Brigade Dynamic Configuration Update
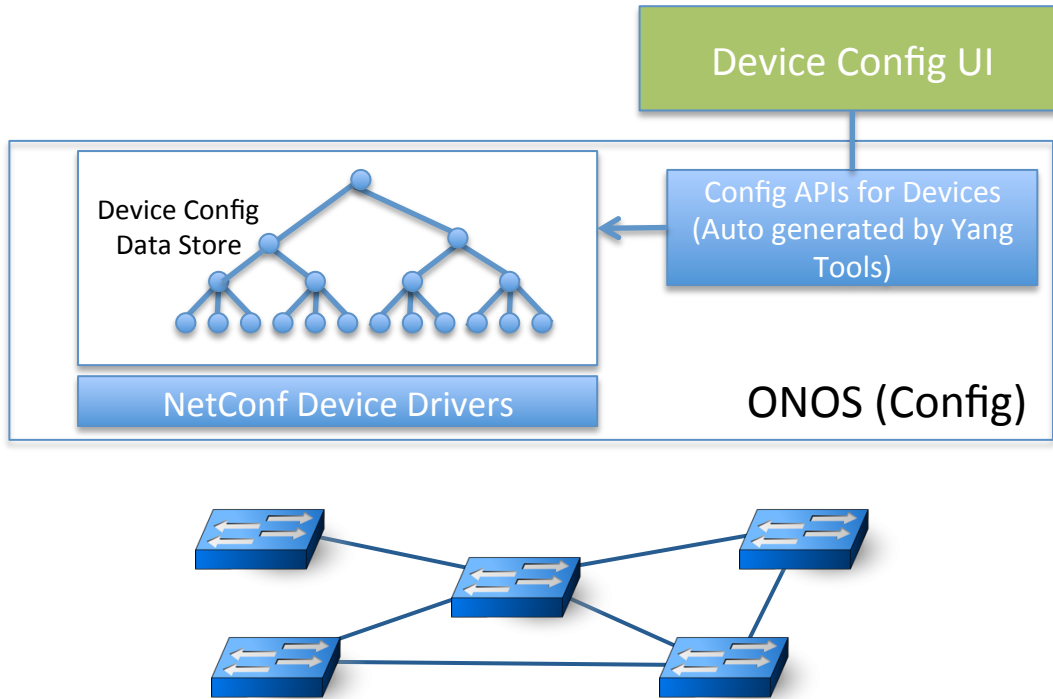
ONOS Build
Paris, Nov 4, 2016

Team Information

- Gigamon : Venkata Narayana Tata;
- Fujitsu: Aarun, Satoshi
- Huawei: Gaurav, Henry, Patrick, Sithara, Vinod
- ON.LAB: Brian, Ray, Thomas
- Verizon: Viswanath KSP, Jay Simha, Venkat Sravan, Vipul Malgotra

- Email: brigade-dynconfig@onosproject.org

Dynamic Configuration of Devices

- Goal: Enable a network operator to seamlessly bring up/down and configure devices from different vendors and to verify the configuration
  - With minimal or no human intervention
- Benefits
  - Network operators: Significant opex savings and vendor independence
  - Vendors: Faster integration of its products into operators' networks and better value prop to its customers (e.g., reduced opex)

Dynamic Configuration of Devices

Device Config UI

Config APIs for Devices (Auto generated by Yang Tools)

Device Config Data Store

NetConf Device Drivers

ONOS (Config)

- ONOS solution based on NetConf and Yang –industry standards
- The Device Config Store contains all device specific config info
  - Schema and data tree
- The Device Config APIs are auto generated by Yang tool chain and device Yang model
- A device config app uses auto generated API to configure a given device
  - The API writes the config parameters into the store
  - The store generates a notification to the driver software to actually program the device via NetConf

4

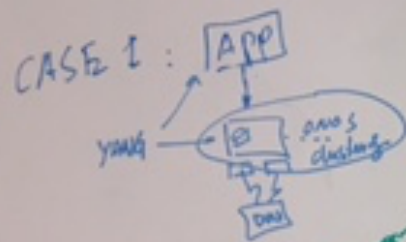# Yang Tool Chain (for ONOS)

Device Yang Model
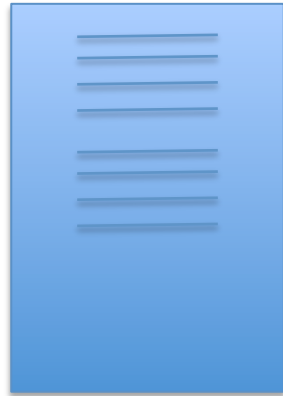
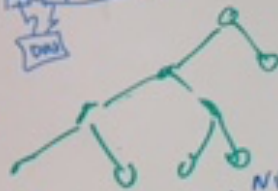Yang Tool Chain

Auto-generated API Java skeleton code

Config Schema

Other Code (e.g. Notification, validation

- Creation of Yang Tool Chain a significant milestone
  - It is difficult to make it completely platform independent
- For every device type, the vendor provides the Yang model which is used to auto create a schema, APIs, skeleton code and validation code: the schema is stored in a repository and the code with the API is added to the code base
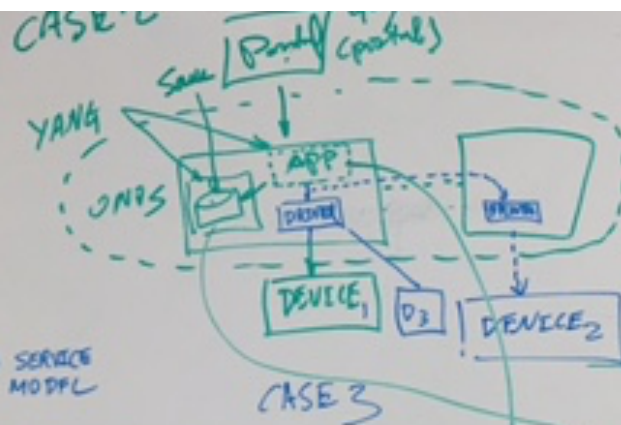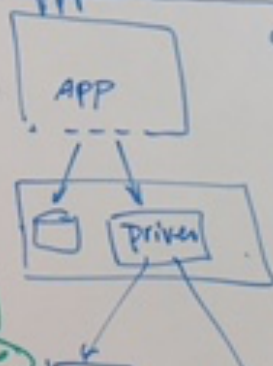- A developer fills in the details in the skeleton code (the device specific logic) to make Java code complete
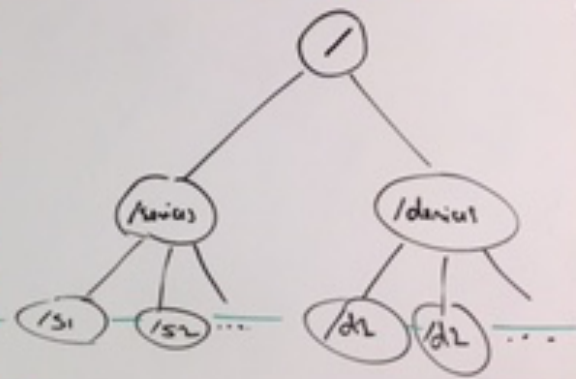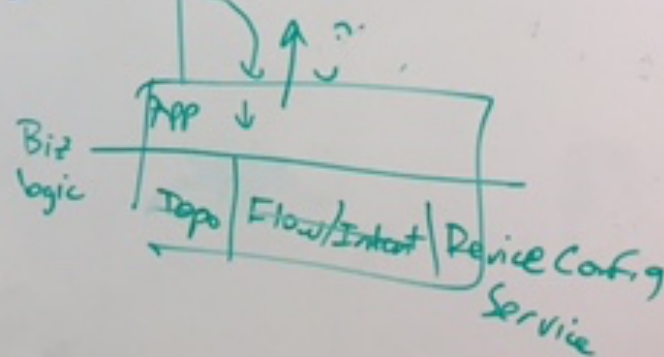
5

CASE 1 : APP

YANG — ONOS clustering
DNS

Agonostic

CASE 2

YANG
Save Purely (partal)
ONOS
APP
DRIVER DRIVER
DEVICE₁ D₂ DEVICE₂

CASE 3

NB SERVICE MODEL

CASE 4
ONOS
APP
Driver

Biz logic
APP
Topo | Flow/Intent | Device Config Service

/
/device
/s1 /s2 ... /d1 /d2 ...

6

CASE 1 : APP

YANG → [☐] ONOS clusters

DMI

AGonostic

CASE 2

GUI/UI (portals)
Portal

YANG →

ONOS
APP
DRIVER   FRUAK

DEVICE₁   D3   DEVICE₂
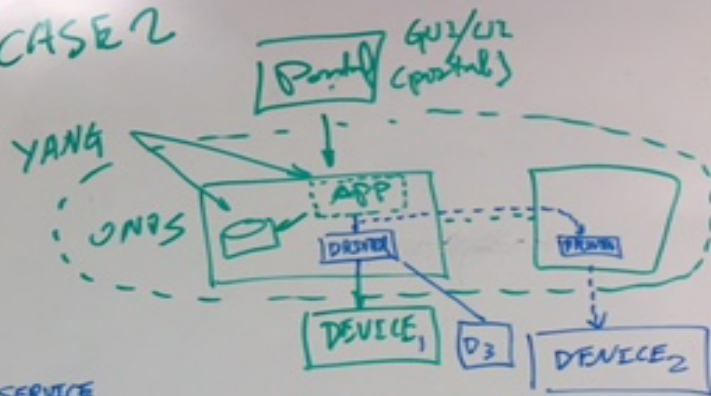
CASE 3

NB SERVICE MODEL

ONOS

CASE 4

APP

Driver

DEVICE₁   DEVICE₂
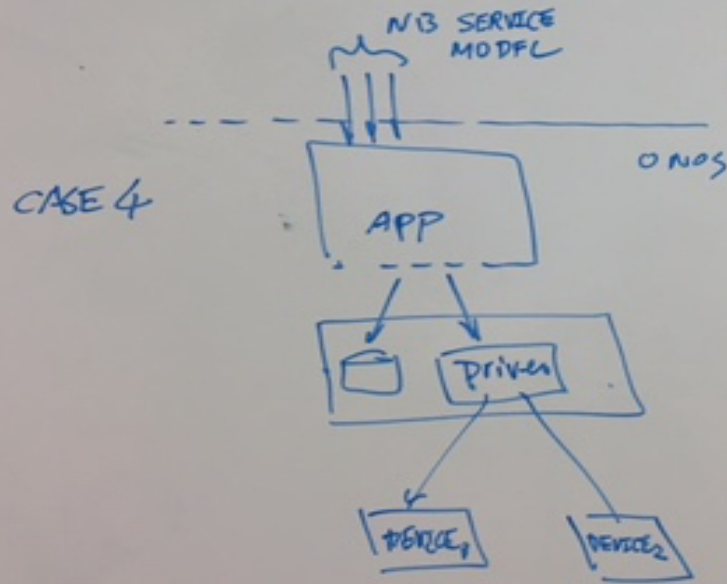
7
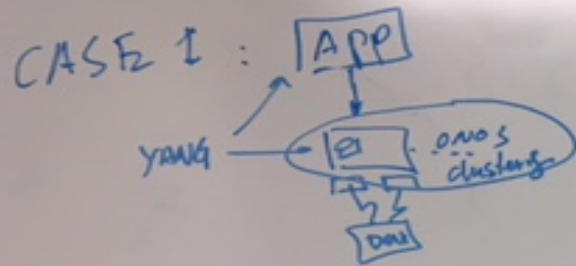
Demo Plan
## (Brigade Meeting In Paris, Nov 2, 2016)

- Demo 1: single vendor's device configuration
  - The configuration is about basic ones for a vendor's router/switch, or other devices.
  - with configuration data persistency (Data Store)
- Demo 2: Single service on Single vendor's device Configuration
  - with configuration data persistency.
  - The initial thought on service is L3VPN.

Demo Plan – cont'
(Brigade Meeting In Paris, Nov 2, 2016)

- Demo 3: Single service (e.g. L3VPN) on multiple vendor devices
  - Similar to Demo 2, but add the multi-vendor devices
- Demo 3+: Common SB Driver mechanism
  - How different Device Model from different vendor is handled without ONOS code change;

Initial Components and Their Relationship Diagram (Draft)
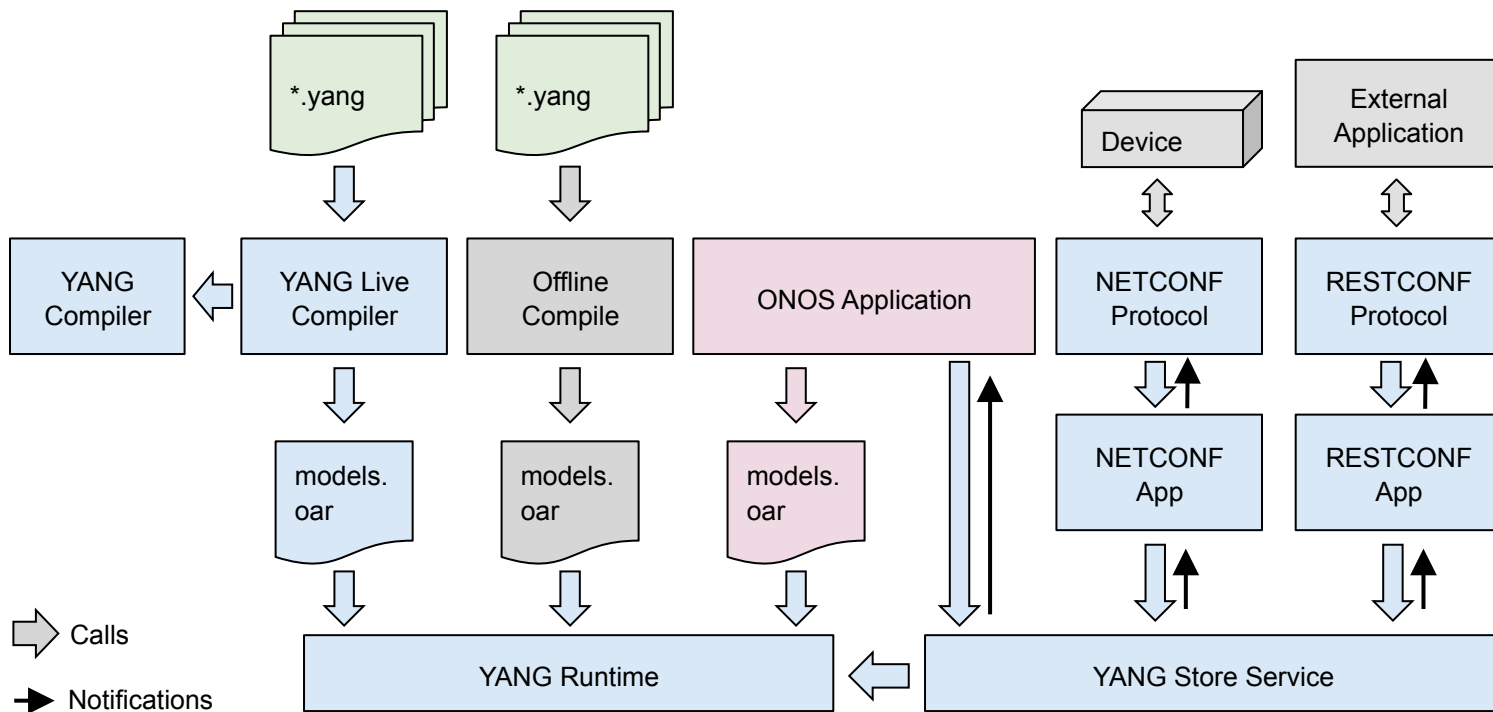
Roadmap

- Phase 1: Target ONS 2017 (April, 2017)

  - Demo 1, Demo 2

  - ONOS Release "I" (now)

  - ONOS Release "J" and "K"

- Phase 2

  – Demo 3 (Multi Vendor Device)

  – ONOS Release "L"

- Future

  – Demo 3+ and More