Fabric & Controllers Track Overview

Co-Chair: Charles Chan (ONF)
Vignesh Ramamurthy (Infosys)
Track Overview (Day 1)

- **4:30-4:40** Track overview and Trellis introduction  
  - Charles Chan (ONF)
- **4:40-4:45** Trellis contributor award  
  - Saurav Das (ONF)
- **4:45-5:30** Integrating Trellis into a carrier-grade NFV platform  
  - Subramanya Datta, Vignesh Ramamurthy (Infosys)
- **5:30-6:00** One size does not fit all – Tungsten Fabric as an enabler of intent-based security in diverse multicloud architectures  
  - Richard Roberts, Ato Sanchez-Monge (Juniper)
- **6:00-6:30** Transforming networks with ONF support  
  - Metin Balci (ULAK)
Track Overview (Day 2)

• 2:00-3:00 **Tuning and Hardening Trellis for Large-scale Deployment**
  - Hariprasad Rajendran (Infosys)

• 3:00-3:30 **VNFs in CNFs Environment**
  - Monika Antoniak, Piotr Skamruk (CodiLime)

• 3:30-4:00 **What Can SDN Do For NFV Cloud Network**
  - Ruixue Wang (China Mobile)
Trellis Introduction

Charles Chan, Ph.D.
Member of Technical Staff, ONF
Trellis
production-ready
multi-purpose leaf-spine fabric
Trellis

SDN-based Bare-metal
Open-source Leaf-Spine Fabric

Trellis Apps
ONOS Cluster
(SDN Controller)

Spine

Leaf (ToR)

Leaf (ToR)

Compute Nodes

Upstream Routers

Access Nodes

Trellis Software Component
Trellis Compliant Bare-metal Hardware
Why Trellis?

• Trellis is designed for service provider edge
  - Traffic types/encapsulations, topologies, ASICs

• SDN simplifies and optimizes existing features
  - Learn more at Trellis booth

• SDN & P4 switches enable new features
  - Learn more at SEBA BNG booth

• Open-source -> ownership & customizability
  - Learn more at Comcast booth
and more...
Use Cases

Enterprise DC Fabric

OpenStack
Kubernetes
VMWare

Trellis Apps
ONOS Cluster (SDN Controller)

Distributed Fabric for Access/Edge Networking

Treillis apps
IPv4/IPv6 unicast/multicast, vlan, MPLS SR, vRouter...

ONOS Cluster

SEBA with Embedded BNG using P4

Chassis Router

Leaf
Leaf
Leaf
Leaf
Leaf
Leaf
Leaf
Leaf
Leaf
Spine
Spine
Spine
Spine

Network Edge Mediator (NEM)

Docker KBs Helm

OLT
ONU

ONU

Stratum

Stratum

P4-Switch

Embedded BNG

P4Runtime, gNMI, gNOI

SDN Controller - ONOS

VOLTHA Apps
Trellis Apps
BNG-control

VOLTHA

Upstream Routers

Servers

Route processor cards

Compute Nodes

Underlay Fabric

Leaf (ToR)
Leaf (ToR)

Overlay Fabric

Backplane

linecards
Looking Forward

- (Ongoing) Scale & Performance improvements
- (Ongoing) Hardening
- (Ongoing) Stratum/P4 integration
- (Ongoing) BNG features (e.g. PPPoE termination, hierarchical QoS)

- Dual homing for Access nodes (like OLTs)
- In Service Software Upgrades (ISSU)
- 5G user plane features
Get Involved

- Website: https://opennetworking.org/trellis
- Mailing list: trellis-dev@opennetworking.org
- Slack: #Trellis on onosproject.slack.com
12 lessons learnt from production deployments
Trellis: Integrating Trellis into a Carrier grade NFV Platform

Vignesh Ramamurthy
Subramanya Datta G
Balaji Thangavelu
INFOSYS
What is carrier grade in the new paradigm of SDN/NFV?

Trellis: Carrier grade NFV platform

1. High Performance
   - Predictable, High performance non-blocking Switching infrastructure

2. Dis-aggregated & cloud-native
   - Fully dis-aggregated, virtualized and de-coupled control Plane

3. Scalable
   - Cloud-style scaling of Control Plane functions

4. HA, Reliable & fault-tolerant
   - Clustered, highly redundant and fault-tolerant virtual Infrastructure

5. Real-time visibility
   - Real-time streaming and Logging pipeline giving deep visibility

6. Agile and Dev-Ops
   - High velocity software delivery and upgrades
Trellis as NFV Fabric for Virtualized Cable Head-end

Real-time streaming Telemetry and Logs

Dis-aggregated, virtualized and containerized CCAP core and SDN Controllers

Trellis controlled SDN fabric provides the switching backplane for the head-end / hub traffic
Trellis as NFV Fabric for Virtualized Cable Head-end: Networking characteristics

**Fabric Underlay Networking:**
- MPLS Segment Routing based forwarding across the fabric
- IPv6 and IPv4 unicast, L2 bridging and Multi-casting
- MAC+VLAN based host recognition and management
- VLAN cross-connect feature

**Container and Overlay Networking:**
- Multiple networks – L2 VLAN based bridging within a leaf switch pair
- Kubernetes Container networking – VXLAN networking via Host CNI
- SR-IOV based VFs managed as a separate L2 VLAN networks
- SR-IOV DPDK used for fast data-path networking for DOCSIS user-plane
- Container end-points managed in the underlay

**PTP 1588v2 Timer:**
- PTP 1588 based Timing / Phase synchronization
- Prioritized Forwarding flows for 1588 packets – Unicast UDP packets
- Peer to Peer Transparent clocks configured on the Ethernet PHY
Trellis as NFV Fabric for Virtualized Cable Head-end: Networking characteristics

- Virtual CCAP container dynamically binds to the R-PHY
- IPv6 Underlay DEPI / UEPI Tunnels between R-PHY to virtual CCAP
- Control Plane traffic (ARP, ND, RIP, DHCPv4, DHCPv6) punted to Controller and handled in ONOS Apps
- Successful DHCP transactions establish CM/CPE host routes in the fabric
- Dynamic fail-over virtual CCAP to a different container on a different host
1. High Performance and Predictability

- 15k modems @ Start-up
- 120k Routes and 250k flows
- ~1000 ARP / ND per second
- 650 DHCP Transactions /sec
- 500 flows per second
- Controller synch - 120 k Routes < 2 min

- Topology specific Route optimizations
- Timers and Flows / Stats Polling Tuning
- Symmetric Probing for dual-homed hosts
- Flow-store back-up synchronization
2. Dis-aggregated and cloud-native

K8S ONOS POD Architecture

- Kubernetes cluster design – 5 or 3 member cluster
- Server Redundancy and POD redundancy
- Servers dual-connected via Multi-chassis LAG to two switches
- Power it with Ansible automation for all POD deployments and installation
2. Dis-aggregated and cloud-native

K8S POD Design Considerations:

Container Networking:
- Multiple Network Interfaces – Flannel and SR-IOV
- Control Plane Interface – SR-IOV

Efficient POD spec:
- Statefulset POD
- Distributed Config store – GlusterFS
- Atomix File store – hostPath mode

Config automation:
- Maximizing Config automation through ConfigMap
- Deployment automation with Ansible plays
- POD liveliness check
- Pod rejoining Scenarios and Orchestration
3. Scalability – Control Plane and data plane scalability

Control Plane Scaling

- Horizontal scaling of control for increased Loads
- Global Network view - fully replicated across the cluster instances
- Master/Backup arrangement for replication of flow table entries with Partitions – advantageous for horizontal scaling
- Seamless Path computation on Data path scaling
- Fine-grain OF Packet-in Filters to handle control plane storms

Data Plane Scaling

- Unhindered addition of RPHY Nodes and Server on the GO
- Introduction of Aggregator Switch (2-level Leafs) for scalability of RPHY devices
- Scaling CMs behind the RPDs on the fly
- Bringing up more Cable Control components on the go
4. High Availability, Reliability and Fault-Tolerance – Data path

**HA in the data-path**

- Servers all dual-homed
- Multi ECMP Links – OF-Any groups
- Linux bonding in active-active mode
- Linux bonding extended to K8S pods with VF Bonding
- Route reprogramming on POD HA
- Pair link between 2 Leaf pairs for switch local failovers
- Redundancy at each link level with dual links between Leaf and Spine

- Switches simultaneously connects to multiple controller instances
- ONOS instances running as stateful set PODs with POD ordinality fixing the identity
- Losing ONOS instance redistributes switch mastership
- ONOS config state maintained in glusterfs and helps reload config on restart
- Redundancy of Server Hardware, Network connectivity and storage
5. Real-time visibility – Telemetry based

- **ONOS Exporter**: A side car with ONOS
  - Collects statics using REST API
  - Polls switches to collect Redfish data

- **Platform Mgmt App**:
  - Detects ONOS POD creation
  - Sends details to Monitoring Probe

- **OOM** collects optical data and sends it out

- **Monitoring Probe** Sends Metrics to Kafka
  - Consumers on cloud propagates data to Prometheus

- **Grafana** using the data on Prometheus plots various dashboards for OPs
5. Real-time visibility – Dashboards and Logs
6. Agile Dev-Ops – CI/CD Pipeline

- Agile project and requirement management
- Continuous integration
  - Code
  - Code quality review
  - Unit test
  - Build & Package
- Release management
- Test and defect management
  - Functional test
  - Load & Performance
  - Longevity & Resiliency
  - Beta Test/Production Test
- Continuous test
  - Functional test
  - Load & Performance
  - Longevity & Resiliency
  - Beta Test/Production Test
- App Configuration and Deployment
- Environment Management
  - Ansible
  - Kubernetes
  - Nexus
  - Robot/Custom Test automation framework
  - GCOV
  - GoogleTest
  - Manual
  - JIRA
  - Nexus

- Integrated development environment
  - C/C++
  - GitLab
  - sonarqube
  - Python
  - docker

- Environment Management
  - Dev
  - Functional Test
  - System Test
  - Production/Beta

- Test and defect management
6. Agile Dev-Ops – CI/CD Pipeline

Continuous Integration

1. High Performance
2. Disaggregated & cloud-native
3. Scalable
4. HA, Reliable & fault-tolerant
5. Real-time visibility
6. Agile and Dev-Ops

Product owner
Create user story/feature task

Developer
Assign the user story/feature task

Developer
Develop, unit test and resolve code quality issues

Developer
Push changes & create pull request

Developer
Create feature branch linked to Agile task when developer assigns

Code reviewer
Review submitted code changes

GitHub
Develop branch
Auto Merge code to develop on pull request approval

GitHub
Feature branch
CI server catches pull request trigger

GitHub
CI server catches merge/push trigger or, a scheduled event

Jenkins
CI Build success

Jenkin
Push built artifact for continuous delivery

Nexus
Artifact/docker repo

Pre-merge
Build, unit test, code quality scan, smoke test – apply objective merge/commit criteria (if fails then developer is notified)

Post merge build

Manual
Automated

3rd Party software

Note: these code quality analysis tools scan automatically as developer write code and report them visually
6. Agile Dev-Ops – CI/CD Pipeline

- Continuous Deployment

- Pulls setup config from SCM

- Jenkins applies the setup config and Triggers ACN

- ACM configures the PODS

- ACM

- Nexus
  - Docker repo

- Monitoring application and alerting

- Health check and proceed with Test execution on success. Roll-back on failure

- Jenkins
  - TestOn/ Automated Test Suite
  - ServerSpec + health check scripts
  - Quali
6. Agile Dev-Ops – CI/CD Pipeline (Image build process)

Components View

- ONOS New Release source Code
- Custom Docker file, Settings file etc.,
- Custom Application code

- Custom ONOS Image

ONOS New Release source Code

Create YY-onos-<rel>-<rcx> branch

Source: ONOS GIT

- Setup Release Jenkins tag to build

Process View

CICD Team

- Source: ONOS GIT

ONOS New Release source Code

- Use base release code from <git>

Design Team

- Source: Custom GIT

Create <rel>-<rcx>-apps-merge branch

- Merge Custom App Code, i.e. Specific changes

- Merge apps_merge branch to main branch

Custom ONOS Image

- Custom ONOS Image YY-onos-<rel>-<rcx>:v.<rel>.0
- Custom ONOS Image YY-onos-<rel>-<rcx>:v.<rel>.1
- Custom ONOS Image YY-onos-<rel>-<rcx>:v.<rel>.2

- Jenkins automatically triggers build

- Jenkins automatically triggers build

- Jenkins automatically triggers build

FAILS

Make submissions to apps_merge branch

- Merge apps_merge branch to main branch

Jenkins automatically triggers build

Custom ONOS Image YY-onos-<rel>-<rcx>:v.<rel>.x
Thank You