Validation as a Service (VaaS)

Wen-Pai Lu, Ph.D.
IEEE & EIT Digital:
Federated Test Beds Workshop
May 3rd, 2016
Let’s Look at the ETSI NFV Framework
What Have Been Focus?

- **NFVI**
  - OpenStack
  - OPNFV
  - Virtualization
  - Data Plane
  - vNIC
  - Chip architecture
- **VNF**
  - FW, LB, vRtr, etc. from various vendors
  - VNFM
- **MANO**
  - OpenSource
  - NFVO
  - VNFM
  - VIM
  - Service Chaining
  - Service Orchestration
  - Resource Orchestration
- **POC, Lab Trials, etc.**
NFVI under Test

Legend
- Function Under Test
- System Under Test
- Test Device
- Test environment
VNF under Test
Management and Orchestration under Test

NFV Infrastructure

Legend

Function Under Test
System Under Test
Test Device
Test environment
Issues and Challenges

• Variations in VNF Functions
  – Vendor specific
  – Associated with their VNFM
  – Performance

• Network Service (NS) behaviour when deployed in the network
  – VNFs from multiple vendors
  – Overall NS system functions
  – Dependency on the NFV infrastructure

• Commodity hardware ≠ ASIC appliance

• Modularization of the NFV components

• Various Test Types
  – Functional tests
  – Performance tests
  – Regression tests
  – Conformance tests

• Manual process in test environment
Solutions Implementation Details

APPROACHES, ISSUES AND LESSONS LEARNED
What Do We Try to Accomplish?

• Automate the Test Environment
  – Template Driven
  – Dynamic
  – Flexible
  – Easy to use and setup

• Tests and Validation Processes

• Integrating different NFV components
  – NFVO
  – VNFM
  – Open-Source
Mapping to NFV Framework (per TST001)

Integrating with OpenVIM via OpenAPI to leverage the high performance data plane.

SUT can be a VNF or the NS.

Common data model to onboard various VNFs.

Every components including Test Devices are VNFs.

Legend:
- Function Under Test
- System Under Test
- Test Device
- Test environment

5/3/16
Solutions Component Overview

VNF and Network Service Validation

• Orchestration & Management – Luxoft SuperCloud
  – Tenant Management
  – Abstraction of network complexity
  – Managing and orchestrating NFV resources
  – VNF Catalog management
  – VNF/NS onboarding and deployment
  – VNF Manager

• Cobham Test Framework
  – TeraVM – Network function tester
  – TeraVM Controller – Network function test control
  – TeraVM Executive – Management role for TVMC and TVM (plus other functions)

• OpenVIM
  – Infrastructure Manager
  – Resource orchestration
Execution Process

- Define test topology as a template
- Using the data model to describe the VNF – build specific VNF and NS descriptor
- Build test topology template using WebUI via GUI
- Onboard and instantiation via service orchestration and resource orchestration
- Instantiate test services – select test cases, test topology and activate the test
- Test results display on dashboard and deliver in different formats
Building Test Topology Template
Solutions Platform Architecture

- SUT (VNF)
- TeraVM (VNF)
- TeraVM-Exec (VNF)
- TeraVM-Cntlr (VNF)
- SuperCloud (NFVO & VNFM)
- REST API
- OpenVIM (VIM)
- SDN-C & vSwitch
- Libvirt Agent
- Hypervisor (KVM-QEMU)
  Ubuntu 14.04
- Hardware/Emulation

NS

NFVI

MANO
Steps and Actions for VNF Validation

• Preliminary Steps by Administrator
  – Create Tester VNF (VNF Descriptor)
  – Create Test Bed Topology Templates
  – Import Test Topology
  – On-boarding Tester VNF (TeraVM)

• Steps Taken by (Test) User
  – Create SUT Template
  – Attach SUT Template to Test Topology
  – Attach Tests to SUT’s Test Topology Interfaces
  – Manage the Test(s)
  – Execute the Test(s)
  – Obtain and view the test results
Lessons Learned

• Template-driven model works
  – An optimal Test Topology (NS) config is saved as a template
  – NS instantiated on-demand
  – All NS configs driven via REST API’s

• New VNF creation was all done manually for the demo
  – It needs to be automated (data model and descriptor) => less time

• Service Orchestrator performed by SuperCloud
• Resource Orchestration can be performed by OpenVIM
• Tests are managed via Test Framework
• NFV components can be modularized
• Adapt to Open Source