LightReading
www.lightleading.com

REPORT
OCTOBER 2015

TESTING PARTNER: EANTC

PARTICIPATING COMPANY: CISCO

Validating Cisco's NFV Infrastructure

Learn what EANTC and Light Reading discovered while evaluating Cisco's NFVi capabilities

Contents

Introduction

Cisco NFVi introduction and test overview

The virtual switch performance test introduction

Test goals

Test methodology and setup

VM-to-VM performance

Full virtual path performance

Virtual Switch FIB Scalability

Evaluating the Cisco IOS XRv 9000

Conclusion of vSwitch Test Findings

Cisco and EANTC evolved vSwitch test methodology

Carrier-grade high availability and reliability of Cisco's NFVi

Automated and validated OpenStack installation

Centralized logging and Runtime Network Regression Testing

High availability

Putting VNFs to the Test

Cisco's Virtual Packet Gateway VPC-DI

Single Pane of Glass management

Introduction

As part of the New IP revolution, NFV (network functions virtualization) is going to change the world of communications networking and services. It's just a matter of when, not if.

Part of the journey towards the "cloudification" of wide area communications networks involves the development of coordinated hardware and software systems that will enable the creation, delivery, management and tear-down of virtual network functions (VNFs). Collectively, those systems are referred to as the NFV infrastructure (NFVi).

A reliable NFVi is critical to the introduction of VNFs into any communications service provider (CSP) environment. So, as you'd expect, there's no shortage of technology companies that want their NFVi to be at the heart of CSP strategies, to be the foundation upon which new, revenue-generating applications can be reliably launched and provisioned.

As a result, it's incredibly important that network operators, as they start to introduce commercial services using New IP technologies, can turn to third-party organizations to find out if an NFVi can deliver what they need.

Together, Light Reading, as an independent and trusted media organization at the heart of the global communications technology community, and its respected test lab partner EANTC are in a prime position to help network operators with their New IP strategies.

Earlier this year, Light Reading asked the EANTC team to visit the San Jose, Calif. labs of Cisco Systems to conduct a series of validation and verification exercises on a number of Cisco cloud, software-defined networking (SDN) and virtualization platforms.

To follow up that successful project, Light Reading asked the EANTC team to return to San Jose in late September to evaluate Cisco's NFVi. This report is the story of how EANTC planned the new project and the outcome and findings
of its evaluation.

The report provides: an overview of the aims of the evaluation; a look at Cisco's NFVi; an in-depth, multi-page performance evaluation of Cisco's virtual switch technology; carrier-grade high availability and reliability; the integration, features and performance of key applications -- Virtualized Video Processing (V2P), including Cloud DVR, and virtual EPC (evolved packet core); and an evaluation of Cisco's "single pane of glass" management capabilities with regards to its NFVi.

— The Light Reading team and Carsten Rossenhövel, managing director, and Balamuhunthan Balarajah, test engineer, European Advanced Networking Test Center AG (EANTC) (http://www.eantc.de/), an independent test lab in Berlin. EANTC offers vendor-neutral network test facilities for manufacturers, service providers, and enterprises.

Cisco NFVi introduction and test overview

At the beginning on 2015, Light Reading commissioned EANTC to evaluate Cisco's NFV architecture by conducting a series of independent tests. The series began in March, when EANTC validated the functionality of major cornerstones of Cisco's NFV architecture:

- Evolved Programmable Network (EPN): Routing and switching components such as ASR 9000 and CRS-4/S, plus UCS data center platform
- Evolved Services Platform (ESP): WAN Automation Engine and Network Services Orchestrator (NSO, enabled by Tail-f)
- Virtualized Network Functions (VNFs): Mobility IQ and WebEx Squared

To find out what was tested, and the results, see Validating Cisco's Service Provider Virtualization & Cloud Portfolio.

Now we have taken the next step: Evaluate the performance, manageability and high availability of Cisco's NFV infrastructure (NFVi) solution — the data center infrastructure for any virtualized services.

Enterprises have used cloud solutions based on virtualization techniques for a number of years. Those solutions are based on enterprise performance and manageability criteria: Enterprises typically require lots of compute power and storage at a large scale, using many servers, grouped in different pools for single functions. Outages, at least during maintenance windows, are typically acceptable and internal customers do not get strict service levels (SLAs). Service provider virtualization requirements are quite different, which is why we focused our evaluation of Cisco's solution on the following criteria:

1. Common infrastructure supporting any type of virtualized services for mobile, business and residential customers.
2. Enabling service level agreements (SLAs) for virtualized network functions by providing reliable and scalable base network services (specifically the virtual switch main interconnection component).
3. High availability support — including both hardware and software — up to service provider requirements.
4. All-in-one platform and services orchestration and management, enabling a network operations center (NOC) to view and control all aspects of the virtualized solution and network from a single console. The management platform should support multivendor NFVi installations such as OpenStack and VMware.

These four criteria are fundamental to any virtualized network functions — at least if a service provider aims to deploy a multi-service virtualized platform. Many of the vendor test reports published so far violate at least one of these criteria: In almost all cases, performance is achieved by bypassing the virtual switch, for example.

EANTC believes it is time to move beyond this stage and look at how vendor solutions provide a 360-degree support of highly available, scalable and manageable NFV deployments.

In addition to these steps, we also investigated some of Cisco's application use cases, namely:

- Virtualized Video Processing (V2P), including Cloud DVR.
- Virtualized Quantum Packet Core (VQPC), Cisco's virtualized version of the 3G/LTE mobile packet core solution.

Cisco NFVi overview

The NFV Industry Specification Group (ISG) hosted by ETSI (European Telecommunication Standards Institute) has defined an "NFV Infrastructure" (NFVi) block in its reference model. The NFVi includes the hardware resources (in this instance, Cisco UCS servers) and virtualization platform (in this evaluation, OpenStack Juno release, including KVM and the virtual switch).

We evaluated both an Open vSwitch (OVS) implementation using Intel's Data Plane Development Kit (DPDK) and a proprietary Cisco vSwitch implementation.

From a functional perspective, the NFVi provides the technology platform with a common execution environment for NFV use cases, such as business, video, residential, and mobility services. NFVi provides the infrastructure that supports one or more of applications simultaneously and is dynamically reconfigurable between these use cases through the installation of different VNFs.

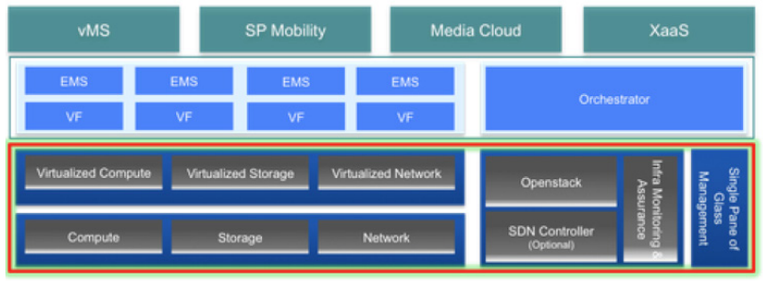

Figure 1: Cisco's NFV infrastructure (NFVi)

The NFVi is typically bundled with an infrastructure manager, or VIM. In Cisco's NFVi, the VIM functions are provided by Openstack, complemented by proprietary Cisco software for management and orchestration (for details, see Part II of the evaluation report).

The orchestration platform and SDN Controller functions — Cisco WAN Automation Engine and Cisco Network Services Orchestrator — were evaluated by EANTC in the previous evaluation, so we did not focus them this time.

The Virtual switch performance test introduction

For the successful deployment of virtualized network functions, the most important aspect is network connectivity — to other VNFs, the rest of the data center, services and customers. Often, VNFs are primarily network data filters or relays: A virtual router, a virtual broadband network gateway, a virtual firewall or even a virtual mobile packet gateway — they are all big packet movers with some add-on functionality.

This is a major difference compared with enterprise cloud data centers. For enterprises, the most important aspect is often the compute and storage function of a virtual machine (VM). Salesforce, Oracle databases, Citrix remote access and others — they are all big data or compute workhorses that require and produce only a relatively small amount of network data.

What is a virtual switch?

The virtual switch (vSwitch) is the broker of all network communications within a server and to the outside world. It connects physical Ethernet ports to virtual services, multiplexes services onto VLANs, and connects virtual services to other virtual services internally (informally called "service chaining" — the official ETSI names are more complex and confusing, so we'll address them later).

By nature of the standard x86 environment, the vSwitch is a software component. In its basic variant, the vSwitch moves packets by copying them around, from Ethernet network interface card (NIC) to kernel memory, from kernel memory to user memory and vice versa. In its most basic, vanilla version, this is an awfully slow process: That an industry which has created ultra-low latency hardware-based data center switches with terabits-per-second of throughput needs to go back to software switching is depressing.

Initially, vendors found a very simplistic solution to the problem to get large-scale throughput — circumvent the vSwitch, allowing the VNF to access the hardware directly. This feature is called "passthrough." It is a great marketing invention, but it does not have a place in an NFV world, as it violates a number of layer abstraction models and, specifically, does not allow service chaining.

vSwitch solution evaluation

Recently, there has been a lot of focus on improving vSwitch performance and validating it. Intel has published the public domain, open source Data Plane Development Kit (DPDK), which provides fast packet processing algorithms. And the Open vSwitch (OvS) project has developed an open source reference implementation for a virtual switch. This implementation supports DPDK as well, improving the performance greatly — it is called OVS-DPDK.

We tested OVS-DPDK performance on Cisco UCS hardware (for the details, see the test configuration tables on the next page), comparing two Intel CPU generations (Haswell and Sandy Bridge). In addition, we compared OVS performance with Cisco's own virtual switching implementation, called Vector Packet Processing (VPP).

How does innovation work (and pay back the innovators' efforts) in an open source world? At the forefront of development, there are often commercial implementations that are later (and/or with limited functionality) released into the public domain. This is what Cisco has done by developing a proprietary, virtualized forwarder — VPP. This technology, as Cisco explained, is included in the VMS (Virtual Managed Services) product and in other Cisco products such as the XR9000v virtual router, covered in this test as well. VPP uses proprietary algorithms to further improve packet forwarding for service provider-specific requirements. VPP runs as a Linux user-space process in host, as well as in a guest VM. When running in host mode, it supports drivers to access NICs over PCI; when running in guest mode, it accesses the network interface cards (NICs) over PCI-passthrough. VPP integrates DPDK poll-mode device driver.

VPP and OVS are not exactly comparable regarding feature sets: OVS is much more of a complete, standalone vSwitch implementation than VPP, which is more of an advanced technology building block. The Cisco team explained that VPP is used in Cisco products where its high performance features are required.

Test goals

We measured the maximum packet forwarding performance of the Open vSwitch and VPP implementations on UCS hardware (both for the Haswell and Sandy Bridge architectures) for the following reasons:

- **Provide per virtual component and system-side performance benchmarking:** While system users do not experience individual component performance, but that of the system as a whole, optimizing the system requires an understanding of the performance of its components. We attempt to provide that understanding.
- **Provide comparison of bare metal and virtualized systems:** A common and important question is, "How much faster or slower will a virtualized system be? What performance tax will I pay for the flexibility of virtualization or what greater speed will I benefit from?" We provide some baseline bare-metal testing to enable this comparison.
- **Establish NFV benchmark best practices that can be used as a reference:** This can be achieved through testing and cooperation with the NFV community, striving to narrow down the number of parameters for further testing. In other words, once we know an option is correct, we do not need to test the incorrect alternatives. And when testing a new combination of elements, we start with a baseline from previous testing. The aim is to automate as much of the testing as possible, for correctness, repeatability and testing efficiency.
- **Develop a reproducible vSwitch testing process:** The NFV community, network designers and engineers, Cisco and the entire industry are all dependent on accurate information on how new NFV-based systems will perform as they scale. This information is currently not available — we thought it was important to contribute to this much needed knowledge base.

Test configuration

Cisco provided a couple of UCS C240 M4SX and UCS C240 M3S blade servers for the testing processes. On the physical hardware Cisco installed Ubuntu 14.04.3 LTS (VMM — virtual machine manager) with KVM for the virtual machine environment, DPDK 2.0 and OpenVswitch 2.4.0.Here are the full details of the hardware and software components:

Table 1: The Evaluation Hardware

Hardware	UCS C240 M4SX
Chipset	Intel® C610 series chipset
Processor	Intel® Xeon® Processor E5-2698 v3 (code-named Haswell) Speed and power: 2.60 GHz, 135 W Cache: 40 MB per processor Cores: 16 cores, 32 hyper-threaded cores QPI: 2x 9.6 GT/s Memory types: DDR4 1600/1866/2133, Reference: http://ark.intel.com/products/81060/Intel-Xeon-Processor-E5-2698-v3-40M-Cache-2_30-GHz
Memory	2133 MHz, 128 GB Total
Local Storage	Cisco 12G SAS Modular Raid Controller, 2x1TB RAID1
PCIe	Gen 3
NICs	3 x Intel® X520-DA2 Adapter (1 used)
BIOS	Version: C240M4.2.0.6a.0.051220151501

Hardware	UCS C240 M3S
Chipset	Intel® C600 series chipset
Processor	Intel® Xeon® Processor E5-2690 (code-named Sandy Bridge) Speed and power: 2.90 GHz, 135 W Cache: 20 MB per processor Cores: 8 cores, 16 hyper-threaded cores QPI: 2x 8.0 GT/s Memory types: DDR3 800/1066/1333/1600 Reference: http://ark.intel.com/products/64596/Intel-Xeon-Processor-E5-2690-20M-Cache-2_90-GHz-8_00-GTs-Intel-QPI
Memory	1333 MHz, 256 GB Total
Local Storage	LSI MegaRAID SAS 9266-8i, 2x500G RAID1, 4x500G RAID5
PCIe	Gen 3
NICs	3 x Intel® X520-DA2 Adapter (1/2 used)
BIOS	Version: C240M3.2.0.3.0 Build: 08/01/2014

Table 2: The Evaluation Software

Software Versions System Capability	
Host Operating System	Ubuntu 14.04.3 LTS Kernel version: 3.13.0-63-generic
VM Operating System	Ubuntu 14.04.3 LTS Kernel version: 3.13.0-63-generic
Libvirt	libvirtd (libvirt) 1.2.2
QEMU	QEMU-KVM 2.2.1
DPDK	2.0.0
Open vSwitch	2.4.0
Cisco VPP	e7cad06e323e31414990f35bc94e2b18 vpe.deb.vpe_nodebug.VMS20_FCS_09_10_2015
XR9Kv	7dbb5327be0395df99e0ca339ec6a1ef xrv9k-full-x.iso.r54.multitm.injfix.sept11

Boot Settings	
Host Boot Settings	Hyper-threading disabled TurboBoost enabled GRUB_CMDLINE_LINUX="isolcpus=1-15 hugepagesz=1G hugepages=64" GRUB_CMDLINE_LINUX="isolcpus=1-31 hugepagesz=1G hugepages=64" GRUB_CMDLINE_LINUX="intel_iommu=on isolcpus=1-15 hugepagesz=1G hugepages=64" GRUB_CMDLINE_LINUX="isolcpus=1-31 default_hugepagesz=1GB hugepagesz=1G hugepages=64" *For XR9Kv performance test:* *TurboBoost enabled* GRUB_CMDLINE_LINUX="intel_iommu=on isolcpus=1-15 nohz_full=1-15 rcu_nocbs=1-15 default_hugepagesz=1GB hugepagesz=1G hugepages=64"
VM Kernel Boot Parameters	*For Intel l2fwd VM:* *GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"* *GRUB_CMDLINE_LINUX="console=ttyS0,115200 isolcpus=1-2 default_hugepagesz=1GB hugepagesz=1G hugepages=2"*

Test methodology and setup

We used Intel's Ethernet frame forwarder tool (l2fwd) as a reference VNF to verify the forwarding performance of the virtual switches. It was installed in the guest operating system (OS) with enabled "hugepages" option, improving the efficiency of the memory management.

Cisco decided to "pin" three CPU cores for the guest OS. Pinning is a technique typically used to statically assign CPU cores to threads that have near real-time requirements; without pinning, the time-sharing of CPU cores between threads would create delay variation in switching and reduce the efficiency. DPDK vhost-user ports (user space socket servers) were configured for the data-plane connectivity between the virtual switch and the guest OS.

We exercised multiple scenarios, as shown in the diagram below. In all cases, traffic was generated by an Ixia load generator with two 10GigE interfaces connected to the UCS system.

Topology 1: VNF-to-VNF, measuring the performance of pure virtual switching — important for service chaining, for example. Traffic was sent by passthrough (bypassing the vSwitch) to l2fwd instances, which interfaced with the vSwitch.

Topology 2: Virtual path performance across NIC, vSwitch and VNF — suitable for regular, single-VNF full-path network scenarios.

Topology 3: Standalone vSwitch forwarding between physical NIC ports, a reference test for vSwitch performance (in this instance, used for lookup tests only).

Topology 4: Standalone VNF forwarding between physical NIC ports without a vSwitch, to get a baseline figure of physical passthrough performance without vSwitch.

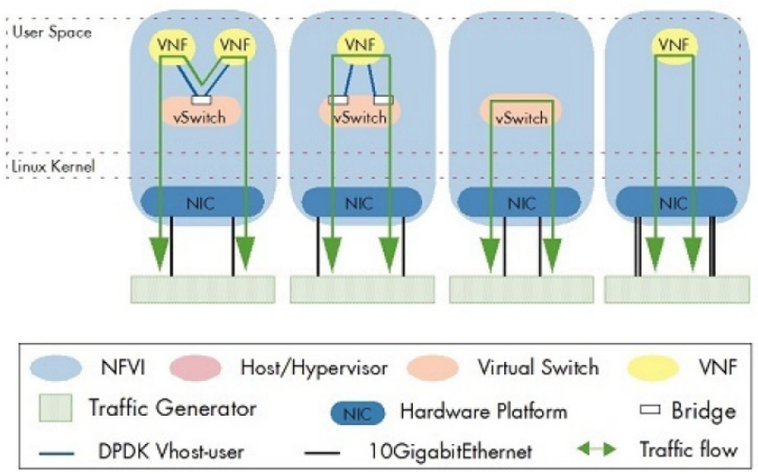

Figure 2: vSwitch Test Topologies (from left to right): VM-to-VM; full virtual path; vSwitch only; passthrough evaluation of Cisco's IOS XRv 9000 virtual router.

All tests were carried out in line with the methodology specified by the OPNFV VSPERF initiative for vSwitch performance testing. These methods are mostly based on one of the foundations of IP benchmarking standards, IETF RFC 2544 (Benchmarking Methodology for Network Interconnect Devices).

In the course of the joint pre-staging, we discovered that virtual switches behave very differently from hardware-based switches, as they are based on non-real-time environments. As a result, we adapted the methodology, including: a) multiple runs to confirm results statistically; b) accepting a minimal loss threshold (0.001 %) for some test runs; c) running linear sweep tests. For more details, see the final page of this report, "Cisco and EANTC evolved vSwitch test methodology."

VM-to-VM performance

Evaluation overview: *Cisco's VPP reached up to 10 Gbit/s, 1.6 Million frames/second throughput, OpenvSwitch up to 7 Gbit/s, 1.09 Million frames/second — each with a single core. VPP showed more predictable behavior than OVS when both were brought to their limits. Performance in SandyBridge and Haswell architectures differed only slightly.*

Test topology 1 verified switching performance between virtual network functions via the virtual switch and their vhost-user virtual links. This will be one of the most important use cases in the future, when multiple VNFs will share host resources.

It is of course crucial that the vSwitch does not take all of the host's compute power away from the VNFs. We discussed with Cisco how many cores should be allocated to a vSwitch and agreed to use just a single core for the vSwitch.

This was a good decision for two reasons. First, this is a good baseline that can be scaled to multiple cores later. Second, OVS 2.4.0 interacting with DPDK 2.0 and QEMU (the virtual queue library) was lacking a specific multi-queue ability required for multi-core switching support. We will report this issue back to the relevant projects.

The throughput performance results were very interesting, so let's discuss the following graphs:

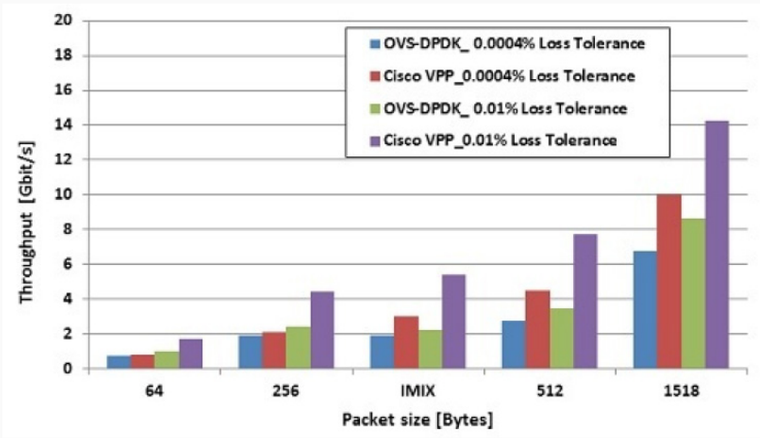

Figure 3: VM-to-VM throughput performance on Haswell architecture.

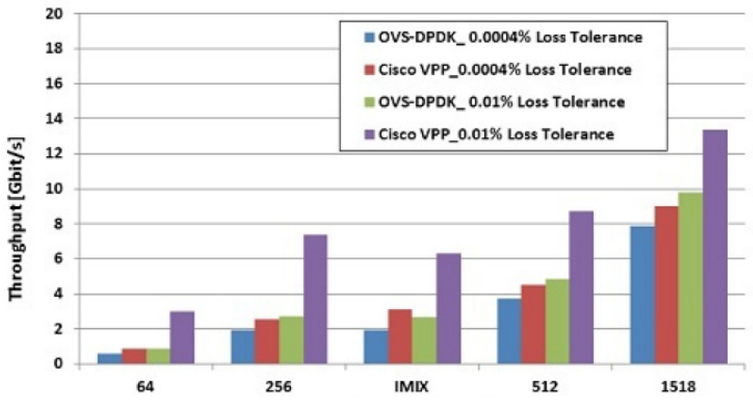

Figure 4: VM-to-VM throughput performance on Sandy Bridge architecture.

With our standard setting accepting 4 parts per million (4 ppm, or 0.0004%) packet loss, Open vSwitch reached up to 1.1 million Ethernet frames per second (Mfps) with small frames, or 7 Gbit/s throughput with large frames. Cisco's commercial platform, VPP, reached up to 1.2 Mfps with small frames, or 10 Gbit/s with large frames. Naturally, performance depends primarily on numbers of frames, so the throughput was much lower with smaller frames of 64 and 256 byte sizes. We also used standard Internet Mix (IMIX) mixed frame sizes which yielded 2 Gbit/s for Open vSwitch and 3 Gbit/s for VPP. These numbers would likely scale with a higher number of cores.

It was a big surprise that the throughput substantially increased when we accepted slightly higher frame loss. We noticed that a non-zero, but small, loss ratio persisted over a broad range of throughput: for example, accepting a higher loss of 0.01% or 100 ppm, Cisco's VPP yielded 14 Gbit/s throughput instead of 10 Gbit/s with 1518 byte sized frames. The reason is that a non-real time system may lose a small number of packets by buffer overflows even if it is not fully loaded. In contrast, hardware switches typically have a real-time approach and a throughput limit beyond which the packet loss rate increases quickly and linearly.

How much packet loss is acceptable depends on the application scenario and it should be noted that the vSwitch is only one component contributing to the end-to-end solution. For example, video-over-IP can typically accept 0.1% or 1000ppm loss (depending on the codec and transport stream settings); a vSwitch contributing 10% of that limit (0.01% loss or 100ppm) will likely be contributing too much loss for that application. (Note, though, that the industry has not converged on specific acceptable loss values yet.)

Next, we measured the forwarding latency as an indication of how long the frames are stuck in any buffers in the system. Naturally, latency was higher for larger packets due to the serialization delays (the time it takes to get a packet off and back on the wire) and the amount of time required to read and copy packets in memory. For small 64-byte sized frames, both Open vSwitch and VPP showed latency of 20 microseconds, which compares with hardware switches. 1518-byte sized frames were handled with an average latency of 100 microseconds. The maximum latency varied quite a bit, though: While VPP serviced all packets within at most 200 microseconds, OVS sometimes took more than 400 microseconds for the same task.

Both the sensitivity to loss and the maximum latency of Open vSwitch showed that the open-source implementation is less predictable at this point. Cisco's VPP technology excelled, behaving in a more controlled way (see graph below).

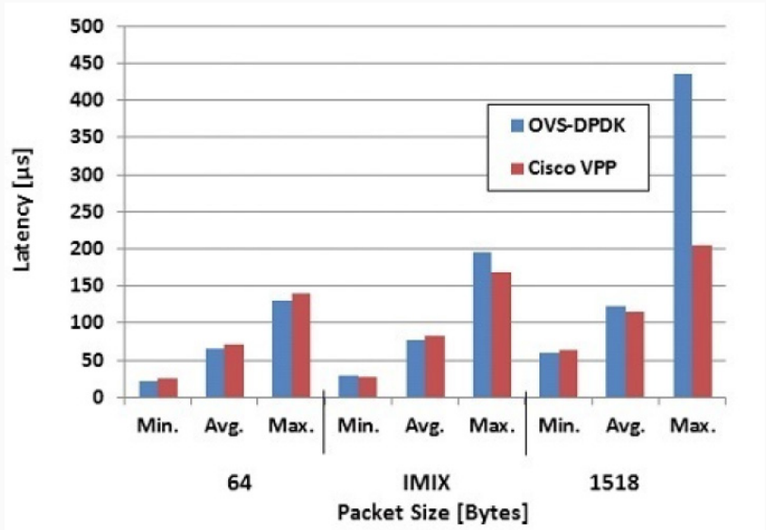

Figure 5: VM-to-VM latency performance comparison at maximum throughput.

Full virtual path performance

Evaluation overview: *In the full virtual path scenario, Cisco's VPP reached up to 20 Gbit/s, 2.5 million frames/second throughput with a single core. Open vSwitch provided between 20-40 Gbit/s, 4-6 million frames/second throughput, varying greatly across measurements.*

While the previous test scenario focused on vSwitch throughput only, we aimed to cover the full virtual path performance in the second scenario. The full virtual path performance includes the hardware (Network Interface Card), the virtual switch and a reference VNF. This scenario is important for single virtual network functions aiming to use the vSwitch, as all VNFs should eventually do (instead of bypassing the virtualization infrastructure). In this scenario, each frame passes the vSwitch twice — in the vSwitch throughput results, we multiply the load generator's frame numbers with two accordingly.

The vSwitch environment was identically configured as before.

We started with linear loss rate and standard RFC2544 lossless single run tests for both VPP and OVS-DPDK solutions.

The VPP implementation yielded stable results up to 20 Gbit/s throughput with large packets and 2.5 Mfps with 64-byte packets.

However, we quickly noticed inconsistencies in throughput performance for OVS-DPDK virtual switch testing. OVS-DPDK showed only 8 Gbit/s throughput with large packets, which seemed unreasonably low. When we reran the test, the results were different. Due to spurious and transient very small packet losses with the OVS-DPDK implementation, the standard RFC2544 results were simply not reproducible and often unreasonably bad. To adapt to the non-real time software environment, we subsequently changed the test methodology to "BestN/WorstN" to yield statistically significant results. We applied five test runs (N=5) for each test scenario. (For more details, see the final page of this report, "Cisco and EANTC evolved vSwitch test methodology.")

The following graphs show the results for BestN/WorstN and for the single run test. As expected, the measured throughput value for single run test was in between the BestN/WorstN throughput range for the VPP performance test. In contrast, OVS-DPDK shows the out-of-range values for the single run test and a large variation of results for lossless throughput. These results show that the OVS-DPDK implementation is currently not optimized for stable lossless Ethernet frame forwarding. Or, viewing things from the other side, one could conclude that RFC2544 old-school testing is not resulting in statistically significant values with virtual switches, such as OVS-DPDK. The results vary wildly, with IMIX between 3-14 Gbit/s in our five test runs.

We suggest using multiple test runs 'N' to determine accurate worst or best performance when OVS-DPDK is used. The exact number of 'N' could subsequently be derived mathematically based on the volatility of results, calculating a confidence interval. (We will save the reader from detailed mathematics here and will follow up with standards bodies.)

BestN/WorstN test results for full path of forwarding performance

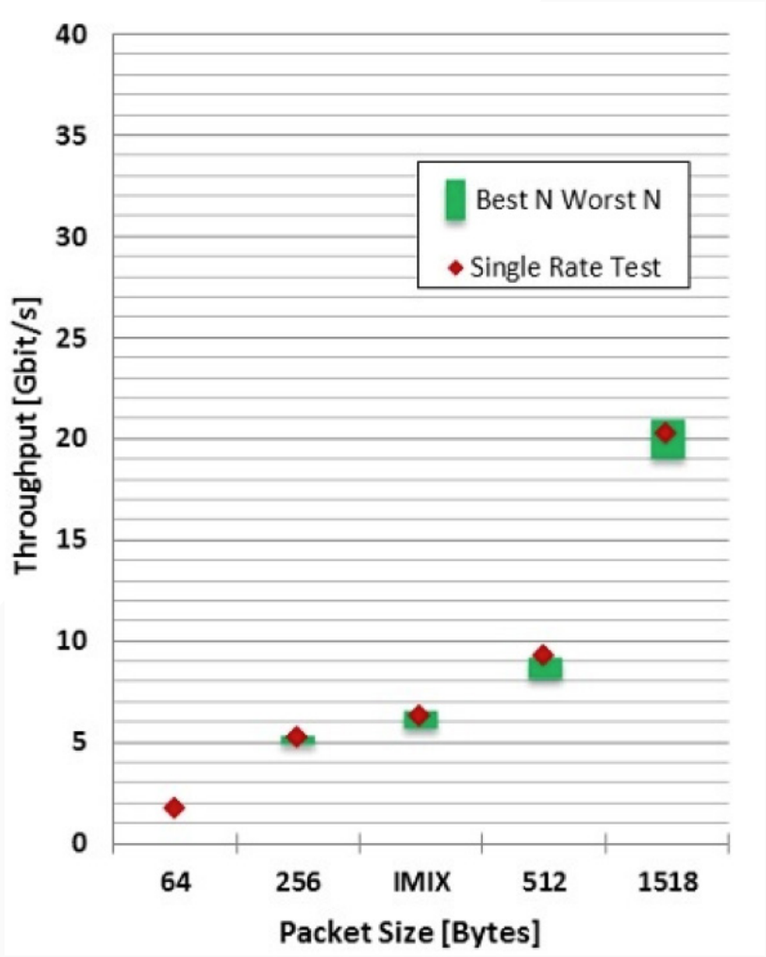

Figure 6: Full virtual path Cisco VPP lossless throughput performance on Sandy Bridge

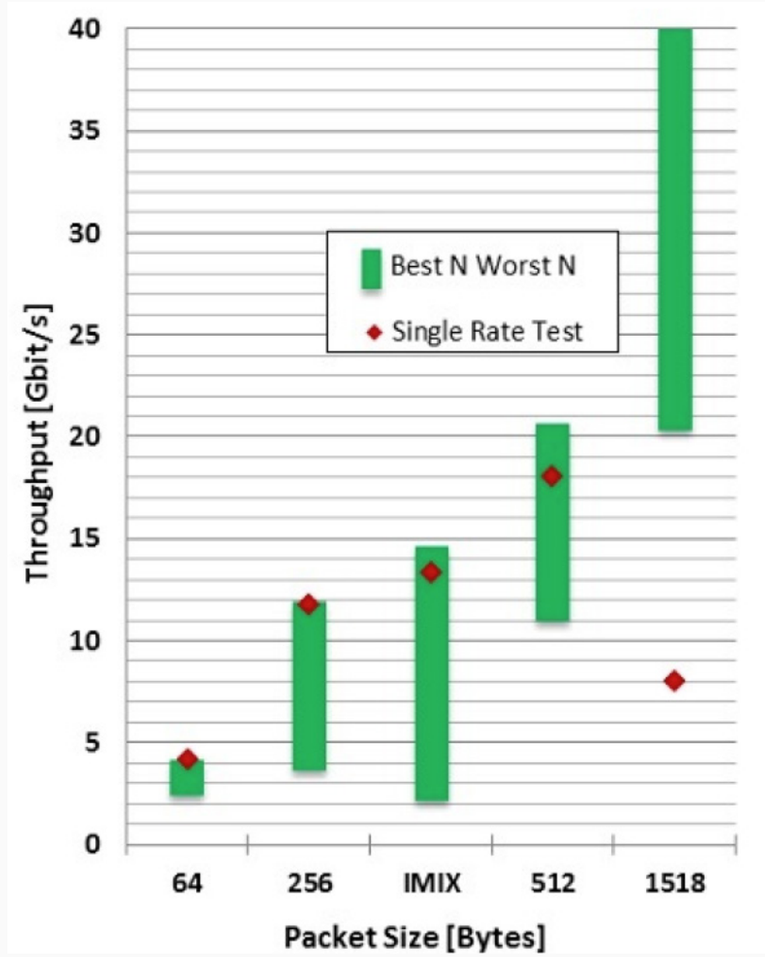

Figure 7: Full virtual path OVS-DPDK lossless throughput performance on Sandy Bridge

OVS-DPDK showed small minimum and average latency results across all packet sizes, however the maximum latency for small packets was very high, with 1,400 microseconds (as the graph below shows). VPP maximum latency was lower, providing more consistent treatment of packets, specifically for small packets and the IMIX packet size mix.

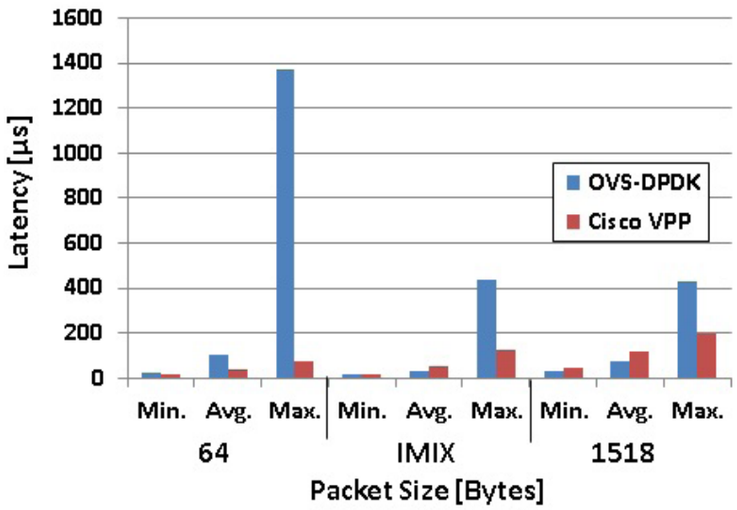

Figure 8: Full virtual path latency performance.

Virtual switch FIB scalability

Evaluation overview: *In a pure virtual switching scenario, VPP showed no throughput degradation when forwarding to 2,000 IPv4 or Ethernet MAC addresses at 20 Gbit/s, less than 1% reduced throughput towards 20,000 MAC addresses and 23% reduced throughput when forwarding to 20,000 IP addresses. OVS IPv4 and Ethernet forwarding was reduced by 81% when forwarding to 2,000 IPv4 addresses.*

Both previous scenarios focused exclusively on Ethernet layer throughput with a small number of flows because they both involved virtual machines. In the third scenario, we evaluated pure virtual switching without any actual application. This is, of course not a realistic application setup: instead it is a reference test of vSwitch properties that need to be determined independently of VNF performance.

One of the most basic and important scalability figures of an Ethernet switch is its ability to handle many Ethernet flows between different endpoints (associated with MAC addresses) in parallel. In a data center, there is usually much more East-West traffic between servers and services directly connected on the Ethernet segment than there is North-South traffic. vSwitches need to enable virtual services to participate in data center communication and need to be able to connect to many Ethernet destinations in parallel.

Separately, vSwitches obviously need to support many IP addresses in their forwarding information base table (FIB) simultaneously, when configured for IP forwarding. If traffic is routed towards a virtual firewall or a virtualized packet filter, there are usually tens of thousands of flows involved from thousands of IP addresses.

We verified forwarding performance of the standalone virtual switch with multiple layer 2/layer 3 FIB table sizes.

VPP showed its strengths based on the optimized, vector-based handling of large tables. It achieved very consistent IPv4 forwarding: The throughput was not dropped at all when forwarding to 2,000 IPv4 addresses compared to a single address scenario; throughput dropped only by 23 % when forwarding to 20,000 IPv4 addresses. The average forwarding latency was largely unaffected by the larger tables, and the maximum IPv4 forwarding latency was still bearable at 400 microseconds for 2,000 address entries and 1,200 microseconds for 20,000 entries.

In contrast, Open vSwitch seems to use less optimized FIB lookups. Throughput dropped from 20 Gbit/s (IPv4) and 8.8 Gbit/s (Ethernet) for the single-entry case down to around 4 Gbit/s in both cases for 2,000 IPv4 and MAC addresses. For 20,000 FIB entries the OVS-DPDK implementation was not usable in its current version as the throughput dropped to almost zero and maximum latency skyrocketed to 37 milliseconds (not microseconds!). We are not complaining — after all, it's free software — and in fact we hope that Cisco will contribute its VPP improvements back to OVS to improve the open source software.

These results indicate much better VPP performance in higher scale network deployments.

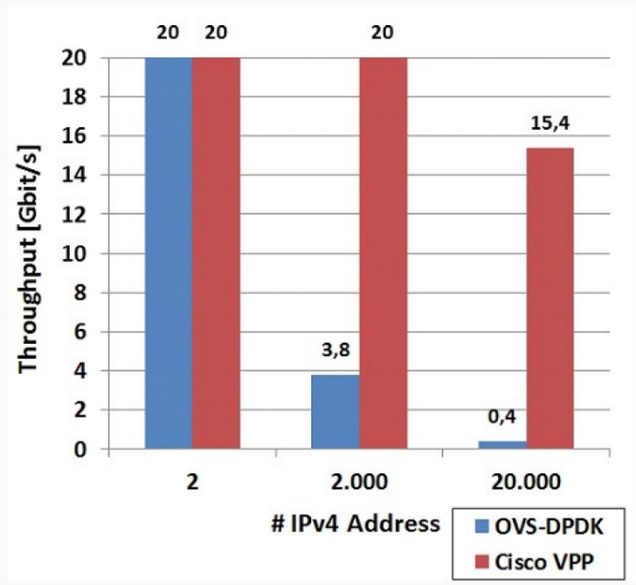

Figure 9: vSwitch-only IP forwarding performance.

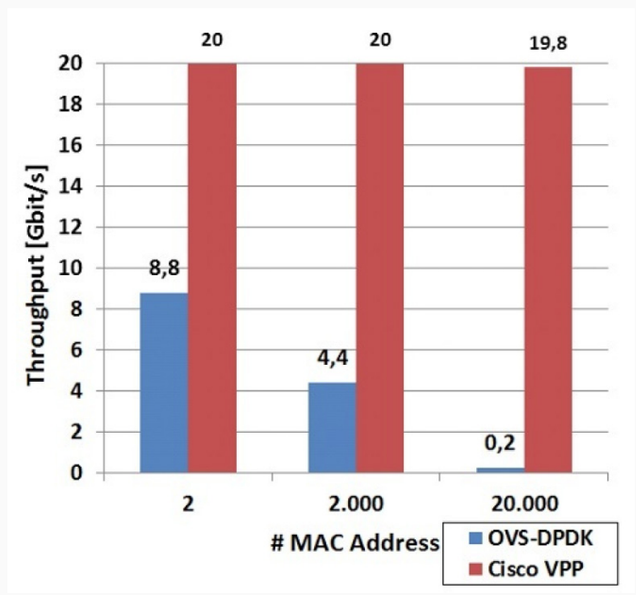

Figure 10: vSwitch-only Ethernet forwarding performance.

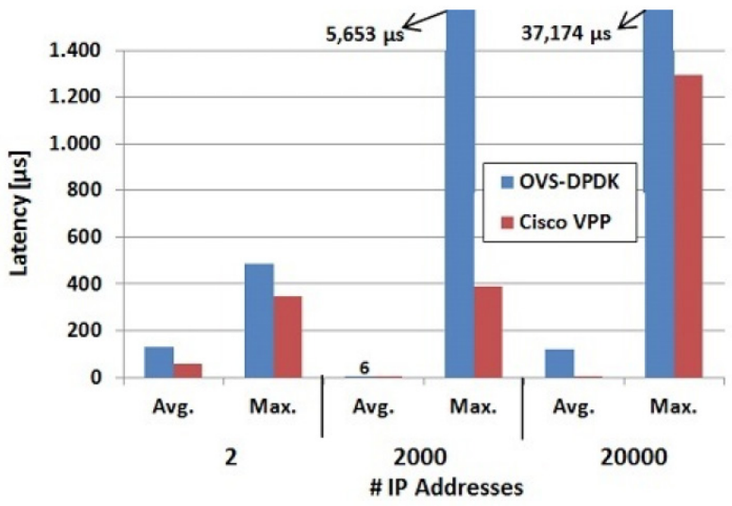

Figure 11: vSwitch-only latency performance.

Evaluating the Cisco IOS XRv 9000

Evaluation overview: *Cisco's IOS XRv 9000 virtual router excelled with up to 35 Gbit/s and up to 8.5 Mpps throughput in a feature-rich configuration, using 14 Haswell cores and PCI passthrough configuration. Forwarding latency was in line with expectations in most cases.*

As the last of four testing scenarios, we evaluated the performance of a commercial virtual router implementation provided by Cisco. As the product name "Cisco IOS XRv 9000" indicates, the virtual router that Cisco supplied to our test is based on IOS XR, the routing system Cisco developed first for the CRS-1 and has used for core and aggregation routers ever since.

For this performance test, Cisco used the UCS C240 M4SX (Haswell) hardware platform. 14 CPU cores (of the total 16 cores) were pinned for the virtual router. This makes sense, as the virtual router is the main virtual service in this case. The test utilized four 10GE interfaces connecting directly with the VM by PCI passthrough technology. In general, EANTC is much in favor of using vSwitches for all applications, but here the purpose of the test was to achieve a baseline performance evaluation of the virtual router without being limited by a vSwitch.

The Cisco team told us they chose the XRv 9000 virtual router from the company's portfolio of virtual routers since it uses VPP technology as well and is a full-featured router implementation for service provider environments.

The XRv 9000 was configured with rich features — ingress ACLs, ingress color-aware hierarchical policing, egress hierarchical QoS with parent-shaping, child-queuing, packet remarking and Reverse Path Forwarding (ipv4 verify unicast source reachable-via any) and a mix of IPv4 and IPv6 traffic. EANTC validated the 174-line IOS XR configuration in detail.

The XRv 9000 achieved lossless throughput of more than 35 Gbit/s for packets sizes of 512 bytes or more and up to 8.5 million packets per second (Mpps) with small packets.

This is a very impressive result that is, of course, influenced by the 14 cores being the workhorses for the virtual router, whereas all previous scenarios had been tested with just a single core. Nevertheless it is reassuring to see that the virtual router can handle 62% of line rate with a realistic mix of packet sizes, and 90% of line rate with large packets of 512 bytes or more. There have been higher throughput values touted before, but let's not forget that the XRv 9000 is a full-featured virtual router and we actually used quite a few of those features in our test. Cisco certainly did not go for the low-hanging fruit in this test.

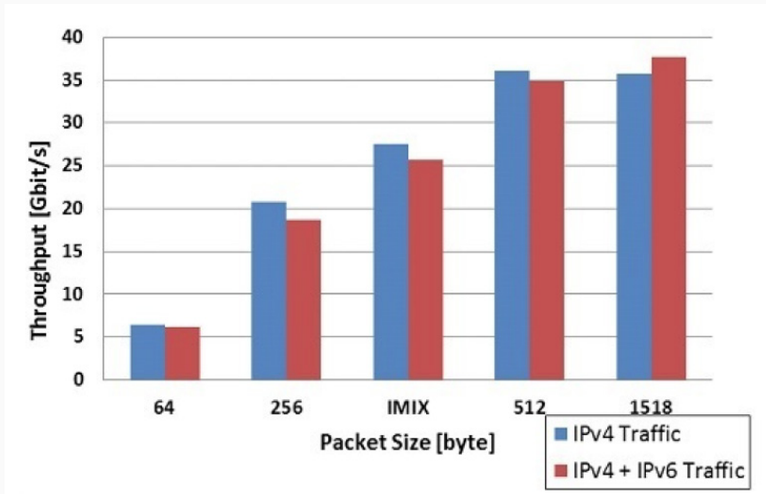

Figure 12: Cisco IOS XRv 9000 IPv4/Ipv6 throughput performance.

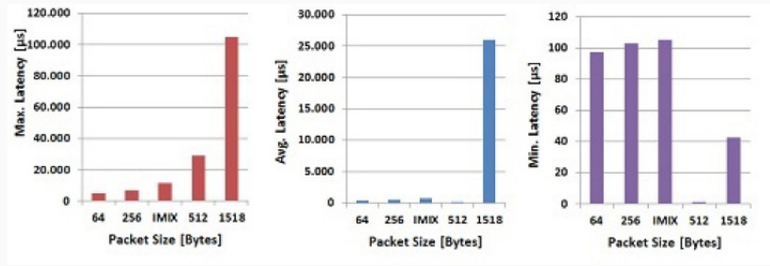

Figure 13: Cisco IOS XRv 9000 minimum, average and maximum latency performance.

With around 100-1,000 microseconds, the minimum and average latencies were well line with our expectations for all packet sizes, except 1518 bytes. The 1518 bytes results, coupled with the fact that the maximum latencies rose to 25-100 milliseconds, indicates an implementation or configuration issue.

Conclusion of vSwitch test findings

At the end of a full week of in-depth vSwitch testing, including overnight and weekend automated tests (those RFC2544 runs consume a lot of time!), we gathered a tremendous amount of data about Cisco's VPP (both as a pure vSwitch and serving in the XRv 9000 virtual router) and the DPDK-enabled version of Open vSwitch.

vSwitch technology and Cisco's virtual router implementation are definitely getting there. We witnessed a single Haswell or Sandy Bridge core achieve up to 20 Gbit/s Ethernet switching throughput and 2.5 Gbit/s virtual routing throughput. This is really a great step for an industry which, let's not forget, is still in the early stages of a new technology development cycle. It underlines the power of open source development, where many vendors cooperate to progress quickly.

At the same time, the VPP performance results were much more consistent and reliable than those of Open vSwitch. Obviously, the traditional vendor model still has advantages when it comes to quality assurance and when reliable software needs to be bundled for use in mission-critical service provider applications.

The EANTC test is one of the first comprehensive, independent and public vSwitch performance evaluations. The results confirm that the concept of virtual switching and routing in the context of the ETSI NFV virtualization model is feasible.

Cisco's commercial VPP implementation used amazing techniques to get more consistent performance out of the system, while it's clear that the open source solution will soon become usable for large-scale deployments once a few more glitches will have been eliminated.

By the beginning of 2016, there should be no more good reason for using the “passthrough" direct network hardware access method that breaks virtualization concepts.

With the vSwitch test done, EANTC can confirm that the machine room performs as needed. In Part 2 of this NFVi evaluation report, we will take the next steps and look at the other main service provider pain points — the manageability and reliability of a virtualized infrastructure.

Cisco and EANTC evolved vSwitch test methodology

All throughput measurements based on RFC 2544 use a binary search algorithm. A binary search allows test teams to find the throughput with a specified resolution in a minimum number of traffic runs. Basically, when searching between 0 and 100%, one starts with 100% throughput; if that run yields zero loss, the test is done, otherwise the next run is at 50%. Subsequently, either 25% or 75% are tested depending on the result of the previous measurement. The next steps continue at half of the previous interval steps until a specified precision has been reached.

We noticed, however, that the Open vSwitch did not always show reproducible or linear behavior. Some measurements at a certain load yielded zero packet loss. Others with the exact same rate resulted in a very small loss (for example, 10 packets out of a 1 million), seemingly an effect of buffer or interrupt management. In other scenarios, the Open vSwitch showed small loss, for example at 12% throughput, but continued to function without loss at 13%, 14% and 15% throughput — probably due to the same non-deterministic small loss behavior.

We extended the test methodology, including the standard RFC 2544, as follows:

Single run test

Single Run (SR) tests execute a single run of RFC2544 binary search algorithm to measure the throughput for defined frame sizes. We used 64, 256, 512 and 1518 bytes frame and as well as IMIX frame with distribution 64 bytes:7, 570 bytes:4, 1518 bytes:1.

Linear loss rate test

This test measured packet loss ratio resulting from linear increase of offered load from 1% of line rate to 100% of line rate, with step of 1% of line rate. IMIX frames were used for the measurement.

BestN/WorstN test

This test uses more samples to drive the binary search and yield statistically more accurate results. This keeps the heart of the RFC2544 methodology, still relying on the binary search of throughput at specified loss tolerance, while providing more useful information about the range of results seen in testing. Instead of using a single traffic run per iteration step, each traffic run is repeated N times and the success/failure of the iteration step is based on these N traffic runs. We defined two types of revised tests — *Best-of-N* and *Worst-of-N*.

Best-of-N: Best-of-N indicates the highest expected maximum throughput for (packet size, loss tolerance) when repeating the test.

Worst-of-N: Worst-of-N indicates the lowest expected maximum throughput for (packet size, loss tolerance) when repeating the test.

Figure 14: Validating Cisco's NFV Infrastructure.

Carrier-grade high availability and reliability of Cisco's NFVi

Naturally, any network infrastructure solution in the telecommunications world is required to be highly available and operationally reliable. Major customers in the wholesale business and in key industries such as the financial sector have long required network availability values beyond 99.9% or "three nines" — this can be achieved only by creating redundancy with hot standby components and alternative paths.

To achieve a three-nines available service, each contributing part of the service — data center, core and aggregation network, access network — needs to be even more reliable, as the reliability figures of each module will multiply statistically: Six components, each 99.99% available, create a service that will be 99.94% available.

This is one of the two reasons why the industry created a goal of "five nines" i.e. 99.999% availability. The other is that service providers' lawyers decided to measure availability in a relaxed a way as possible, evaluating over the course of a full year. A customer requiring at the very most just one hour of end-to-end service downtime needs to demand 99.99% availability, the equivalent to about 52 minutes per year: This is why a data center implementing a virtualized component of a network service must be able to provide 99.999% availability.

At EANTC, we have tested high availability features of many Cisco network components before — core, aggregation and edge routers, data center switches, mobile core, etc. — and we are confident that it is possible to build highly available network infrastructures. But we had not previously evaluated NFVi high availability prior to this evaluation.

High availability (HA) comprises (at least) three aspects:

Design: The end-to-end infrastructure should be designed holistically, eliminating any single point of failure and taking all layers into account — hardware, virtualization infrastructure (NFVi), network services, management and end-to-end connectivity.

Co**nfiguration:** The HA solution must be configured correctly at installation time, including its failover mechanisms, which will rarely be exercised in the production network in the best case, so without testing in advance there would be no guarantee they actually work at all.

Security: It is critical to put security mechanisms in place to reduce the risk that the solution's configuration might be compromised.

The HA solution's correct functionality and performance should be audited frequently as a health check and regression test.

Cisco presented a number of products (modules) for evaluation that contribute to these goals. In contrast to the vSwitch performance tests reported earlier, these Cisco environments were set up to focus on functionality, not performance or actual high availability figures.

Automated and validated OpenStack installation

Currently, any OpenStack installation is a tedious, highly manual and thus error-prone process, specifically when it comes to more elaborate high availability configurations. Another issue is that pure OpenStack configuration is insufficient to get an NFVi deployed: There are additional data-plane components that the administrator needs to deploy and integrate, such as the hypervisor, storage platform and switching options. Cisco provides scripts around this installation process, automating it and allowing efficient verification of the configuration.

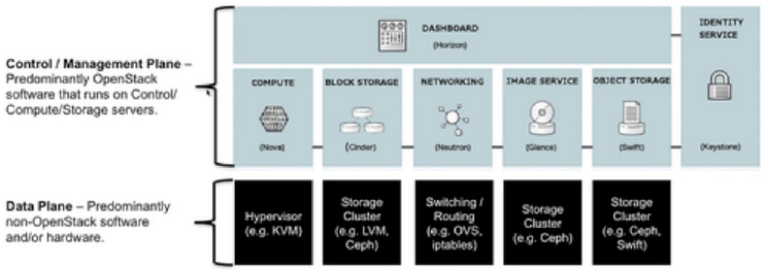

Figure 15: OpenStack and additional NFVi components.

Cisco's deployment capabilities include user input and configuration validation, bare-metal and Openstack installation. Cisco demonstrated a command-line installer offering six modes of operation, as shown in the following screenshot:

```
[root@flm-server-23 bootstrap]# ./installer/runner/runner.py -l
args:  []
Mercury Installer:
==================
+---------------+--------------+
| Operations    | Operation ID |
+---------------+--------------+
| VALIDATION    |      1       |
| BAREMETAL     |      2       |
| COMMONSETUP   |      3       |
| CEPH          |      4       |
| ORCHESTRATION |      5       |
| VMTP          |      6       |
+---------------+--------------+
[root@flm-server-23 bootstrap]#
[root@flm-server-23 bootstrap]#
[root@flm-server-23 bootstrap]#
[root@flm-server-23 bootstrap]# ./installer/runner/runner.py -s 2,3,4,5,6
args:  []

Skipping steps ['2', '3', '4', '5', '6']. Continue (Y/N)Y
```

Figure 16: Cisco installer choices.

The operations 'BAREMETAL,' 'COMMONSETUP' and 'ORCHESTRATION' had been pre-deployed by Cisco prior to our evaluation. EANTC was able to check that the deployment had been made.

Cisco explained to us that the 'VALIDATION' function detects improper parameters before starting with any installation, so eliminating unexpected issues during deployment.

Input parameters for the installer (both mandatory and optional ones) are specified in a YAML file. The installer verifies the presence of all mandatory parameters. It also verifies that the values provided for both mandatory and optional parameters are valid.

The validation method ran swiftly, taking around half a minute. It showed Linux boot-style checks:

```
#########################################
                Mercury Installer
#########################################
1/6][VALIDATION:INIT]                                          [ DONE! ]   0 min 1 sec
1/6][VALIDATION:Check server config]                           [ DONE! ]   0 min 2 secs
1/6][VALIDATION:Check NTP servers]                             [ DONE! ]   0 min 12 secs
1/6][VALIDATION:Check PROXY/DNS servers]                       [ DONE! ]   0 min 32 secs
1/6][VALIDATION:Check the IP pool range]                       [   |   ]   0 min 33 secs

User Input Validations!
+-----------------------------------------+--------+-------+
| Rule                                    | Status | Error |
+-----------------------------------------+--------+-------+
| Check Section Presence                  |  PASS  | None  |
| Check Total Network Count               |  PASS  | None  |
| Rack Placement of Controller            |  PASS  | None  |
| Check Section Info                      |  PASS  | None  |
| Check Cobbler Input details             |  PASS  | None  |
| Control node check                      |  PASS  | None  |
| Compute node check                      |  PASS  | None  |
| NTP servers Check                       |  PASS  | None  |
| PROXY/DNS servers Check                 |  PASS  | None  |
| Volume Driver check                     |  PASS  | None  |
| check servers in ROLES and SERVERS match|  PASS  | None  |
| Server Common Info Check                |  PASS  | None  |
| Check Redhat Subscription details       |  PASS  | None  |
| Check Network provided for ['storage']  |  PASS  | None  |
| Check Default Route ['storage']         |  PASS  | None  |
| Check VLAN provided for ['storage']     |  PASS  | None  |
| Check Gateway provided for ['storage']  |  PASS  | None  |
| Check Network provided for ['tenant']   |  PASS  | None  |
| Check Default Route ['tenant']          |  PASS  | None  |
| Check VLAN provided for ['tenant']      |  PASS  | None  |
| Check Gateway provided for ['tenant']   |  PASS  | None  |
| Check Network provided for ['api']      |  PASS  | None  |
| Check Default Route ['api']             |  PASS  | None  |
| Check VLAN provided for ['api']         |  PASS  | None  |
| Check Gateway provided for ['api']      |  PASS  | None  |
| Check Network provided for              |  PASS  | None  |
| ['management', 'provision']             |        |       |
| Check Default Route ['management',      |  PASS  | None  |
| 'provision']                            |        |       |
| Check VLAN provided for ['management',  |  PASS  | None  |
| 'provision']                            |        |       |
| Check Gateway provided for              |  PASS  | None  |
| ['management', 'provision']             |        |       |
| Check Segment Info                      |  PASS  | None  |
| Check External VIP and Internal VIP     |  PASS  | None  |
| IP Pool Check                           |  PASS  | None  |
| Cobbler API Server Status               |  PASS  | None  |
| Kibana Web Server Status                |  PASS  | None  |
| ElasticSearch Web Server Status         |  PASS  | None  |
| Logstash Web Server Status              |  PASS  | None  |
+-----------------------------------------+--------+-------+
1/6][VALIDATION:Check the IP pool range]                       [ DONE! ]
1/6][VALIDATION:Logstash Web Server Status]                    [ DONE! ]   0 min 34 secs
```

Figure 17: Cisco install validation tool in action.

To check if it would detect any failures, we created an issue intentionally and ran the installer once more:

```
User Input Validations!
+-----------------------------------------+--------+-----------------------------+
| Rule                                    | Status | Error                       |
+-----------------------------------------+--------+-----------------------------+
| Check Section Presence                  |  PASS  | None                        |
| Check Total Network Count               |  PASS  | None                        |
| Rack Placement of Controller            |  PASS  | None                        |
| Check Section Info                      |  PASS  | None                        |
| Check Cobbler Input details             |  PASS  | None                        |
| Control node check                      |  PASS  | None                        |
| Compute node check                      |  PASS  | None                        |
| NTP servers Check                       |  PASS  | None                        |
| PROXY/DNS servers Check                 |  PASS  | None                        |
| Volume Driver check                     |  PASS  | None                        |
| check servers in ROLES and SERVERS match|  PASS  | None                        |
| Server Common Info Check                |  PASS  | None                        |
| Check proxy_port provided               |  FAIL  | Section: REDHAT_SUBSCRIPTION No |
|                                         |        | proxy_port provided         |
| Check Network provided for ['storage']  |  PASS  | None                        |
```

Figure 18: Cisco's install validation tool detecting an error.

In general, Cisco's installation tool is able to conduct a bare-metal installation as an automated bootstrap. It pulls all packages from the build server. Cobbler is used for PXE booting of a Linux VM. As the next step, it deploys OpenStack services in Docker containers. All the containers are started and maintained as Linux system processes.

Centralized logging and runtime network regression testing

All components of the NFVi create their own logs — including OpenStack, the vSwitch, storage solutions and others. It is a standard and longstanding challenge for any IT systems administrator to gain a holistic overview of what's going on by aggregating system logs. When there is an issue, probably one of the many system log files will report it. The difficult issue is to monitor all these logs and triage their messages into those that are critical, important or less important.

Cisco presented a solution that enables enhanced centralized logging using the ELK stack, a public domain solution. All logs are collected by a component called LogStash Forwarder. A second component called Kibana Dashboard is used for gaining an overview of all logs, and a third component called ElasticSearch allows heuristic searches across the log database.

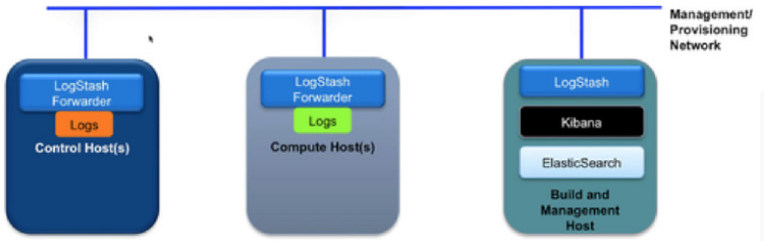

Figure 19: ELK logging overview

Figure 20: Kibana logging dashboard.

During the demonstration, live logs were shown to the EANTC team. Cisco provided a few basic searches across the database that did not show any major issues with the installation.

Runtime network regression testing

As we know, the fact that a solution worked at installation time does not mean it will run correctly forever. There are always reconfiguration activities taking place, either administratively driven or by intentional or unintentional hardware issues. It is important to check the functionality and performance of the NFVi frequently.

Cisco's VMTP is a Python application that contributes to this regression testing activity, covering the network connectivity. It automatically performs ping tests, round trip time measurement (latency) and TCP/UDP throughput measurements on an OpenStack cloud.

VMTP can be deployed and run by a command-line installer. This tool may be used to perform automated data path validation between VMs of a single tenant, between VMs of different tenants and between VMs in different LANs.

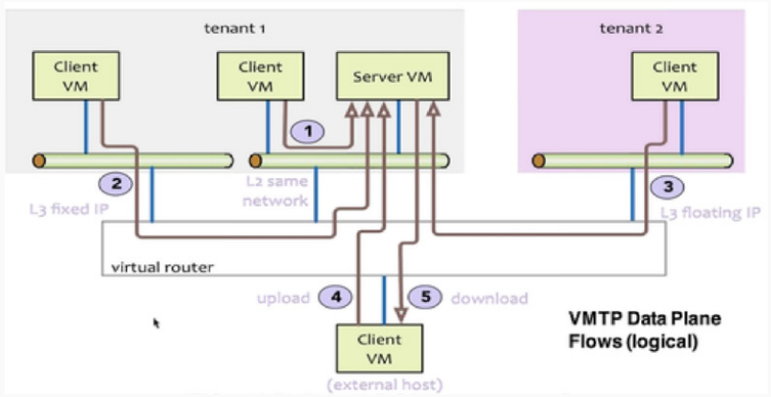

Figure 21: VMTP overview.

During our test session at Cisco's labs, we were able to witness VMTP running a number of preconfigured test scripts as shown in the following screenshot:

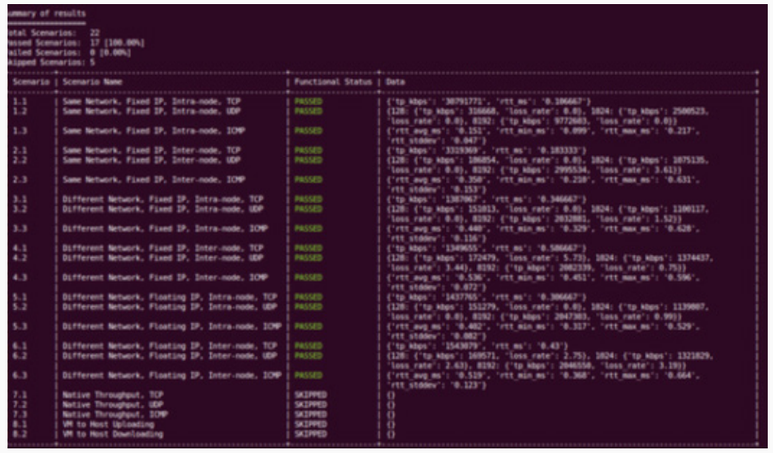

Figure 22: VMTP in action.

VMTP performed well in this standard situation. The EANTC team did not have the chance to inject an error to validate VMTP's ability to actually report issues, nor did we create a configuration deviating from Cisco's preconfigured demo environment.

High availability

The standard OpenStack high availability configuration is to deploy three OpenStack control nodes as a cluster. The control nodes are configured as active/standby mode using a "HA proxy" and a keep-alive daemon.

All OpenStack services except storage are deployed in active-active mode behind HA proxy. The HA proxy statistics page provides a consolidated view of the HA services and their status. It also gives overview of load balancing. During our session, Cisco showed the statistics report of a preconfigured system. We were able to witness at a glance that all services were running without issues.

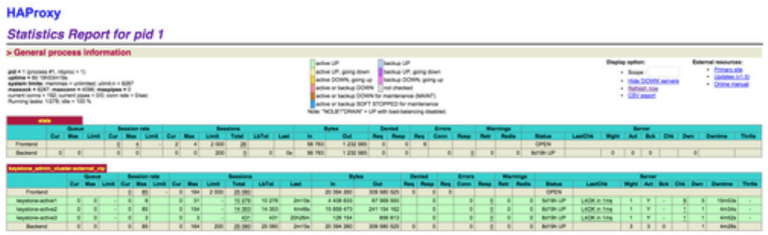

Figure 23: Cisco HAProxy statistics screenshot.

The EANTC team did not have a chance to play around with high availability configuration. We did not validate how the HAProxy behaves if there is an actual issue in one of the components that it is supposed to report on.

CloudPulse health check

CloudPulse is an open source tool developed by Cisco that conducts health checks of the cloud environment. As Cisco explained, CloudPulse tests can be configured on both operator and application level. Tests can be configured to run periodically or on demand.

CloudPulse operator tests can:

- Monitor the status of the cloud infrastructure
- Monitor API endpoint and functionality
- Monitor status of OpenStack services

In our session, Cisco demonstrated a number of preconfigured CloudPulse tests aimed at checking the health of OpenStack components (cinder, keystone, neutron, nova and so on). They were configured to run periodically. Since the OpenStack system was running correctly in our session, the tools did not report any issues, as shown in the following screenshot:

```
[root@flm-server-23 bootstrap]# cloudpulse result
+--------------------------------------+------+--------------------+----------+---------+
| uuid                                 | id   | name               | testtype | state   |
+--------------------------------------+------+--------------------+----------+---------+
| 39d0ce9d-2113-4cf5-9385-e066e5695483 | 7077 | cinder_endpoint    | periodic | success |
| fcb85f33-5757-4d93-8323-ed6715d39b5c | 7078 | keystone_endpoint  | periodic | success |
| 5466d37c-6000-43e1-9a10-93ce822bc7a2 | 7079 | neutron_endpoint   | periodic | success |
| 84c5df3a-e551-4739-b752-9a652489c3e0 | 7080 | glance_endpoint    | periodic | success |
| 153f1b3f-78d2-4af0-80bb-a8051733a25f | 7081 | nova_endpoint      | periodic | success |
| 30494bbd-cb33-4189-a680-14a8816729db | 7082 | cinder_endpoint    | periodic | success |
| 5f26f103-d09f-42ce-ac5c-9f3b83ded9e4 | 7083 | keystone_endpoint  | periodic | success |
| b33b9a4b-777b-4243-b88e-e0603774bb41 | 7084 | neutron_endpoint   | periodic | success |
| 0fa4b304-2ec1-4b7c-80f9-948d465252dd | 7085 | glance_endpoint    | periodic | success |
| 1ee53aa3-bfd1-48f5-b2b4-ad1ee38093bb | 7086 | nova_endpoint      | periodic | success |
+--------------------------------------+------+--------------------+----------+---------+
```

Figure 24: CloudPulse in action.

Cloud99 Pre-Deployment OpenStack Validation

Another tool contributing to Cisco's toolbox of configuration checks is Cloud99. It can be used to verify the behavior of the cloud once the administrator has deployed it and before it is put into production. This point in time is the best opportunity to run actual high availability and performance checks without affecting the production tenants negatively if anything fails.

In our session, Cisco demonstrated Cloud99's ability to perform high availability testing of OpenStack services and core infrastructure components.

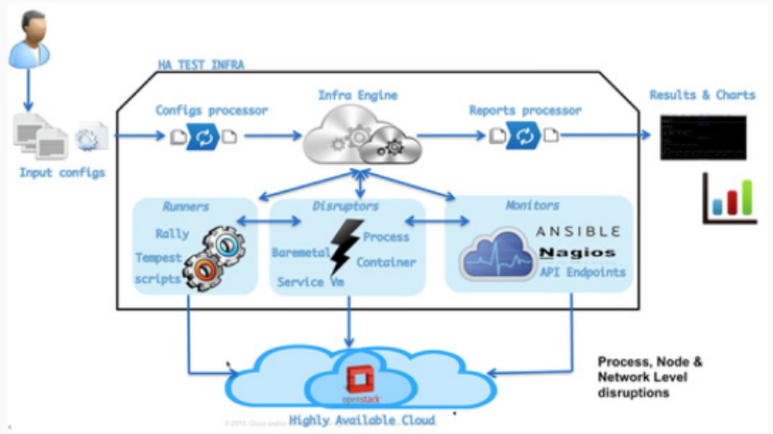

Figure 25: Cloud99 solution overview.

Cloud99 combines OpenStack's internal test runners driven by Rally (an OpenStack testing language) with test harness tool Ansible and monitoring tool Nagios.

We verified its features by disrupting the OpenStack Nova API while running scale/performance tests through Rally. The artificial issue injected in the test caused the Nova API to restart frequently. Nagios and the OpenStack Health API were unaware of the disruption, but the Ansible tests caught it as expected.

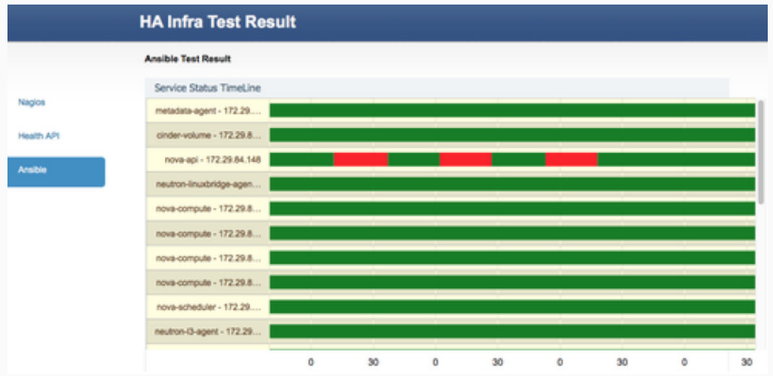

Figure 26: Cloud99 Ansible dashboard.

Summary

EANTC witnessed a number of Cisco tools complementing OpenStack that are designed to improve the consistency of deployment, simplify the high availability options and improve operational checks. These tools are an interesting and worthwhile approach to provide added value on top of OpenStack, partly with the help of other public domain tools. Cisco users are traditionally CLI-savvy, as are Linux-oriented IT administrators. They will probably like the versatility of Cisco's toolbox.

Putting VNFs to the test

As part of the first phase of our test of Cisco's cloud and virtualization portfolio, published in March this year, we looked at a small number of virtual network functions (VNFs) implemented by Cisco. (See Validating Cisco's Service Provider Virtualization & Cloud Portfolio)

Back then, Cisco claimed to support 60 VNFs — now the company says its ecosystem now includes more than 100 VNFs.

When the EANTC team returned to San Jose in September to conduct the NFVi tests, we took the opportunity to get some insight into two key VNFs: The Cloud DVR, a media service platform also known as the Virtualized Video Processing (V2P) platform; and the VPC-DI, the latest release of Cisco's virtual packet gateway for mobile networks.

Virtualized Video Processing (V2P)

As the vendor's team explained, the V2P "is Cisco's next-generation media service and applications hosting platform that provides the tools, frameworks, and containers required to host and manage standard media data plane functions. It includes software infrastructure containers that enable application decoupling and metadata storage. The applications range from Multi-Screen Live to Cloud DVR."

While this sounds relatively unexciting for the uninitiated, V2P solves a real operational problem that has bugged content providers: The media world becomes more dynamic every day, with new content channels sprouting all the time, yet the physical headend infrastructure to receive and transcode video streams and to stream the video data remains cumbersome to install, configure, scale and upgrade. As a virtualized service it would be much easier to bring up additional channels and to scale content streaming portfolios: It's that capability that Cisco invited us to verify, focusing specifically on the V2P Controller, one of the building blocks of V2P.

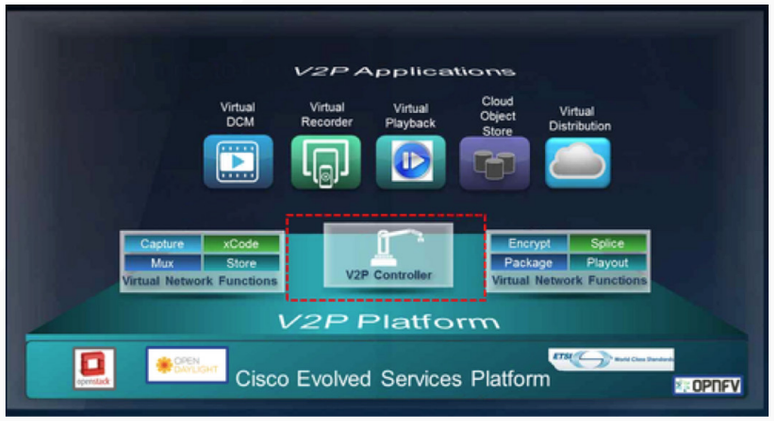

Figure 27: Virtualized Video Processing (V2P) applications overview.

Cisco explained that V2P includes functions such as real time ingestion of live channels and dynamic ingestion of video-on-demand (VoD) content, generating common manifests and index file formats, store media contents and media delivery.

The media functions comprise server instances called nodes, and logical clusters called endpoints. Each endpoint is composed of one or more nodes, and defines the (compute and storage) resources that are available in that cluster. The figure below shows the high-level architecture and functionality of the endpoints such as Media Capture Engine (MCE), Media Playback Engine (MPE) and Application Engine (App Engine), while a centralized log server (CLS) collects logging messages from each node.

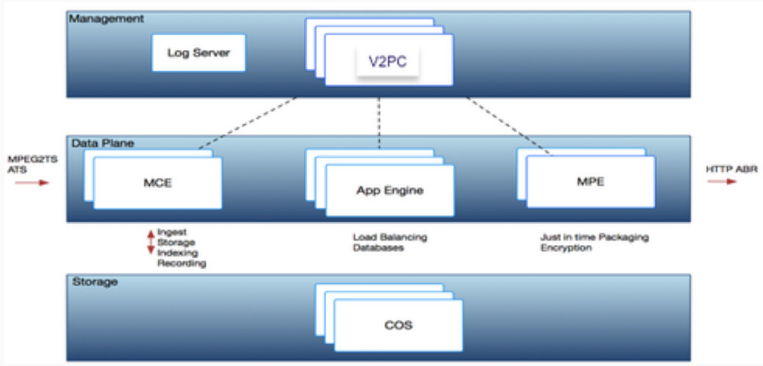

Figure 28: High-level Cloud V2P architecture.

EANTC validation of V2P

In the test session at Cisco's labs, we used V2P to create a live video channel, experiencing the flexibility of configuration. We used a graphical workflow that integrated virtual resource pools of capture engines and playout engines. The V2P platform was deployed on an OpenStack environment.

Once we verified the V2P deployment on OpenStack, we started to register Cisco's Cloud Object Store (COS) cluster to the V2P controller. COS is a storage middleware implementation that can interact with hardware drives and software. Once the COS cluster was configured, the Cisco team executed a script (called 'cosinit') on the storage drive to register the storage node with the V2P Controller.

Figure 29: V2P GUI after storage connection establishment.

Next, we created a channel media source by entering a streaming type (ATS or TS) and source URL and then created a channel lineup referencing the media source previously created. Channel lineup is a workflow that allows a content provider to configure channels that can be ingested to certain subsets of users.

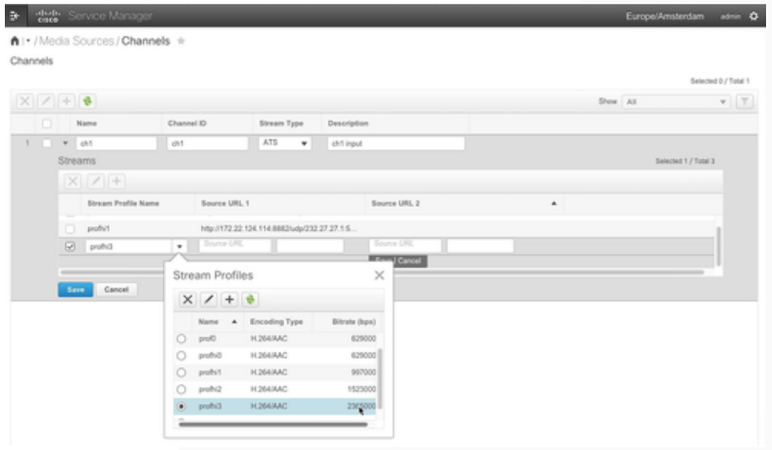

Figure 30: Stream profiles in V2P

Then we created a 'publish template' and an HTTP header policy for the channel: Publish templates allows the addition of HLS, HHS, HDS, CIF and CIF-DASH-TS transcoder formats.

Figure 31: V2P's publish templates.

Finally, we created a live channel by configuring live asset workflows via the drag and drop GUI:

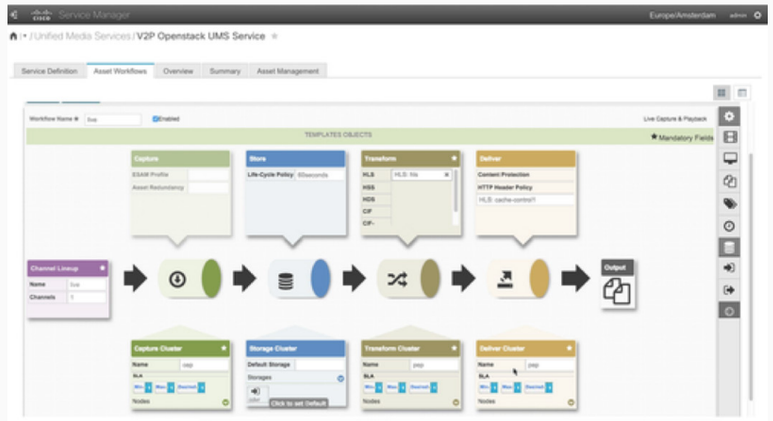

Figure 32: The V2P graphical user interface (GUI).

The GUI functioned as expected, and the detailed channel asset management view showed that the full configuration we entered correctly. We validated that our entries had been correctly accepted.

The templates were a very efficient way of configuring video content services. Given the limited time to investigate V2P, we were not able to verify the actual functionality or performance of the solution. Cisco asked us to see their application-layer configuration options, based on the OpenStack NFVi that we had evaluated in more detail in the previous sections.

We created templates for both HLS (for Apple IOS) and HSS (for Windows) and watched the channel simultaneously with both Apple and Windows-driven devices. In addition, we verified Cloud DVR recording, which allowed us to pause or rewind the video content.

Cisco's Virtual Packet Gateway VPC-DI

EANTC has evaluated Cisco's mobile core components multiple times — starting with Light Reading's test of the full mobile portfolio in 2010, which included the physical packet gateway ASR5000. Later, we ran a performance and functionality test of the first virtualized version in 2013. (See Testing Cisco's Mobile Network, Part I.)

Back then, Cisco had ported the ASR5000 directly, keeping the configuration interfaces and source code.

Meanwhile, Cisco has implemented the next step with the 'VPC-DI' where DI stands for Distributed Instance — an indication that the solution can now scale out per virtual component more flexibly. It is based on StarOS software that operates as a fully distributed network of multiple VMs. VPC-DI consists of two components:

- Control Function (CF): Two control function VMs act as an active:standby (1:1) redundant pair
- Service Function (SF): Service function VMs provide service context (user I/O ports) and handle protocol signaling and session processing tasks. A VPC-DI instance can contain up to 46 SF VMs. A minimum configuration for a VPC-DI instance requires four SFs — two active, one demux and one standby.

We verified that vPC-DI could be brought up with Day 0 configuration using Cisco's Network Services Orchestrator (NSO). For this demonstration Cisco used a pre-configured blueprint to create a vPC-DI instance. The instantiation process was completed by spinning up SF and CF VMs. According to Cisco, the day1 or day2 configuration of VPC-DI is possible via NSO.

However, it was not shown during the demonstration and Cisco used CLI commands and scripts for vPC-DI final configurations.

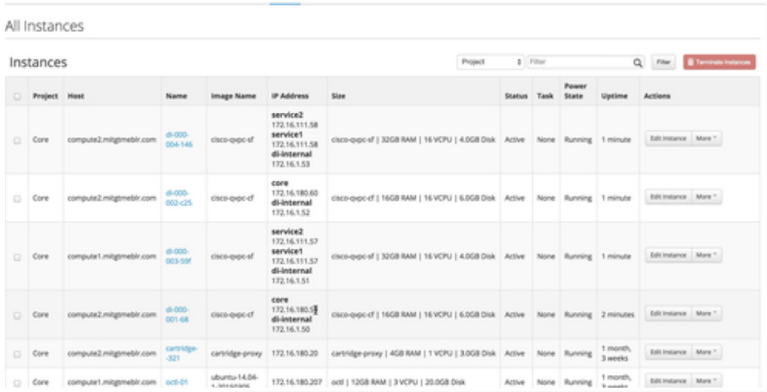

Figure 33: Platform orchestration overview of Cisco's NFVi.

We verified the system resiliency by performing card migration via command-line interface. In parallel, we verified that a previously initiated call was not terminated during the migration.

```
Monday September 28 17:16:20 EDT 2015
Slot        Card Type                                Oper State     SPOF  Attach
----------  ---------------------------------------  -------------  ----  ------
 1: CFC     Control Function Virtual Card            Active         Yes
 2: CFC     Control Function Virtual Card            Standby        -
 3: SFC     2-Port Service Function Virtual Card     Active         No
 4: SFC     2-Port Service Function Virtual Card     Standby        -
 5: SFC     2-Port Service Function Virtual Card     Active         No
 6: SFC     2-Port Service Function Virtual Card     Active         No
 7: SFC     None                                     -              -
 8: SFC     None                                     -              -
 9: SFC     None                                     -              -
10: SFC     None                                     -              -
11: SFC     None                                     -              -
```

Figure 34: Cisco VPC-DI virtual slots overview.

Next, we terminated one VM (active Service Function) forcefully via NSO — a scenario that might happen if the virtual service breaks down for whatever reason. It was auto recovered to standby mode — as expected. The existing call continued to operate, at least on the control plane (there was no test equipment for the data plane connected).

Finally, we performed live migration of vPC SF (VM) from one compute node to another using a CLI command. To maintain the active call previously established, Cisco used the Inter Chassis Session Recovery (ICSR) as an availability mechanism. In fact, the existing call was switched over to new chassis and continued to function. We did not measure how long (if at all) the data path was interrupted.

Summary

Having a powerful portfolio of VNFs ready is an important part of a cloud and virtualization strategy. Cisco quickly demonstrated two VNFs that contribute to this portfolio. The Cloud DVR/Virtualized Video Processing (V2P) platform looks like it can streamline a lot of operational functions for the content provider world, while its VPC-DI demonstrated functions ready to scale in a more granular way and to support VM operations. Both are worth a more detailed look from performance and high-availability levels in the future.

Single Pane of Glass management

The scalability of virtualized network services for really large network solutions is still a big unknown, hence service providers are experimenting quite a bit. Traditional enterprise cloud infrastructures may scale very well, but these usually lack the multivendor, open source approach that the telecommunications industry requires.

In addition, service provider VNFs are more complex because they need to support highly available network service chains across multiple data center locations.

So how can a network operator manage all this complexity? Only two years ago, this question would have attracted the disrespect of any seasoned Cisco operator, since the only viable answer was “using the CLI, of course!”

In the eyes of most network designers and operators, the complexity of Cisco's feature-rich products was never met sufficiently by the company's graphical management tools.

Well, the situation has definitely changed. As our journey across Cisco virtualized solutions in this article has shown, the multi-layered complexity of these solutions, the vast number of solution modules involved, and the lack of integrated configuration and troubleshooting interfaces in an open source world makes it virtually impossible to master the configuration using a command-line interface.

So, surprisingly, there is both an opportunity and an overwhelming need for Cisco to come up with an integrated network management solution for its virtualized services portfolio.

Multi-data center, multi-VIM service chain configuration

During our visit to Cisco's labs in San Jose, the EANTC team verified a quite complex scenario that Cisco had preconfigured for our evaluation:

- There were two data centers that had to be managed using a single orchestrator (NFVO) and a single interface (management terminal).
- A network service with multiple components (‘service chain’) had to be provisioned using a graphical user interface, based on a catalog of service functions. Cisco chose a VNF service chain of vASA firewall and CSR router.
- The service chain function had to be coupled with a VPN network service that supports firewall security by chaining the required components. Cisco chose a dynamic client-to-site VPN as a Service.
- A network service with multiple components (‘service chain’) had to be provisioned using a graphical user interface, based on a catalog of service functions. Cisco chose a VNF service chain of vASA firewall and CSR router.
- Finally and most importantly, all functions were to be handled by a “Single pane of glass" and all provisioning activities were to be integrated.

The following figure illustrates the test bed architecture that was based on Cisco's vMS services:

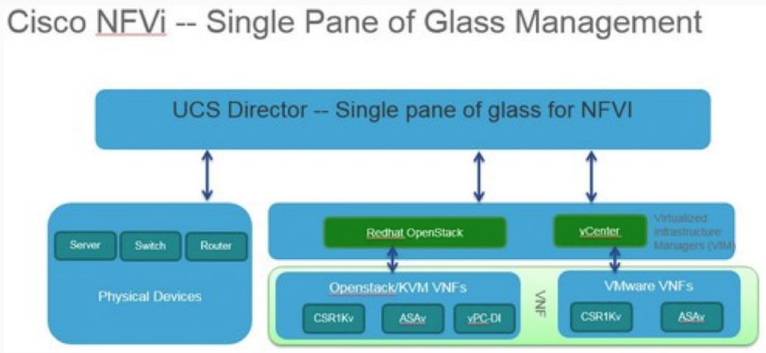

Figure 35: Cisco's Single Pane of Glass management architecture.

The top-level component was Cisco's UCS Director (UCSD), its management system for NFV infrastructure. In addition to the virtual infrastructure, UCSD had to manage a physical test network:

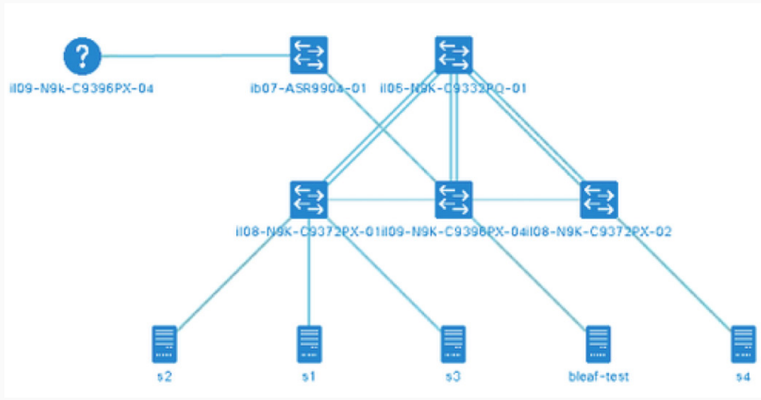

Figure 36: Physical test bed for UCSD evaluation

To prepare for the scenario, Cisco had already preconfigured configuration templates in the UCSD catalogue. We used the first one in the diagram below — the ASA and CSR virtual machine provisioning.

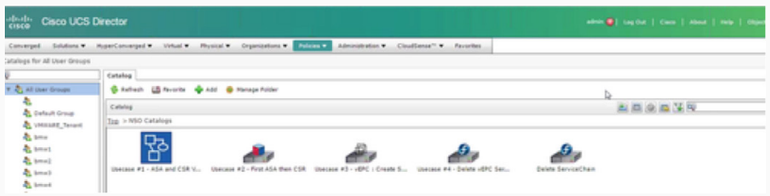

Figure 36: UCS Director catalog.

In this catalogue entry, it was easily possible to choose the virtual infrastructure management (VIM) type as either OpenStack or VMware vCenter. Being able to manage both vastly different solutions from the same graphical user interface looked like a big simplification to us:

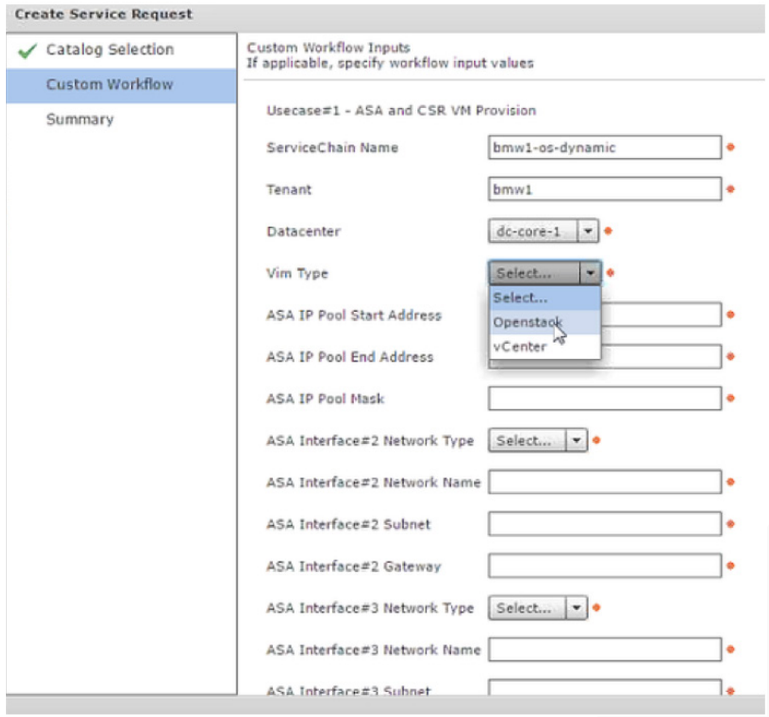

Figure 37: UCSD VIM selection in catalog entry.

We continued with the use case evaluation, configuring services across two data centers – one that supported OpenStack VMM environment, while the other supported vCenter.

Based on the preconfigured catalogue entries, UCSD simply did its job in a very straightforward manner. All resources and components were configured properly as expected.

We validated the successful completion of provisioning using the UCSD interface:

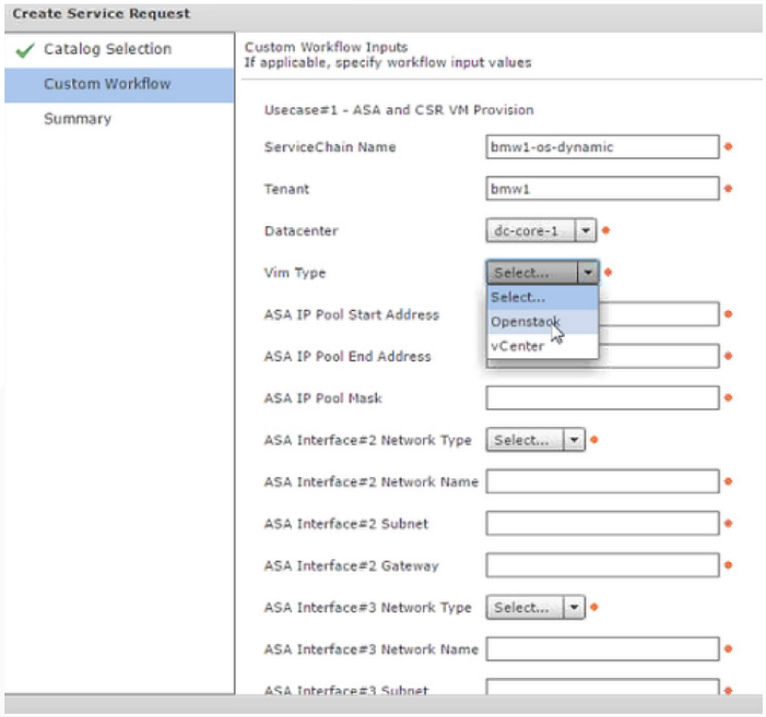

Figure 38: UCSD service completion status.

In addition, we double-checked proper configuration using Cisco's network service orchestrator log files and OpenStack logs.

Summary

Cisco was able to demonstrate that its UCSD can function as a "single pane of glass" virtual services provisioning tool across multiple types of virtual infrastructures.