Title

Network Hardware Virtualization for Application Provisioning in Core Networks

Journal

IEEE Communications Magazine, 55(2)

ISSN

0163-6804

Authors

Gumaste, Ashwin

Das, Tamal

Khandwala, Kandarp

et al.

Publication Date

2017

DOI

10.1109/mcom.2017.1500488cm

Peer reviewed

Network Hardware Virtualization for Application Provisioning in Core Networks

[1]Ashwin Gumaste, [1]Tamal Das [1]Kandarp Khandwala, and [2]Inder Monga
[1]Department of CSE, Indian Institute of Technology, Bombay, India.
[2] Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Abstract: Service providers and vendors are moving towards a network-virtualized (NV) core, whereby multiple applications would be treated on their own merit in programmable hardware. Such a network would have the advantage of being customized for user requirement and allow provisioning of next generation services that are built specifically to meet user needs. In this paper, we articulate the impact of NV on networks that provide customized services and how a provider's business can grow with NV. We outline a decision map that allows mapping of applications with technology that is supported in NV-oriented equipment. Analogies to the world of virtual machines and generic virtualization show that hardware supporting NV will facilitate new customer needs while optimizing the provider network from cost and performance perspectives. A key conclusion of the paper is that growth would yield sizable revenue when providers plan ahead in terms of supporting NV-oriented technology in their networks. To be precise, providers have to incorporate into their growth plans network elements capable of new service deployments while protecting network neutrality. A simulation study validates our NV-induced model.

I. INTRODUCTION

Provider revenues are growing primarily based on provisioning next generation services such as video, cloud, mobile-backhaul and data-center. Applications that dominate provider revenues are becoming aggressive in their network requirements [1]. If service providers do not reinvent themselves to meet application requirements, their revenue will decrease due to over-the-top vendors capturing much of the newfound e-commerce revenue. For example, video distribution *Over-The-Top* (OTT) vendors like Netflix, Amazon, Dropbox or Salesforce are cashing in on raw bandwidth pipes provided by network operators, creating a constant feud between network providers and application providers. In the worst-case scenario, a network provider could impede good quality service to application providers as they do not share revenues, given that the network is merely seen as a basic bandwidth pipe. This feud must be resolved for the larger sake of the ecosystem.

Another aspect of this feud is the drive to protect network neutrality (NN). Shown in [2] are multiple aspects of NN. While not throttling someone's service is a given, a more important aspect is how to create a new service that better facilitates the OTT operator, while protecting NN. It is not a question of how long would it take for service providers (SPs) to support OTT services, but rather a question as to *how* to support such a service. As elaborated in [1], it is about routing money, not packets.

This paper studies the interaction between network providers and application providers through the use of network hardware virtualization. Network Virtualization (NV) manifests itself as an excellent way to resolve this feud by facilitating the partitioning of the network hardware into qualitative domains that are responsible for providing specific service to the application provider. We see NV as an intermediate enabler for NFV, and in direct conjunction with SDN white-boxes.

In this paper, we propose NV as an enabler towards solving the paradox between network operators and OTT application providers. Network operators reason that they have to invest in the network infrastructure, license and maintain the network, while application providers use the network and earn revenue from consumers. Customers of the network provider at times overuse the liberties provided by the network provider. Application providers, on the other hand, treat the network as a bunch of bandwidth pipes that pre-exist and do not see the reason to share their revenue. There are merits in both arguments from the perspectives of network and application providers. The deadlock needs to be resolved for both parties to maximize profit as well as serve the end-user better.

This deadlock can be technically resolved by implementing NV. The idea is: *by using NV in the network, a service provider can now customize services that suite the OTT application*. An OTT application provider has now the incentive to share revenue or buy a specific related service that better drives his application to his end-user (the consumer).

The next obvious question is: *how to implement NV in a network operator?* We begin by understanding application requirements at a broad level and mapping them to possible capabilities of networks to offer customized services. Webb et. al. [3] described ways by which an application can communicate to the network in terms of customization required for a particular application. However, rather than real-time application-level changes, most OTTs have specific and well-known requirements from the network [4]. *So can we model a network based on such requirements, mapping these requirements to NV partitions?*

To do so, we first understand if it indeed is feasible to model OTT requirements over a SP network, by isolating key services that would have: (a) strong business case for implementing NV, and, (b) have key requirements that a provider can fulfill. To this end, Section II presents a table that manifests OTT requirements from the network including network technology choices [5]. For the sake of brevity, we focus only on the metro and core network, assuming that network pipes are essentially static entities with little scope for technology enhancement due to voluminous users, though with NFV even the access gear could be virtualized. Section III presents a method for the Application Service Provider (ASP) to interact with the SP and shows how this can be implemented in four different technology classes, each using a software defined control plane. Section IV shows how SDN can be made to function in such a scenario and the relationship between SDN and NV pertaining

to the technology solutions. Section V captures results from a simulation model that validates our hypothesis.

II. Disparate Networking Requirements

Table 1: Service-Technology Matrix

Domain/OTT Service	Requirement from Network	Technology
Video Services	Guaranteed Bandwidth Low Jitter	IP/MPLS/ WDM/CE
Mobile VAS	Unconstrained Bandwidth Low Packet Drop	MPLS/ OTN/CE
Video Advertising & Merchandise Delivery	Bandwidth on Demand Low Jitter	MPLS/CE
Real-time Events & Entertainment Delivery	Extreme Multicast Bandwidth on Demand	MPLS/CE/ WDM
Healthcare and Telemedicine	Low Downtime, High Bandwidth, Security, Low Latency	MPLS/CE
Defence Networks	Minimal Downtime, Low Latency, Security, Virtualization, Multicast, Bandwidth	MPLS/OTN/ CE/WDM
Finance and Banking	Virtualization, Minimal Latency, Security	IP/MPLS/ CE/OTN
Educational Networks	Multicast, High Bandwidth	WDM
IT Virtualization	Fast switching, resiliency	MPLS
Gaming Services	Extreme interaction, multicast, low latency	MPLS/CE/IP

In this section, we discuss application-level requirements of various domains and how these can be mapped to network equipment through NV. Shown in Table 1 is a list of key revenue-generating OTT services. For each service, Table 1 lists network-centric specifics desired by an ASP and plausible technology options to provision the service. Table 1 considers ASP businesses currently valued at 1 billion USD+ [6]. The key driver towards ASP traffic is video. Since we ignore the access network, it is safe to say that the traffic is largely B2B in nature, but can without loss of generality be extended to a B2C model. For many of the applications, there are multiple technology solutions possible and the ones that are commercially viable in a tier-1 provider network have been illustrated in column 3.

The key question that Table 1 highlights is: *how a SP can provision a particular service requirement in the network?* To this end, a system must be designed that orchestrates interaction between the provider and the OTT ASP, adhering to tenets of NN. This interaction must be mapped onto network hardware so that service provisioning is possible. Our proposal is to create an SDN controller that would facilitate interaction between incoming traffic requests from ASPs mapping these onto provider hardware that adheres to NV principles. *The key challenge in this approach is to: (a) map the incoming demand into network-specific parameters that can be used for traffic engineering, bandwidth brokering, provisioning and service support, and, (b) enable the network hardware to be able to provision new services with specific OTT needs.* The challenge in the latter is to be able to create services and differentiate them at the hardware layer.

The next section describes a solution using NV principles to partition SP-hardware to meet ASP service goals. The advantage of NV is that it enables an SDN controller to realize the full potential of an SDN-centric network.

III. Building a Solution with Virtual Network Equipment Partitio (VNEP)

In this section, we describe a method to implement NV to meet specific ASP requirements. We assume that a request for a service arrives into a SP domain and a *network management system* (NMS) communicates to an SDN controller that would provision services. The NMS can abstract specific requests into network-centric parameters with the goal of provisioning services. The NMS maps a service request onto an abstracted network topology by considering specific service parameters. These parameters are then mapped onto all the network elements (NEs) in the path to check service provisioning feasibility. To check feasibility, there must be a parameterized relationship between incoming service requests and the equipment deployed. The SDN controller maps an incoming request to a network-virtualized hardware. The idea is that every piece of hardware is further divided into service supporting modules that are parameter-driven and have a direct relation with an SDN controller populated flow table. Virtualization happens by the creation of multiple (virtualized) instances of the data-plane at each NE. Each such instance of the data-plane enables OTT-service-specific feature implementation.

III.A. Method to implement NV in SP-ASP (OTT) interaction

We now describe how to implement NV in a provider network. A request that enters the network is provisioned through a network interface supported by the NMS. For each new incoming request, the NMS computes the optimal network resources to be allocated. To this end, the following steps are envisaged at the centralized NMS:

- A route is computed based on service requirements. Actual bandwidth allocation is computed along the route depending on the specified request and other requests at that instance.
- Each element along the computed route is examined from a service support perspective, whether it can satisfy *specific* requirements of the service.
- To compute the specific requirements of the service request, we propose the concept of *VNEP* or *Virtualized Network Equipment Partitions* that enables a network equipment (such as a switch or router) to be partitioned to satisfy specific service parameters. An example on VNEP is provided in Section III.B.
- If VNEPs are possible along the path to provision the request, then all the network equipment are provisioned to meet the new request by the NMS through the SDN controller. Otherwise, an alternate path that maximally conforms to the VNE requirement (partially, if not fully) is provisioned.
- A VNEP created at a node may be moved to another node depending on resource availability over a period of time.

VNEP computation is now described in detail.

III. B. VNEP Computation

A VNEP is represented by the virtual partitioning of hardware such that each of the partitioned elements corresponds to fully functional entities, capable of performing all the

functions as the larger hardware, but specific for a service request. The key to VNEP creation is to note that the overlaid software creates partitions by allocating hardware resources within a larger NE. Partitions could be created in switching elements, network processors, buffers and packet classifiers. Partitions correspond to hardware resources as defined by the software and are made available strictly for a particular service or function.

Our conjecture (based on an analysis of existing network gear) is that a networking element can be divided into partitions, such that a partition can act as a completely independent networking element. We argue that the sum of parts – i.e. the union of all *partitions* – does not necessarily add up to *the original element* for that *particular parameter*. Throughput, average latency, packet-loss rate are examples of a parameter.

Let us consider an example: Assume a 60Gbps switch-fabric with VOQ (virtual-output-queued) buffers, with 6-input lines and 6-output lines – all at 10Gbps. Assume one of the lines is sending data at 2Gbps, the average packet-size is 250bytes and the VOQ memory to store packets for contention resolution is 3Mb. The maximum ingress-to-egress latency is observed as 300µs. However, we aim to estimate average latency, which is a function of the provisioned services at the other 5-ingress ports, the nature of the traffic, and type of switch-fabric (cut-through, store-and-forward, shared memory, etc.).

Since the latency of a flow through a switch also depends on other flows, one way to control it is to bound the number of flows through the switch. A simple 4×4 cross-bar with VOQ (essentially a 12×4) switch (each port at 1Gbps) can take 4-flows each with 250byte average packet-size at full line-rate (wirespeed operation), resulting in $1.2\mu s$ switching, while the same switch will result in 2.4µs port-to-port latency, if the average packet-size is 128bytes [7]. Similarly, the switch will result in a latency of 3µs, if the packet-size is 64bytes [7]. The switch behavior becomes more erratic, when the standard-deviation between flows across multiple ports increases [8]. For example, the switch results in a port-to-port latency of 12µs for multicast traffic if the packet-size is 64bytes, and remaining ports have provisioned flows with packet-size of 1500bytes. The above discussion highlights the complex relationship between packet-sizes, port-counts, traffic distribution (random/unicast/multicast), etc. implying that for carrier-class services, i.e. with desired deterministic parameters of delay and jitter, predicting switch behavior is important-but-difficult. Even intricate queuing models (i.e. those deploying G/G/1 queues) tend not to converge in real-time.

So our approach is to provision services without getting involved in the intricacies of computing switch-specific parameters in real-time. *Our approach is technology-specific, given the enormous amounts of technology deployments*.

In our approach, we partition a switch/router/optical-cross-connect into VNEPs that can individually provision services. The idea is to dynamically create a VNEP that will adhere to all the system-wide parameters for a particular service, with the constraint that the sum of all the VNEPs in a NE is less than the total capacity of the switch. The union of VNEPs is not linear. This implies that the system leads to over-provisioning, which though undesired, is necessary to maintain many of the carrier-class attributes desired for OTT services.

VNEP creation and sizing involves the following steps:

1. A NE is viewed as the number of instances q_i of a particular parameter i such that $f(qi)$ denotes the performance criteria (such as bounded latency) for parameter i.
2. The value $f(qi)$ also takes into consideration another parameter j whose performance criteria is $f(qj)$ is the number of instances of supporting j and which impacts $f(qi)$.
3. Note that it is mathematically non-trivial to compute $f(qi)$ and hence worst-case provisioning metrics are used as acceptable practices.

The second point is supported by an example. Let i denote the service parameter for port-to-port latency. Assume a 60Gbps switch-fabric supports 6×10Gbps connections with 250bytes average packet-size and $f(qi) = 3$µs. The same fabric will have a $f(qi) = 12\mu s$ latency for the same number of flows if the average packet-size reduces to 64bytes. The delay increases sizably ($f(qi) = 50$) if the number of flows increases to 60×1Gbps flows. So, now if we have to provision a service of 1Gbps with a latency within 3µs, and another service of 5Gbps with a latency also within 3µs, then how do we do so given that the packet-size of the first service is say 128bytes and the second one is say 64bytes? Obviously, the second service will require more over-provisioning as compared to the first one, i.e. to say that though the second service is 5× of the first service, in order to achieve similar parameters, the second service may have to be provisioned through the switch with 12× resources (buffers primarily) so that the switch can meet provisioning requirements. Now how do we arrive at the number 12×? This number is a function of both volume and quality: volume, as in how much more would the service take in every parameter's domain, and quality, as in what would be the impact of the service provisioning in other parameter domains.

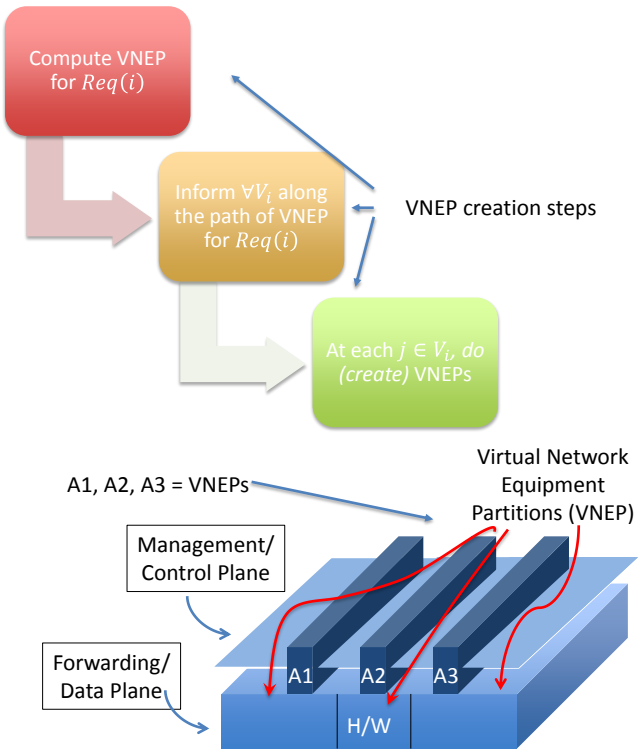

Figure 1: VNEP computation (top, a) and VM migration analogy (bottom, b).

Shown in Figure 1a is the actual process for creating and allocation VNEPs. From an incoming request (*Req(i)*), we compute the corresponding partition's impact on other partitions. The SDN controller computes VNEPs for each service at each NE. The controller then sends specific information to each node to partition itself according to its VNEP computation, based on the four use-cases discussed at the end of this section.

Given that there are a large number of protocols deployed leading to a variety of equipment such as IP/MPLS routers, OTN cross-connects, Carrier Ethernet (CE) switches and WDM gear, a key question is how to implement partitioning. It is publicly known that many vendors are in the process of SDNizing their current gear. The question we want to answer: *how can equipment vendors achieve network virtualization at the data plane*?

To this end, we have identified network equipment from 10 vendors who are known to be committed to SDNizing their product portfolio. These 10 vendors combined, have products across the aforementioned technologies such as IP/MPLS etc. 7 of these vendors have products in the layer-2/3 (L2/L3) space, while 3 products are from the optical space.

On studying the equipment of these 7 chipsets as well as corresponding patents, it appears the architecture follows one or a combination of the following three strategies: (a) A FPGA-based switching core or an FPGA as a processing element; (b) an ASIC or merchant silicon-based switching core with an FPGA or a processor guiding the ASIC; and (c) a network processor (NP)-based switching core.

In Table 2, we captured the key chipsets that are used for creation of the products for the various equipment vendors. The table also shows how the datapath can be partitioned.

Table 2: VNEPs in pragmatic network elements.

	FPGA 1	FPGA 2	NP1	NP2	ASIC 1	ASIC 2	ASIC 3
Slice-able or not	yes (425K logic blocks)	yes (693K logic blocks)	Yes	Yes	Yes	Yes	Yes
Min. SW granuality IO (Mbps)	1	≤ 1	1	0.064	0.128	1	0.064
SW granularity every component (Gbps)	240	800	120	640	40	1280	50
How many parallel lines (10Gbps)	24 (16 standard 8FX)	80 (GTX)	12×10G OR 3×40G	64×10G and 16×40G	4x10G and 24×1G and 12×2.5G	128×10G	2×25G
Switch capacity	360 Mpps	1600 Gbps	≤ 2 GHz	1.25 GHz	60 Mpps	1440 Mpps	30 Mpps
Avg. Latency	400ns	500ns	NA	NA	NA	150-650ns	120 ~750ns
Protocol	L2/3/4	L2/3/4	L2/L3	L2/L3/L4	L2/L3	L2/L3	L2
Memory Capacity	50 Mb	52 Mb	4MB	7.5 MB	8.5 MB	12Mb	-
Number of switching blocks	Variable	Variable	120000	240000	40000	1280000	50000

Shown in Table 2 are 7 implementations of a switching plane used for L2/L3 equipment. Additionally we have also considered 3 ROADM implementations using liquid crystal on silicon (LCOS)-based wavelength selective switches (WSS) of $1 \times M$ and $M \times N$ configurations and another WSS based on digital lightwave processing technology. The initial 7 L2/L3 cases are shown in the table. Two types of FPGAs (FPGA1-2), two types of network processors (NP1-2) and three types of ASICs with FPGAs and NPs (ASIC1-3) are compared.

The key takeaway from Table 2 is to show that irrespective of the technology deployed, it is indeed possible to create VNEPs. To this end, Table 2 showcases the sliceability parameter – at what granularities can we slice a fabric. The impact of slicing is on the throughput (speed-of-the-device) and latency. The memory capacity also has a direct impact on the throughput – more the slicing, more the memory required and hence latency suffers. Larger number of flows require either more interconnected fabrics (multi-card designs) or use of large ASICs (column #7, 8). The latency gets impacted with sliceability as well as protocol (QoS, more processing etc.)

VNEP Partitioning analogous to Virtual Machine (VM) Creation and Migration: VNEP creation and NV using VNEPs is analogous to VM creation and migration in hardware. Shown in Figure 1b is the analogy of the forwarding plane in a NE with a VM hypervisor. VMs can be dynamically created in a processing environment. The same analogy is used for VNEP creation, whereby VNEPs are like VMs – created on-the-fly and use the switch fabric resources independently. As in Figure 1b, VNEPs are created by the control plane (SDN-based), and implemented within the NE through NV.

From the perspective of Table 2, we can create VNEPs as slices in different implementations of L2/L3 equipment or as independent optical switches virtually superimposed on a ROADM as shown next.

III. C. Use-cases

Use-case 1: *IP/MPLS-over-WDM* – For IP/MPLS overlay and WDM ROADM underlay, IP/MPLS label switched routers (LSRs) are partitioned based on supported flows and WDM ROADMs are partitioned to support non-blocking connections. VNEPs in the ROADM require support of colorless, directionless and contentionless (CDC) and gridless properties. A VNEP in an LSR is an MPLS tunnel.

Use-case 2: *MPLS-over-OTN with WDM* – In the case of MPLS-over-OTN with WDM underlay, VNEP partitions take into consideration OTN pipes at MPLS-LSR interfaces that further feed into a WDM network. We assume services are sub-wavelength granular implying wavelength assignment as a multi-service aggregation and provisioning problem. Partitioning happens at the LSR forwarding plane and OTN-based ODU (optical data unit) switch-fabric.

Use-case 3: CE+OTN-over-WDM – In this case, we partition the CE switch-fabric into discrete switching chunks so that an Ethernet Switched Path (ESP) is mapped onto an OTN ODU port. The VNEPs are portions of the CE switch-fabric implemented.

Use-case 4: IP-over-CE+OTN-over-WDM – In this case, IP routers are at select locations as an overlay with a CE underlay, all over a ROADM-based WDM network. Whenever a service has granularity that is near to a wavelength's full capacity (10/100Gbps), it is routed all-optically by the ROADM. Whenever a service can be routed at layer-2 through the use of an ESP, it is done so using the CE network used for aggregation and switching. However, when layer-2/1 provisioning is not possible, then the service is handled exclusively through the IP-layer. VNEP information created by the centralized controller is used to partition switching resources at any or all of the CE/IP layers that use FPGAs/ASICs/NPs.

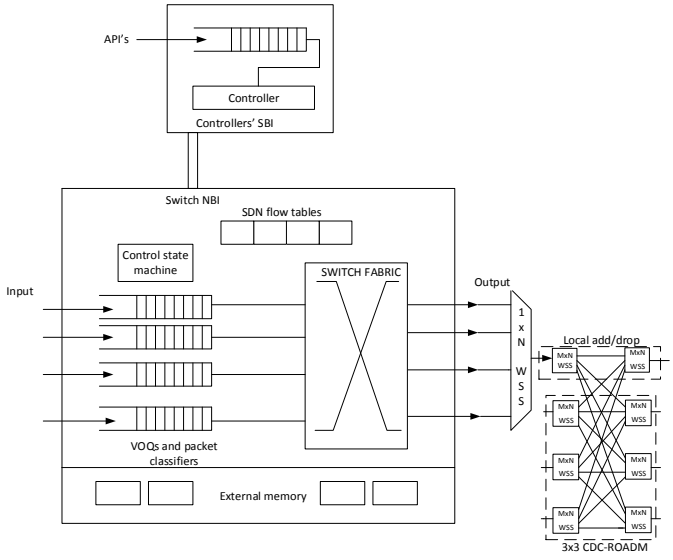

Figure 2: Switch Architecture to Implement SDN.

IV. Interaction between SDN and NV

Figure 2 shows a switch architecture to implement SDN with NV with a controller connected to the switch's northbound interface. The switch could support L2/L3 protocols and the interfaces would be mapped onto wavelengths. Incoming flows are segregated at the input buffers (which are further segregated to support VOQs). Flow headers are worked upon by a control-state-machine (CSM) that also populates SDN tables. Entire protocol functioning happens at the controller. To support scalability, we assume that the controller runs on a VM.

The architecture in Figure 2 can have multiple manifestations including use of FPGAs/ASICs/NPs. In one embodiment, we assume an IP/MPLS LSR in which, the CSM, SDN flow-tables (SDNFT) are implemented in an FPGA, while other modules are implemented in an ASIC. In another CE design, the SDNFT, CSM, VOQs are implemented in a NP, while the switch fabric and memory are implemented in an ASIC. Yet another design includes a smaller CE device that has the entire design except the SDNFT in an FPGA, with the flow-tables in a TCAM ASIC.

We propose following three policies for VNEP partitioning:

Policy 1: *Throughput maximization*. In this policy, VNEP computation maximizes the throughput at every NE. This is a non-carrier-class policy implying that the port-to-port latency per NE is non-deterministic. This implies an additive increase of throughput, and hence, whenever a new request arrives at the SDN control plane, a VNEP is created with a view to maximize network-wide throughput. The CSM partitions the hardware as per the specifications of Table 2.

Policy 2: *Latency-bounded partitioning*. In this policy, a VNEP is created such that the corresponding service is guaranteed to meet end-to-end latency requirement through every NE by bounding latency. This policy requires double optimization: route selection and associated appropriate amount of partitioning at a node.

Policy 3: *Latency sensitive-service maximization* (*LSSM*): In this globally active policy, the approach is to maximize the number of services through a NE. The controller creates VNEPs such that they balance each other in terms of parameterized requirements. For example, services with similar delay and bandwidth requirements are load-balanced. The controller also provides for equal cost multiple paths (ECMP) to load balance the service.

In our simulation model, we rationalize service requirements based on their utility to the network (revenue for the provider), and normalize the utility over delay-constraints. We then provision services such that the delay-constraints are met, while bundling as many services together. The LSSM policy is a greedy heuristic and its complexity is of the fourth-order polynomial in terms of number of links in the network and hence its functioning depends on graph size.

V. Simulation Model and Hypothesis Verification

A simulation model was built to test our VNEP hypothesis as a method to facilitate interaction between SPs and ASPs. We model a provider network with two autonomous systems (AS), five metropolitan regions, with each region divided randomly into 20, 40, 60, 80 and 100 access regions. The backbone and metro networks use fiber, while the access networks could be wireless/fiber/coaxial cable-based. Our goal is to evaluate the impact of NV over different technologies by provisioning OTT services. To this end, the simulation model implements each technology solution using proposed VNEP creation policies.

Each access region has between 10,000 and 100,000 subscribers and is connected to a metro network with multiple metros backhauled to a core network (wholly viewed as a single AS). The point of presence (POP) connecting the access to the metro supports ROADMs. The overlay depends on the technology being simulated and we study IP/MPLS, MPLS, OTN and CE technologies and the 7 cases of Table 2 are deployed randomly. The control plane is implemented as an SDN overlay that consists of controllers, one for an AS of 10K users and hierarchically arranged thereafter.

The simulation model works as follows: randomly generated service requests have specific QoS parameters. Services are organized into two levels – services and sessions. Services are exponentially distributed with a mean holding time of 6-months, while session holding time is exponentially distributed with a mean time equivalent to a 100MB video-file download session. The service is guided to the appropriate controller, which uses one of the three VNEP creation policies and evaluates whether provisioning is possible. Services are lumped through pre-assigned aggregation policies. Once a service is provisioned, we compute service and switch statistics.

Load is computed as average occupancy of all the services to the maximum allowable input-rate across all the ingress ports. MPLS LSRs have 1Gbps and 10Gbps interfaces and a net switching capacity of 640Gbps [9]; CE switches have 1Gbps and 10Gbps interfaces and 80Gbps fabric that is stacked to create a 640Gbps node [7]. ODU switching, is assumed at ODU0/1/2e [10]. Transport wavelengths can be generated by MPLS/CE/IP forwarding plane and support 10Gbps, 40Gbps and 100Gbps. Cost is computed as in [11] for both CAPEX and OPEX, while we assume that for provisioning OTT services, the OTT ASP shares 20% of its revenue with the SP.

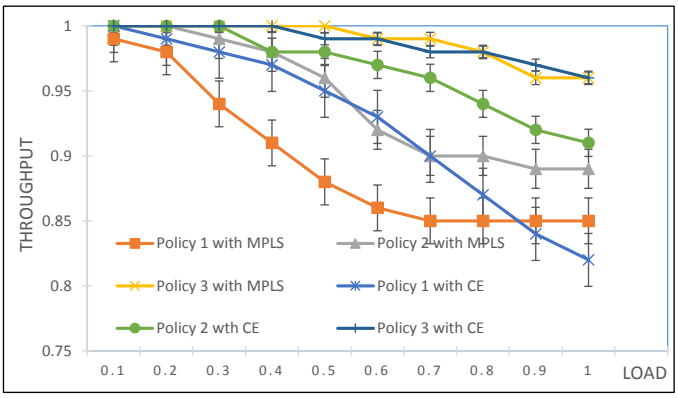

Figure 3: Throughput as a function of load for different policies.

Figure 3 compares all the three policies used for VNEP computation using MPLS and CE technologies using FPGAs, FPGAs+ASIC and NP+ASIC approaches. We show throughput versus load with error bars indicating stability of results. MPLS and CE were chosen as likely candidates for cost considerations. A peculiar behavior is that policy 3 has the best throughput, while being able to take service latency into consideration.

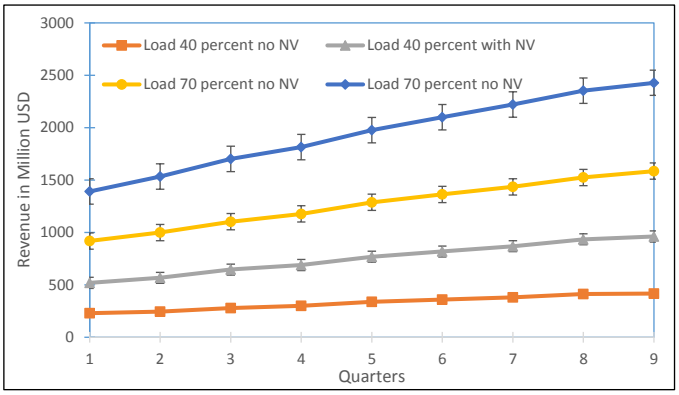

Figure 4: SP revenue with and without ASP revenue-sharing through NV.

Figure 4 highlights the effect of ASP revenue-sharing through NV on the SP revenue. It shows that there is sizable incentive for ASPs to share their revenue as the providers would be able to grow the network, thereby facilitating larger and qualitatively superior reach for the ASPs. Figure 4 is generated as follows: We first measure ASP revenue without NV, and no revenue sharing. NV is implemented using the most popular approach – FPGA+ASIC and uses the LSSM policy. We compute the revenue by pegging each service at 30-40% higher price than before. For example, a 12Mbps HD-video pipe was priced 20 dollars per month with no revenue-sharing and hence no NV-support. The same pipe with guaranteed bandwidth (no packet loss), is priced at 26 dollars, while it is priced at 30 dollars with bounded latency and 50ms restoration of service in case of fiber cut/equipment failure.

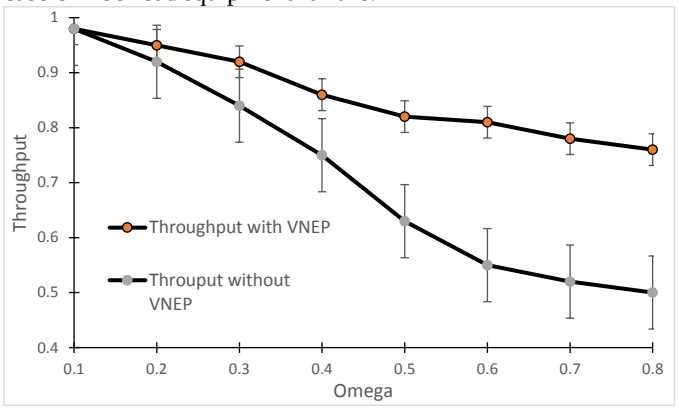

Figure 5: Throughput versus ratio of MP2MP/P2P traffic.

Figure 5 studies the impact of VNEP on throughput in the network as a function of the ratio (defined as Omega) of: *multipoint-to-multipoint* (MP2MP) traffic to point-to-point traffic. The graph is generated for the case of MPLS. As Omega increases, the throughput without VNEP decreases rapidly, while that with VNEP decreases gradually. This is a critical result showing the maximum benefit of the use of VNEP, which is modeled as implemented in FPGA+ASICs. The graph shows how VNEPs can impact new service support such as multicast services that are poorly handled at higher loads.

VI. Conclusion

We presented an approach to integrate OTT application providers with service providers using network virtualization in

hardware. Our work is inspired by [9, 12]. We propose the concept of virtual network equipment partitions (VNEPs) that enable a NE to be partitioned as per service requirement, thereby benefiting from programmability of the control plane. Policies to partition a NE are discussed. Results from a simulation study show the benefit for ASPs in a provider network using NV-compliant hardware.

REFERENCES

[1] V. Mishra, "Routing Money, Not Packets" Communications of the ACM, Vol. 58 No. 6, Pages 24-27

[2] FCC Report: Protecting and Promoting the Open Internet, FCC 15-24, GN Docket No 14-28.

[3] K. C. Webb, A. C. Snoeren, and K. Yocum. Topology switching for data center networks. In Proc. Hot-ICE, 2011

[4] J. Mogul and L. Popa, "What We Talk About When We Talk About Cloud Network Performance," ACM proc. of Sigcomm 2013, Chicago.

[5] A. Gumaste and S. Akhtar, "Evolution of Packet-Optical Integration in Backbone and Metropolitan High-Speed Networks: A Standards Perspective" in IEEE Communications Magazine 2013 Vol. 51, No 11 pages 105-111. Nov. 2013.

[6] Infonetics Research, "Data Center and Enterprise SDN Hardware and Software Report", 2013.

[7] S. Bidkar, S. Mehta, R. Rathore and A. Gumaste, "On the Design, Implementation, Analysis, and Prototyping of a 1- μs, Energy-Efficient, Carrier-Class Optical-Ethernet Switch Router" IEEE/OSA Journ. of Lightwave Tech. Vol. 32 No 17. pp 3043 - 3060.

[8] S. Das, G. Parulkar, N. McKeown, "Rethinking IP Core Networks," IEEE Journ. of Optical Communications and Networking, December 2013.

[9] P. Bossharty, "Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN," ACM Proc. of Sigcomm 2013, Hong Kong, China.

[10] Per Harald Knudsen-Baas, "OTN switching (Masters Thesis)," Department of Telematics, Norwegian University of Science and Technology, June 2011.

[11] A. Mathew, T. Das, P. Gokhale, A. Gumaste, "Multi-layer high-speed network design in mobile backhaul using robust optimization" IEEE/OSA Journ. of Opt. Comm. and Networking, Vol. 7. No. 4. pp 352-367. April 2015.

[12] E. Haleplidis, et al., "Network Programmability with ForCES," IEEE Commun. Surveys. & Tutorials, Vol. 17, No. 3, 2015