VMready

Implementation of a Profile Based Virtual Port Switching Framework

Vijoy Pandey
Blade Network Technologies
2350 Mission College Blvd, #600
Santa Clara, CA 95054
+1-408-931-3803

Jay Kidambi
Blade Network Technologies
2350 Mission College Blvd, #600
Santa Clara, CA 95054
+1-408-886-5040

Abstract

This paper presents an overview of VMready [1] – an implementation of a server-side virtualization aware network switch. The VMready network switch performs switching at the virtual port layer instead of at the physical ports as in traditional network switches. This paper describes the software and hardware architectures required to adapt network switching to this paradigm, discusses the tradeoffs made, and explores the future direction for this architecture.

1. Introduction

Virtualization of physical resources has changed some of the assumptions that we take for granted in designing, configuring and managing the data center. Elements that can be virtualized are any physical resource that can be abstracted and provisioned. Some examples of these elements are the CPU, network I/O and storage I/O.

Virtualization presents unique challenges to enterprise data center administrators, such as, provisioning of resources for each of the virtualized entities per physical host; provisioning the storage and network connectivity of various host-side virtual entities spanning multiple hosts; automatic provisioning of host virtual entities as per the SLAs requiring dynamic creation and deletion of these virtual entities. This may require dynamic reallocation of physical resources such as CPU, memory and I/O attributes; and provisioning for effective redundancy and failover of these physical and virtual entities for uninterrupted service. For efficient operation, all the elements of the data center should operate coherently.

Technologies such as hypervisor-assist CPUs and Single (or Multi) Root I/O Virtualization [11] (SR-IOV) converged network adapters (CNA) bring virtualization to the physical host in a data center.

The solution described in this paper takes the next step, and brings the dynamic nature of server-side virtualization to the networking elements of the data center; particularly by expanding the intelligence of the network fabric to adapt to the dynamic nature of the virtualized end host.

Through the remainder of this paper, we will concentrate on two virtualization technologies on the physical host – hypervisors and the resulting virtual machines (VMs), and CNAs

In Section 2, we briefly describe the virtualized server environment and the problems it creates for the network in a traditional data center. Section 3 will describe the overview of our VMready network switch, which attempts to solve the issues outlined, and Section 4 will discuss the underlying data path and control path technologies in further detail. Section 5 enumerates practical limitations and tradeoffs in our implementation. In Section 6 we offer our acknowledgements, and provide a summary of related literature and standardization work currently taking place in this area. Finally, Section 7 discusses future directions for VMready and offers our conclusions.

2. Network I/O in a Virtual Host Environment

In a non-virtualized host environment, each port on an edge network switch[1] is connected to one host or one interface in case of a multi-homed host. In effect, this implies that a port-based switch configuration on an

[1] An edge network switch is defined as a network switching element that is directly connected to a physical host

edge switch in a non-virtualized environment corresponds directly to a physical host based switch configuration. In other words, network policies[2] that are applicable to a certain physical host are assigned to a particular port on the edge network switch. Once configured, the physical host rarely moves, but the in the event that it does move, the effort involved in manually migrating network policies is minimal. This model was very successful in a non-virtualized host environment, but breaks down immediately in a virtualized host environment as physical hosts, and hence edge switch ports, no longer have a one-to-one mapping to servers or services. The virtualization of a physical host such that it can execute multiple virtual machines, changes the traditional networking model in the following ways:

i. Each CNA will carry storage (Fibre Channel over Ethernet or iSCSI) and network traffic, and will potentially require different switching policies. Storage traffic requires special treatment from an Ethernet network because traditional Ethernet does not provide some of the features such as congestion management and zero loss that higher level storage protocols expect the fabric to provide.
ii. Each CNA can be provisioned into multiple partitioned CNAs, where each partitioned CNA appears as a standalone CNA within an OS or hypervisor running on a host. A physical partition on a CNA refers to a PCI physical function (PF) [11].
iii. An SR-IOV CNA is also capable of exposing PCI virtual functions. A PCIe Virtual Function (VF) is an independent, isolated, secure and "light-weight" PCIe function that is directly accessible by a virtual machine. Each PF supports one or more VFs. A trusted software intermediary (e.g., Hypervisor) configures VF setup and assignment.
iv. Each one of these CNA physical or virtual functions will potentially require separate policies depending on their function
v. Each VM can run a full featured OS and has to be configured and managed as such. Since one physical machine can support multiple VMs, network, security and storage policies should now be applied to each VM on this physical network port
vi. Additionally, as VMs are dynamically migrated across the data center, their policies have to be dynamically migrated to the destination switch. Such movement occurs much more frequently than physical host moves, and as such it is impossible to manage the network policies for migrating VMs manually.
vii. In order to provide network management of various VMs hosted by a single hypervisor running on a single physical host, the hypervisors provides a virtual switch that provides connectivity between the various VMs running on the same physical host. The external edge switch has no insight into the management or control of this hypervisor virtual switch.

Hence a port on the edge switch no longer identifies a unique machine because multiple VMs and/or multiple partitions of a CNAs are now connected to a single port of the switch.

3. VMready – A Server Virtualization Aware Network Switch

We implemented a mechanism whereby the edge network switch, and furthermore, the upstream network switches, become aware of the virtualized and dynamic nature of the physical hosts in the data center. Such a virtualization aware network switch, henceforth called a VMready switch, would at a minimum provide a virtualized-host view of network I/O to the network administrator, where all network, security, and storage policies can be performed at the virtual port granularity instead of the physical port, and hence, physical switch port granularity.

Consider Figure 1 below, which shows a networking fabric (a set of network switches) connected to physical or virtual end-hosts, or host adapter extensions (network switch ports that behave like end-host ports).

A virtual port (v-port in short) is defined as a logical subdivision of a physical network port. A virtual port could be defined on a single physical port as shown in Figure 1, based on:

i. Association with a different type of traffic on a CNA such as storage (FCoE, iSCSI) or network traffic.
ii. Association with a VM network adapter or a VM storage adapter.
iii. Association with a physical or virtual function (partition) of a CNA.
iv. Association with a host I/O extension port of a network switch. Host I/O extension ports

[2] Network policies are network layer 2 and 3 switching configuration on the switch, including, but not limited to, VLAN configuration, multicast configuration, QoS and bandwidth management policies, ACLs and filters, security and authentication policies, load balancing and traffic steering configuration, and redundancy and failover configuration.

appear as end host ports (physical or virtual) for a set of back-end servers. For example, a set of Fibre Channel (FC) ports on an FC switch can proxy as host FC extension ports for a set of FC storage adapters through a mechanism known as N_Port ID Virtualization (NPIV) [7].

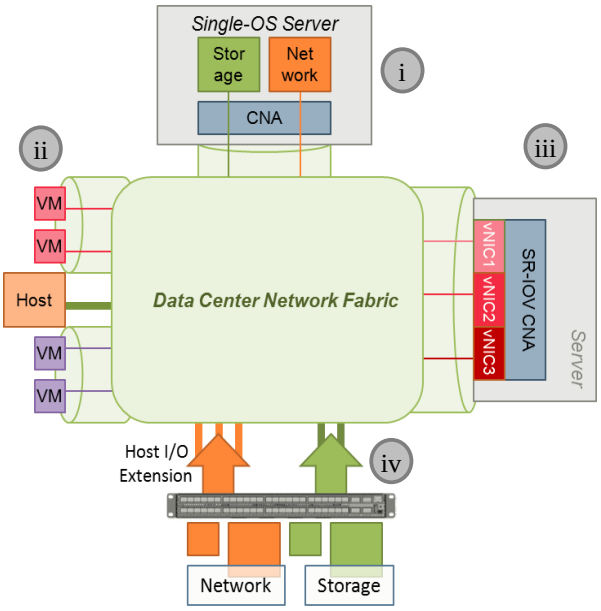


Figure 1: Virtual Ports for Switch Fabric

The VMready switch discovers, instantiates, and displays such defined virtual ports to be used in policy definitions by the network administrator as shown in the Figure below.

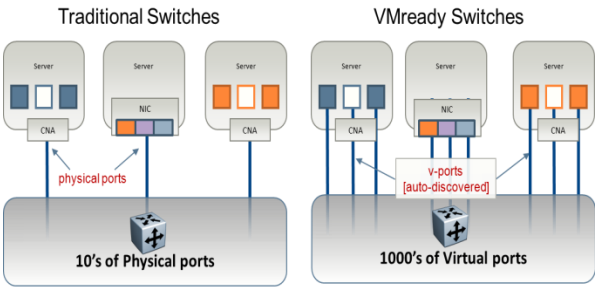


Figure 2: Network Administrator View of a VMready Switch

With such an explosion of virtual ports in the data center, the creation and management of policies per virtual port would become an arduous task for the network administrator. Moreover, a typical network, storage or security policy is usually applied based on workload characteristics. For example, all database workloads (physical or virtual) for a particular application would inherit the same end-host port-level policy, and therefore can be *grouped* together in a profile, say the "database profile". Similarly, all web server physical or virtual ports could be grouped together in a "web server profile".

3.1 Managing a VMready Switch

Because of the automated discovery and instantiation of virtual ports, the VMready switch provides a single pane of management from which a network administrator can configure policies for network-side physical ports, VM network adapters or VM storage adapters, regardless of the physical location of the VM and the partitions on the CNAs. It does this by:

- Enabling grouping of virtual ports with similar function (e.g., database workloads in one group; test workloads in another group; web server workloads in a separate group). Administrators would define common Layer 2 or Layer 3 network policies within a profile, e.g., higher Quality of Service (QoS) and better Layer 2 redundancies for the "database profile" versus best-effort QoS and no Layer 2 redundancy for the "web server profile".

- Enabling profiles to be applied per group of virtual ports, especially VM storage or network ports, regardless of the physical location of the VM. This enables the source VMready switch to migrate these profiles automatically to the target VMready switch when a VM migration event occurs in the data center.

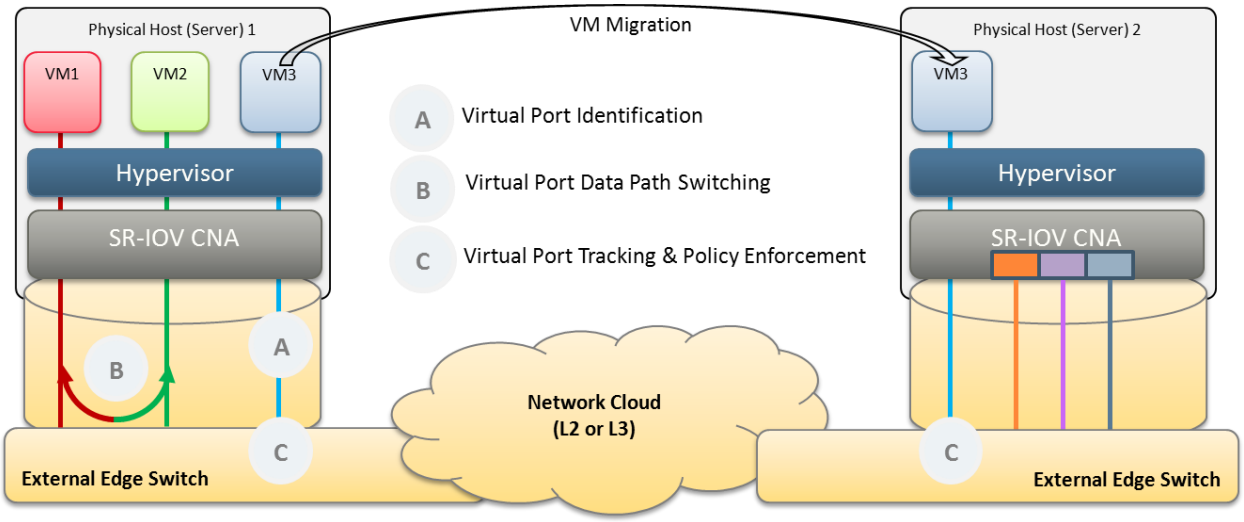
- Providing a service-level application-aware health check framework for the *group* to provide failover and redundancy at the workload level and not just the physical port level

3.2 Data Center wide VMready

Whereas virtual port profiles and *group* policy configurations can exist as part of a switch's database[3], an external database-based approach is extensible across the whole data center, and potentially across data centers. VM Remote Access Database (VMrad) Extensions to a Network Management System (NMS) database enabled such profile-based policy enforcement on VMready switches across a multi-

[3] Current VMready switch-resident profile databases extend across the domain of an 8-10 unit virtual switch chassis.

Figure 3: Key Issues to Solve to Enable Virtual Port Switching



vendor data center network, as shown in the Figure below. The Figure shows two key components – the VM management server, and the central policy database. The VM management server allows an administrator to create, start, stop and migrate VMs across the datacenter. The central policy database contains the network profiles of every virtual port in the datacenter, and the mapping between the profile and the VM(s) it is assigned to.

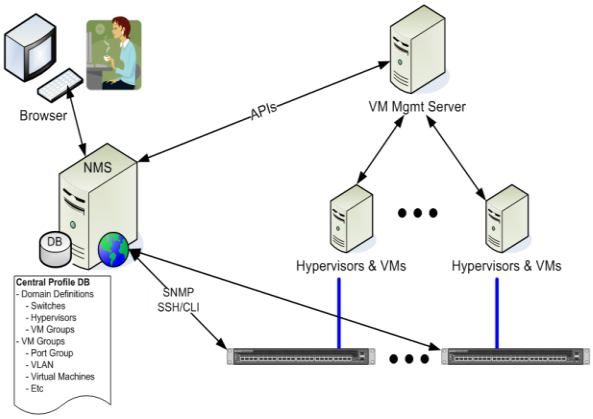


Figure 4: VMrad - Data Center wide VMready

By keeping the profiles in a central database, VMready switches can synchronize virtual port policies with this database as and when virtual ports are created or moved dynamically across the data center.

4. Implementation Details

Consider Figure 4, which shows two physical servers (1 and 2) with 3 VMs on Server 1, and none on Server 2. These servers are connected to an edge physical network switch, such as a blade or top-of-the-rack switch, which themselves connect to each other via a network cloud. These edge network switches are VMready enabled. There are three key problems to solve when building a virtual port switching framework, as shown in Figure 4:

A. Identifying, discovering and instantiating virtual ports on the VMready switch
B. Enabling data path switching between the virtual ports
C. Applying defined policy profiles to virtual ports as they are dynamically created and migrated

There is standardization work under way as part of IEEE 802.1Qbg [2] and 802.1Qbh [3], and as part of the DMTF [10], that attempts to define a base level specification for doing parts A through C. Since VMready predates this standardization work, we will describe how VMready solves these three problems today, and in the following Section, describe how the standardization work will influence future versions. Additionally, there are parts of the solution that are not covered by any standardization work.

A. Virtual Port Identification

Since there was no standard way of hypervisors and CNAs to communicate the dynamic creation and migration of virtual ports to external edge switches, we could only rely on one or more of the following three mechanisms to discover virtual ports:

a. By passively snooping the traffic between the hypervisors and the VM management stations to discover the VMs and hypervisors on the physical port. Typically, this would be done via learning the VM-specific or hypervisor-specific MAC addresses on the physical host port.

b. By querying the attributes of the VMs and hypervisors (UUIDs, MAC addresses, IP addresses, VM names etc.) with the VM management console.

c. By implementing proprietary versions of data center protocols such as Data Center Bridging capability eXchange (DCBX) protocol between the VMready edge switch and the virtualization provider on the host.

Mechanism (c) has been used in the VMready switch to discover FCoE virtual ports as well as virtual ports due to partitioned CNAs.

Mechanisms (a) and (b) have both been used to discover virtual ports due to VM network and storage adapters. Mechanism (b) has been implemented by enabling a hypervisor-vendor-specific (HV) agent on the VMready switch that communicates with the VM management consoles via web services APIs and transports offered by VMware vCenter or Microsoft System Center.

Mechanism (b), in addition to discovering VM virtual ports, also allows for a secure method of cross-checking the validity of a VM's traffic and hence avoiding a MAC-spoof attack. It is also used to provide a richer user experience for network administrators on the VMready switch by correlating the information presented on each virtual port with the information available at the VM management console.

As more mechanisms become available, or as parts of these mechanisms get standardized, the virtual port identification layer will need to be adjusted. Expecting this requirement, the VMready software architecture (Figure 4) implemented a *Virtual Port Identification (VPI)* Plugin (*) layer, where additional plugins could be added over time for the discovery and instantiation of virtual ports to the upper switching layers. Placing a Virtual Port Switching Abstraction layer (**) above the VPI enabled us to keep the switching protocols intact without much change.

B. Data Path Switching

There were three critical changes made on both the control plane as well as the data plane VMready switch for implementing virtual port data path switching.

First was the implementation of a virtual port switching layer, both in the switch silicon as well as the switching software. On the VMready switch, this layer (Figure 5) was implemented in a way so as to not disrupt the protocol code sitting above it. Both control and data planes are required to implement this layer since protocol and switching decisions are made at both places:

- Link Aggregation Groups could be formed and traffic can be load-shared between a set of virtual ports

- Multicast groups can be formed per virtual port, where certain virtual ports on the same physical port can be part of the group while others are not.

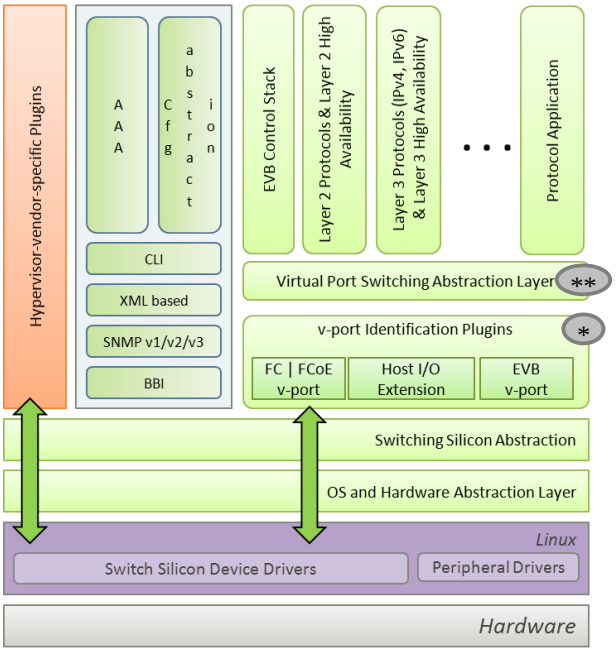


Figure 5: VMready Software Architecture

Second was the extension of the hypervisor-vendor-specific (HV) agent to configure the hypervisor resident soft switch and maintain consistency between the policies applied on the VMready switch and the soft switch.

Third was the enablement of a *hairpin* mode of operation (Reflective Relay) on the switch silicon physical port. This mode is enabled for only those port and hypervisor vendors for which our VMready HV agent can configure the soft switch to send all VM traffic out to the edge switch.

C. Virtual Port Tracking and Policy Enforcement

By virtue of implementing the VPI plugin layer, a VMready switch could easily learn about the creation and migration of virtual ports. This dynamic learning is then verified via the HV agent communicating with the VM management console. Once verified, administrator defined policy profiles can be applied to the newly created or migrated virtual port. As discussed in Section 3, the policy can be enforced within an 8-10 unit virtual switch stack without the use of an external database, or across the data center using the VMrad external database.

5. Implementation Issues and Tradeoffs

Since VMready was implemented in the 2007 timeframe, there wasn't much available in terms of merchant network switching silicon support for virtual ports. We had to mimic each virtual port or a group of virtual ports, via the use of S-VLANs or Q-in-Q stacked VLANs [6]. Switching decisions in the hardware were then made to operate on these stacked VLANs rather than physical ports on customer VLANs.

As the space has matured, so has the support for virtual ports in merchant switching silicon. Even so, there are a few areas where feature support for virtual ports will never match the feature support for physical ports, simply due to the limitations imposed by cost, real estate and power. First and foremost, the number of virtual ports supported per silicon will have capacity limitations. Additionally, features such as bandwidth metering are areas which require significant silicon expense, especially as the number of virtual ports per physical port grows in the data center, and switching silicon becomes denser in terms of number of physical ports supported. Tradeoffs will be made whereby certain virtual ports can be metered individually, whereas other virtual ports can only be metered *in-bulk* across a *group* of virtual ports. Switch vendors will have to expose this tiered handling, and network administrators will have to be cognizant of this differentiation.

Another area that is slowly maturing is the soft switch implementations in the hypervisors and the associated management interfaces to these entities. Some vendors provide comprehensive APIs to configure and manage these entities while others are still building up their interfaces. This diversity has resulted in VMready having a richer user experience with some hypervisors. Moreover, even if these interfaces exist, the capabilities of the soft switch itself varies significantly from vendor to vendor, giving rise to network-vendor-specific soft switches, such as the Cisco 1000v [4]. Again, as the space matures and becomes pertinent, the landscape is changing for the better.

Finally, even though the definition of virtual ports in VMready, as discussed in Sections 3 and 4 have a common underlying framework, the actual mechanisms to instantiate one differs based on the type of the virtual port. This would be true even after the standardization work in IEEE 802.1Q is complete, simply because any standard only goes so far as to define a common base specification. Implementation of any variation to this specification is usually left to the system vendor. The VMready VPI plugin layer has been specifically designed to handle such variations in virtual port discovery and instantiation.

6. Acknowledgments, Related Work and Future Directions

VMready was initially developed by Blade Network Technologies over the late-2007 to mid-2008 time period for VM-specific virtual ports across a virtual switch stack. Over the next 2 years, support was added for cross data center policy enforcement via VMrad, and for the definition of virtual ports for FCoE traffic and for partitioned CNAs. The first implementation of FCoE virtual ports was enabled with QLogic CNAs, while the first implementation of partitioned CNAs was enabled with Emulex CNAs, both under partnership with IBM [8]. All the pieces of VMready described in this paper, except for the future work identified, is currently shipping as a software load on BLADE Network Technologies 10G Ethernet switches.

The problem of switching between virtual ports has been tackled by other vendors too, namely, Cisco's VN Link technology solves the problem using a combination of the 1000v soft switch and the VN Tag Ethernet tag [9].

There is standardization work around VM-specific virtual port switching in the IEEE as 802.1Qbg [2] and 802.1Qbh [3]. We can map the key issues to solve (A, B and C) highlighted earlier to the protocols being developed in the IEEE:

A. VM discovery and identification is provided using the Channel Discovery and Control Protocol (CDCP).
B. Data path switching is assisted via a Virtual Ethernet Bridge (VEB) or a Virtual Ethernet Port Aggregator (VEPA) entity on the host. A soft switch in the hypervisor is a VEB.

C. Virtual Station Interface (VSI) Discovery [and Control] Protocol (VDP) provides profile-based policy enforcement mechanism across the data center.

VMready is being extended to comply to the IEEE 802.1Qbg version 0 specifications [2]:

(A) With the use of S-VLANs to identify virtual ports, VMready is already compliant to the way CDCP builds virtual ports. Enhancements will have to be made in the LLDP code to enable CDCP Type-Length-Value (TLV) parsing.

(B) With its HV agent communicating with the hypervisor soft switch, VMready already complies with the VEB mode of operation for data path switching. We are adding support for more soft switches.

We are also working with a partner who is enabling the VEPA mode of operation in their hypervisor. The switch side functions to support this, such as Reflective Relay (RR), S-VLAN based virtual port creation and profile management, and EDCP for enabling VEPA and RR modes, are being expanded upon. The other aspects of IEEE 802.1Qbg, such as profile-based configuration, are common between both VEPA and VEB modes of operation.

(C) Enhancements are currently in progress to enable VEPA mode. (C) VMrad provides a profile database driven policy enforcement on VMready switches today. Work is currently underway to change the underlying handshake method to VDP.

7. Summary and Conclusions

This paper describes the implementation of VMready – a virtual port switching framework. Through this paper we describe how we solved the three key issues of enabling virtual port switching, namely, (A) discovering, identifying and instantiating virtual ports on the switch based on storage or network traffic classification, VM-resident adapters, and partitioned CNAs; (B) enabling data traffic switching between virtual ports which are either co-located on the same physical port, or are located across different physical ports; and (C) creation and enforcement of virtual port network, storage and security policies as these ports are dynamically created and migrated in the data center.

As we walk the reader through the implementation overview, architecture and details, we also enumerate the practical limitations and tradeoffs in implementing such a framework. Finally, we discuss related work in the industry, the efforts under way towards standardizing key parts of the solution, and how VMready will adapt to these directions moving forward.

8. References

[1] BLADE Network Technologies, *"VMready : A Virtualization Aware Network Switch"*, http://www.bladenetwork.net/vmready.html

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

[3] IEEE 802.1, *"802.1Qbh – Bridge Port Extension"*, http://www.ieee802.org/1/pages/802.1bh.html

[4] Cisco Systems, *"Cisco Nexus 1000v Series Switches"*, http://www.cisco.com/en/US/products/ps9902/

[6] IEEE 802.1ad Specifications, http://www.ieee802.org/1/pages/802.1ad.html

[7] T11, "NPIV Functionality Protocol", ftp://ftp.t11.org/t11/member/fc/da/02-340v1.pdf

[8] IBM, *"Virtual Fabric for the IBM BladeCenter"*, http://www-07.ibm.com/systems/in/bladecenter/hardware/openfabric/virtualfabric.html

[9] Cisco Systems, *"Cisco VN-Link: Virtualization Aware Networking"*, http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns892/ns894/white_paper_c11-525307_ps9902_Products_White_Paper.html

[10] DMTF, *"Distributed Management Task Force"*, http://www.dmtf.org/home

[10] PCI-SIG, *"PCI-SIG I/O Virtualization Specifications"*, http://www.pcisig.com/specifications/iov/.