nSwitching: Virtual Machine Aware Relay Hardware Switching to improve intra-NIC Virtual Machine Traffic

Jim Bardgett
Harris Corporation
1025 W. NASA Blvd.
Melbourne, FL 32919

Cliff Zou
University of Central Florida
4000 Central Florida Blvd.
Orlando, FL 32816

***Abstract* – Recent development on Ethernet switching to provide Single Root I/O Virtualization (SR-IOV) on network interface cards (NICs) improves Ethernet throughput for Virtual Machines (VMs) and lowers CPU loads. SR-IOV creates multiple receive queues on a NIC, directly accessible by VMs for frames coming from sources external to the Ethernet port. This virtualization of Ethernet ports and the presentation of frames directly to VMs eliminates a major cause for CPU loading by reducing the interrupts for receipt of inbound frames. However, SR-IOV cannot provide switching support for two VMs on the same computer; the only existing switching option is software-based switching in the hypervisor, which limits throughput and results in high CPU utilization. New industry standards 802.1Qbg and 802.1Qbh assist Ethernet traffic between VMs, but they require costly replacement of both Ethernet NICs and the data center external physical switch infrastructure. In this paper, we propose a new design by integrating a new Ethernet switching functionality into the NIC, which is called nSwitch, to enable hardware-based switching for inter-VM traffic on a single computer that has a single or multi-socket, multi-core CPU. Compared with software-based switching in the hypervisor, this enhancement greatly reduces CPU utilization and permits efficient traffic monitoring for on-board inter-VM I/O. Furthermore, it eliminates the back-and-forth usage of external port or channel bandwidth for internal VM communications.**

Keywords: Virtualization; Virtual Machine switching; CPU utilization; vSwitch

I. INTRODUCTION

Traditional data center network switching architecture involves the definition of switching platforms, port bandwidth, physical medium connectivity, virtual local-area network (VLAN) and Internet Protocol (IP) addressing, fail-over mechanisms, port bonding, quality of service (QoS) and security. With the introduction of improved switching protocols and virtualization, increased utilization of hardware has imposed many design challenges. Trill [5], a recent replacement for spanning tree, eliminates unused redundant ports across data center switches and permits continuous up time during network technology refresh or maintenance. A more dramatic change is the virtualization of Operating Systems (OS) which pushes CPU and I/O utilization to levels unachievable without Virtual Machines (VMs). Hypervisors and VM clients are improving hardware reduction, beyond that achieved by stacking of multiple applications on a single computer.

With the instantiation of multiple VMs on a single server, it is important to consider the frequent switching of frames between VMs on the same machine. The initial solution, a virtual switch (called "vSwitch") was hypervisor integrated software for VM to VM (VM-VM) switching. Fig. 1(a) shows a vSwitch.

However, Virtual Switching (vSwitching) from one VM to another on the same server was found [10, 11] to greatly increase CPU load for all but modest level's of I/O. I/O Virtualization[1] in vSwitch has been analyzed since 2008 [1, 9, 19]. Upgraded approaches, such as Xen's OpenvSwitch [24] integration, still had limited success as we describe in this paper and also added traffic rate limiting. Additional functionalities such as access lists on the switch in the hypervisor lead to dramatic breakdown in throughput and are currently costly to implement. Another major problem is that practical packet capture, which is critical in troubleshooting virtualization in the data center, is not supported by vSwitch.

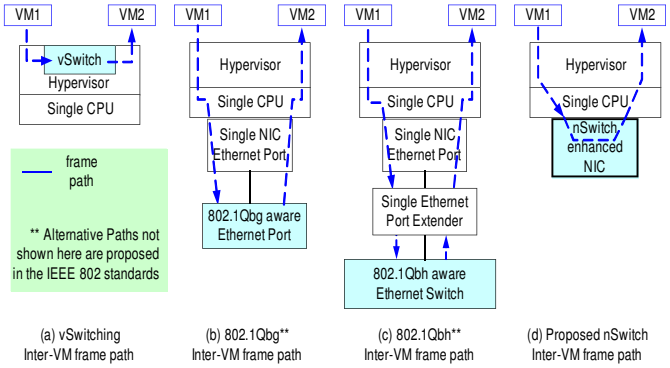

Figure 1: High level architecture of single computer inter-VM frame paths: a). Virtual Switch; b & c). Physical Switch; and d). Our proposed nSwitch.

As implemented, Spanning Tree Protocol (STP) prevents a switch from redirecting a VM-VM frame out of the port through which it entered. The introduction of reflective relay and VEPA in two new IEEE protocol 802.1 standards, 802.1Qbg (Qbg) and 802.1Qbh (Qbh), permits reflective relay, i.e., a hairpin turn at the switch port. However, Qbg and Qbh require modification of the NIC and reflective relay upgrade to the external network hardware to switch VM-VM frames originating and terminating on the same physical Ethernet port[2].

In this paper, we propose the nSwitch architecture to improve the VM-VM switching performance for traffic in the same computer across multiple CPUs and sockets. nSwitching is compatible with the SR-IOV[3] specification without any

1 I/O Virtualization is defined by [21] as "the capability of a single physical I/O unit to be shared by more than one system image".
2 Qbh requires hardware and software changes to the external physical switch where Qbg requires only software changes.
3 SR-IOV is the sharing of a single physical Ethernet device, appearing as

Table 1: Physical Switch comparison with the proposed nSwitch

Method	Standard	Implemented	Concern	Details
Proposed nSwitch	SR-IOV plus our proposal	Ethernet NIC	Not yet implemented.	Low CPU utilization. Low VM load not proportional to I/O. Fast, high throughput. Data does not traverse the pSwitch port, does not affect VM resources. nSwitch simplifies the selection of bridge port based on MAC Address selection supported by VF on bridge.
pSwitch with Reflective Relay	802.1Qbg/ Qbh	Physical Switch	Latency, Bandwidth, External switching.	Well understood traffic patterns. Throughput must contend for available physical port bandwidth. pSwitch requires multiple external devices in some cases, modifies frame and requires modification of existing switches. It adds additional latency.

Ethernet frame alteration. By defining the nSwitching as an enhancement to SR-IOV, the architecture will enable all high load inter-VM traffic without the overhead introduced by 802.1Qbg or 802.1Qbh. Our approach allows a VM to select the nSwitch Reflective Relay group it participates in by MAC Address, thus enabling the VM management software to provide additional dynamic control. nSwitch allows CPU optimization across multiple CPU cores and our proposed implementation provides a tool for additional research in VM optimization due to switching based on MAC Address. The method used also allows traffic switched through nSwitch to be monitored, allowing visibility into the network traffic that is currently only provided by a sniffer used in conjunction with a physical switch (called "pSwitch"). No pSwitch modification is required by our proposed implementation.

Although we mainly focus nSwitch design and evaluation for wired networks in this paper, nSwitch is also applicable and suitable for advanced wireless networking architectures that are implemented with virtual machines. There is a considerable need for virtualized, low power, mobile network nodes to reduce their power and channel utilization, for which the proposed nSwitch technique could be very useful. An example would be the virtualization of mobile routing nodes in ad-hoc networks such as those deployed for dynamic search-and-rescue networks in remote areas. Reduction of CPU utilization reduces power consumption and nSwitching will reduce precious wireless channel bandwidth as well. The contributions of this paper are:

- Present a new switching architecture called nSwtich to enable hardware based switching for inter-VM traffic on a single computer
- Conduct performance comparison of the nSwitch, vSwitch and pSwitch.

The rest of the paper is organized as follows. Section II introduces background on the physical and virtual switching in data centers. Section III introduces proposed single and multiple Ethernet port nSwitch designs. Section IV evaluates existing vSwitch and simulates proposed nSwitching and not yet available pSwitching. Summary is presented in section V.

II. Background

Before virtualized servers (no hypervisor or virtual machine monitor), the network architect's scope ended with physical server connectivity. The physical external switch delivered switched Ethernet frames to the server's OS. The network edge and switch fabric was bounded by the physical Ethernet switch which switched between servers[4]. The same process occurred for a blade server with a pass through Ethernet NICs. Traditionally this switching occurred in the directly connected pSwitch which passed frames in one direction or the other without reflection[5]. The switch fabric edge and thus the network edge was well defined. In this paper, we confine our discussion to single VLAN non-trunked switch ports, independent of virtual port channel or chassis.

VM servers on the same single physical device require frame switching which is contrary to the spanning tree [6] based switching design. Traffic between VMs on a single NIC card would not pass traffic unless Virtual Switching in the hypervisor existed [7]. Fig.1 shows VM-VM communications.

However, virtual switches in hypervisors, like those provided by KVM, VMware and Xen, have had many problems and proposed solutions [1,10,11]. Evolution of OpenvSwitch has shown recent improvement. The vSwitch makes monitoring of protocols or bandwidth usage complicated or impossible [8]. Open-vSwitch does have rate limiting but not QoS (e.g. 802.1p). Concerns like limited I/O bandwidth and the additional skill development for server administrators makes managing the vSwitch complex [1,8,9]. In addition, vSwitch could cause very high CPU loads [1,9,10] with software switching [11].

After the concern with software switching in the hypervisor, subsequent analysis allowed for hypervisor redesign, new VM queuing and SR-IOV. SR-IOV creates receive queue's for externally received frames [12] and eliminates the packet receive interrupt for the receiving computer. However, VM-VM traffic on the same computer is not considered by SR-IOV and currently there is no existing hardware solution.

For VM-VM traffic, only the vSwitch shown in Fig. 1(a) has been realized. vSwitches have proprietary and at least one open source implementation [7]; the major limitation is determined by the CPU loads which is created by I/O Virtualization.

multiple separate physical devices through virtualization.

4 Using only a MAC layer broadcast or a look-up provided by the MAC Address table or multicast functionality.

5 Based on the implementation of the spanning tree algorithm invented by Radia Perlman while at Digital Equipment Corporation.

There are two unimplemented switching methods defined by IEEE, which will deliver VM-VM frames on the same network, logic board and switch port. Each of the standards proposes a different method for enabling pSwitch Reflective Relay: Edge Virtual Bridging (EVB) also called 802.1Qbg[6] and Bridge Port Extension (BPE) also called 802.1Qbh[7] Reflective relay [13] and another pass through device known as a port extender in addition to the pSwitch is required for 802.1Qbh. pSwitch requires hardware and software changes to support Qbg or Qbh. Depending on the implementation, the frame may have to continue upstream across fiber or copper until it is returned to the same Ethernet port for processing by the same NIC it exited. This traversing of three devices and two interconnecting fiber and associated port interface connection devices like SPF+ requires a large number of external devices required to switch a packet from one VM to another on the same CPU. Fig. 1(b)(c) show the inter-VM frame path for a frame going from one VM to another on the same device in and out of the same Ethernet port for Qbg/Qbh, irrespective of which CPU is the frame destination.

III. Proposed nSwitch Design

We present two designs of nSwitch which reflect VM-VM traffic on the same computer. These designs differ in terms of implementation complexity and functionality. Design 1, shown in Fig. 2, is a single Ethernet port NIC. Design 2, shown in Fig. 4, has two Ethernet ports. Both designs support multiple CPUs and multiple socket logic boards.

A. *nSwitch Design for Single Port SR-IOV NIC Architecture*

Fig. 2 shows the nSwitch design with the SR-IOV architecture for NICs with single Ethernet port. We show a single Ethernet NIC which builds on the SR-IOV functions (e.g. use of SR-IOV VFs for frames and use of SR-IOV PFs for reflective relay). The implementation, shown in Fig. 2, is fully compatible with SR-IOV and utilizes an existing core [16] as a reference structure. Virtual Machine 1 (VM1) has a virtual MAC Address (vMAC1) and is supported by an SR-IOV aware hypervisor on a Virtualization Aware CPU. The NIC shown has the Physical Functions (PFs)[8] associated with the physical interfaces and Virtual Functions (VFs)[9] which represent the physical interface to VMs; PFs and VFs are defined in detail in [18, 20, 23]. The Configuration PF 0, which configures the functionality of PF 0 Ethernet, has been modified to allow nSwitch functionality within the NIC.

Pseudo code for the single PF, corresponding to a single

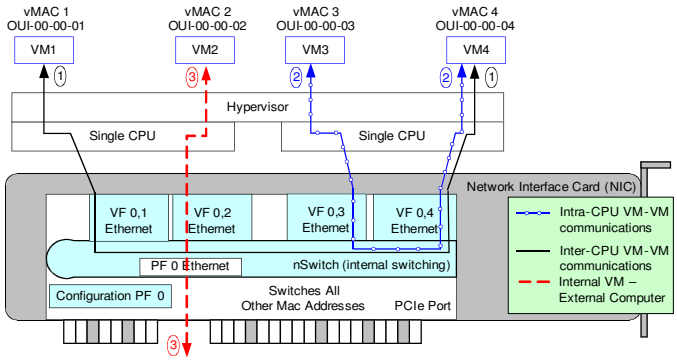

Figure 2. nSwitch architecture and frame path for a single Ethernet port NIC

Ethernet Port design is shown in Fig. 3. The nSwitch functionality is in addition to the same routing functionality for SR-IOV as shown in the DesignWare IP datasheet by Synopsys [16]. Modifying the Configuration PF 0 structure with the proposed pseudo-code in Fig. 3 allows the Routing Function [15] to permit the VF to VF communication using reflective relay.

Initial state

- VF 0, 3 associated with VM3; - VF 0, 3 associated with VM3; - PF 0 associated with PCIe routing function allocating bandwidth data path and switching functionality between VMs; Create table space (Table A) to associate a MAC Address with a given VF

Initialization of VM interfaces:

- vMAC offered to VMs will have a consistent MAC Address OUI based on the PF number (e.g. VF 0,3 assigns vMAC3 an OUI F0-F0-F0, VF 0,4 assigns the same to vMAC4); Insert MAC Address into Table A and associate with a given VF

Steady State

- Upon receipt of frame from a VF, compare source and destination OUI, prioritize based on 802.1p marking from VM.

- Case 0: equal source and destination OUI, look up the destination VF and route packet to that VF for the associated VM; Case 1: unequal source and destination OUI, send to PCIe port; Case 2: Follow SR-IOV for receipt of a frame from PCIe port

Figure 3. Pseudo Code for nSwitching in the Synopsys (SR-IOV) core.

B. *nSwitch Design for Multiple Port SR-IOV NIC*

For a NIC card with multiple Ethernet ports, the nSwitch design will be slightly different. Fig. 4 shows the design with multiple SR-IOV architectural PF elements in a multi-port Ethernet NIC which builds on the SR-IOV functions. In Fig. 4, we see the additional frame paths available for communications between VM1 and VM4 which reside on different external physical ports (represented by different PFs). VM1 has been assigned with VF0,1[10]. VF0,1 is presented to the Guest OS on VM1 because the hypervisor is

6 It was found in a description of 802.1Qbg that a miniaturization of 802.1Qbg may be able to be performed on the NIC.

7 Vendor presentation slides may suggest an implementation of NIC switching but not as what we propose.

8 Physical Functions (PFs) are the same as PCIe functions with a full configuration space and a supervisory role over the associated VM. PF Numbering requires a single digit.

9 SR-IOV specification introduces Virtual Functions (VFs) associated with a Physical Functions (PFs). When refering to a VF a PF is required, so the number pair is the PF, VF of format VF0,1 for VF1 associated with PF0 [20].

10 This is the VF number 1 associated with PF 0.

SR-IOV aware. In this case the role of VF0,1 is to receive a frame from VM1 and the PF 0 Ethernet switches the frame to PF 1 using the pseudo code as seen in Fig. 3. The design shows a connection between the two physical functions supported by configuration of the data path in the Synopsys core. SR-IOV Virtual Functions are enhanced by our pseudo code [17, 18]. As the nSwitch enhancement is applied to the routing function, the change to the NIC software permits inter-VF communications through an existing switch structure in the core [16].

The core by Synopsis [16] shows considerable promise for the ability to integrate the nSwitch functionality with minimum effort. Based on the datasheet [16], it appears only a software modification to the core is needed to implement nSwitching. With PCIe 3.0 32-bit PIPE and 8.0 GigaTransfers/second (GT/s), this core can support significant transfers from VM-VM internally. Properly constructed code for internal VF-VF routing are envisioned as the logic necessary for nSwtiching in the NIC.

Fig. 4 shows the frame path for a multiple port SR-IOV card with multi-VM, multi-CPU architecture for the proposed nSwitch. It associates each PF, which is correlated to a physical port on the NIC, to a separate nSwitch function to provide an individual VF for each VM. The figure shows the frame path for VM-VM communications on a single CPU (the No. 2 blue path), the path for VM-VM communications across CPUs (the No. 1 black path) and the frame path for VM to External Traffic (the No. 3 red path). For external traffic, PF 0 is used and all PCIe structures are used for physical frame transmission including internal scheduling, framing, encoding, signal generation, etc.

The VF Routing Identifier (RID) [16] structure in the Synopsys core should support the switching of packets with the appropriate code implementation. Each nSwitch instance corresponds to a single PF. VF Switch Identifiers (SIDs) will be expressed as an offset of the PF SIDs [14].

C. Benefits of nSwitching

Without the nSwitch implementation, Ethernet cards with EVB, BPE and SR-IOV strictly address frames coming from an External Physical Ethernet Switch (pSwitch) using Direct I/O to the VM. The addition of nSwitching to SR-IOV will reduce CPU loads and eliminate the need for bandwidth between the NIC and pSwitch for inter-VM traffic internal to the server.

Compared with the software-based vSwitch, there are many benefits to switching in hardware by using nSwitching:

- Eliminate CPU utilization increase caused by inter-VM I/O traffic and remove NIC bandwidth constraint
- Enable application of Access Lists (ACLs) and Quality of Service (802.1p) without CPU performance hit

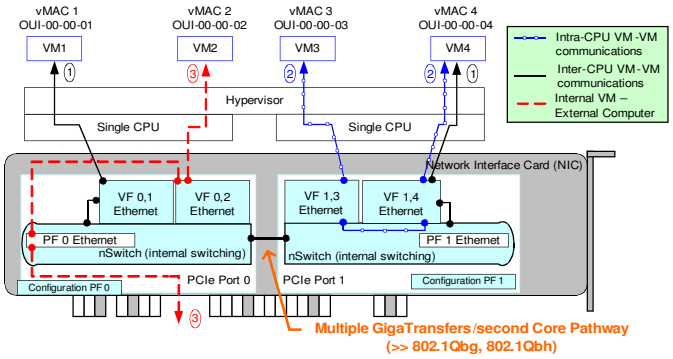

Figure 4. nSwitch Logical Functions possibly implemented via BIOS in Synopsys Core for NICs with multiple Ethernet ports.

- Enable VM-VM frame monitoring and control using MAC Address Organizational Unit Identifier (OUI)

Without using nSwitch, all bandwidth in and out of every VM must be accounted for in the NIC and Ethernet switch port speed, as well as upstream device if required. We propose that for heavy VM workloads between VMs on the same Ethernet port, Ethernet switching components of 802.1qbg should be integrated into the SR-IOV and MR-IOV NIC designs. This will eliminate the CPU workload problems created by inter-VM switching in the hypervisor or vSwitch, and the bandwidth, latency and reliability problems created by switching in the pSwitch.

IV. EVALUATION

Software, hardware, platform profiling tools and VM with several operating systems were used for switching methods evaluation. Investing capital in any new core in silicon would be cost prohibitive without the intent to produce and sell the product, thus real implementation is beyond the scope of this paper. In this paper, we compare existing vSwitch with an approximation of 802.1Qbh and the proposed nSwitch.

A. Testing Software, Hardware, Profiling Tools and VMs

Software: Citrix(r) XenServer(tm)[11] 5.6 FP1 with Open vSwitch [7] was chosen for accelerated I/O virtualization and a paravirtualized guest. The VM operating systems used were Redhat Beta 6 and Ubuntu 10.10 Maverick Meerkat.

Hardware: Directed I/O, Virtualization Technology for Directed I/O (VT-D) and SR-IOV were integrated in the main board, Ethernet card and Processor. The hardware was special built by us as the features are not yet combined in a single platform.

Platform profiling tools: Linux top, dstat, md5sum for load and CPU Limit. Xen [22] uses Open vSwitch. The Redhat VMs were given 1Gigabyte RAM and 4 GB of hard drive. VMs also used Ubuntu 10.10. Fig. 6(a) shows the setup.

11 Note in all cases in this document all Registered or Unregistered Trademarks are the property of their respective owners, designated or not.

B.

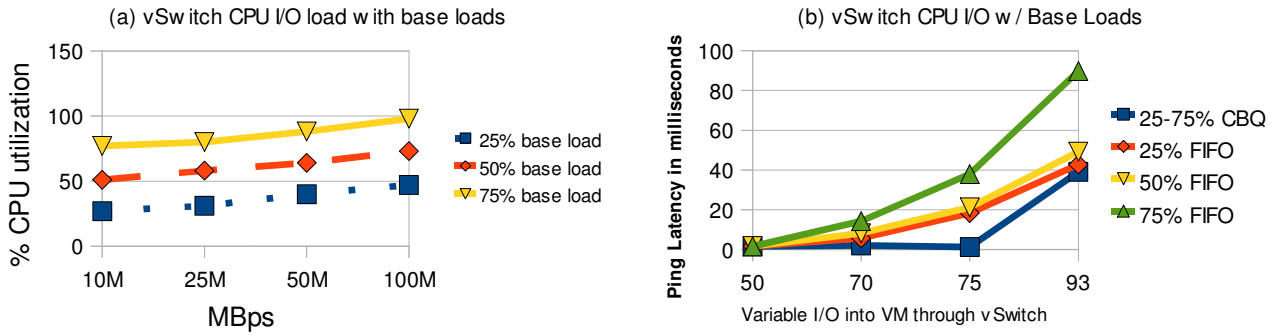

Fig. 5. CPU I/O load and delay for VM-VM data transfer with consistent loads
(a) 25%-75% starting base load (b) Data transfer drives CPU load (CB = class based)

vSwitch: Bandwidth, Delay and CPU Load with 2 VMs

We seek the combined CPU load and delay imposed by the vSwitch when transferring with different traffic rates (10 MBps to 93 MBps) from VM2 to VM1 in figure 1(a). We test three fixed CPU loads on the receiving virtual machine, VM1. We also seek to know the:

- Minimum consistent I/O load required to drive CPU utilization to 100%
- Switching latencies due to I/O driven CPU load
- Maximum vSwitch transfer rate between VMs.

Fig. 5(a) shows the testing results of vSwitch time delay (latency) and CPU loading. Tests were executed with 2 VMs, a varying load on VM1 using md5sum controlled with CPU Limits software when VM2 sent a varying bandwidth through the vSwitch to VM1. The total CPU load was the sum of system load and the CPU soft interrupt load**.** The results are shown in Fig. 5(a) under three different CPU base loads. As I/O increases from 10 MBps to 93 MBps, a 20% CPU load increase would be produced due to the vSwitch. In Fig. 5(b), traffic loads over 75 MBps can be seen to dramatically increase the latency between VM2 and VM1 with a vSwitch. The maximum throughput of the Xen vSwitch tested with 0% CPU base load is 93 Mega Bytes per second, irrespective of the number of VMs. Our proposed nSwitch can increase this maximum throughput without using external NIC bandwidth.

C. pSwitch: Bandwidth, Delay and CPU Load of 802.1Qbh

For pSwitching approximation shown in Fig. 1(c) we use three VMs with no fixed CPU load. VM3 sends data using the external 802.1Qbh switch to VM1 and VM2 sends traffic to an external computer through the SR-IOV Ethernet NIC. We show our simulation of Fig. 1(c) in Fig. 6 (b). Fig. 6(b) shows the test setup for a single Gigabit Ethernet port and one Cat 6 cable. VM1 communicates with VM2 by exiting an Ethernet port then re-entering the same Ethernet port after traversing a Port Extender and an 802.1Qbh-aware switch. This emulation is required because there is no existing 802.1Qbh software for NIC cards and no Qbh software for the pSwitch exist at this time. We illustrate the port bandwidth limitation in Fig. 7. We seek to find whether any load created on VM3 will affect either the CPU load or the transfer rate from VM1 to VM2.

From our tests we determine that processing time (t_p) is not a significant factor. For bandwidth investigation, the red oval in Fig. 6 (b) shows that all VM-VM communication must pass through the 'port extender' to the NIC. This fact keeps the maximum bandwidth between all VMs to be bounded by the Ethernet's physical bandwidth at 1 Gigabit per second. Fig. 7 shows the results when traffic is passing from VM3 to VM1 as VM2 gradually increases traffic sending to an external device over SR-IOV. Fig. 7 also shows the limitation of 802.1Qbh: port speed limits the upper bound of all VM communications, and intra-VM communication will reduce the bandwidth available for external communication.

D. Proposed nSwitch approximation: Bandwidth, Delay and CPU Load testing

From Fig. 1(d) we see that the frame path is between two VMs using the NIC as the switching mechanism. The internal mechanisms are shown in Fig. 2 and 4 and pseudo code in Fig. 2. The latencies for this switch path, shown in Fig. 8, are defined as the sum of the time to the NIC (t_1), the time from the NIC (t_4) and the time through the NIC(t_p). Thus we have a way to approximate nSwitch latency. To make an estimate of

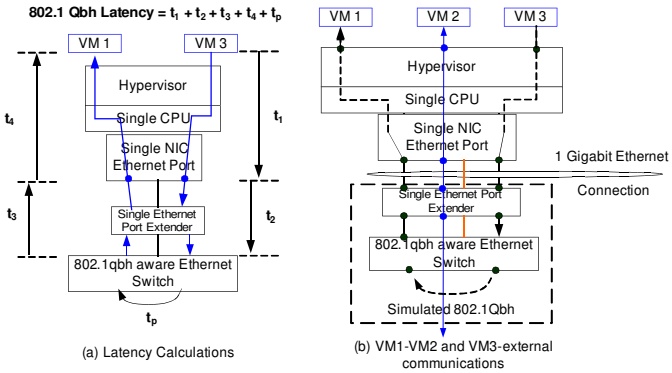

Figure 6. Expanded 802.1Qbh Approximation: Frame path and latency associated with 802.1Qbh
(a) Qbh Latency = $t_1 + t_2 + t_3 + t_4 + t_p$ (b) Maximum bandwidth = port speed.

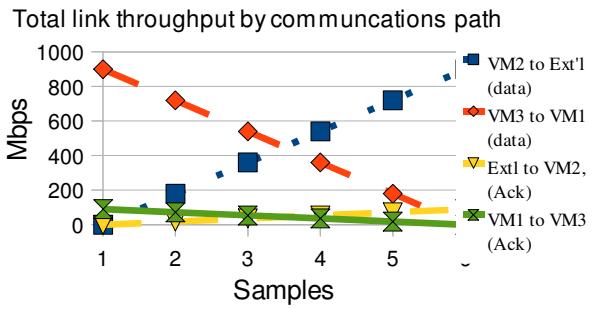

Figure 7. VM to VM and VM to External device throughput showing path traffic is constrained by the physical port and proportional Ack traffic

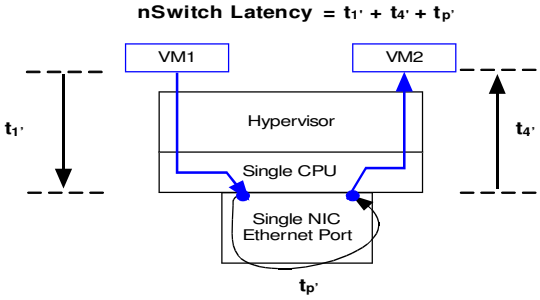

Figure 8. VM 1 to VM 2 contributions to nSwitch Latency = $t_{1'} + t_{4'} + t_{p'}$

the latency in the nSwitching architecture, we simulate latency with 1 VM pinging the IP address assigned to the SR-IOV NIC. We find the nSwitch approximate latency is on average about 0.009 ms. Fig. 8 subscripts are shown with a prime ['] as they are shorter than the latencies for 802.1Qbh which passes through the entire NIC.

With nSwitch there is no port bandwidth limitation; available channel bandwidth in the NIC determines the bandwidth available between virtual machines. As SR-IOV mechanisms are used for packet receipt in nSwitch there are no CPU loads created by switching in the NIC.

VI. Conclusion

nSwitch is shown to be able to reduce CPU utilization over the vSwitch and decrease latency. Comparing with 802.1Qbh or Qbg, inter-VM transmission speed will not be limited by the Ethernet port speed. At least one core was shown to eliminate the port bottleneck using PCIe 3.0 32-bit PIPE and 8.0 GigaTransfers/second, and can support significant VM-VM transfers on an nSwitch-enabled Ethernet card. We have presented a method of using SR-IOV functions in the nSwitching design and proposed that it is feasible to investigate the detailed implementation nSwitching in existing SR-IOV core structures. One of the primary benefits for nSwitching is that it eliminates any load created on the CPU due to switching in the hypervisor and the changes to the switch infrastructure as required by other edge switching technologies.

References

[1] Santos, J. R., Turner, Y., Janakiraman, G., Pratt, I. "Bridging the Gap between Software and Hardware Techniques for I/O Virtualization". USENIX '08: 2008 USENIX Annual Technical Conference. 2008.

[2] IEEE Standard, "802.1Qbg - Edge Virtual Bridging". http://www.ieee802.org/1/pages/802.1bg.html

[3] IEEE Standard, "802.1Qbg - Edge Virtual Bridging". http://www.ieee802.org/1/pages/802.1bh.html

[4] IEEE Standard, "802.1w-2001 - IEEE Standard for Local and Metropolitan Area Networks - Common Specifications - Part 3: Media Access Control (MAC) Bridges: Amendment 2 - Rapid Reconfiguration". http://standards.ieee.org/findstds/standard/802.1w-2001.html

[5] Touch, J., & Perlman, R. Request for Comments (RFC) 5556. May, 2009. IETF Network Working Group.

[6] IBM. "Spanning Tree Protocol and the Virtual Switch". (). z/VM V5R3.0 Connectivity SC24-6080-05. http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zvm.v53.hcpa6/hcsc9b2190.htm

[7] Open vSwitch. "About: What is Open vSwitch?". (April, 2011). http://www.openvswitch.org/

[8] McClellan, R., Metzler, J. (June, 2010). http://searchnetworking.techtarget.com/tip/Networking-for-virtualization-Virtual-network-switch-problems-abound

[9] Wood, T., Cherkosova, L., Ozonat, K., Shenoy, P. "Profiling and Modeling Resource Usage of Virtualized Applications". (2008). Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. (2008)

[10] Cherkasova, L., Gardner, R. "Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor" (April, 2005) Proceedings Of USENIX Annual Technical Conference (2005)

[11] Menon, A. Renato Sanots, J. Yoshio, T., Janakiraman, G. Zwaenepoel, W. "Diagnosing Performance Overheads in the Xen Virtual Machine Environment" (2005) USENIX: First International Conference on Virtual Execution Environments 2005 (VEE'05).

[12] LAN Access Division, Intel. "Intel(r) 83576 SR-IOV Driver Companion Guide" (June, 2009). http://download.intel.com/design/network/desguides/322192.pdf

[13] Blade Network Technologies, et. al. "Standardizing Data Center Server-Network Edge Virtualization" (October, 2010). http://h30499.www3.hp.com/hpeb/attachments/hpeb/bladesblog00/130/1/VEPA-EVB%20Industry%20Whitepaper.pdf

[14] Varma, A. "Single Root IOV Endpoint Implementation" (2007). PCI-SIG.

[15] Dong, Y., Yu, Z., Rose, G. "SR-IOV Networking in Xen: Architecture, Design and Implementation". (2008). USENIX, First Workshop on I/O Virtualization (WIOV'08).

[16] Synopsys. "Datasheet: DesignWare IP PCI Express Single-Root I/O Virtualization" (February, 2011). https://www.synopsys.com/dw/doc.php/ds/c/dwc_pci_express_iov.pdf

[17] Dong, Y., Jiang, Y., Tian, K. "SR-IOV Support in Xen". (June, 2008) Xen Summit Boston 2008.

[18] PCI-SIG. "Single Root I/O Virtualization and Sharing Specification 1.0. "(September, 2007). http://m1.archiveorange.com/m/att/hrTVj/ArchiveOrange_PD2t5zHbVLVi1Wd9XFMQb1SUkS0a.pdf

[19] Shafer, J. "I/O Virtualization Bottlenecks in Cloud Computing Today". (March, 2010) Workshop on I/O Virtualization.

[20] Knowlton, S. "Using PCI Express for I/O Virtualization". (2011). http://www.chipestimate.com/techtalk.php?d=2010-05-11

[21] Krause, M., Recio, R. "I/O Virtualization and Sharing" (2006). Microsoft WinHEC 2006..

[22] Xen. Xen Wiki: XenNetworking. (2011). http://wiki.xensource.com/xenwiki/XenNetworking

[23] Intel. "PCI-SIG Single Root I/O Virtualization (SR-IOV) Support in Intel® Virtualization Technology for Connectivity: Efficient Native Sharing of I/O Devices with Virtual Machines for enhancing I/O Performance" (June, 2008).

[24] Plaff, B., et. al. (2009). "Extending Networking into the Virtualization Layer". Hotnets 2009.