Virtual Switching in an Era of Advanced Edges

Justin Pettit jpettit@nicira.com
Jesse Gross jesse@nicira.com
Ben Pfaff blp@nicira.com
Martin Casado casado@nicira.com
Nicira Networks, Palo Alto, CA, 650 473-9777

Simon Crosby
simon.crosby@citrix.com
Citrix Systems, Ft. Lauderdale, FL, 954 267-3000

***Abstract*—CPU virtualization has evolved rapidly over the last decade, but virtual networks have lagged behind. The greater context and ability to apply policies early make edge switching attractive. However, the perception of a lack of features and high CPU utilization can cause it to be prematurely dismissed. Advanced edge switches, such as Open vSwitch, answer many of the shortcomings in simple hypervisor bridges. We revisit the role of edge switching in light of these new options that have capabilities formerly only available in high-end hardware switches. We find that edge switching provides an attractive solution in many environments. As features are added and the ability to offload more to hardware improves, these advanced edge switches will become even more compelling.**

I. Introduction

Compute virtualization is having a profound effect on networking. It challenges the traditional architecture by creating a new access layer within the end-host, and by enabling end-host migration, joint tenancy of shared physical infrastructure, and non-disruptive failover between failure domains (such as different subnets). It also promises to enable new paradigms within networking by providing richer edge semantics, such as host movement events and address assignments.

Traditional networking presents many adoption hurdles to compute virtualization. Classic IP does not support mobility in a scalable manner. Flat networks such as Ethernet do not scale without segmenting into multiple logical subnets, generally through VLANs. Network configuration state such as isolation, QoS, and security policies does not dynamically adapt to network topology changes. Thus, not only does virtual machine (VM) migration break network configuration, physical network policies still demand per-box configuration within cloud environments – a far cry from the *resource pool* model which is so common in scale-out cloud deployments. New protocols such as Trill [1] address some of these issues, but still fail to meet all of the challenges posed by virtualization, such as managing end-to-end configuration state during mobility.

Over the last few years, both academia and industry have proposed approaches to better adapt networks to virtual environments. In general, these proposals add additional context to packets, so that the network can begin enforcing forwarding and policy over the logical realm.

Network architecture must change in two ways to accomplish this. First, network elements must be able to identify the logical context. This can be accomplished by adding or overloading a tag in a packet header to identify a virtual machine or a virtual network. Second, the network must adapt to changes in the virtualization layer, either reacting to changes inferred from network activity or to signals from the virtualization layer.

The existing approaches to making these changes can be characterized by the network entity that does the first lookup over the logical context. In one approach, the end-host implements the virtual network control logic and the hypervisor forwards packets [2], [3], [4]. In the other approach, packets immediately pass to the first-hop switch, bypassing the end-host's networking logic [5], [6]. Both approaches can use advanced NIC (Network Interface Card) features for acceleration.

The rationale for performing packet lookup and policy enforcement at the first hop switch is simple: switches already contain specialized hardware to enforce complex network functions at wire speeds. Thus, much of the recent public discourse and standardization efforts have focused on this approach.

However, the increase in core count, processor speed, and the availability of NICs with on-board network logic, coupled with the flexibility of software makes switching at the end-host an attractive alternative. Yet, little has been written about the relative merits of this approach.

In this paper, we draw on our experiences implementing and deploying Open vSwitch [4] to present the case for edge-based switching. It is not our intention to argue that this is preferable to other approaches; there are good arguments on either side. Rather, we focus on the implications of virtual switch design, and discuss how developments in NIC technology will continue to address performance and cost concerns. We believe that the combination of software switching with hardware

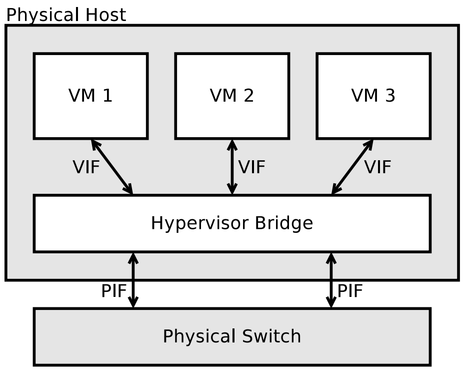

Fig. 1. Structure of a virtual network.

offload (either in the NIC or first-hop switch), strikes a practical balance between flexibility and performance.

We construct the discussion as follows. The next section provides some background on the emergence of the virtual networking layer. Section III discusses switch design within the hypervisor, using Open vSwitch as a concrete example. We then discuss cost and performance issues in Section IV. Section V looks at alternatives to edge switching, followed by speculation about future trends in Section VI. We conclude in Section VII.

II. Evolution of the Edge

On any particular virtualized server, the virtual network resources of VMs must share a limited number of physical network interfaces. In commodity hypervisors, an individual VM has one or more virtual network interface cards (VIFs). The network traffic on any given VIF is bridged by the hypervisor, using software or hardware or both, to a physical network interface (PIF). Figure 1 shows how physical and virtual network resources are related.

The 80*x*86 hypervisors introduced in the late 1990s implemented simple network bridges in software [7]. These bridges had no network manageability features and no ability to apply policies to traffic. For this purpose, the most troublesome traffic was that between VMs on the same physical host. This traffic passed directly from VM to VM, without ever traveling over a physical wire, so network administrators had no opportunity to monitor or control it.

This drawback was only a minor issue at the time, because the low degree of server consolidation meant that the complexity of virtual networks was limited. Since then, however, the number of VMs per server has increased greatly, and now 40 or more VMs on a host is not uncommon. At this scale, network and server administrators must have tools to manage and view virtual networks.

This paper discusses the *advanced edge switch* approach to solving this virtual network management problem. This approach takes advantage of the unique insight available to a hypervisor bridge, since as a hypervisor component it can directly associate network packets with VMs and their configuration. These switches add features for visibility and control formerly found only in high-end enterprise switches. They increase visibility into inter-VM traffic through standard methods such as NetFlow and mirroring. Advanced edge switches also implement traffic policies to enforce security and quality-of-service requirements.

To make management of numerous switches practical, advanced edge switches support centralized policy configuration. This allows administrators to manage a collection of bridges on separate hypervisors as if it were a single distributed virtual switch. Policies and configuration are centrally applied to virtual interfaces and migrate with their VMs.

Advanced edge switches can use NIC features that improve performance and latency, such as TCP segmentation and checksum offloading now available even in commodity cards. More advanced technologies such as support for offloading GRE and IPsec tunnels are becoming available, while SR-IOV NICs allow VMs to directly send and receive packets.

Advanced edge switches are making inroads in the latest generation of hypervisors, as the VMware vSwitch, Cisco Nexus 1000V, and Open vSwitch. The following section highlights Open vSwitch, an open source virtual switch containing many of the advanced features that we feel define this new generation of edge switches.

III. Open vSwitch

Overview

Open vSwitch is a multilayer virtual switch designed with flexibility and portability in mind. It supports the features required of an advanced edge switch: visibility into the flow of traffic through NetFlow, sFlow, and mirroring (SPAN and RSPAN); fine-grained ACLs (Access Control Lists) and QoS (Quality-of-Service) policies over a 12-tuple of L2 through L4 headers; and support for centralized control. It also provides port bonding, GRE and IPsec tunneling, and per-VM traffic policing.

The Open vSwitch architecture has been previously described [8]. Since then, a configuration database has been introduced, which allows for the use of multiple simultaneous front-ends. The primary interfaces use JSON-RPC to communicate over local sockets on the system or to remote systems through a SSL tunnel.[1] We anticipate adding other front-ends, including an IOS-like CLI, SNMP, and NETCONF.

[1]This remote interface will likely be standardized as the OpenFlow switch management protocol in the near future. Combined with the already defined datapath control protocol, the pair would comprise the OpenFlow protocol suite.

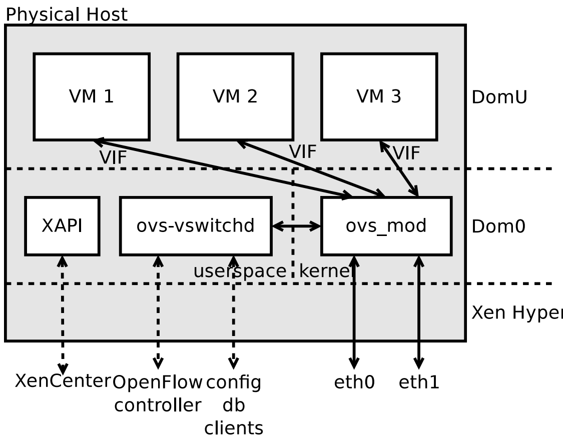

Fig. 2. Open vSwitch integration with XenServer. Each unprivileged virtual machine (DomU) has one or more virtual network interfaces (VIFs). VIFs communicate with the Open vSwitch "fast path" kernel module `ovs_mod` running in the control VM (Dom0). The kernel module depends on the userspace `ovs-vswitchd` "slow path," which also implements the OpenFlow and configuration database protocols. XAPI, the XenServer management stack, also runs in Dom0 userspace.

Open vSwitch is backwards-compatible with the Linux bridge, which many Linux-based hypervisors use for their edge switch. This allows it to be a drop-in replacement in many virtual environments. It is currently being used in deployments based on Xen, XenServer, KVM, and VirtualBox. The Xen Cloud Platform and upcoming versions of XenServer will ship with Open vSwitch as the default switch.

Open vSwitch is also being ported to non-Linux hypervisors and hardware switches, due to its commercial-friendly license, modular design, and increasing feature set.

Integration with XenServer

Open vSwitch works seamlessly with XenServer, as shown in Figure 2. As mentioned earlier, future versions of XenServer will ship with Open vSwitch as the default. It is also fully compatible with XenServer 5.6, the currently shipping version.

XenServer is built around a programmatic interface known as XAPI [9] (XenServer API). XAPI is responsible for managing all aspects of a XenServer, including VMs, storage, pools, and networking. XAPI provides an external interface for configuring the system as well as a plug-in architecture that allows applications to be built around XenServer events.

XAPI notifies Open vSwitch of events that may be of interest. The most important of these are related to network configuration. Internal and external networks may be created at any time. External networks support additional configuration such as port bonding and VLANs. These networks are essentially learning domains for which virtual interfaces may be attached.

XAPI notifies Open vSwitch when bridges should be created and interfaces should be attached to them. When it receives such a notification, Open vSwitch queries XAPI to determine greater context about what is being requested. For example, when a virtual interface is attached to a bridge, it determines the VM associated with it. Open vSwitch stores this information in its configuration database, which notifies any remote listeners, such as a central controller.

Centralized Control

A common component of advanced edge switches is the ability to be centrally managed. Open vSwitch provides this through support of the OpenFlow [10] protocol suite. OpenFlow is an open standard that permits switch configuration and dataplane manipulation to be managed in a centralized manner.

At Synergy 2010 [11], Citrix demonstrated an application that treats all the Open vSwitches within a XenServer pool as a single distributed virtual switch. It supports the ability to implement policies over the visibility, ACL, and traffic policing support outlined earlier.

These policies may be defined over pools, hypervisors, VMs, and virtual interfaces in a cascading manner. Appropriate policies are generated and attached to virtual interfaces. As VMs migrate around the network, the policies assigned to their virtual interfaces follow them. Due to the tight integration with XenServer, Open vSwitch is able to work in tandem with the control application to enforce these policies.

IV. Implications of End-Host Switching

In this section, we look at the implications of end-host switching. We focus on issues of cost, performance, visibility, and control associated with such an approach.

Cost

Advanced edge switching is primarily a software feature. Deploying it requires installing software in the hypervisor layer (assuming it is not installed by default), and upgrades require only a software update.

Some deployments of advanced edge switches may benefit from purchasing and deploying advanced NICs to increase performance. We expect these NIC features to naturally migrate into lower-end hardware over time, as has happened with hardware checksumming and other features once considered advanced. As edge switches continue to add new features, some of these may not be initially supported in hardware, but the switch can fall back to software-only forwarding until a newer NIC is available.

Otherwise, deployment of advanced edge switching has minimal hardware cost. Implementation of advanced edge switching using a flow-based protocol, such as OpenFlow, requires adding an extra server to the network to act as the controller. Depending on deployment size, a VM may be suitable for use as a controller.

Performance

Edge switches have been demonstrated to be capable of 40 Gbps or higher switching performance [12]. However, raw forwarding rates for edge switches can be deceptive, since CPU cycles spent switching packets are CPU cycles that could be used elsewhere. An edge switch's greatest strength is in VM-to-VM traffic, which never needs to hit the wire. The communications path is defined by the memory interface, with its high bandwidth, low latency, and negligible error rate, rather than network capabilities. Throughput is higher, errors due to corruption, congestion, and link failure are avoided, and computing checksums is unnecessary.

The CPU cost of switching at the edge can be minimized by offloading some or all of the work to the NIC. Already common techniques such as TCP segmentation offloading (TSO) provide some assistance, while newer methods such as SR-IOV allow virtual machines to directly access the hardware, completely avoiding any CPU load from switching. Future increases in performance and newer types of offloading will further diminish the impact of network processing on the end host. Section VI includes a discussion of future directions for NIC offloading.

The edge's visibility into the source of virtual network traffic gives it the ability to affect traffic before it enters the network, protecting oversubscribed uplinks. It is particularly important to be able to throttle traffic for untrusted VMs, such as those in a commercial cloud environment.

Visibility

Traditional hypervisor switches provide network administrators very little insight into the traffic that flows through them. As a result, it is logical to use hardware switches for monitoring, but VM-to-VM traffic that does not pass over a physical wire cannot be tracked this way.

Advanced edge switches provide monitoring tools without a need for high-end physical switches. For example, Open vSwitch supports standard interfaces for monitoring tools, including NetFlow, sFlow, and port mirroring (SPAN and RSPAN). Software switches can inspect a diverse set of packet headers in order to generate detailed statistics for nearly any protocol. Adding support for new types of traffic requires only a software update.

Independent of the capabilities of different pieces of hardware and software, some locations are better suited to collecting information than others. The closer the measurement point is to the component being measured, the richer the information available, as once data has been aggregated or removed it cannot be recovered. An edge switch directly interacts with all virtual machines and can gather any data it needs without intermediate layers or inferences. In contrast, hardware switches must rely on a hypervisor component for certain types of data or go without that information. A common example is the switch ingress port: packets must be explicitly tagged with the port or else it must be inferred from the MAC address, a dangerous proposition given untrusted hosts.

Control

Managing edge switches is potentially a daunting task. Instead of simply configuring a group of core switches within the network, each edge becomes another network device that needs to be maintained. As mentioned earlier, advanced edge switches provide methods for remote configuration, which can make these disparate switches appear as a single distributed virtual switch.

Given the visibility advantages of the edge described in the previous section, these switches are in a prime position to enforce network policy. The available rich sources of information give network administrators the ability to make fine-grained rules for handling traffic. In addition, by applying policy at the edge of the network it is possible to drop unwanted traffic as soon as possible, avoiding wasting the resources of additional network elements.

Advanced edge switches run on general-purpose computing platforms with large amounts of memory, enabling them to support a very large number of ACL rules of any form. While complex sets of rules may have a performance impact, it is possible to tailor the expressiveness, throughput, and cost of a set of ACLs to the needs of a particular application.

V. Related Work

Advanced edge switching solves the problem of separation of traffic on the virtual network from the policies of the physical network, by importing the network's policies into the virtual network. It is also possible to take the opposite approach: to export the virtual traffic into the physical network.

The latter approach is exemplified by VEPA [13] (Virtual Ethernet Port Aggregator), which changes the virtual bridge to forward all traffic that originates from a VM to the *adjacent bridge*, that is, the first-hop physical switch. This is a simple change to existing bridges in hypervisors.[2] In VEPA, the adjacent bridge applies the physical network filtering and monitoring policies. When traffic between VMs in a single physical server is sent

[2]A proposed patch to add VEPA support to the Linux kernel bridge added or changed only 154 lines of code [14].

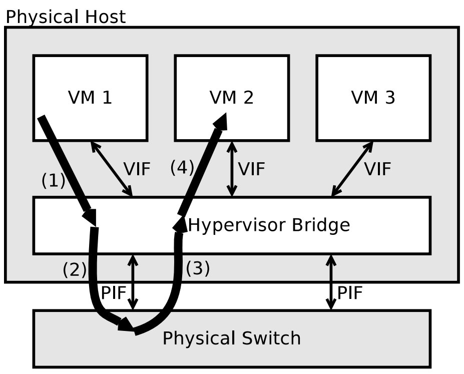

Fig. 3. Path of a packet from VM 1 to VM 2 in hairpin switching: (1) VM 1 sends the packet over a VIF to the hypervisor, (2) the hypervisor passes the packet to the adjacent bridge, (3) the bridge passes the packet back to the hypervisor, (4) the hypervisor delivers the packet to VM 2.

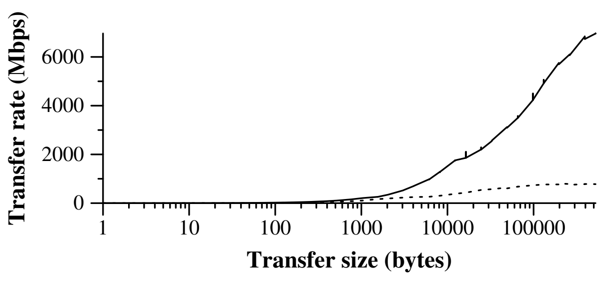

Fig. 4. Throughput versus flow size in XenServer virtual machines on the same host with Open vSwitch (solid) and VEPA (dotted). An advantage of edge switching is that the hypervisor does not send traffic off-box and can avoid the overhead of actually handling every packet twice.

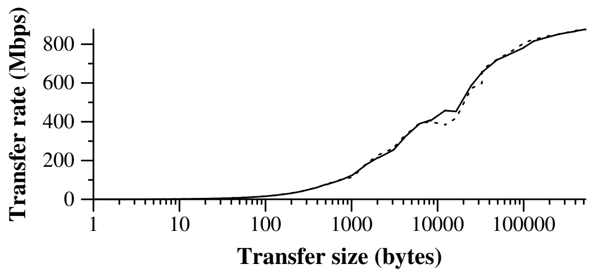

Fig. 5. Throughput versus flow size in XenServer virtual machine and remote bare-metal host with Open vSwitch (solid) and VEPA (dotted). Performance was roughly equivalent across all transfer sizes.

to the adjacent bridge, it then sends that traffic back to the server across the same logical link on which it arrived, as shown in Figure 3. This *hairpin switching* phenomenon is characteristic of the approach, so we adopt it here as the general term that includes VEPA and similar solutions, such as VN-Link from Cisco [6].

Sending packets off-box for processing allows the end-host to be simpler. Processing of packet headers is avoided and NICs can utilize low-cost components, since advanced functionality is handled by the adjacent bridge. If advanced NICs are used, SR-IOV makes moving packets from VMs to the adjacent bridge even more efficient by bypassing the hypervisor altogether.

Without the use of a technology such as SR-IOV, VEPA will have similar performance to an edge switch for off-box traffic. However, VM-to-VM traffic will tend to be worse, since it must traverse the physical link. In addition to throughput limitations and potential congestion, performance may be affected by the need to effectively handle twice as many packets. Even though no actual switching happens in the hypervisor these packets still must be copied to and from the network card, have their existence signaled with interrupts, and incur other overhead associate with actually sending and receiving packets.

Figures 4 and 5 show the effective throughput that Open vSwitch achieves for various flow sizes versus VEPA on the same hardware. The measurements were executed using NetPIPE 3.7.1 on Debian 5.0.5 VMs within Citrix XenServer 5.6.0. For the off-box test in Figure 5, Ubuntu Server 10.04 was running on bare-metal. VEPA support was based on the previously mentioned patch applied to the XenServer bridge source. The hairpin switch was a Broadcom Triumph-2 system with the MAC put into line loopback mode. All off-box communication was done over a 1 Gbps link.

Hairpin switching reduces the number of switching elements within the network. In the absence of centralized configuration of edge switches, this eases configuration for the network administrator. It also makes configuration consistent for networks that use a single source for their equipment.

Some hairpin solutions encapsulate frames sent to the first-hop switch with an additional tag header, e.g. VEPA provides the option of using S-Tag and VN-Link defines the new VNTag header. The tag provides additional context about the source of the packet, in particular the virtual ingress port. Without this information, the switch must rely on fields in the packet header, such as the source MAC address, which are easily spoofed by a malicious VM.

Tagging protocols introduce new issues of their own. A tag distinguishes contexts, but it does not say anything about those contexts. Most network policies cannot be enforced accurately without more specific information, so extra data (*port profiles*) must be distributed and kept up-to-date over additional protocols. The form of this information is not yet standardized, so users may be tied to a single vendor.

Besides this inherent issue, the specific proposals for tagging protocols have weaknesses of their own. Tags in VNTag and S-Channel are only 12 bits wide [15], [13], which is not enough to uniquely identify a VM within a scope broader than a server rack. This limits its usefulness beyond the first-hop switch. S-Channel tagging additionally does not not support multicast or broadcast across tags, which can potentially increase the overhead of sending packets to multiple guests. Tags may also introduce problems with mobility, even across ports in a single switch.

Hairpinning effectively turns an edge switch into an aggregation device. The first-hop switch must then scale to support an order of magnitude or more greater number of switch interfaces with features enabled. The hardware resources of top of rack switches, primarily designed to handle approximately 48 interfaces, may prove to be insufficient for the services the user wished to enable on a particular VM interface. Consider ACL rules for example: switch hardware is often constrained to about 4000 rules. This number may be adequate to define similar policies across most members of the network. However, it may become constraining if fine-grained policies are needed for each guest. For example, a switch with 48 physical ports and 20 VMs per port would allow roughly 4 rules per VM.

Hairpin switching is an attractive solution in environments that need to apply similar policies over all clients in the network or enforce them in aggregate. Edge switches provide more flexibility and fine-grained control at the possible expense of end-host CPU cycles. Fortunately, the approaches are not mutually exclusive and most environments will likely find that a combination of the two provides the best solution.

VI. Future

We predict that virtual networking as a field will continue to rapidly evolve as both software and hardware shift away from the model based on a more traditional physical network. We have already seen the beginning of this trend with software switches becoming more sophisticated. It will only accelerate as the line between the hardware switches used for physical networks and the software switches in virtual networks begins to blur.

As the trend continues, virtual switches will add more features that are currently available only in higher-end hardware switches. Current virtual switches support management tools such as NetFlow and ACLs. The next generation will add SNMP, NETCONF, a standard command line interface, and other features that will make software and hardware switches indistinguishable. Given the naturally faster pace of software development compared to hardware, parity will be reached before long.

The question of edge networking versus switching at the first hop is often framed as a debate between software and hardware. In reality, the two issues are orthogonal. Hardware has long been an accepted component of "software" switches, via checksum offloading, TCP segmentation offloading, and other acceleration features of NICs that are now common even in low-end computers. These provide a good blend of flexibility and performance: the speed of hardware under the control of software at the edge of the network where there is the most context.

Network cards are adding new types of offloading that are particularly suited to the types of workloads encountered with virtualization. Intel 10 GbE NICs [16], for example, can filter and sort packets into queues based on a variety of the L2, L3, and L4 header fields used by monitoring tools. Switch software can set up these programmable filters to accelerate forwarding taking place in the hypervisor. While it is unlikely that all traffic can be handled in hardware due to limitations in matching flexibility and the number of rules and queues, it is possible for the switch software to trade performance against flexibility as needed based on the application. The most active traffic flows can often be handled in hardware, leaving the software switch to deal with the edge cases.

Network cards offered by Netronome [17] go a step further by offering a fully programmable network flow processing engine in the card. These allow the complete set of rules to be expressed and pushed down into the hardware. Along with I/O virtualization that allows virtual machines to directly access a slice of the physical hardware, all packet processing can be done without the help of the hypervisor. The NIC does the heavy lifting, but the hypervisor must still provide the management interfaces and knowledge of the system that is used to configure the hardware.

As more advanced forms of offloading become available in networking cards, virtual edge switches will be able to balance the capabilities of software and hardware dynamically based on the application and available resources. These features become available in higher end network cards first, but they will trickle down over time. A hybrid stack of hardware and software aids in this process because it is possible to balance performance with cost and upgrade while maintaining a consistent control plane.

Just as hardware is fusing with software at the edge of the network, platforms designed for virtualization are moving into the core. Several switch vendors are currently porting Open vSwitch to their hardware platforms. This will provide a unified control interface throughout the network. Since the switches included in hypervisors have generally been custom-built for that platform, there was previously very little commonality with hardware switches. As the relationship between software edge switches and physical first-hop switches is currently asymmetric, it is not a large leap to something like VEPA. However, with switches like Open vSwitch and the Cisco Nexus 1000V providing a common management interface, the disadvantages of having very different switching abstractions are much greater. As long as any policy must be applied at the edge (as with untrusted tenants), it is better to be able to use a consistent set of management tools if at all possible.

VII. Conclusion

CPU virtualization has evolved rapidly over the last decade, but virtual networks have lagged behind. The switches that have long shipped in hypervisors were often little more than slightly modified learning bridges. They lacked even basic control and visibility features expected by network administrators. Advanced edge switches, such as Open vSwitch, answer many of their shortcomings and provide features previously only available in high-end hardware switches.

The introduction of these advanced edges changes the virtualized networking landscape. Having reached near parity with their hardware brethren, a fresh look at the relative merits of each approach is warranted. Hairpin switching can offer performance benefits, especially in situations with little VM-to-VM traffic and heavy policy requirements. Yet with their greater context, early enforcement of policies, and ability to be combined with other edges as a single distributed switch, the new generation of edge switches has many compelling features. Advances in NIC hardware will reduce the performance impact of edge switching making it appealing in even more environments.

VIII. Acknowledgements

The authors would like to thank Andrew Lambeth, Paul Fazzone, and Hao Zheng.

References

[1] J. Touch and R. Perlman, "Transparent Interconnection of Lots of Links (TRILL): Problem and applicability statement," RFC 5556, May 2009.

[2] VMware Inc., "Features of VMware vNetwork Virtual Distributed Switch," http://www.vmware.com/products/vnetwork-distributed-switch/features.html, May 2010.

[3] Cisco, "Nexus 1000V Series Switches," http://www.cisco.com/en/US/products/ps9902, Jul. 2009.

[4] Nicira Networks, "Open vSwitch: An open virtual switch," http://openvswitch.org/, May 2010.

[5] P. Congdon, "Virtual Ethernet Port Aggregator Standards Body Discussion," http://www.ieee802.org/1/files/public/docs2008/new-congdon-vepa-1108-v01.pdf, Nov. 2008.

[6] "Cisco VN-Link virtual machine aware networking," http://cisco.biz/en/US/prod/collateral/switches/ps9441/ps9902/at_a_glance_c45-532467.pdf, April 2009.

[7] R. Malekzadeh, "VMware for Linux Networking Support," http://web.archive.org/web/19991117164603/vmware.com/support/networking.html, 1999.

[8] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *HotNets*, 2009.

[9] Citrix Systems, "Overview of the XenServer API," http://docs.vmd.citrix.com/XenServer/5.6.0/1.0/en_gb/sdk.html#sdk_overview, May 2010.

[10] OpenFlow Consortium, "OpenFlow specification, version 1.0.0," http://www.openflowswitch.org/wp/documents/, Dec. 2009.

[11] *Synergy*. San Francisco, CA: Citrix Systems, May 2010.

[12] VMware Inc., "Achieve breakthrough performance with enterprise application deployment," http://www.vmware.com/solutions/business-critical-apps/performance.html, May 2010.

[13] Hewlett-Packard Corp., IBM *et al.*, "Edge virtual bridge proposal, version 0, rev 0.1," IEEE, Tech. Rep., April 2010.

[14] A. Fischer, "`net/bridge`: add basic VEPA support," http://lwn.net/Articles/337547/, June 2009.

[15] J. Pelissier, "Network interface virtualization review," http://www.ieee802.org/1/files/public/docs2009/new-dcb-pelissier-NIV-Review-0109.pdf, Jan 2009.

[16] "Intel 82599 10 GbE contoller datasheet," http://download.intel.com/design/network/datashts/82599_datasheet.pdf, April 2010.

[17] Netronome Systems, Inc., "Netronome network flow processors," http://www.netronome.com/pages/network-flow-processors, May 2010.