ONOS: SDN OS for Service Provider Networks

- Scalability, High Availability & Performance
- Northbound & Southbound Abstractions
- Modularity
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
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!
Distributed Architecture

Apps

NB Core API

Distributed Core
(state management, notifications, high-availability & scale-out)

SB Core API

Adapters

Protocols

Protocols

Protocols

Protocols

Apps
Distributed Architecture

Apps

NB Core API

Distributed Core
(state management, notifications, high-availability & scale-out)

SB Core API

Adapters

Protocols

Apps

Protocols

Adapters

Protocols

Adapters

Protocols

Adapters

Protocols
ONOS Architecture Tiers

Northbound Abstraction:
- network graph
- application intents

Core:
- distributed
- protocol independent

Southbound Abstraction:
- generalized OpenFlow
- pluggable & extensible

Northbound - Application Intent Framework
(policy enforcement, conflict resolution)

Distributed Core
(scaleability, availability, performance, persistence)

Southbound
(discover, observe, program, configure)

Apps

OpenFlow  NetConf  ...

Diagram of network nodes connected with arrows indicating data flow.
Application Intent Framework

- Application specifies high-level intents; not low-level rules
  - focus on *what* should be done, rather than *how* it should be done
- Intents are compiled into actionable objectives which are installed into the environment
  - e.g. *HostToHostIntent* compiles into two *PathIntents*
- Resources required by objectives are then monitored
  - e.g. link vanishes, capacity or lambda becomes available
- Intent subsystem reacts by recompiling intent and re-installing revised objectives
Distributed Core

- Distributed state management framework
  - built for high-availability and scale-out

- Different types of state require different types of synchronization
  - fully replicated
  - master / slave replicated
  - partitioned / distributed

- Novel topology replication technique
  - *logical* clock in each instance timestamps events observed in underlying network
  - *logical* timestamps ensure state evolves in consistent and *ordered* fashion
  - allows rapid convergence without complex coordination
  - applications receive notifications about topology changes
ONOS Core Subsystem Structure
Modularity Objectives

● Increase architectural coherence, testability and maintainability
  o establish tiers with crisply defined boundaries and responsibilities
  o setup code-base to follow and enforce the tiers

● Facilitate extensibility and customization by partners and users
  o unit of replacement is a module

● Avoid speciation of the ONOS code-base
  o APIs setup to encourage extensibility and community code contributions

● Preempt code entanglement, i.e. cyclic dependencies
  o reasonably small modules serve as firewalls against cycles

● Facilitate pluggable southbound
ONOS Modules

- Well-defined relationships
- Basis for customization
- Avoid cyclic dependencies
ONOS 1.0.0 Release Priorities

- Release ONOS with coherent and modular architecture
- Enable and demonstrate high-availability operation
- Enable sustained pursuit of performance and scale objectives
- Enable development of apps and engagement of SDN community
- Demonstrate SDN-IP & Packet-Optical use cases
- User Interface
ONOS Priorities for Feb. Release

● Prove out performance at scale
  o current release falls short in our own view
  o testing with real hardware
  o provide comprehensive assessment

● Continue to increase robustness
  o defects, edge-cases, additional failure scenarios
  o continuous testing framework

● Segment Routing use-case
  o port to the released version of ONOS

● REST API & Security

● Support for multiple-tables
Performance Objectives

● Throughput of proactive provisioning actions
  o path flow provisioning
  o global optimization or rebalancing of existing path flows

● Latency of responses to topology changes
  o path repair in wake of link or device failures

● Throughput of distributing and aggregating state
  o batching, caching, parallelism
  o dependency reduction

● Controller vs. Device Responsibilities
  o defer to devices to do what they do best, e.g. low-latency reactivity, backup paths
What’s available in ONOS today?

- **ONOS with all its key features**
  - high-availability, scalability*, performance*
  - northbound abstractions (application intents)
  - southbound abstractions (OpenFlow adapters)
  - modular code-base
  - GUI

- **Open source**
  - ONOS code-base on GitHub
  - documentation & infrastructure processes to engage the community

- **Use-case demonstrations**
  - SDN-IP, Packet-Optical

- **Sample applications**
  - reactive forwarding, mobility, proxy arp