PCI-SIG SR-IOV Primer

An Introduction to SR-IOV Technology

Intel® LAN Access Division

321211-002

Revision 2.5

January 2011

Legal

Revisions

Date	Revision	Description
April 2008	1.0	Initial release. Confidential.
December 2008	2.0	Confidential status removed.
January 2011	2.5	Updated Material.

Contents

1 Introduction

IBM* introduced virtualization on the VM/370 mainframe 30 years ago as a way to run multiple environments on one system. At the time, computing resources were expensive and allowing multiple environments to coexist meant that businesses didn't have to invest in a second mainframe.

Today there is a different motivation driving the trend toward virtualization. With IA-based (Intel Architecture) servers, the price structure has changed so much that hardware cost is no longer the issue. Bigger problems are physical space in the data center, power/cooling costs, and management. The performance of the IA-based servers has also increased to a point where there is a large amount of idle capacity.

With virtualization, IT departments can utilize that spare capacity rather than adding a new physical server to support a new environment.

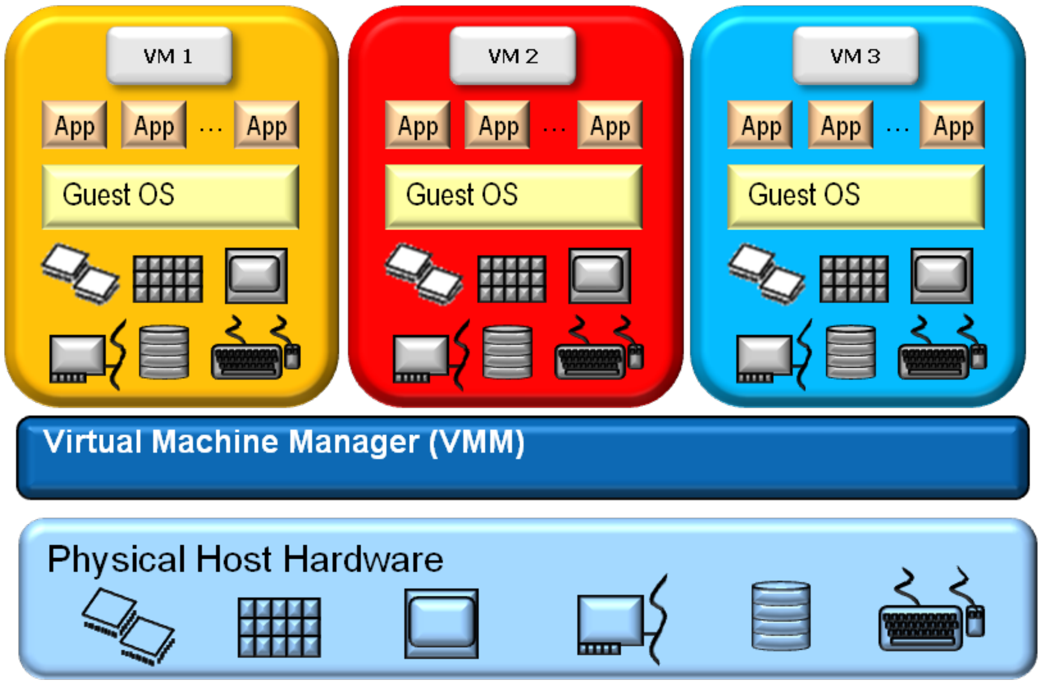

Figure 1. Common Virtualized Environment

Adding the virtualization layer introduces overhead that consumes some of the idle capacity. There has been significant effort in the computing industry to address this overhead. Intel has introduced a set of processor, chipset and Ethernet Controller enhancements defined under Intel Virtualization Technology (Intel® VT) toward this end. Intel® VT improves the performance, efficiency, and robustness of virtualized platforms.

The latest trend is to develop complementary mechanisms in the I/O subsystem. The PCI-SIG* has developed the *Single Root I/O Virtualization Specification (v1.1)* to address sharing of I/O devices in a standard way.

2 Background

2.1 Virtualization Trends

There are numerous trends in virtualization that impact the I/O Virtualization solutions:

- Platform performance is increasing
    - Multi-Core CPU’s are perfect for virtualization
    - Many platforms are underutilized.
- Virtualization increases the utilization of hardware resources by consolidating multiple workloads onto one physical machine
- I/O is getting faster
    - 10GbE is rapidly gaining popularity
- The number of virtual machines is increasing
    - Many have one Virtual Machine per physical CPU Core
- The performance/watt ratio is of growing importance
    - Everybody wants to be ‘Green’ and reduce power consumption & costs

2.2 I/O Virtualization Goals

I/O virtualization solutions need to provide the same isolation that was found when the environment was running on a separate physical machine. Solutions need to provide scalability to support the number of virtual machines (VMs) necessary to take advantage of idle resources. They should also provide near native performance for I/O operations.

Isolation should provide separation of memory space. Isolation also includes the need to separate I/O streams, interrupts, and (in the case of shared devices) the ability to isolate control operations, I/O operations and errors.

With software sharing approaches, isolation is enforced in by the virtualization layer. For shared hardware devices where the virtualization layer is bypassed for I/O, other mechanisms have to enforce isolation.

For software sharing, scalability is a function of the virtualization layer at the cost of CPU utilization. For shared hardware, cost is potentially weighted by the necessity of additional hardware for I/O.

2.3 I/O Sharing Approaches

I/O Virtualization (IOV) involves sharing a single I/O resource between multiple virtual machines. Approaches for IOV include models where sharing is done using software, where sharing is done in hardware, and hybrid approaches.

2.3.1 Software-Based Sharing

Software based sharing utilizes emulation techniques to provide a logical I/O hardware device to the VM. The emulation layer interposes itself between the driver running in the guest OS and the underlying hardware. This level of indirection allows the VMM to intercept all traffic issued by the guest driver.

Emulation software can parse the I/O commands, translate guest addresses into host physical addresses, and ensure that all referenced memory pages are present in memory. Software must resolve the multiple I/O requests from all the virtual machines and serialize them into a single I/O stream that can be handled by the underlying hardware.

Common approaches to software-based sharing are (1) device emulation and (2) the split-driver model:

1. Device emulation models mimic widely supported real devices (such as an Intel 1Gb NIC) and utilize existing drivers in the guest OS. The VMM emulates the I/O device to ensure compatibility and then processes I/O operations before passing them on to the physical device (which may be different). A potential problem is that I/O operations then have to traverse two I/O stacks, one in the VM and one in the VMM.
2. The split-driver model takes a similar approach but, instead of emulating a legacy device, the split-driver uses a front-end driver in the guest that works in concert with a back-end driver in the VMM. These drivers are optimized for sharing and have the benefit of not needing to emulate an entire device. The back-end driver communicates with the physical device.

2.3.1.1 Drawbacks to Software-Based Sharing

Both the device emulation and split-driver (para-virtualized driver) provide a subset of the total functionality provided by physical hardware and may as a result not have the ability to take advantage of advanced capabilities provided by the device.

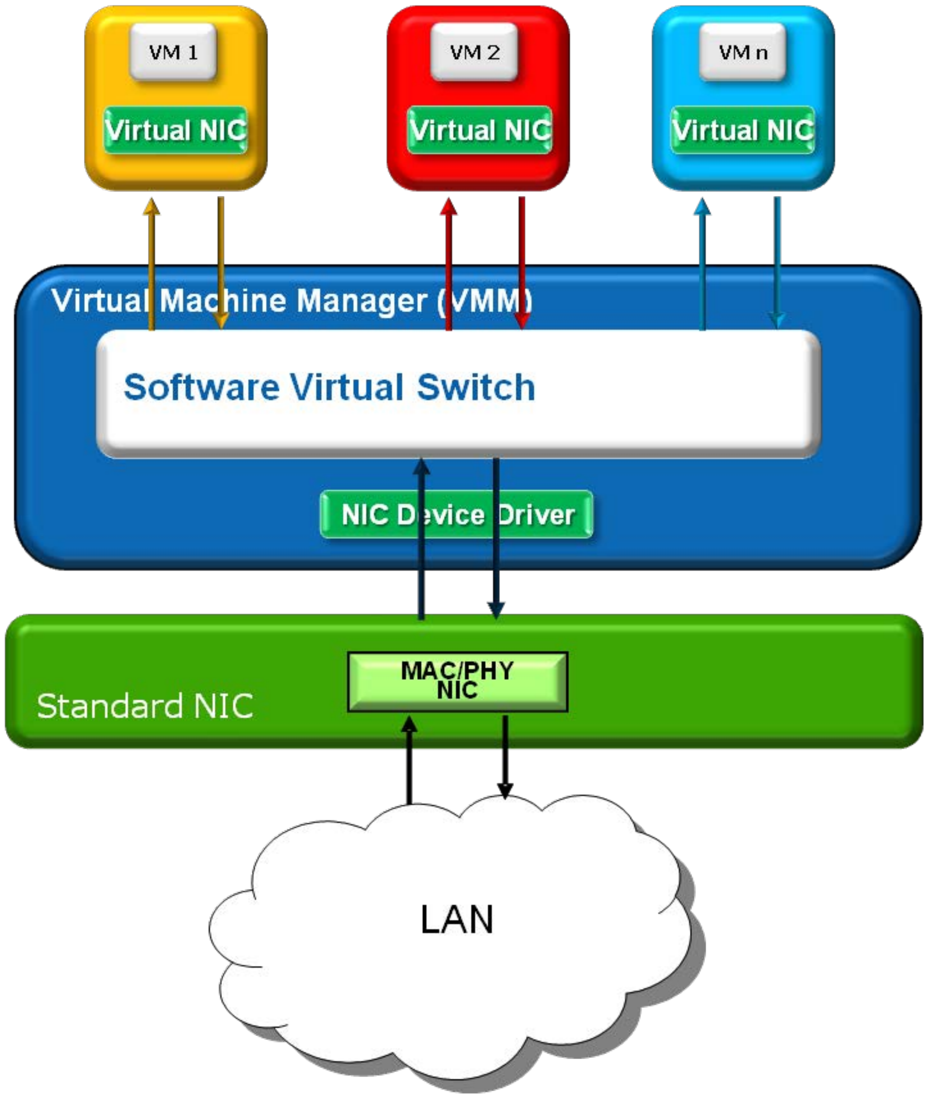

Figure 2. Software Based Sharing

In addition, significant CPU overhead may be required by the VMM to implement a virtual software-based switch that routes packets to and from the appropriate VMS. This CPU overhead can (and generally does) reduce the maximum throughput on an I/O device. As an example, extensive testing has shown using only device emulation, a 10Gbps Ethernet controller can achieve a maximum throughput of 4.5 to around 6.5 Gbps (the range varies with the architecture of the server being tested on).

One reason that line rate, or near line rate cannot be achieved is because each packet must go through the software switch and that requires CPU cycles to process the packets.

2.3.2 Direct Assignment

Software-based sharing adds overhead to each I/O operation due to the emulation layer between the guest driver and the I/O hardware. This indirection has the additional affect of eliminating the use of hardware acceleration that may be available in the physical device. Such problems can be reduced by directly exposing the hardware to the guest OS and running a native device driver.

Intel has added enhancements to facilitate memory translation and ensure protection of memory that enables a device to directly DMA to/from host memory. These enhancements provide the ability to bypass the VMM's I/O emulation layer and can result in throughput improvement for the VMs.

See Figure 3. A feature of Intel® VT-x technology allows a VM to have direct access to a physical address (if so configured by the VMM). This ability allows a device driver within a VM to be able to write directly to registers IO device (such as configuring DMA descriptors). Intel® VT-d provides a similar capability for IO devices to be able to write directly to the memory space of a virtual machine, for example a DMA operation.

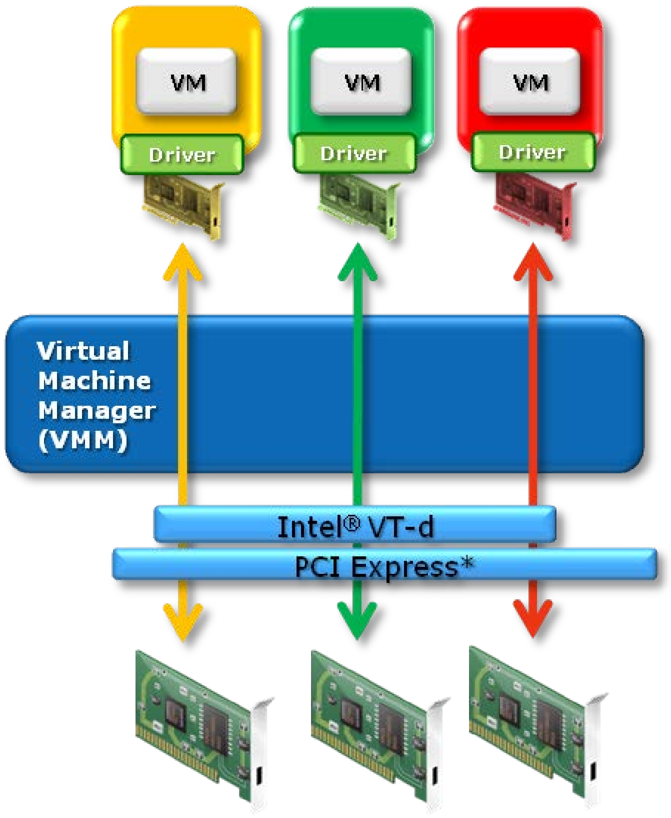

Figure 3. Direct Assignment

The mechanism for doing direct assignment varies from one vendor to another. However, the basic idea is that the VMM utilizes and configures technologies such as Intel® VT-x and Intel® VT-d to perform address translation when sending data to and from an IO device.

2.3.2.1 Drawbacks to Direct Assignment

One concern with direct assignment is that it has limited scalability; a physical device can only be assigned to one VM. For example, a dual port NIC allows for direct assignment to two VMs (one port per VM). However, there is still a fundamental limit to the number of I/O devices that can be placed in one system.

Consider for a moment a fairly substantial server of the very near future – it may have 4 physical CPU’s, with say 12 Cores per CPU. If you use the rule of thumb of one VM per core, that is potentially 48 VM’s running. If you wanted to have Direct Assignment to each of those VM’s, you would need 48 physical ports.

3 PCI-SIG Single Root I/O Virtualization and Sharing

Current I/O virtualization techniques have their advantages and disadvantages. None are based upon any sort of industry standard.

The industry recognizes the problems of alternative architectures and is developing new devices that are natively shareable. These devices replicate resources necessary for each VM to be directly connected to the I/O device so that the main data movement can occur without VMM involvement.

Natively shared devices will typically provide unique memory space, work queues, interrupts, and command processing for each interface they expose while utilizing common shared resources behind the host interface. These shared resources still need to be managed and will typically expose one set of management registers to a trusted partition in the VMM. See Figure 4.

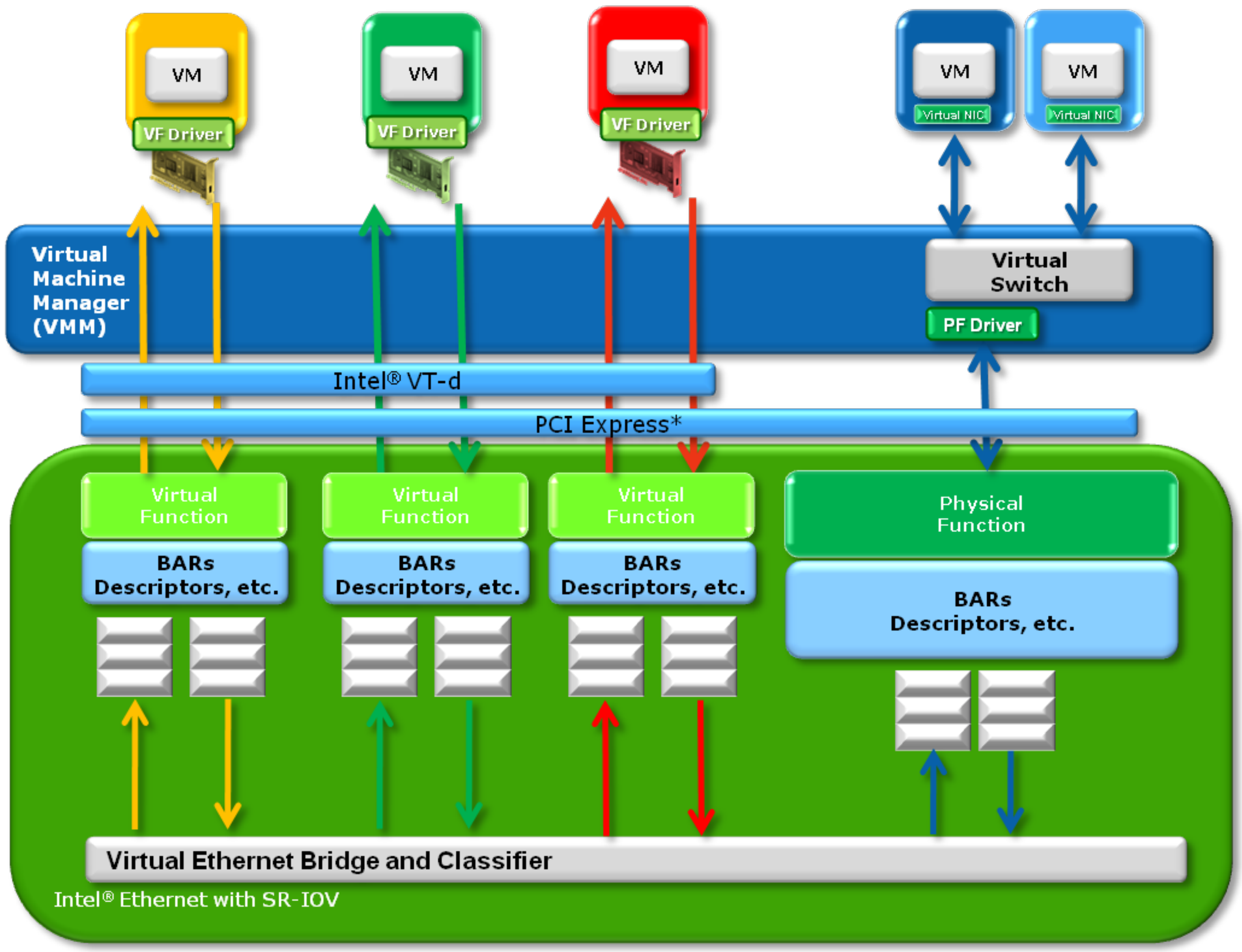

Figure 4. Natively and Software Shared

By having separate work queues and command processing, these devices are able to simultaneously receive commands from multiple sources and merge them together before passing them to the secondary fabric (for example, an Ethernet or SAS link). The virtualization software no longer has to multiplex the I/O requests into a serial stream.

Natively shared devices can be implemented in numerous ways, both standardized and proprietary. Since most of these devices are accessed over PCI, the PCI-SIG decided to create a standard approach. The PCI-SIG *Single Root I/O Virtualization and Sharing* (SR-IOV) specification defines a standardized mechanism to create natively shared devices.

Figure 4 shows a possible example configuration where three VM's are natively (directly) accessing dedicated resources within the Ethernet controller via Virtual Functions, while at the same time the Physical Function also has its own resources and can still be used for the emulated path as described in Section 2.3.1.

3.1 SR-IOV Goals

The goal of the PCI-SIG SR-IOV specification is to standardize on a way of bypassing the VMM's involvement in data movement by providing independent memory space, interrupts, and DMA streams for each virtual machine. SR-IOV architecture is designed to allow a device to support multiple Virtual Functions (VFs) and much attention was placed on minimizing the hardware cost of each additional function.

SR-IOV introduces two new function types:

- Physical Functions (PFs): These are full PCIe functions that include the SR-IOV Extended Capability. The capability is used to configure and manage the SR-IOV functionality.
- Virtual Functions (VFs): These are 'lightweight' PCIe functions that contain the resources necessary for data movement but have a carefully minimized set of configuration resources.

3.2 SR-IOV Overview

The Direct Assignment method of virtualization provides very fast I/O. However, it prevents the sharing of I/O devices. SR-IOV provides a mechanism by which a Single Root Function (for example a single Ethernet Port) can appear to be multiple separate physical devices.

A SR-IOV-capable device can be configured (usually by the VMM) to appear in the PCI configuration space as multiple functions, each with its own configuration space complete with Base Address Registers (BARs). The VMM assigns one or more VFs to a VM by mapping the actual configuration space the VFs to the configuration space presented to the virtual machine by the VMM. See Figure 5.

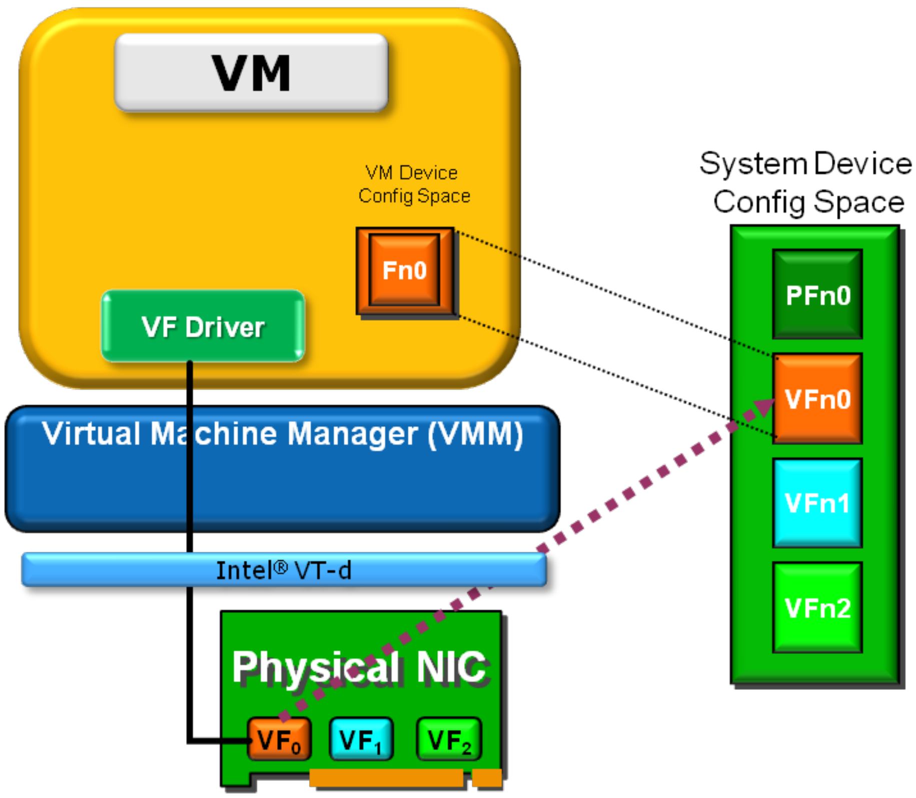

Figure 5. Mapping VF Configuration

SR-IOV-capable devices provide configurable numbers of independent VFs, each with its own PCI Configuration space. The VMM assigns one or more VF to a virtual machine. Memory Translation technologies such as those in Intel® VT-x and Intel® VT-d provide hardware assisted techniques to allow direct DMA transfers to and from the VM, thus bypassing the software switch in the VMM.

4 Related Technologies

This section introduces technologies that are complimentary to SR-IOV.

4.1 Alternative Routing-ID Interpretation (ARI)

Alternative Routing ID Interpretation (ARI) is an ECN (Engineering Change Notification) to the PCIe v2.0 (5GT/s); ARI provides a mechanism to allow a single PCI Express component to support more than eight functions. The ECN was approved in August 2006.

15 --------------------------------- 8	7---------------------------3	2-----------------0
Bus #	**Device #**	**Function #**

Table 1. Traditional Routing

ARI-compliant switches and endpoint devices reinterpret the Device Number field in the PCI Express Header as an extension to the Function Number field. This gives each component the ability to support up to 256 functions. I/O devices, switch ports and bridges upstream of ARI-compliant switches and endpoint devices must also implement ARI features for ARI to function.

15 ------------------------------------- 8	7 --- 0
Bus #	**Identifier**

Table 2. Alternate Routing

SR-IOV was created with the architectural headroom to support hundreds (if not thousands) of VFs and has the ability to allow devices to request the VMM to allocate more than one bus number so that the I/O component can scale beyond 256 functions.

ARI has been added to the PCIe v3.0 specification.

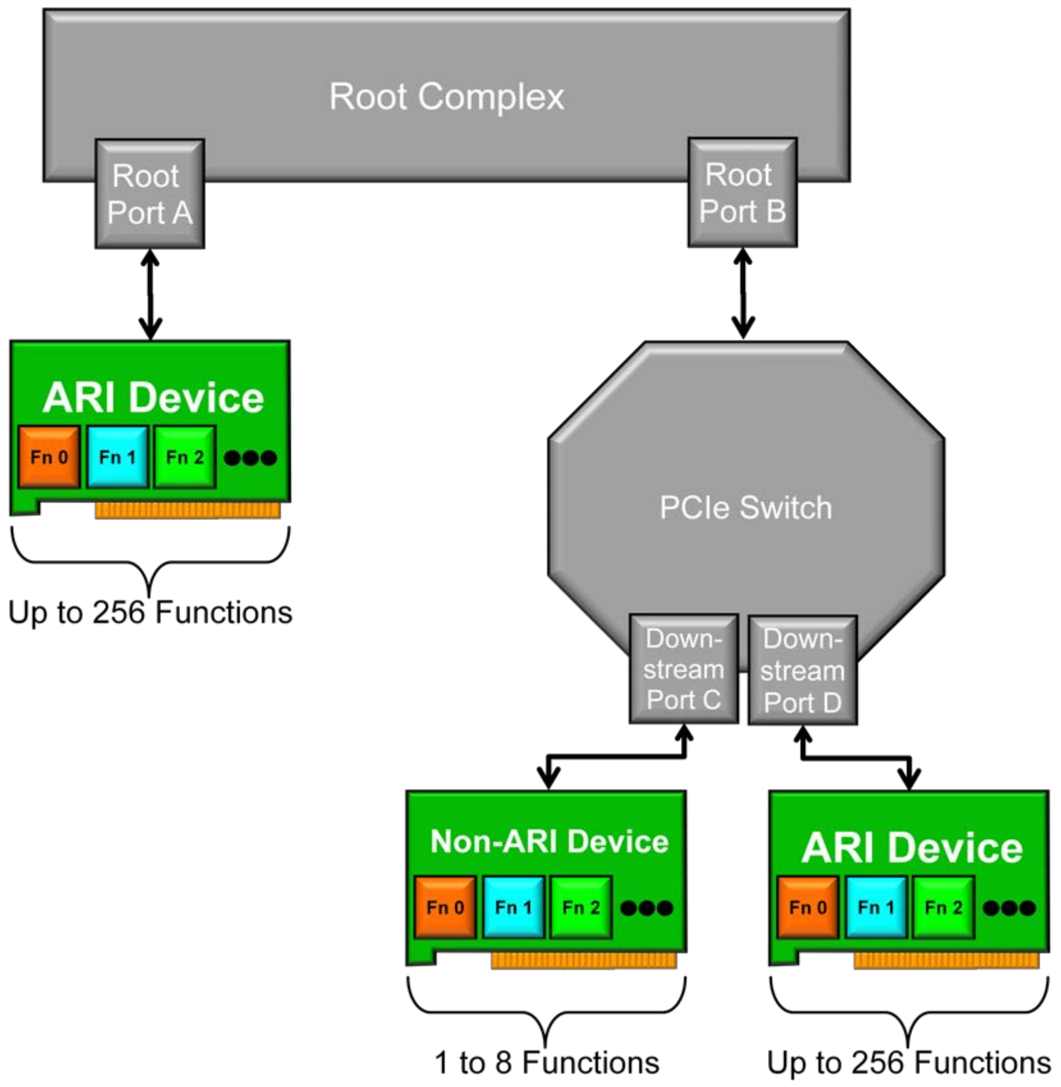

Figure 6 Example Topology with ARI and Non-ARI devices

4.2 Address Translation Services (ATS)

Address Translation Services (ATS) provides a mechanism allowing a virtual machine to perform DMA transactions directly to and from a PCIe Endpoint (in this case, an Intel Ethernet Controller). From a high-level view, this is done by utilizing look-up tables to map a virtual address that the VM is accessing (reading from or writing to) to a physical location.

ATS also allows a PCIe Endpoint to perform DMA transactions to memory locations in a virtual machine by using the same mechanism.

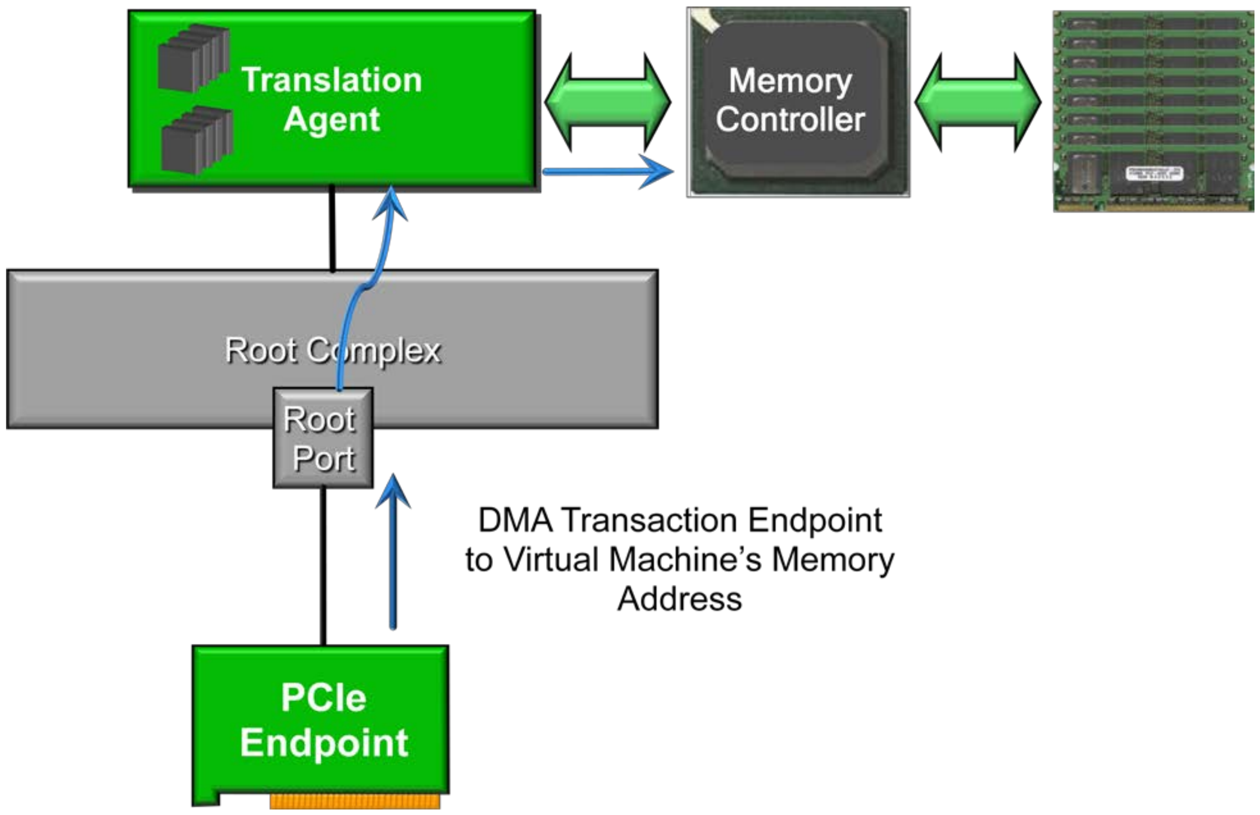

Figure 7 Address Translation Service

ATS is an optional capability and is not required for direct assignment of an I/O device to a VM. Intel chipsets that support Intel® VT-d typically size their translation caches large enough to accommodate the majority of workloads.

4.2.1 Intel® Virtualization Technology for Directed I/O (Intel® VT-d)

Intel® Virtualization Technology for Directed I/O (Intel® VT) provides a hardware based mechanism to translate addresses for DMA transactions issued by I/O devices.

This translation mechanism is important in a virtualized environment where the address space seen by the guest OS is not the same as the underlying physical address of the host machine. When a guest OS talks directly to an I/O device, it provides the Guest Physical Address (GPA) in the commands and buffer descriptors. The GPA is used by the I/O device when it issues the DMA transaction and must be translated to a Host Physical Address (HPA) so that DMA transactions can target the underlying physical memory page that has been configured as the DMA buffer.

Intel® VT-d utilizes the Address and Requestor ID (RID) in the PCI Express Transmission Layer Packet (TLP) as an index to a lookup table that is created by the VMM. The RID corresponds to one of the directly assigned functions and identifies the associated VM. By identifying the VM context and using VT-d tables, the chipset can translate the DMA address so that it targets the correct physical page and it can apply protection mechanisms to ensure that DMA operations can not affect memory space of unrelated virtual machines.

4.3 Access Control Services (ACS)

The PCIe specification allows for peer-to-peer transactions. This means that it is possible and even desirable in some cases for one PCIe endpoint (say a Virtual Function – or even a standard PCIe Function) to send data directly to another endpoint without having to go through the Root Complex.

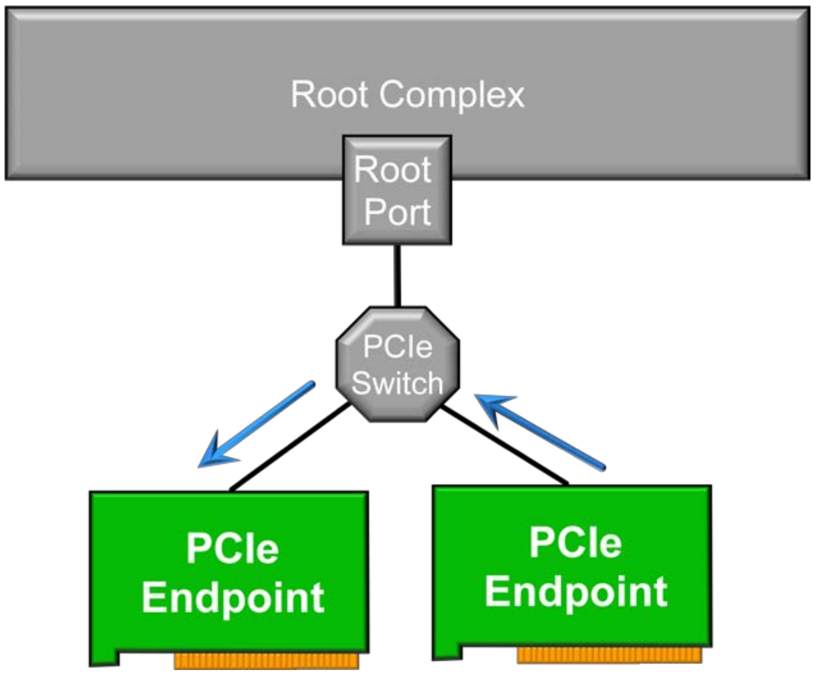

Figure 8 Peer-to-Peer PCIe Transaction

In the most recent generation of Intel Architecture servers, most designs do not have a PCIe switch between the PCIe Endpoint and the PCIe Root Complex. There are some PCIe Endpoint devices however that do have a PCIe Switch within them. One example of this are some quad-port Ethernet NICs, where the NIC is composed of two dual-port Ethernet devices connected via a PCIe switch on the NIC itself.

In a virtualized environment it is generally not desirable to have peer-to-peer transactions that do not go through the root complex. With both Direct Assignment (see Section 2.3.2) and SR-IOV, which is also a form of Direct Assignment, the PCIe transactions should go through the Root Complex in order for the Address Translation Service to be utilized.

Access Control Services (ACS) provides a mechanism by which a Peer-to-Peer PCIe transaction can be forced to go up through the PCIe Root Complex. ACS can be thought of as a kind of gate-keeper - preventing unauthorized transactions from occurring.

Without ACS, it is possible for a PCIe Endpoint to either accidentally or intentionally (maliciously) write to an invalid/illegal area on a peer endpoint, potentially causing problems.

Access Control Service began as an ECN to the PCI specification; it is now part of the PCI specification itself.

NOTE: The latest generation of PCIe switches include this capability – however many older ones do not. Intel has worked with the major VMM vendors so that they now probe the PCIe Endpoints and any PCIe switches between the Endpoint and the Root Complex. If there are any PCIe switches along this path that do not support ACS, the VMM will not enable Direct Assignment or SR-IOV for those Endpoints.

5 Eco-System Requirements

We will now take a closer look at each software entity and discuss how its existing operation is enhanced to support the new device architecture.

5.1 BIOS

The BIOS performs a role in partitioning Memory Mapped I/O and PCI Express Bus numbers between host bridges.

In many systems, mechanisms for allocating PCI resources to the host bridges are not standardized and software relies on the BIOS to configure these devices with sufficient memory space and bus ranges to support I/O devices in the hierarchy beneath each host bridge.

BIOS enumeration code needs to be enhanced to recognize SR-IOV devices so that enough MMIO (Memory Mapped IO) space is allocated to encompass the requirements of VFs. Refer to the PCI-SIG SR-IOV specification for details on how to parse PCI Config space and calculate the maximum amount of VF MMIO space required.

5.2 Virtual Machine Monitor (VMM)

SR-IOV defines new Virtual Functions that are lightweight versions of a traditional PCI Functions; it also introduces a new software entity called the Single Root PCI Configuration Manager (SR-PCIM or simply PCIM). SR-PCIM is responsible for configuration and management of the VFs. It is a conceptual model whose definition is outside the scope of the PCI-SIG efforts; the expectation is that this software will be incorporated into the virtualization layer by VMM vendors.

VFs implement a subset of traditional configuration space and share some configuration space with the associated Physical Function. It is the responsibility of SR-PCIM to intercept all configuration accesses and use the information in the Physical Function to present a complete PCI configuration model to a guest OS running in the VM. This section will cover some issues surrounding the use of SR-IOV. For details, refer to the SR-IOV specification.

Intel continues to work with the major VMM vendors to add support for SR-IOV. SR-IOV support in the Linux kernel has been available since the 2.6.30 release in June of 2009. Several distributions now have SR-IOV support.

5.2.1 Virtual Function (VF) Creation

At power on, VFs do not exist and can not be accessed through configuration space. Prior to accessing or assigning VFs, they must be configured and enabled through the SR-IOV capability located in associated Physical Functions.

The SR-IOV capability structure in the Physical Function PCI Configuration space includes a System Page Size field that the VMM should set to the size supported by the platform.

Memory Spaces for the VFs are concatenated together in a contiguous memory space located by the VF Base Address Register. In order to ensure isolation of the individual memory spaces, VFs must align their memory resource to the page protection boundaries provided by the system.

The ARI Capable Hierarchy (see Section 4.1) bit may impact the location of the maximum number of VFs. The VMM should enable the ARI capability in the Root Ports and Switches and set the ARI Capable Hierarchy bit in the SR-IOV capability. Note that BIOS may have already enabled ARI.

5.2.2 VF Discovery

Once the VF Enable field is set in SR-IOV, VFs are created and will respond to configuration transactions. They will not, however, be discovered automatically by legacy enumeration software.

A new mechanism exists in SR-IOV that allows SR-IOV compliant software to locate VFs. The First VF Offset and VF Stride create a linked list that starts at a Physical Function that can be used to identify the location of all the VFs associated with a particular Physical Function.

SR-IOV allows a device to implement 100s of VFs. To do this, a SR-IOV device may request the software to allocate more than one bus number (in order to support more that 256 functions).

Refer to the SR-IOV specification for details.

5.2.3 Support for VF Driver to Physical Function Driver Communication

VMMs can support a mechanism to create shared pages that enable communication between the VM Driver and the Master Driver (MD). See Section 6.

5.2.4 VF Assignment to Virtual Machines

Once VFs are created and configured, they can be assigned to a VM so that I/O operations can occur directly between a VM and hardware. The SR-IOV specification was written with the intent that all VFs on a device will be identical, meaning they all have the same PCI capabilities advertised in the PCI Configuration.

However, by having the Master Driver involved in the assignment of the VFs, the hardware may have the opportunity to provide additional features or differing levels of service depending on the needs of system administrators. By defining the appropriate mechanisms, the VMM could ask the Master Driver for a VF capable of meeting a specific level of service (for example, requiring 2Gbps of Ethernet performance).

6 Master Driver (MD)

The Master Driver (MD; also referred to as the Physical Function Driver or PFD, or PF Driver) is a specialized driver that manages global functions for SR-IOV devices and is responsible for configuring shared resources. MD is specific to the VMM and is expected to operate in a more privileged environment than a typical VM driver. MD contains all the traditional driver functionality to provide access to the I/O resource for the VMM; it can also be called upon to perform operations that impact the entire device. MD must be in a persistent environment, loaded before any VM Driver and unloaded (if necessary) after all VM drivers.

Device specific operations that globally affect all VMs should only be accepted from the MD. To achieve this, it is necessary for the driver in the VM (the Virtual Function Driver) to communicate control operations to the MD.

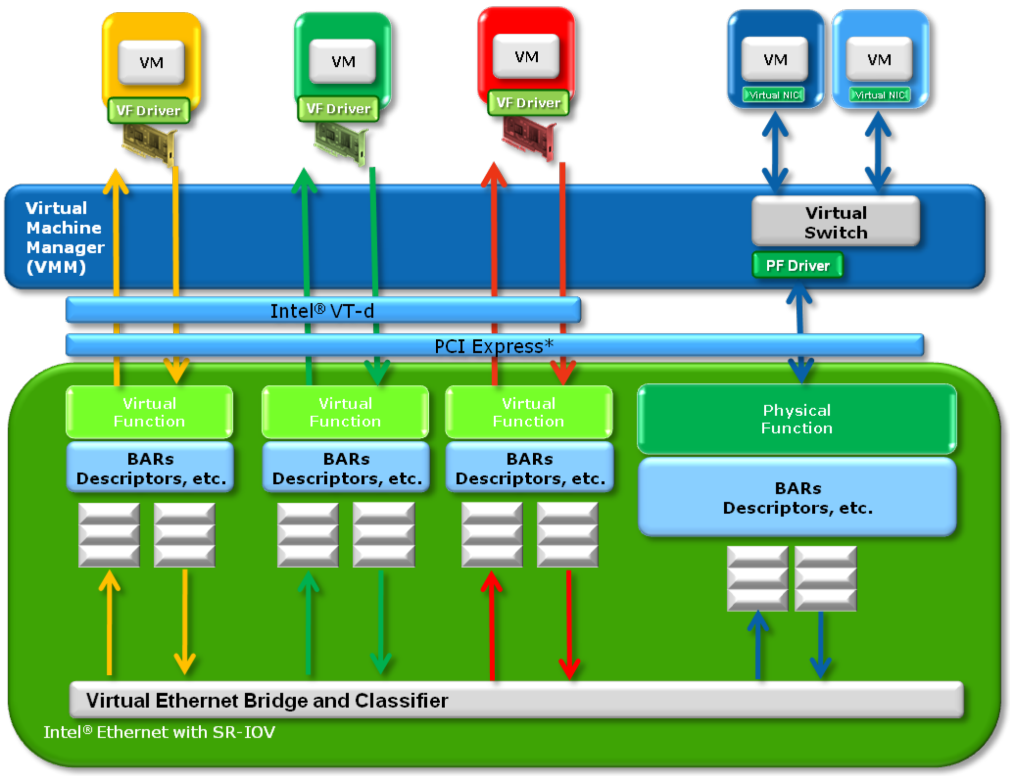

Figure 9 SR-IOV Blocks

An example of where this communication mechanism might be utilized is a link status or a MTU change for an Ethernet device. Suppose a VF Driver is used to communicate to the MD to determine the link status. At this point, the MD could return any state it chooses. If the MTU changes on the physical device, the MD could communicate this to each VF Driver so that the network stack could make appropriate changes. See Section 6.2 for more information on this communication path.

6.1 Virtual Function Drivers

The VF is a 'lightweight' PCIe function, containing the resources necessary for data movement. It is not a full-fledged PCIe device, basically it provides a mechanism to transfer data in and out. The VF driver residing in the VM should be a para-virtualized driver (aware that it is in a virtualized environment), and perform only the actions available to it.

In general the VF provides the ability to send and receive data and the ability to perform a reset. This reset only affects the VF itself, not the entire physical device. For actions that are beyond the VF reset or sending and receiving data, the VF driver needs to communicate with the Master Driver.

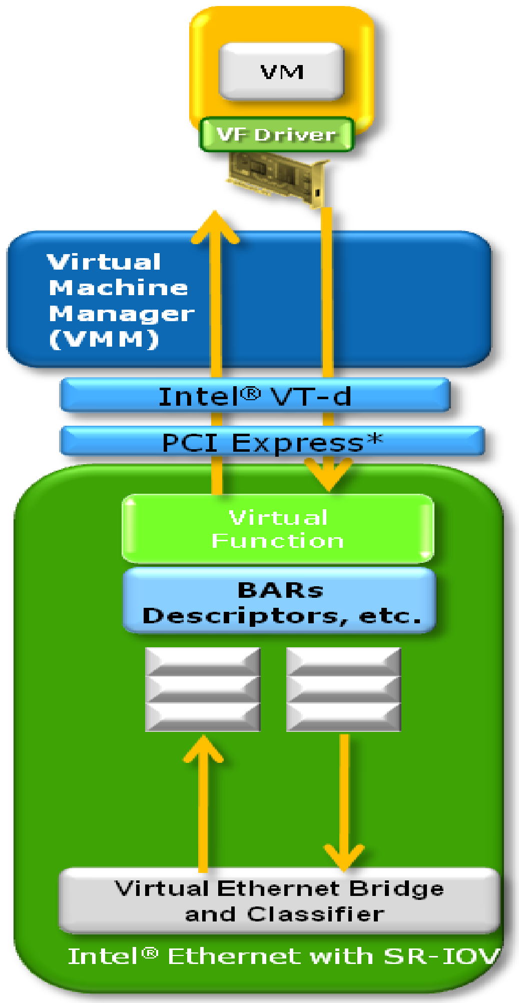
VM
VF Driver
Virtual Machine Manager (VMM)
Intel® VT-d
PCI Express*
Virtual Function
BARs Descriptors, etc.
Virtual Ethernet Bridge and Classifier
Intel® Ethernet with SR-IOV

Figure 10 VF Stack

The VF driver is a specialized driver – it ‘realizes’ that it has only certain functionality available to it, such as being able to configure the DMA Descriptors in the VF, configuring MAC addresse(s), VLAN tags, etc.

Each VF has its own dedicated resources within the I/O device. An Ethernet device for example will likely have dedicated Tx and Rx queues associated with dedicated BARs descriptors etc. The VF Driver will interact with and configure the BARs and descriptors.

6.2 VF Driver to Master Driver Communication

A necessary component for device sharing is the ability of the VF driver to communicate with the MD to request operations that have global effect. This channel needs this ability to pass messages and generate interrupts. SR-IOV does not define a mechanism for this communication path. It is the responsibility of the designer of the Master Function Driver, Physical Function Drivers and the VMM to create this communication path.

The simplest approach (and one supported by Intel SR-IOV capable Ethernet controllers) is to utilize a set of mailboxes and doorbells within the I/O device for each VF.

Many VMM Vendors are planning to implement this communication mechanism within the VMM. At this time, the communication path is specific to the vendor.

6.3 Theory of Operations

So how does this all work together? Let us walk through a potential example flow of how an incoming Ethernet packet is sent to a VM that has been assigned a VF from an Intel® Ethernet device.

1. The Ethernet packet arrives at the Intel® Ethernet NIC.
2. The packet is sent to the Layer 2 sorter/switch/classifier.
   - This Layer 2 sorter is configured by the Master Driver. When either the MD or the VF Driver configure a MAC address or VLAN, this Layer 2 sorter is configured.
3. After being sorted by the Layer 2 Switch, the packet is placed into a receive queue dedicated to the target VF.
4. The DMA operation is initiated. The target memory address for the DMA operation is defined within the descriptors in the VF, which have been configured by the VF driver within the VM.
5. The DMA Operation has reached the chipset. Intel® VT-d, which has been configured by the VMM then remaps the target DMA address from a virtual host address to a physical host address. The DMA operation is completed; the Ethernet packet is now in the memory space of the VM.
6. The Intel® Ethernet NIC fires an interrupt, indicating a packet has arrived. This interrupt is handled by the VMM.
7. The VMM fires a virtual interrupt to the VM, so that it is informed that the packet has arrived.

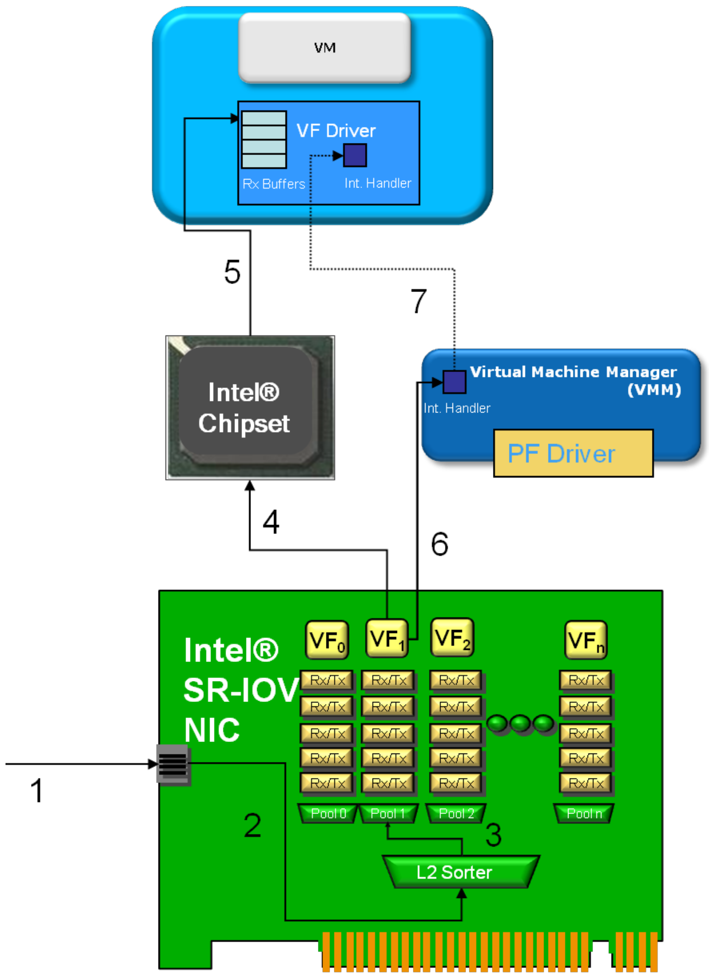

Figure 11 Flow of an incoming Ethernet packet

6.4 Summary

Many components within the system and software need to work in conjunction to provide a robust SR-IOV solution. The platform must have some sort of Address Translation Service support (such as the Intel® VT-d technology) to allow SR-IOV-capable devices to DMA data directly to the memory space of a virtual machine.

Some components in the system (the BIOS or VMM) must understand and configure the PCI configuration space. The mechanism to find VFs within the PCI configuration is new and different.

The VMM must work in conjunction with the Physical Function driver and the VF drivers to provide the necessary functionality, such as presenting PCI Configuration space to VMs and allowing a VM to perform PCI Resets on a VF.

Intel Corporation actively participated in the PCI-SIG forum in the development and ratification of the PCI-SIG SR-IOV* specification. All current and future generation Intel Server platforms include Intel® VT-d technology. Intel will continue support and improve Intel® VT-x technology.

The Intel 82576 Gigabit Ethernet Controller supports SR-IOV functionality, as does the Intel 82599 10 Gigabit Ethernet Controller. The first half of 2011 will see the Intel LAN Access Division release the next generation of both Gigabit and 10 Gigabit Ethernet Controllers for servers that also support SR-IOV.

Intel continues to work with the major Hypervisor* vendors to enable SR-IOV functionality. The Intel PF and VF drivers for Intel Gigabit and Intel 10 Gigabit Ethernet controllers are available as part of the more recent Linux* Kernel and also available on Source Forge*.

About the Author

Intel Technology Leader Patrick Kutch is a Technical Marketing Engineer (TME) for Intel Server I/O Virtualization and Manageability technologies. As a senior TME he works with customers, providing both educational materials and answering questions ranging from technical to architectural. He has worked in nearly every facet of Server Manageability, from contributing to the IPMI specification to writing management software over a span of 12 years at Intel. Patrick has split his time between Manageability and I/O Virtualization for the past 4 years. Patrick frequently blogs about his technologies at http://communities.intel.com/community/wired.

7 Additional Resources

PCI-SIG Single Root I/O Virtualization 1.1 Specification:
http://www.pcisig.com/specifications/iov/single_root

PCI-SIG Address Translation Services 1.1 Specification:
http://www.pcisig.com/specifications/iov/ats

PCI-SIG Alternative Routing-ID Interpretation (ARI):
http://www.pcisig.com/specifications/pciexpress/specifications/ECN-alt-rid-interpretation-070604.pdf

Intel Technology Journal - Intel® Virtualization Technology:
http://www.intel.com/technology/itj/2006/v10i3/1-hardware/5-architecture.htm

Intel® Virtualization Technology for Directed I/O (Intel® VT-d):
http://www.intel.com/technology/magazine/45nm/vtd-0507.htm

Intel® Ethernet SR-IOV Toolkit:
http://download.intel.com/design/network/Toolkit/322191.pdf

Intel® SR-IOV Explanation Video:
http://www.youtube.com/watch?v=hRHsk8Nycdg

* * *

NOTE: This page intentionally left blank.