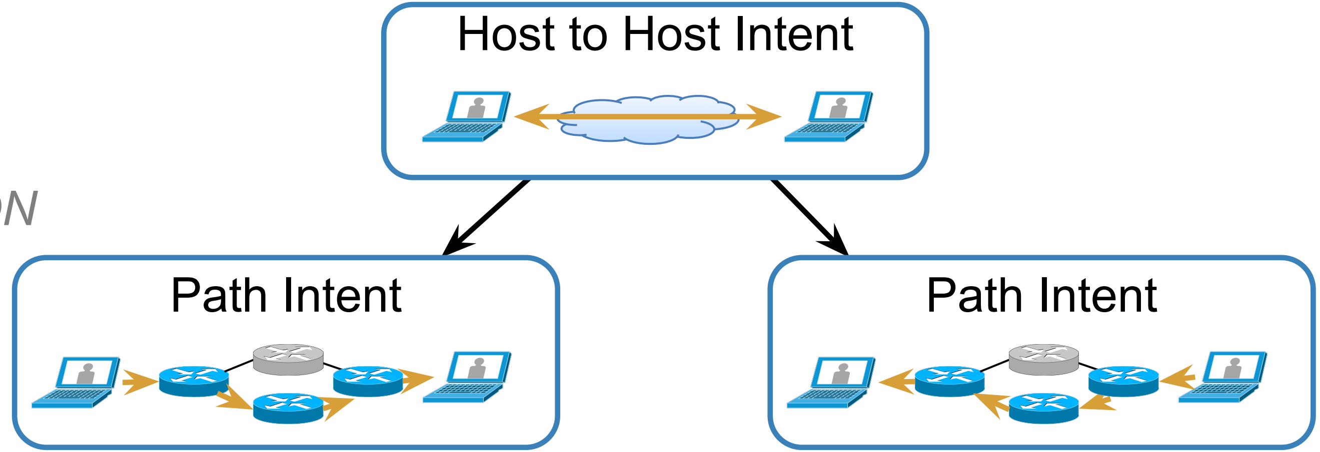


Intent Example

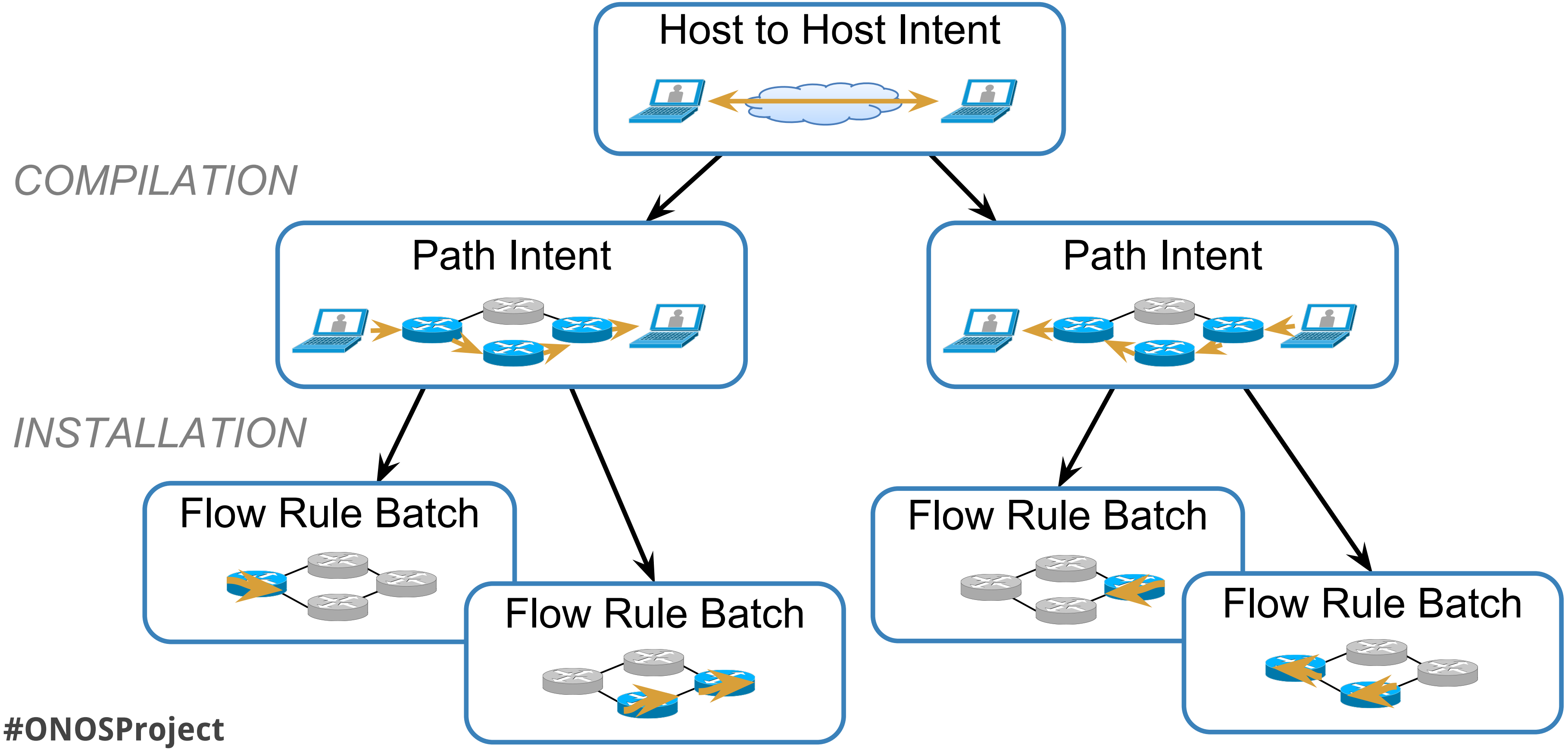


Intent Example

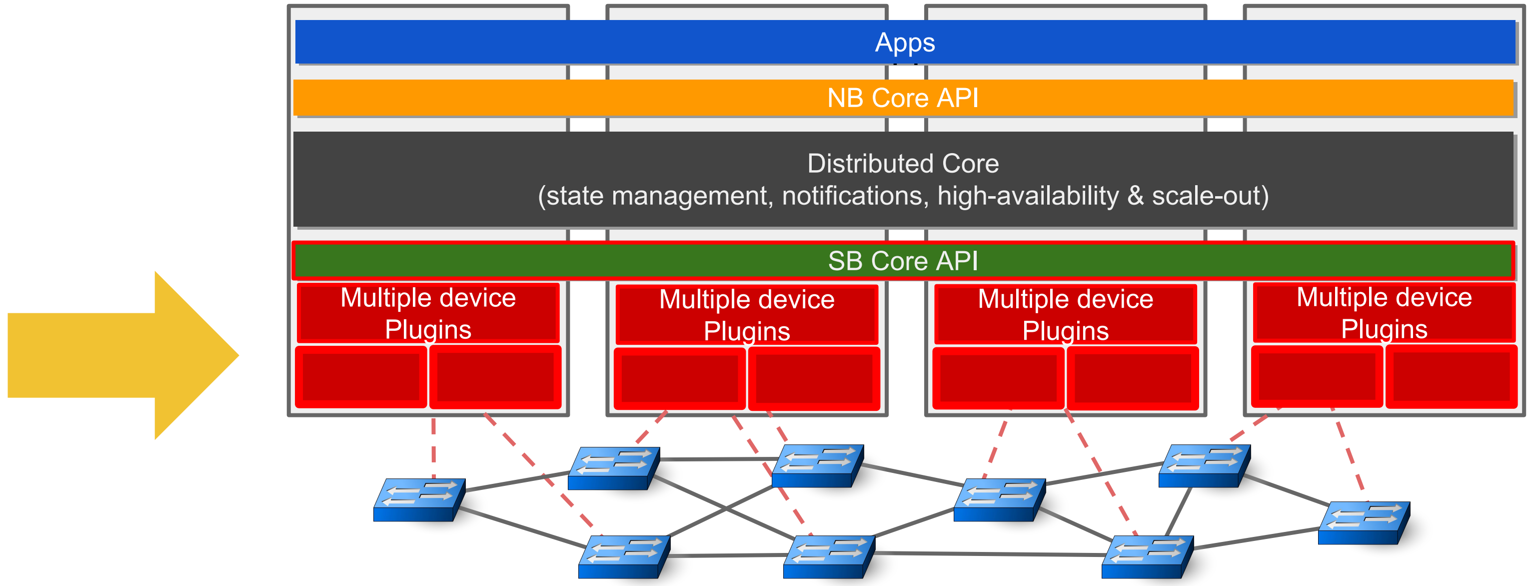
COMPILATION



Intent Example



Southbound

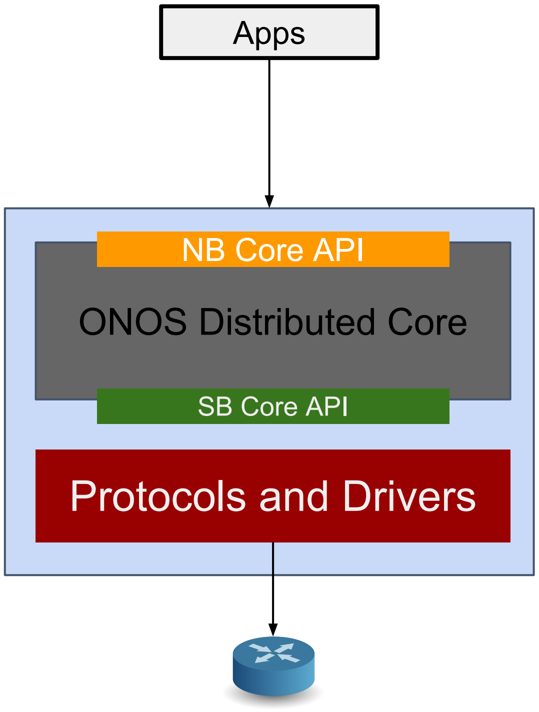


Apps
NB Core API
Distributed Core
(state management, notifications, high-availability & scale-out)
SB Core API
Multiple device Plugins
Multiple device Plugins
Multiple device Plugins
Multiple device Plugins

Southbound overview

Southbound protocols:

- OpenFlow 1.0-1.3
- OVSDB
- NETCONF + YANG
- SNMP
- P4 → bmv2
- BGP, ISIS, OSPF
- PCEP
- REST
- LISP



ONOS SB architecture outline

Driver

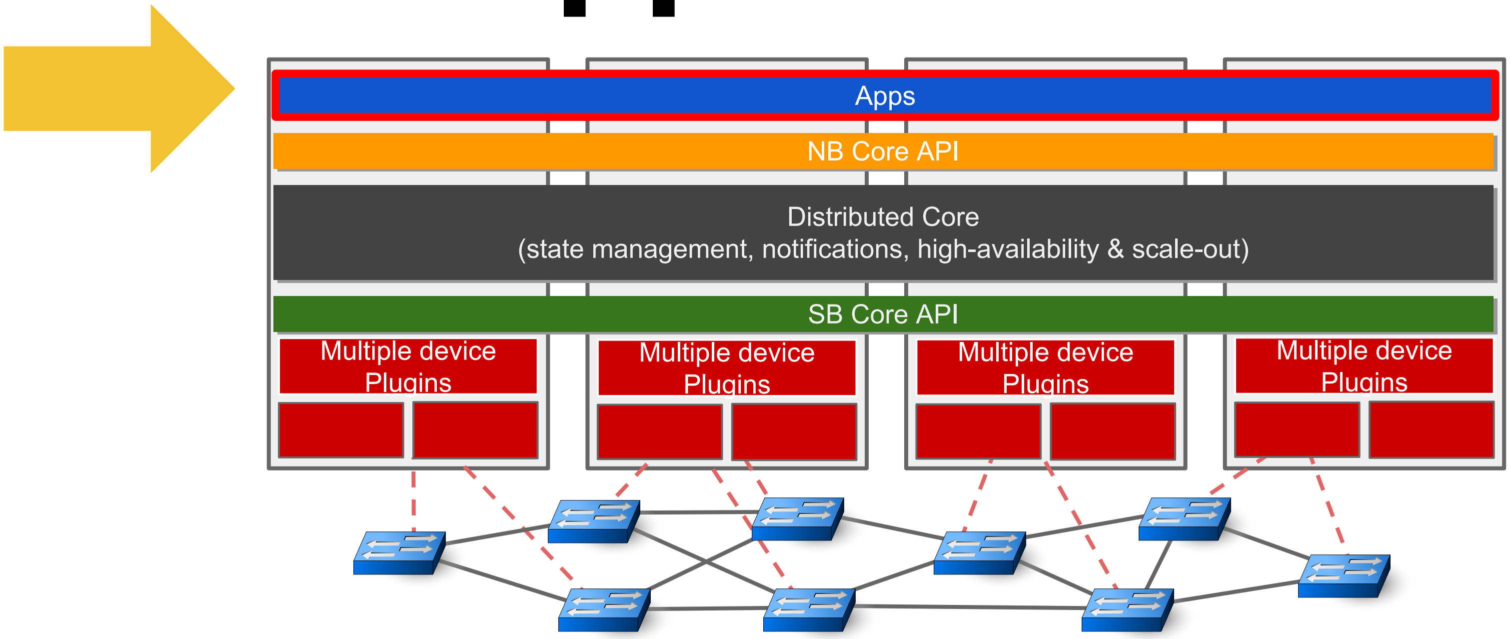
- On-demand activation
- Define device’s capabilities
- Encapsulate specific logic and code

```
<driver name="default "manufacturer="ON.Lab"
     hwVersion="0.0.1" swVersion="0.0.1">
       <behaviour api=InterfacePath
                   impl=ImpementationPath />
</driver>
```

Goals of ONOS southbound:

- Abstractions, modularity, interoperability
- Live use of new devices
- Customization without changing the core
- Hidden complexity to upper layers

Applications



Developing ONOS applications

ONOS applications:

- Interact with the northbound Java or REST interface
- Device and protocol agnostic
- Augment ONOS though modularity
- Provide GUI, REST, CLI and distributed stores.
- Shape the network.
- Easy to start with auto generated basic code via maven archetypes.

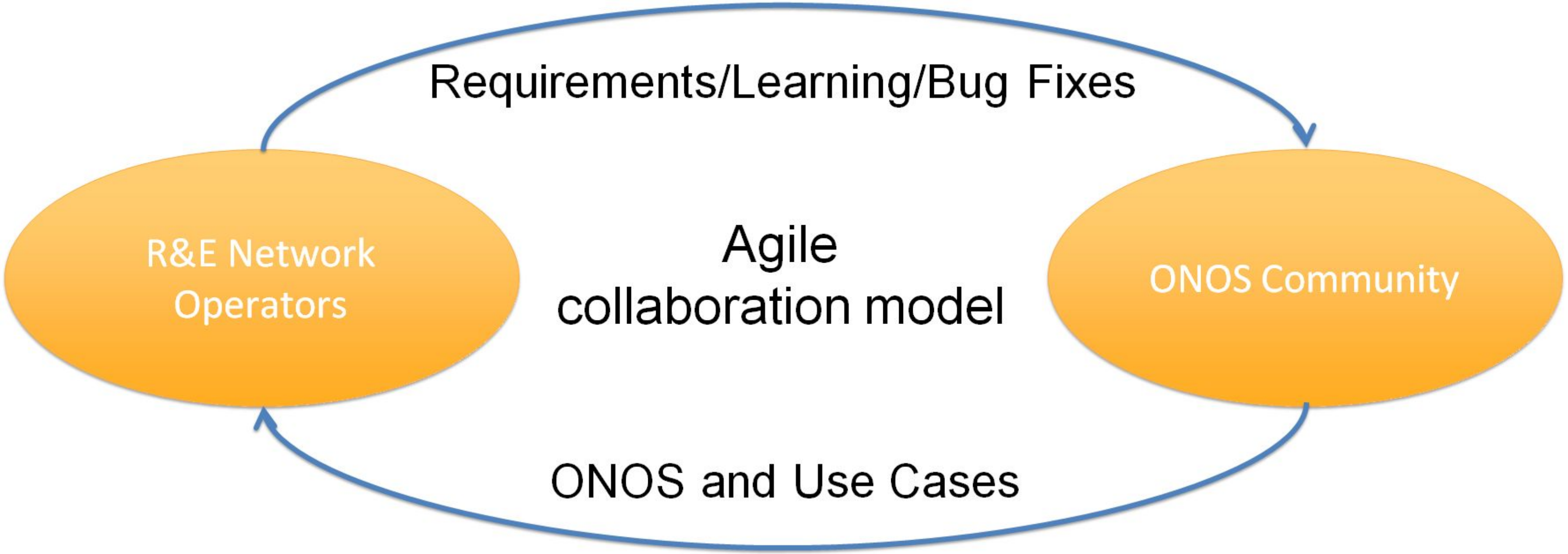
Example Applications

- SDN-IP Peering
  - Abstracts the SDN network as a BGP Autonomous System
- Video Streaming / IpTV
  - Establish multicast forwarding from a sender to set of receivers
- Virtual Network Gateway (vBNG)
  - Provide connectivity between a private host and the Internet
- Bandwidth Calendaring
  - Establish tunnels with bandwidth guarantees between two points at a given time
- Multi-level (IP / Optical) Provisioning
  - Provision optical paths/tunnels with constraints

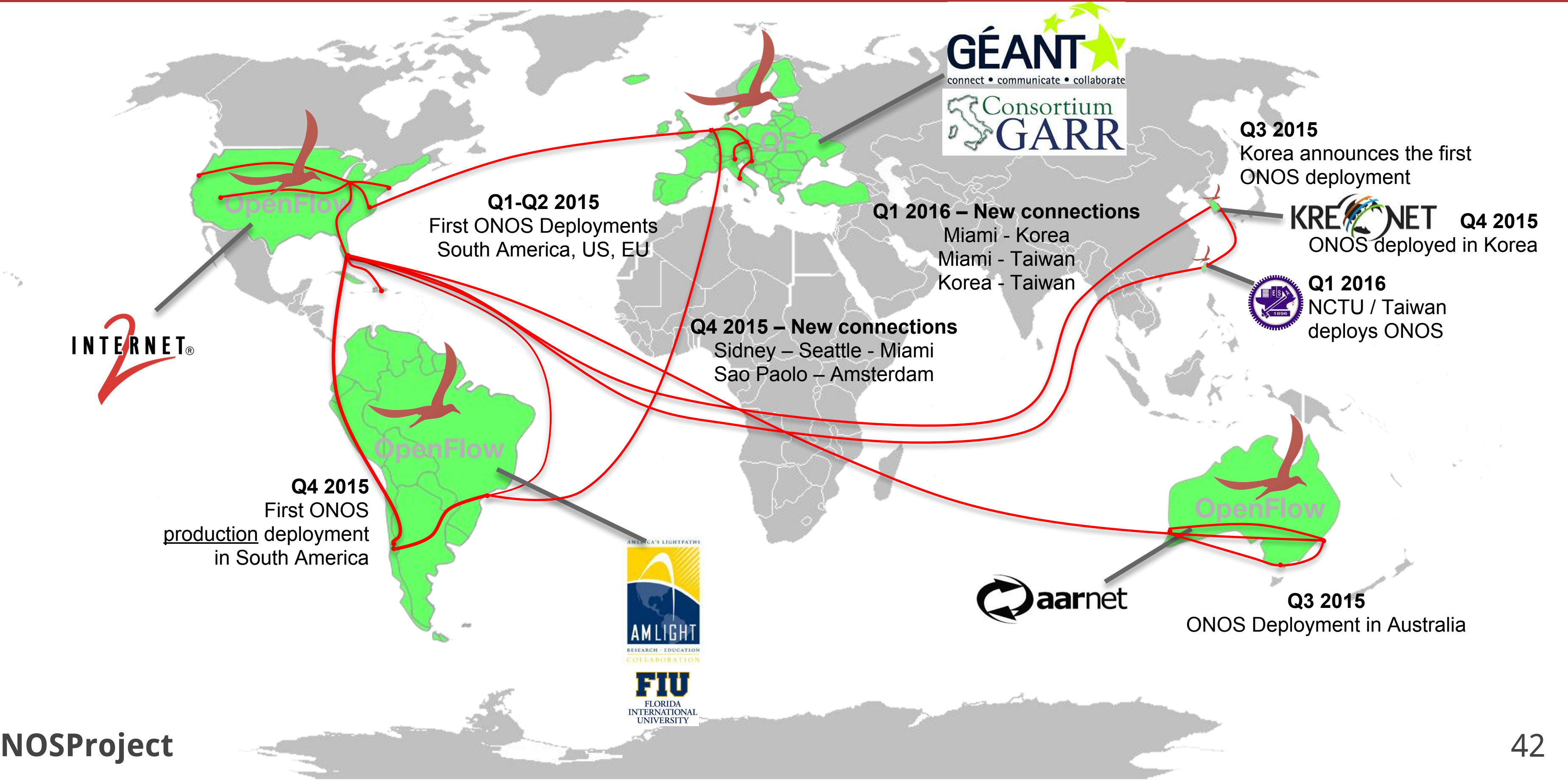
Demo

Deployments
&
Use Cases

Motivation and Goals



Global SDN Deployment Powered by ONOS



Enabling network innovation with new apps

Castor

- Provides L2/L3 connectivity for Internet Exchange Points (SDXs).
- Developed and deployed in AARNET.

SDX L2/L3

- Provides L2/L3 connectivity for Internet Exchange Points (SDXs).
- Developed and deployed by GEANT.

VPLS

- L2 broadcast overlay networks on demand.
- Ready to be deployed at AmLight.

SDN-IP

- Transforms a SDN into a transit IP network.
- SDN AS uses BGP to communicate with neighbors.
- L3 connectivity without legacy routers.
- Deployed by AmLight, Internet2 (upgrading), KREONET, NCTU.

CORD: Central Office Re-architected as a Datacenter

CORD:

- Combines SDN, NFV, Cloud with commodity infrastructure and open building blocks to deliver datacenter economies of scale and cloud-style agility to service provider networks
- Allows service providers to build an underlying common infrastructure in Central Office with white boxes, ONOS (SDN Control Plane), OpenStack (Virtual infrastructure mgmt), XOS (Services mgmt), open commodity hardware, OF-enabled OLT MAC and G.fast DPU
- Enables organizations to build the services and solutions for their customers.
- R-E-M-A variants upon the CORD platform.

CORD Mission: Deliver to Service Providers

CO is a service provider's "gateway" to its customers

- CO represents a great vantage point for a service provider: it enables new services to users!

Economies of a datacenter

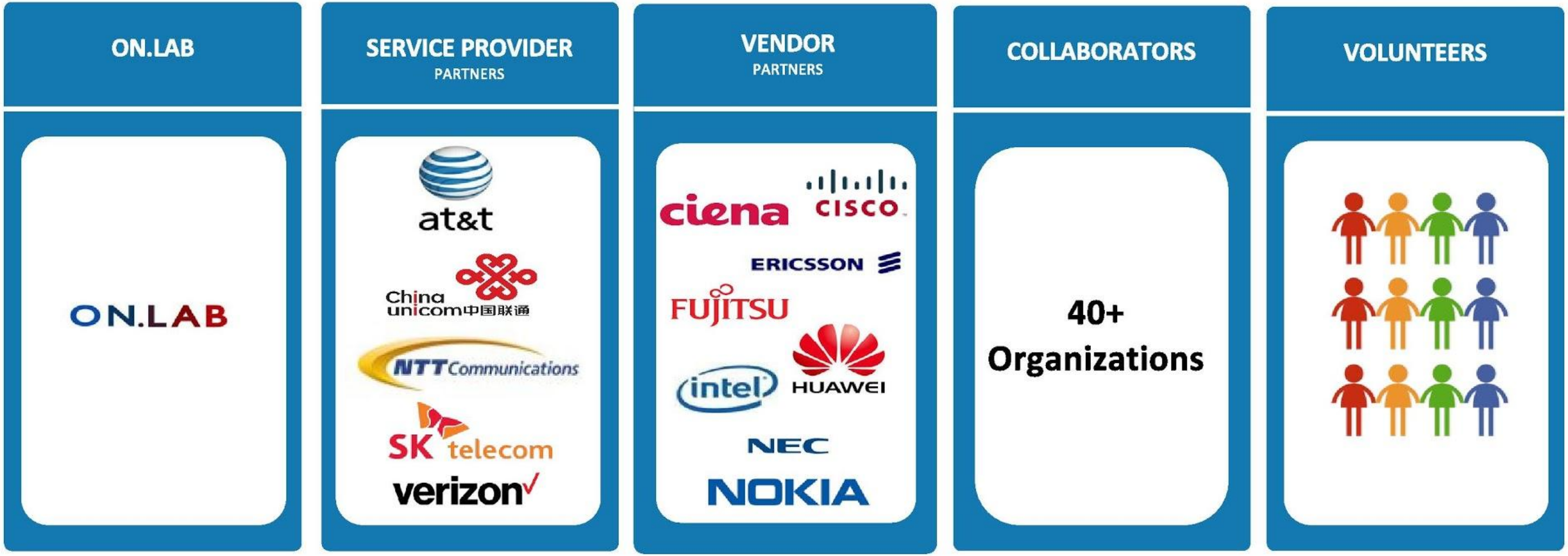
- Infrastructure built with a few commodity building blocks using open source software and white box.

Agility of a cloud provider

- Software platforms that enable rapid creation of new services.

Community

ONOS Ecosystem



- ON.Lab provides architecture shepherding and core engineering with focus
- Leading service providers make ONOS & SDN/NFV solutions relevant to them
- Leading vendors help make ONOS and SDN/NFV solutions real: ready for deployment
- Collaborating organizations help grow the community and grow the impact

Quarterly Releases

Quarterly ONOS releases:

- **Avocet** (1.0.0) - 2014-12
- **Blackbird** (1.1.0) - 2015-03
- **Cardinal** (1.2.0) - 2015-06
- **Drake** (1.3.0) - 2015-09
- **Emu** (1.4.0) - 2015-12
- **Falcon** (1.5.0) - 2016-03
- **Goldeneye** (1.6.0) - 2016-06
- **Hummingbird (1.7.0)** - 2016-09

Currently working on
Ibis - 1.8.0

How to get involved

- **Open Source software** → scratch your own itch
- **Bug Bounty** → start small with a simple bug
  - Jira bugs
- **Application or Use Case** → create your own app to deploy your use case
  - Creating and deploying and ONOS App and Template application tutorial
- **Brigades** → dynamic configuration, virtualization, GUI, deployments
  - Brigades wiki
- **Collaborator proposal** → create, use and maintain your own ONOS subsystem

Ask us:

Andrea Campanella → andrea@onlab.us

Carmelo Cascone → carmelo@onos-ambassadors.org

Andrea Biancini → andrea.biancini@onos-ambassadors.org

Further reading

ONOS website:

http://onosproject.org

Tutorials, documentation and general reading at:

https://wiki.onosproject.org/

ONOS Github:

https://github.com/opennetworkinglab/onos

Setup Tutorial

https://wiki.onosproject.org/display/ONOS/Installing+and+Running+ONOS

Screencasts:

https://wiki.onosproject.org/display/ONOS/Screencasts

ONOS

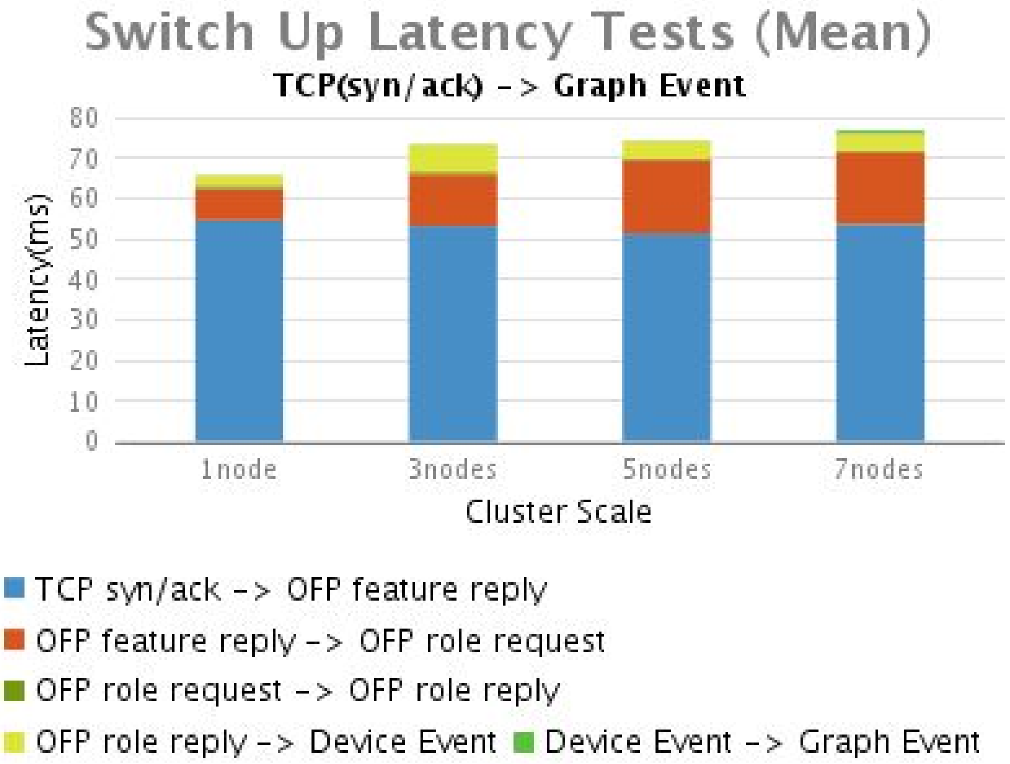
Open Network Operating System

Software Defined Transformation of Service Provider Networks

Join the journey @ onosproject.org

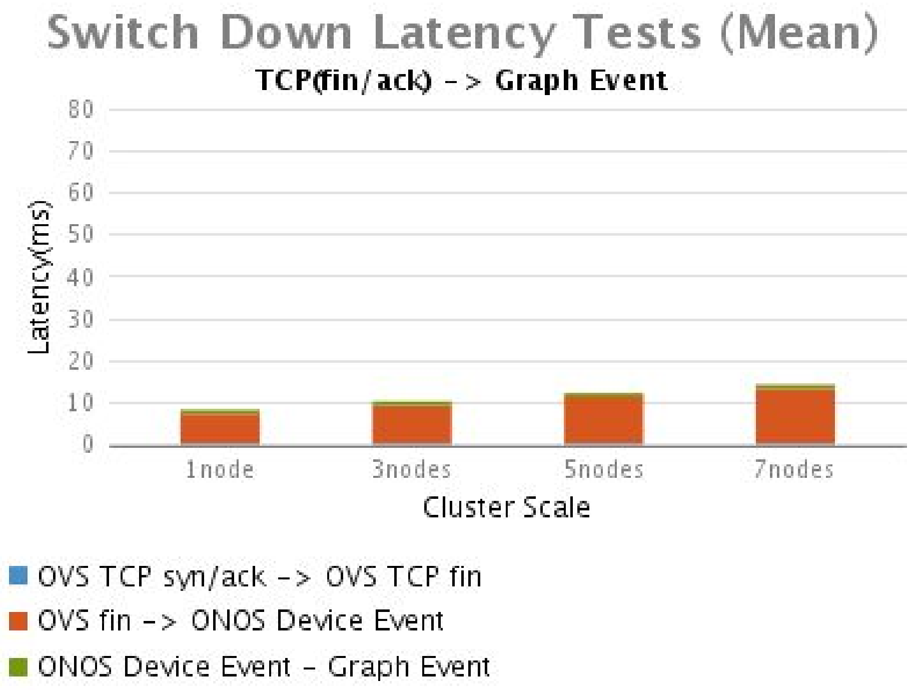
Performance

Switch Up Latency



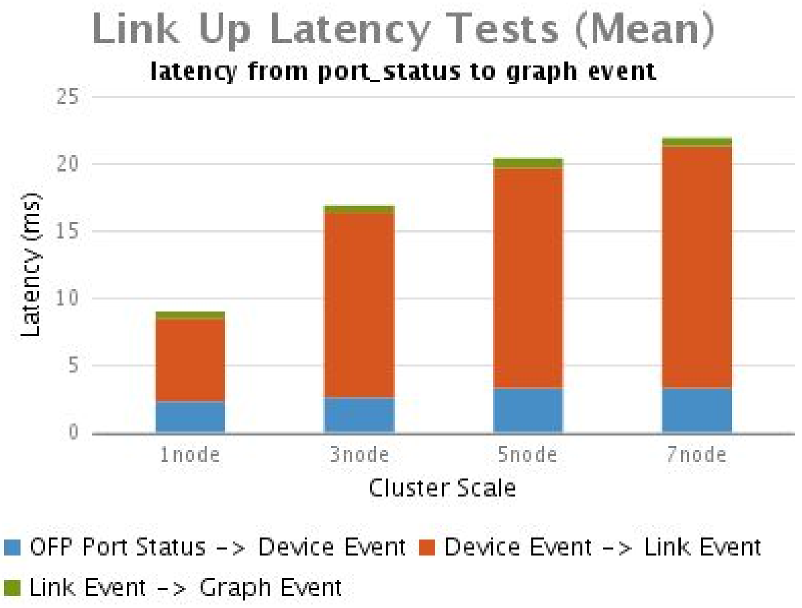
- Most of the time is spent waiting for the switch to respond to a features request. (~53ms)
- ONOS spends under 25ms with most of it's time electing a master for the device.
  - Which is a strongly consistent operation

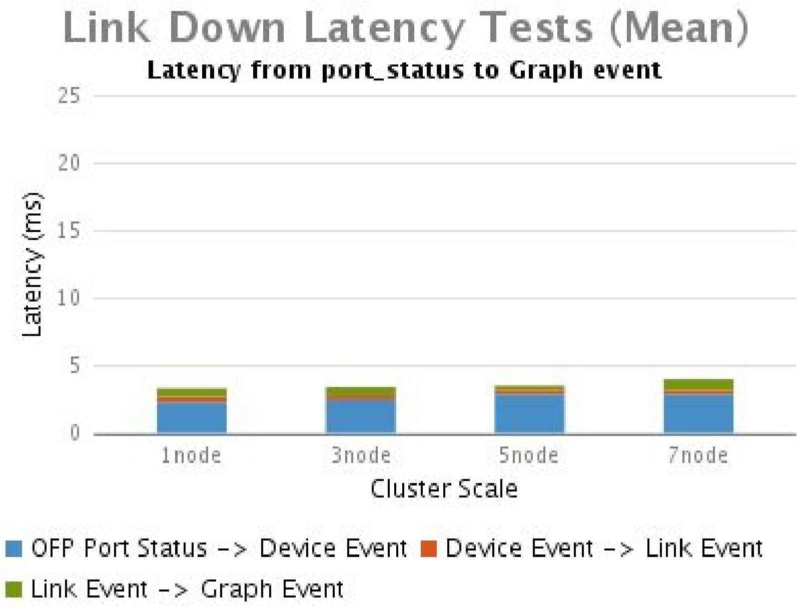
Switch Down Latency



- Significantly faster because there is no negotiation with the switch
- A terminating TCP connection unequivocally indicates that the switch is gone

Link Up/Down Latency

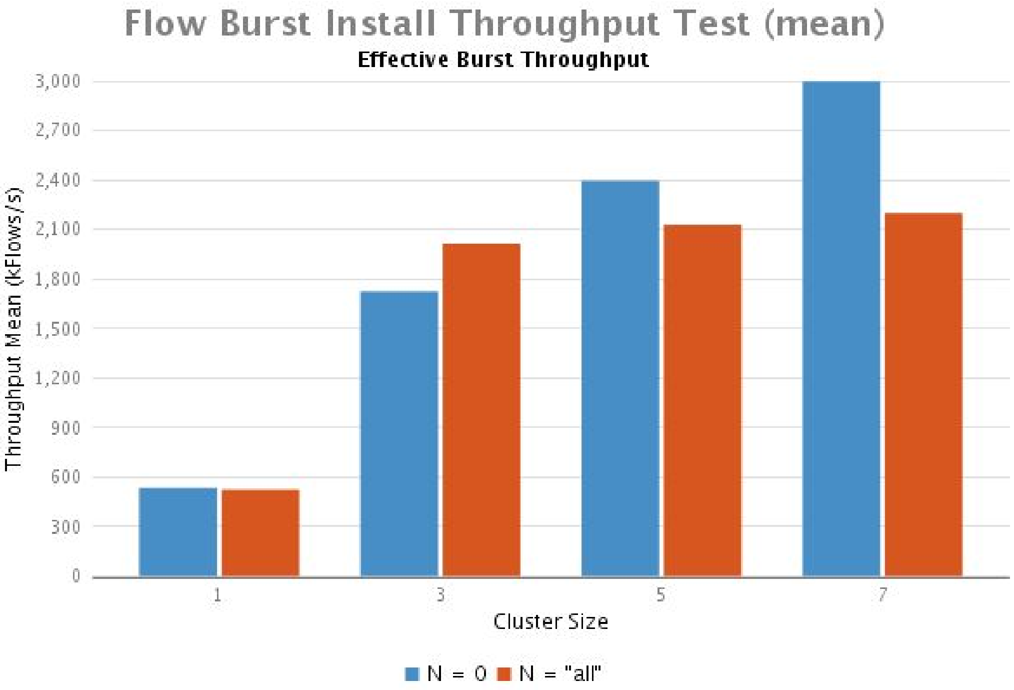




- The increase from single to multi instance is being investigated
- Since we use LLDP to discover links, it takes longer to discover a link coming up than going down

- Port down event trigger immediate teardown of the link.

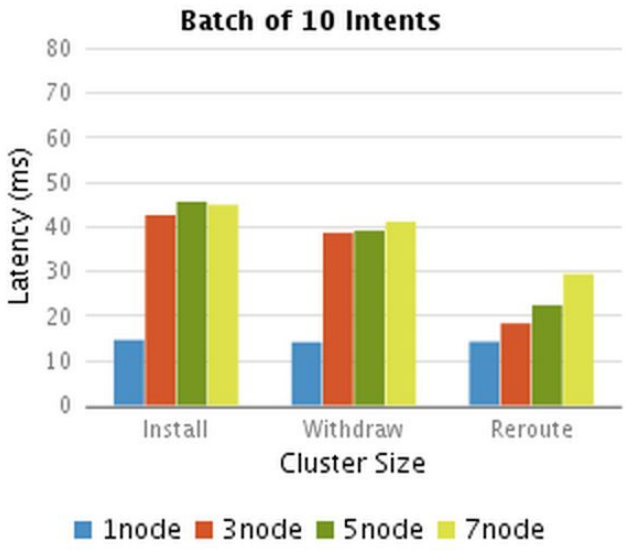
Flow Throughput results

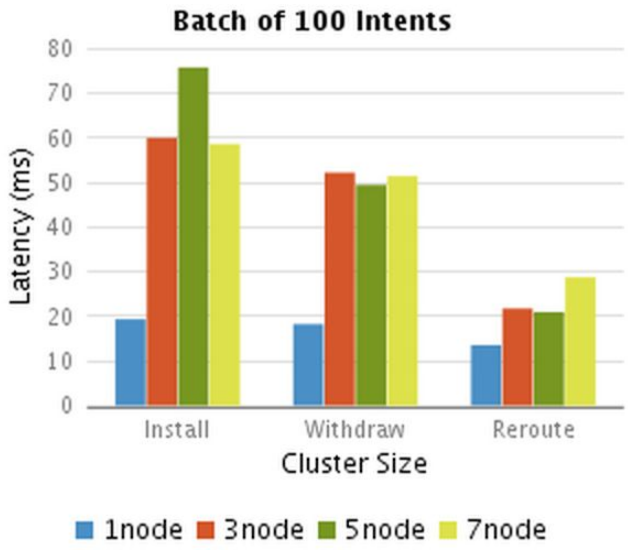


- Single instance can install over 500K flows per second
- ONOS can handle 3M local and 2M non local flow installations
- With 1-3 ONOS instances, the flow setup rate remains constant no matter how many neighbours are involved
- With more than 3 instances injecting load the flow performance drops off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

Intent 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





Intent Throughput Results

● Process