ONOS Virtualization Brigade

Yoonseon Han

DMC R&D Center, Samsung Electronics, South Korea
ONOS Virtual Network Brigade, ON.Lab, US
Objectives

Goal
Enable the creation of SDN capable virtual networks

Short-term focus
Define Virtualization API
Design a virtualization subsystem for ONOS
Port-based virtualization features
  Address virtualization, Topology Virtualization, Control function virtualization
Multi southbound protocol support

Long-term focus
External Connectivity (e.g. Internet) for virtual networks
Virtual network embedding
Virtual Network pausing and snapshotting
Openstack integration
Members

Mentor
Thomas Vachuska (ON.Lab)

Brigade Core Members
Claudine Chiu (Lead)
Yoonseon Han
Nithyanantham
Sangyun Han
Harold Huang
Jovana Vuleta
Eric Tang
Doyoung Lee

Brigade Test and QA Members
Matteo Gerola
Mariano Osabe
Jovana Vuleta
Adam Pavlidis
Jordi Ortiz
Frank Wang
Harold Huang
Background

- **Software Defined Networking (SDN)**
  - Separates the control plane from the data plane
  - Centralized management
  - Rich programmability

- **OpenFlow (OF)**
  - The de facto standard protocol to communicate between control and data planes
  - Provides remote administration of forwarding tables of switches
  - Current SDN technologies are heavily dependent on OF

- **Benefits of SDN**
  - Programmability
  - Agility
  - Flexibility
  - CAPEX/OPEX saving
  - Vendor neutrality

[Flow Table Diagram]

Source: SDN Market Size Report, [SDxCentral 2015]
Development Objectives

Develop a Network Virtualization Framework

- Multi-tenancy
- Isolated data and control plane per VN
- Distributed architecture
- Heterogeneous network support
- On-platform apps
- Service APIs for VNs
SDN NV Approaches

• Overlay Network-based Approaches
  – Create and manage VNs using Virtual switches and tunnels
  – Most of commercial data center solutions (VMWare NSX, Microsoft Hyper-V, Cisco ACI)
  – Hypervisor-based solution, easy to deploy

• Direct Programming Approaches
  – Directly program physical switches (network slices) to create and manage VNs
  – OpenVirtex, FlowVisor, FlowN, VerTIGO
  – Network-based solution, powerful QoS management, transparency
Target use-cases

NV Use-cases

• Creating virtual SDN networks for tenants
• Slicing regions of networks for use by different tenants (M-CORD)
• Federation - exposing abstracted view to peer/parent controllers (E-CORD)
Overall Architecture

- **Off-platform Apps**
  - External SDN controller
  - Off-Platform apps

- **On-platform Apps for VNs**
  - OFAgent
  - OpenStack Integration

- **Translates virtual things into physical things**
  - Network Addressing
  - VLANs

- **Abstracts interfaces to control the physical network**
  - LISP
  - OpenFlow

- **Encode/decode protocol messages**
  - OVSDB
  - Yang/Netconf
Virtual Network Subsystem

• Extended subsystem structure

• Implemented as an ONOS subsystem

• Virtual Managers

• Virtual providers
VN Model Implementation

VN Abstraction Model

ONOS provides a rich set of network abstractions

But, designed for a single physical network

Extending existing abstraction model

Allow the instantiation of multiple services

Introduce new attributes for VNs
Virtual Network Administration Service

Virtual Network Service
Query service to obtain virtual objects

Virtual Network Admin Service
Create/remove/modify virtual objects

Diagram showing the components and interactions involving:
- VirtualNetwork
- VirtualDevice
- VirtualLink
- VirtualHost
- TenantId
- VirtualPort
- VirtualNetworkManager Component
- VirtualNetworkProviderService
- VirtualNetworkProvider Component
- VirtualNetworkService
- VirtualNetworkListener
- DistributedVirtualNetworkStore Component
- VirtualNetworkStoreDelegate
- VirtualNetworkProviderRegistry

Code snippet showing:
- VirtualNetworkService
  - getVirtualNetworks(TenantId)
  - getVirtualDevices(NetworkId)
  - getVirtualHosts(NetworkId)
  - getVirtualLinks(NetworkId)
  - getVirtualPorts(NetworkId, DeviceId)
  - getPhysicalDevices(NetworkId, VirtualDevice)
  - get(NetworkId, Class<T>)
  - getDirectory()
  - getVirtualNetworkApplicationId(NetworkId)
- VirtualNetworkAdminService
  - registerTenantId(TenantId)
  - unregisterTenantId(TenantId)
  - getTenantIds()
  - createVirtualNetwork(TenantId)
  - removeVirtualNetwork(NetworkId)
  - createVirtualDevice(NetworkId, DeviceId)
  - removeVirtualDevice(NetworkId, DeviceId)
  - createVirtualHost(NetworkId, HostId, MacAddress, VlanId, HostLocation, Set<TenantId>)
  - removeVirtualHost(NetworkId, HostId)
  - createVirtualLink(NetworkId, ConnectPoint, ConnectPoint)
  - removeVirtualLink(NetworkId, ConnectPoint, ConnectPoint)
  - createVirtualPort(NetworkId, DeviceId, PortNumber, ConnectPoint) virtualPort
  - bindVirtualPort(NetworkId, DeviceId, PortNumber, ConnectPoint) virtualPort
  - removeVirtualPort(NetworkId, DeviceId, PortNumber) void
Virtual Network Core Services

VN Core services
dedicated and independent service processing per VN
managed by *Virtual Network Service*
Translators

Pluggable and Extensible Translators
- Convert virtual object into physical object
- Provides topology, address, and control functions virtualization
- Various algorithms can be applied

Topology Virtualization
- Allows decoupling of VN topology from physical networks
  - End-to-end path only
  - Substrate networks
  - Identical (clone) networks
Scheme-based Flexible Virtual Providers

Virtual Providers (Translators)
   Implement logics to translate virtual objects

Virtual Provider Services
   A callback service to get notified requests/changes

Virtual Provider Registry Service
   A match making service to connect Virtual Providers and Provider Services
Distributed Virtualization Operations

Distributed operations and stores

Synchronize VN information among the multiple instances

High availability
Load balancing

Apply sync. mechanism according to the data type

![Diagram of distributed virtualization operations](image-url)
Virtual Network Event Delivery

Event delivery service
Used to notify changes of VN objects
From a Store to subscribing Event Listeners
Shared event delivery mechanism using event encapsulation

Application #1

Service Interface

Store Delegate

Generate events

Notify events

Event Delivery Service

Encapsulate and decapsulate events for virtual network service

Service #2

Event Listener Interface

post events

post events

Application #1

Service Interface

Store

Generate events

Notify events

Event Listener Interface

post events

post events

Service #2

Event Listener Interface

post events

post events

Application #1

Service Interface

Store Delegate

Generate events

Notify events

Event Delivery Service

Encapsulate and decapsulate events for virtual network service

Service #2

Event Listener Interface

post events

post events
Distributed Stores

Mastership service
- Responsible for determining mastership of an element

Distributed Stores for VNs
- Structural Information (-> Consistent Map)
  - Topology, Address mapping
- Operational Information (-> Cluster Communication)
  - Packet, Flow rule, Flow objectives, Meter, Group, …
- Kyro serializer, CopyCat (RAFT consensus), Netty

```java
networkIdVirtualNetworkConsistentMap = storageService.<->consistentMapBuilder()
  .withSerializer(SERALIZER)
  .withName("onos-networkId-virtualnetwork")
  .withRelaxedReadConsistency()
  .build();
networkIdVirtualNetworkConsistentMap.addListener(virtualNetworkMapListener);

clusterCommunicator.unicast(new VirtualFlowRuleBatchOperation(networkId, operation),
                          APPLY_BATCH_FLOWS,
                          serializer::encode,
                          master)
```

A part of Virtual Network Store (ConsistentMap)
A part of Virtual Flow Rule Store (Cluster Communication)
Application Model

On-Platform Apps

- Consume VN core services
- Almost similar to developing apps for SDN controllers
- Benefits:
  - Low latency
  - A rich set of services
  - Tenant-awareness

Off-platform Apps

- Using northbound interfaces
- CLI
- REST
- Protocols (e.g. Openflow, NetConf)
Applications

Application Development Environment
Can be implemented using JAVA
Consume ONVisor Core Services by referencing
Running as a OSGi bundle
  Support dynamic activation/deactivation

@Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
protected DeviceService deviceService;

Referencing ONOS core service

@Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
protected VirtualNetworkService vnetService;

... DeviceService deviceService = vnetService.get(networkId, DeviceService.class);

Referencing VN core service
Summary and Contributions

Summary
A network virtualization framework is presented
Provide SDN based programmable VNs
The evaluation results showed a small performance degradation
Available on https://github.com/opennetworkinglab/onos

Future work
Automated and Optimal Virtual Network Embedding
Virtual Network Snapshotting and Migration
Virtual Network Resilience