A bit of history

- OpenFlow Controller
- SDN Controller
- SDN Framework
- Network Operating System
A bit of history

NOX (C++ & Python)
  NOX-MT (C++)
  POX (Python)
  Trema (Ruby & C)
  Beacon -> Floodlight (Java)
Ryu (Python)
  OpenDayLight (Java + OSGi)
  ONOS (Java + OSGi)

2008  2010  2011  2012  2013  2014
Background

• Networks have been managed and configured using
  - lower level, device-specific instruction sets
  - mostly closed proprietary NOSs
    ✓ e.g., Cisco IOS and Juniper JunOS

• SDN by means of the logically centralized control offered by a NOS is promised to
  - facilitate network management
  - ease the burden of solving networking problems

• With NOSs, to define network policies
  - a developer no longer needs to care about the low-level details of data distribution among routing elements
How many flows exist in real networks/datacenters?

• NOX
  - handles around 30k flow initiation events per second
  - while maintaining a sub-10ms flow install time.

• Kandula et al. found that
  - a 1500-server cluster has a median flow arrival rate of 100k flows per second.

• Benson et al. show that
  - a network with 100 switches can have spikes of 10M flows arrivals per second in the worst case.
Centralized Controllers

• A centralized controller
  - a single entity that manages all forwarding devices of the network
  - naturally, a single point of failure and with scaling limitations

• Centralized controllers are designed as highly concurrent systems to achieve required throughput
  - i.e., multithreaded design for multicore computer

• Beacon can deal with more than 12 million flows per second by using Amazon cloud service.

• List of centralized controllers:
  - NOX-MT, Maestro, Beacon, Floodlight, Trema, Ryu, Meridian, ProgrammableFlow, Rosemary
Effect of Multi-threading on Throughput

Distributed Controllers

• A distributed NOS can be scaled up to meet the requirements of potentially any environment.

• Most distributed controllers offer weak consistency semantics
  - which implies that there is a period of time in which distinct nodes may read different values

• Another common property is fault tolerance.
  - When one node fails, another neighbor node should take over the duties and devices of the failed node.

• SDN resiliency as a whole is an open challenge.

• List of distributed controllers:
  - Onix, HyperFlow, HP VAN SDN, ONOS, DISCO, yanc, PANE, SMaRt-Light, Fleet
## Architectural and Design Elements of SDN Controllers

<table>
<thead>
<tr>
<th>Component</th>
<th>OpenDaylight</th>
<th>OpenContrail</th>
<th>HP VAN SDN</th>
<th>Onix</th>
<th>Beacon</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base network services</strong></td>
<td>Topology/Stats/Switch Manager, Host Tracker, Shortest Path Forwarding</td>
<td>Routing, Tenant Isolation</td>
<td>Audit Log, Alerts, Topology, Discovery</td>
<td>Discovery, Multi-consistency Storage, Read State, Register for updates</td>
<td>Topology, device manager, and routing</td>
</tr>
<tr>
<td><strong>East/Westbound APIs</strong></td>
<td>—</td>
<td>Control Node (XMPP-like control channel)</td>
<td>Sync API</td>
<td>Distribution I/O module</td>
<td>Not present</td>
</tr>
<tr>
<td><strong>Integration Plug-ins</strong></td>
<td>OpenStack Neutron</td>
<td>CloudStack, OpenStack</td>
<td>OpenStack</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>Management Interfaces</strong></td>
<td>GUI/CLI, REST API</td>
<td>GUI/CLI</td>
<td>REST API Shell / GUI Shell</td>
<td>—</td>
<td>Web</td>
</tr>
<tr>
<td><strong>Northbound APIs</strong></td>
<td>REST, REST-CONF [200], Java APIs</td>
<td>REST APIs (configuration, operational, and analytic)</td>
<td>REST API, GUI Shell</td>
<td>Onix API (general purpose)</td>
<td>API (based on OpenFlow events)</td>
</tr>
<tr>
<td><strong>Service abstraction layers</strong></td>
<td>Service Abstraction Layer (SAL)</td>
<td>—</td>
<td>Device Abstraction API</td>
<td>Network Information Base (NIB) Graph with Import/Export Functions</td>
<td>—</td>
</tr>
<tr>
<td><strong>Southbound APIs or connectors</strong></td>
<td>OpenFlow, OVSD, SNMP, PCEP, BGP, NETCONF</td>
<td>—</td>
<td>OpenFlow, L3 Agent, L2 Agent</td>
<td>OpenFlow, OVSD</td>
<td>OpenFlow</td>
</tr>
<tr>
<td>Name</td>
<td>Architecture</td>
<td>Northbound API</td>
<td>Consistency</td>
<td>Faults</td>
<td>License</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------</td>
<td>----------------</td>
<td>-------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Beacon [186]</td>
<td>centralized multi-threaded</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPLv2</td>
</tr>
<tr>
<td>DISCO [185]</td>
<td>distributed</td>
<td>REST</td>
<td>—</td>
<td>yes</td>
<td>—</td>
</tr>
<tr>
<td>ElastiCon [228]</td>
<td>distributed</td>
<td>RESTful API</td>
<td>yes</td>
<td>no</td>
<td>—</td>
</tr>
<tr>
<td>Fleet [199]</td>
<td>distributed</td>
<td>ad-hoc</td>
<td>no</td>
<td>no</td>
<td>—</td>
</tr>
<tr>
<td>Floodlight [189]</td>
<td>distributed</td>
<td>RESTful API</td>
<td>weak</td>
<td>yes</td>
<td>—</td>
</tr>
<tr>
<td>HP VAN SDN [184]</td>
<td>distributed</td>
<td>RESTful API</td>
<td>weak</td>
<td>yes</td>
<td>C++</td>
</tr>
<tr>
<td>HyperFlow [195]</td>
<td>distributed</td>
<td>weak</td>
<td>yes</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Kandoo [229]</td>
<td>hierarchically distributed</td>
<td>—</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Onix [7]</td>
<td>distributed</td>
<td>VNP NBAPI</td>
<td>weak, strong</td>
<td>yes</td>
<td>commercial</td>
</tr>
<tr>
<td>Maestro [188]</td>
<td>centralized multi-threaded</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>LGPLv2.1</td>
</tr>
<tr>
<td>Meridian [192]</td>
<td>centralized multi-threaded</td>
<td>extensible API layer</td>
<td>no</td>
<td>no</td>
<td>—</td>
</tr>
<tr>
<td>MobileFlow [222]</td>
<td>distributed</td>
<td>SDMN API</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Mul. [230]</td>
<td>centralized multi-threaded</td>
<td>multi-level interface</td>
<td>no</td>
<td>no</td>
<td>GPLv2</td>
</tr>
<tr>
<td>NOX [26]</td>
<td>centralized</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPLv3</td>
</tr>
<tr>
<td>NOX-MT [187]</td>
<td>centralized multi-threaded</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPLv3</td>
</tr>
<tr>
<td>NVP Controller [112]</td>
<td>distributed</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>commercial</td>
</tr>
<tr>
<td>OpenContrail [183]</td>
<td>distributed</td>
<td>REST API</td>
<td>no</td>
<td>no</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>OpenDaylight [13]</td>
<td>distributed</td>
<td>REST, RESTCONF</td>
<td>weak</td>
<td>no</td>
<td>EPL v1.0</td>
</tr>
<tr>
<td>ONOS [117]</td>
<td>distributed</td>
<td>RESTful API</td>
<td>weak, strong</td>
<td>yes</td>
<td>—</td>
</tr>
<tr>
<td>PANE [197]</td>
<td>distributed</td>
<td>PANE API</td>
<td>yes</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>POX [231]</td>
<td>centralized</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPLv3</td>
</tr>
<tr>
<td>ProgrammableFlow [232]</td>
<td>distributed</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>C</td>
</tr>
<tr>
<td>Rosemary [194]</td>
<td>centralized</td>
<td>ad-hoc</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Ryu NOS [191]</td>
<td>centralized multi-threaded</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>SMuRtLight [198]</td>
<td>distributed</td>
<td>RESTful API</td>
<td>yes</td>
<td>yes</td>
<td>—</td>
</tr>
<tr>
<td>SNAC [233]</td>
<td>centralized</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPL</td>
</tr>
<tr>
<td>Trena [190]</td>
<td>centralized multi-threaded</td>
<td>ad-hoc API</td>
<td>no</td>
<td>no</td>
<td>GPLv2</td>
</tr>
<tr>
<td>Unified Controller [171]</td>
<td>distributed</td>
<td>REST API</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>yunc [196]</td>
<td>distributed</td>
<td>file system</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Many Different SDN Controllers

- NOX/POX
- Ryu
- Floodlight
- OpenDaylight
- Pyretic
- Frenetic
- Procera
- RouteFlow
- Trema

<table>
<thead>
<tr>
<th>Controller</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ryu (NTT)</td>
<td>• Apache license</td>
</tr>
<tr>
<td></td>
<td>• Python</td>
</tr>
<tr>
<td>NOX/POX (ONRC)</td>
<td>• GPL</td>
</tr>
<tr>
<td></td>
<td>• C++ and Python</td>
</tr>
<tr>
<td>Beacon (Stanford Univ.)</td>
<td>• BSD-like license</td>
</tr>
<tr>
<td></td>
<td>• Java-based</td>
</tr>
<tr>
<td>Maestro (Rice Univ.)</td>
<td>• GPL</td>
</tr>
<tr>
<td></td>
<td>• Based on Java</td>
</tr>
<tr>
<td>Trema (NEC)</td>
<td>• GPL 2.0</td>
</tr>
<tr>
<td></td>
<td>• Written in C and Ruby</td>
</tr>
<tr>
<td>Floodlight (Big Switch)</td>
<td>• Apache license</td>
</tr>
<tr>
<td></td>
<td>• Java-based</td>
</tr>
<tr>
<td>OpenDayLight (Linux Foundation)</td>
<td>• Eclipse Public License</td>
</tr>
<tr>
<td></td>
<td>• Java-based</td>
</tr>
</tbody>
</table>
NOX: Overview

• First-generation OpenFlow controller
  - Open source, stable, widely used

• Two “flavors” of NOX
  - NOX-Classical: C++/Python
    ✓ No longer supported
  - NOX: the “new NOX”
    ✓ C++ only
    ✓ Fast, clean codebase
    ✓ Well maintained and supported (?)

http://www.noxrepo.org/
NOX: Characteristics

• Users implement control in C++
• Supports OpenFlow v.1.0
  - A fork (CPqD) supports 1.1, 1.2, and 1.3
• Programming model
  - Controller registers for events
  - Programmer writes event handler

When to Use NOX
• You know C++
• You are willing to use low-level facilities and semantics of OpenFlow
• You need good performance
POX: Overview

• NOX in Python
  - Supports Open Flow v.1.0 only

• Advantages
  - Widely used, maintained, supported
  - Relatively easy to read and write code

• Disadvantages
  - Performance

When to Use POX
• You know Python
• You are not concerned about controller performance
• Rapid prototyping and experimentation
Ryu

- Open source Python controller
  - Supports OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5, Nicira extensions
  - WorkswithOpenStack

- Aims to be an “Operating System” for SDN

- Advantages
  - Open Stack integration
  - OpenFlow 1.2, 1.3, 1.4, 1.5
  - Good documentation

- Disadvantages:
  - Performance

http://osrg.github.io/ryu/

Ryu is pronounced "ree-yooh".
Floodlight

http://www.projectFloodlight.org/floodlight/

- Open-source Java controller
- Supports OpenFlow v.1.0 and v.1.3
- Fork from the Beacon Java OpenFlow controller
- Maintained by Big Switch Networks

Advantages
- Good documentation
- Integration with REST API
- Production-level, OpenStack/Multi-Tenant Clouds

Disadvantages:
- Steep learning curve
OpenDaylight Consortium

• Heavy industry involvement and backing

• Focused on having an open framework for building upon SDN/NFV innovations
  - Not limited to OpenFlow innovations
Hydrogen Release
Beryllium Release
Java, Maven, OSGi, Interface

• Java chosen as an enterprise-grade, cross-platform compatible language
• Maven – build system for Java
• Open Service Gateway Initiative (OSGi):
  - Allows dynamically loading bundle
  - Allows registering dependencies and services exported
  - For exchanging information across bundles
• Java Interfaces are used for event listening, specifications, and forming patterns
OpenDaylight Web Interface
Importance of ODL

• Advantages:
  - Supports a wide variety of SBI protocols versions
  - Active community
  - Aligned with vendors and telcos
  - Easy proposal of projects
  - Easy deployment (OSGi)

• Disadvantages:
  - Not so good documentation:
  - Development of modules requires a deep knowledge of ODL

ONOS – A Carrier Grade Controller

• Features of Open Network Operating System (ONOS)
• Highly available
• Modular
• extensible
• Distributed
• Scalable
• multi-protocol controller infrastructure
• Supports OpenFlow(1.0,1.3), NETCONF, OVSDB
• Protocol and device behavior independence
  - Written in Java
  - Apache 2.0 license
ONOS Board

• Service providers
  - AT&T, ChinaUnicom, NTT Communications, SK Telecom, Verizon

• Vendors
  - Alcatel Lucent, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, NEC, ON.Lab
ONOS Architecture
## ONOS Releases

<table>
<thead>
<tr>
<th>Release Name</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avocet 1.0.0</td>
<td>December 5, 2014</td>
</tr>
<tr>
<td>Blackbird 1.1.0</td>
<td>February 28, 2015</td>
</tr>
<tr>
<td>Cardinal 1.2.0</td>
<td>May 31, 2015</td>
</tr>
<tr>
<td>Drake</td>
<td>September 18, 2015</td>
</tr>
<tr>
<td>Emu</td>
<td>December 18, 2015</td>
</tr>
<tr>
<td>Falcon</td>
<td>March 10, 2016</td>
</tr>
<tr>
<td>Sparrow 2.2.0</td>
<td>Aug 30, 2019</td>
</tr>
<tr>
<td>Toucan 2.3.0</td>
<td>Jan 27, 2020</td>
</tr>
<tr>
<td>Uguisu 2.4.0</td>
<td>Jun 5, 2020</td>
</tr>
</tbody>
</table>

https://wiki.onosproject.org/display/ONOS/Downloads
ONOS Web GUI
Taiwan on ONOS
Importance of ONOS

• Advantages:
  - Supports a wide variety of SBI protocols versions
  - Active community
  - Aligned with vendors and telcos
  - Good documentation
  - Easy deployment (OSGi)

• Disadvantages:
  - Still in its early phases (some projects are still under development and not fully supported)

ODL vs. ONOS

- Cloud vs. Carrier-grade networks
- Legacy vs. “Pure” SDN
- Private companies vs. Academic
- Both in Linux Foundation!
  - ONOS and ODL focused on different problems.
  - ONOS has focused on service providers’ needs,
    ✓ which landed it a role as a local controller for AT&T.
  - ODL was created to be the Linux of networking: one platform to have a very long life and enable people to build a wide range of solutions to solve a wide range of problems.
    ✓ AT&T is using ODL framework as the basis for its global SDN controller.
Thanks for your attention!