

# **An Introduction to Intel Flexible Port Partitioning Using SR-IOV Technology**

---

Intel® LAN Access Division

*September 2011  
Revision 1.01*

# Legal

---

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families.

This document as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an ordering number and are referenced in this document or visit Intel's website at <http://www.intel.com>.

Intel and Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\*Other names and brands may be claimed as the property of others.

Copyright © 2008-2011. Intel Corporation. All Rights Reserved.

# Revisions

---

Date	Revision	Description
Sept 14, 2011	1.0	Initial Release
Sept 19, 2011	1.01	Added link additional resources.

# Contents

---

1	Introduction.....	5
2	Traditional Traffic Partitioning Using Discrete 1GbE Ports.....	7
3	10 Gigabit Ethernet.....	9
4	Using PCI SIG SR-IOV Standard for Traffic Partitioning.....	11
5	Intel Flexible Port Partitioning.....	13
5.1	Bandwidth Sharing Fairness.....	13
5.2	VLANs .....	13
5.3	MAC Address Assignment .....	14
5.3.1	Configuring This Goodness.....	14
5.3.1.1	System Requirements .....	14
5.3.1.2	Operating System Requirements.....	14
5.3.1.3	Specifying VF's to OSs.....	14
6	Flexible Port Partitioning Using Intel 1GbE.....	16
7	Summary .....	18
8	What's Next .....	19
8.1	About the Authors.....	19
9	Additional Resources .....	20

# 1 Introduction

The transition to 10 Gigabit Ethernet from 1 GB Ethernet environments has begun. As the price of 10 Gigabit components continues to come down and the demand for high speed networking continues to grow, this transition will accelerate.

It has been a common practice for a long while to physically segment different types of traffic (such as iSCSI, NFS, Backup, Management, etc.) using discrete 1GbE ports. Often times such designs also have a redundancy requirement, necessitating additional ports.

This practice of physically separating Ethernet traffic types using multiple 1GbE ports also ensures the bandwidth required for the different traffic types do not interfere with other Ethernet tasks.

Consider the configuration in Figure 1. This example is of a database server that has an SLA requirement of 1Gbps for database traffic; it also has dedicated ports for Backup, iSCSI and server management.

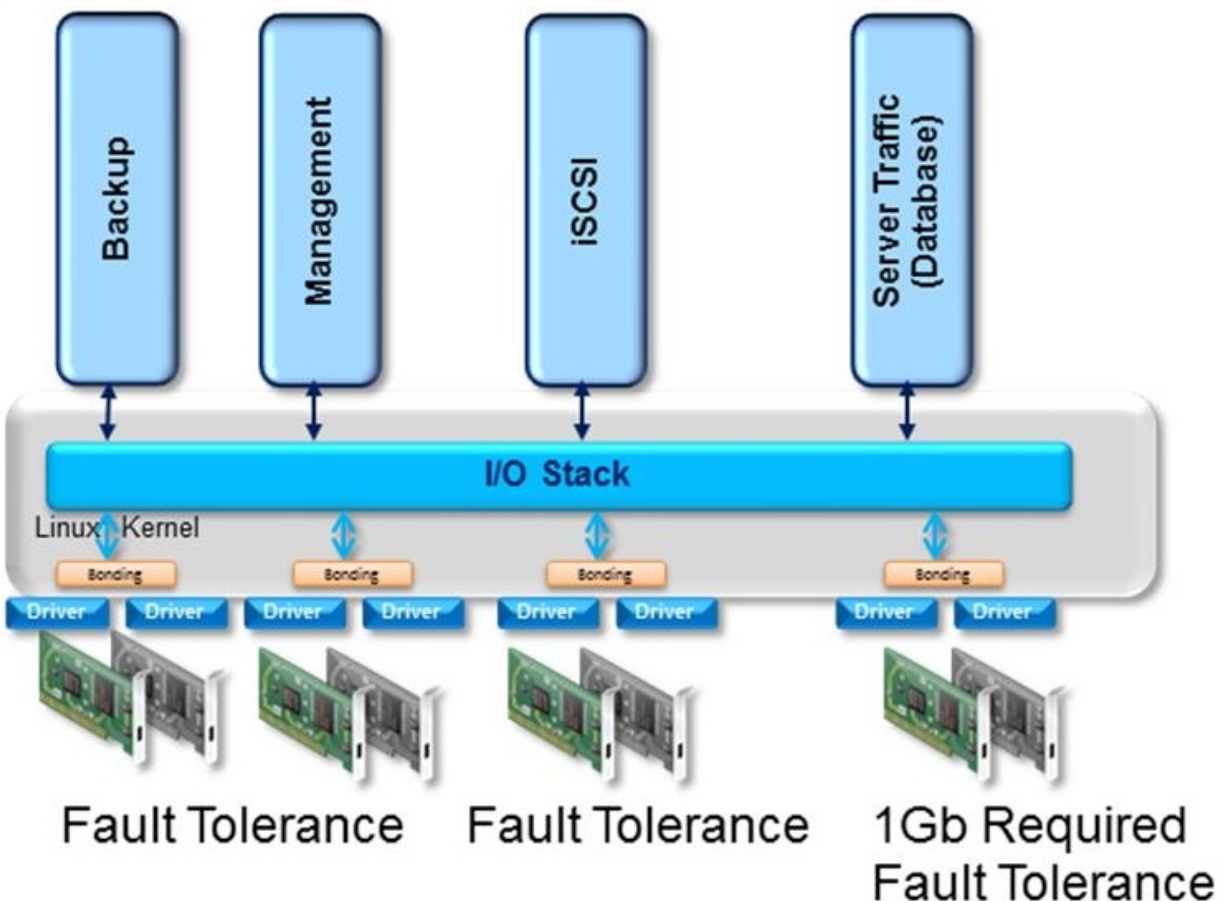


Figure 1 Physical Separation of Traffic Types

Many (or all) of these ports require a fault tolerance port for failover. In this example, eight ports are required for connectivity and fault tolerance. One could use two quad port Ethernet devices to achieve the necessary number of ports.

This is a fairly standard configuration. The configuration provides for the separation of different traffic types and ensures that one type of traffic does not starve other processes. It also ensures the SLA of a 1Gbps minimum traffic is available, with fault tolerance for database traffic.

Although this configuration meets the requirement for the separation of different traffic types over discrete devices for QoS (Quality of Service) and redundant ports for failover, it also requires four additional Ethernet ports (on 'standby'). Consider that for just a moment – while the server, switch and Ethernet cables are all functioning without any kind of failure, there are four Ethernet ports connected to the switch, drawing power that won't ever be utilized until some kind of failure on another port occurs.

## 2 Traditional Traffic Partitioning Using Discrete 1GbE Ports

In a traditional datacenter, designs are likely to physically partition kinds of traffic (such as NFS, iSCSI and Backup) on different physical uplinks. This allows for inherent Quality of Service (QoS), as the different traffic types utilize discrete physical uplinks. It also ensures that no one traffic type can monopolize the Ethernet connectivity with head-of-line blocking.

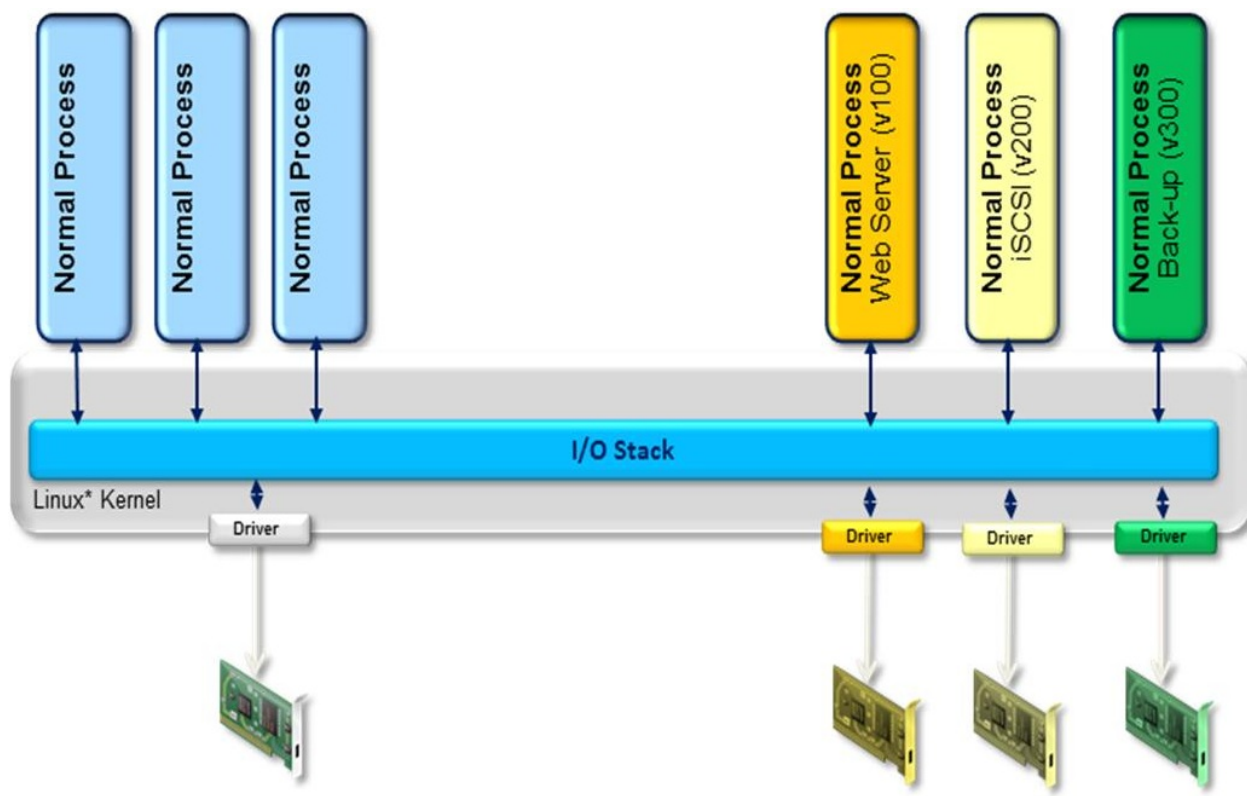


Figure 2 Traditional Ethernet Partitioning – Discrete Ports

Many times more than one physical port is assigned to a given traffic type, for fault tolerance, bandwidth aggregation, or both. This can quickly lead to a large number of physical 1GbE uplinks from a server.

In a 10GbE environment, it would likely be prohibitively costly to physically segment different kinds of Ethernet traffic using dedicated physical ports. So then the question is how can a designer achieve QoS and fault tolerance without the physical segmentation mechanism used in traditional 1 GbE configurations? That is the \$64k question, or the 64VF question (don't worry, the poor joke will make sense later).

This paper describes a method to achieve the desired goal of partitioning a 10 Gigabit Ethernet connection into segments using components available today.

# 3 10 Gigabit Ethernet

In a few years, there will be more 10 Gigabit Ethernet ports in datacenters than 1 Gigabit ports. The method of physically partitioning traffic is likely to be cost prohibitive. This requires that a new paradigm come into being.

Having 10Gbps of bandwidth is useful. But if you have a process (such as Backup or NFS) that utilizes more bandwidth than desired, this still can lead to significant management challenges.

We have heard from customers who like the bandwidth capabilities and cable management advantages of utilizing 10 Gigabit Ethernet. The same customers, however, say that there are challenges they face if various traffic types (such as NFS, iSCSI and Backup) cannot be partitioned.



Figure 3 Unruly Backup Process

Figure 3 shows a configuration where several potentially high bandwidth processes are active (iSCSI, Backup, NFS and two Virtual Machines). In this example, the Backup process is utilizing greater than 9 Gbps on a 10Gb connection. There is also management traffic (such as SSH, VNC etc).

Because the Backup process is utilizing so much bandwidth, the other processes are being 'starved' for bandwidth. 10 Gigabits of bandwidth is great, however an unruly processes or two could have some very undesirable consequences. A couple of misbehaving processes such as this could potentially use so much bandwidth as to render the remote management interfaces (such as SSH or Remote Desktop) all but useless for lack of Ethernet bandwidth.

OK – so I might be exaggerating a bit, however I think the point is clear – a nice big 10 Gigabit pipe is great, however there really needs to be a way to ensure some kind of QoS to make sure misbehaving processes can't negatively affect other processes needing Ethernet connectivity.

**Author Note:** The GUI shown in Figure 3 is not a mock-up; it monitors throughput utilizing data from the ethtool tool with the '-S' option.

# 4 Using PCI SIG SR-IOV Standard for Traffic Partitioning

---

When you hear 'SR-IOV' you likely think of "Single Root I/O Virtualization" and the specification for it from the PCI SIG; or Intel Blogs, videos or whitepapers on the topic. Intel has been working with the Open Source community for several years to add SR-IOV support to the Linux Kernel. As a result, there are a growing number of distro's that have support for SR-IOV.

Note that the specification from the PCI-SIG is actually entitled "Single Root I/O Virtualization and Sharing Specification". It is this "and Sharing" part that most seem to have neglected. SR-IOV need not be limited to a virtualized environment.

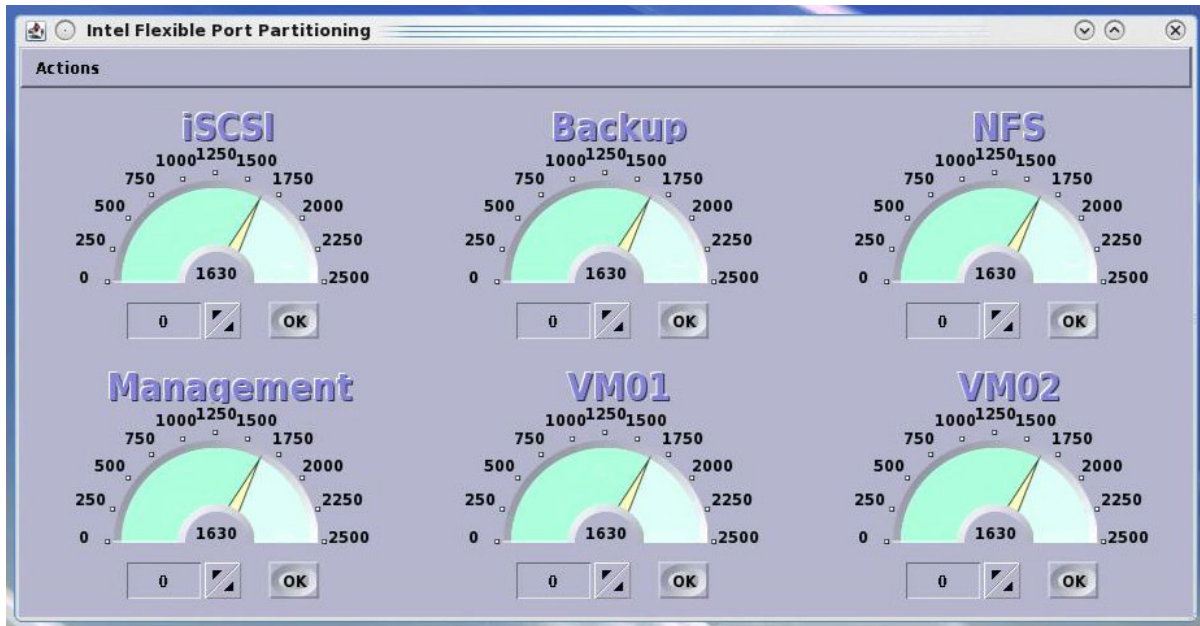
Let me say that again, as it is very important – SR-IOV need not be limited to a virtualized environment. For years now most everybody has been talking about how with SR-IOV you can directly assign a VM to a VF to get optimal performance and gain some additional security for you VM. I've been one of those out there promoting this mindset.

Well I'm here to tell you that there is an alternate, very interesting and appealing use for SR-IOV in a non-virtualized, or a mixed environment.

Intel has been working with the Open Source community for years now to enable the ecosystem for SR-IOV support. This means that SR-IOV support is now within the Linux Kernel. Now what this really means is that if you enable SR-IOV on an Intel Ethernet Controller in your Linux based OS (recent kernel), what you will get are just the Intel Ethernet devices showing up in your system, you will also get Virtual Functions to appear in your OS.

These VF's will be seen by the OS, and assigned to standard Linux Ethernet devices. Which can in turn be assigned IP Addresses, VLAN tags etc. So now you have many 'virtual' Ethernet devices where before you had one. What would happen do you think if you segmented your traffic types across these virtual Ethernet devices?

Figure 4 demonstrates exactly what can happen. I took the same traffic that was all running over a single port from Figure 3 and spread it across six VF's or virtual Ethernet devices.



**Figure 4 Flexible Port Partitioning**

See how now all of the 6 processes each now have an equal share of the Ethernet bandwidth? All of these processes are still utilizing the same physical switch uplink (the same single 10 Gigabit Ethernet connection) How is this possible? What happened?

The answer is SR-IOV. That little "and Sharing" part of the specification is quite powerful. By assigning each of these processes to its own Ethernet device, which underneath has a Virtual Function associated with it, an immediate QoS is achieved. This is due to the fact that each of the VF's on an Intel Ethernet device with SR-IOV enabled has dedicated hardware resources, and that each of these VF's is serviced using a hardware based round-robin scheduler built into the Intel Ethernet device.

By enabling SR-IOV and assigning different processes to VF's, this immediate QoS is achieved with absolutely no CPU overhead, as it is all done within the hardware of the Intel Ethernet device.

# 5 Intel Flexible Port Partitioning

---

Utilizing the SR-IOV features of Intel Ethernet devices, one can create multiple virtual functions from a single physical port. The virtual functions appear in the operating system as normal Ethernet devices.

Consider the following output of 'lspci : grep 82599':

```
06:00.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
06:00.1 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
06:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:10.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:11.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:11.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
```

In this example we have two physical ports, and five virtual functions (0-4). Each of these are standard **Eth devices** within the operating system:

```
driver: ixgbevf
version: 1.0.19-k0
firmware-version: N/A
bus-info: 0000:06:10.2
```

This is the output of 'ethtool -i ethn'

Since each one of the virtual functions are standard Eth devices as far as the operating system is concerned, the designer can assign an IP address to each, along with VLAN tags if so desired.

Intel 82599 Ethernet controllers support up to 63 virtual functions per port.

## 5.1 Bandwidth Sharing Fairness

Intel Ethernet controllers employ a round-robin scheduling system within the hardware of the Ethernet Controller, ensuring each VF is given a fair share of the available bandwidth.

This means that at **minimum** each process assigned to a VF will get a 1/n share of the bandwidth, where n is the number of VF's created and being used.

If there is no contention, then processes have additional bandwidth available. For example, if the VM's running in Figure 3 were not using assigned bandwidth, then bandwidth not used would be equally split amongst other active processes.

This is highly flexible. No static partitioning, no user configuration, no special switches are required.

## 5.2 VLANs

Virtual LANs (VLANs) are an essential part of the modern network infrastructure. The Flexible Port Partitioning technology provides the capability of configuring VLANs for each of the Virtual Functions. Intel Ethernet Adapters use standard Linux tools for the configuration.

Example:

```
# vconfig add eth10 100
```

In this case eth10 is a Linux Ethernet device that happens to be a VF. One uses the same exact syntax in setting up a VLAN on any kernel Ethernet device.

## 5.3 MAC Address Assignment

When a virtual function is instantiated, it is given a synthetic MAC address based upon a software algorithm. This occurs each time the VF is instantiated; meaning that between reboots, the address will change. Any configuration done to that eth device is likely lost.

One can manually assign a MAC address to a VF using standard Linux tools. This is recommended.

The following example configures VF 0 on the eth2 physical port to have the MAC address of DE:AD:DE:AD:BE:EF

```
# ip link set eth2 vf 0 mac DE:AD:DE:AD:BE:EF
```

### 5.3.1 Configuring This Goodness

#### 5.3.1.1 System Requirements

Flexible Port Partitioning uses Intel Ethernet implementation of the PCI-SIG SR-IOV specification standard as the underlying technology. One must have a SR-IOV capable and enabled platform. Most major server manufactures are now selling SR-IOV capable systems.

#### 5.3.1.2 Operating System Requirements

At the time of publication, only Linux based Operating Systems support SR-IOV and Intel Flexible Port Partitioning. In order to have the functionality discussed in this paper, a 2.6.39 (or newer) kernel is required.

#### 5.3.1.3 Specifying VF's to OSs

Intel has worked with the open source community to enable SR-IOV within the Linux kernel. Once you have your SR-IOV capable server configured for SR-IOV and the Linux based OS installed, most of the work is done.

Consider the following:

```
# lspci | grep 82599
06:00.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
06:00.1 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
```

Here, the script looks for Intel 82599 10GbE controllers on the server. Both ports on the 82599 show up (this server has an Intel Ethernet Server Adapter X520 installed).

To load up a number of VF's, unload the ixgbe driver. Then reload it with a max\_vfs parameter from 1 to 63. The following example creates 5 VF's:

```
#modprobe ixgbe max_vfs=5
#lspci | grep 82599
06:00.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
06:00.1 Ethernet controller: Intel Corporation 82599EB 10-Gigabit Network Connection (rev 01)
06:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:10.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:11.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
06:11.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
```

Now, in addition to the 2 physical ports on the Intel Ethernet Server Adapter X520, there are 5 virtual functions that have been created and are ready for use.

# 6 Flexible Port Partitioning Using Intel 1GbE

Recall Figure 1 in which 4 different traffic types are physically separated by using discrete Ethernet ports. Each of these traffic types requires fault tolerance, forcing the number of ports used to 8. In the example, only 4 of ports are active. The remaining ports are not being used, but still draw power and use switch ports.

Let's see how Flexible Port Partitioning may be able to provide a solution that efficiently uses all ports, while possibly reducing the number of ports utilized.

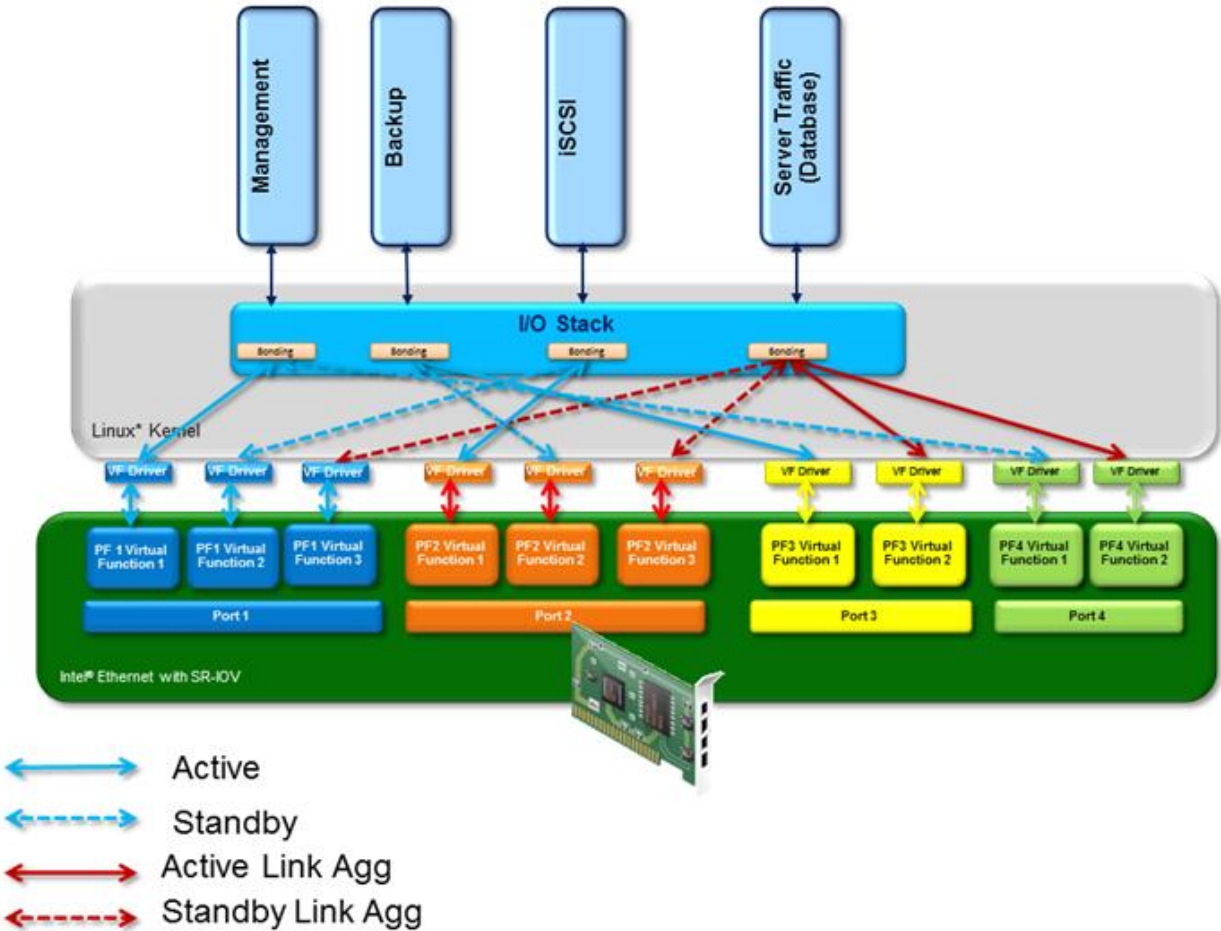


Figure 5 FPP in Action

Figure 5 illustrates a possible alternate configuration. The example uses the Linux bonding driver for both fault tolerance and for link aggregation. Recall from Section 5.1 that each VF will get at minimum an equal share of the bandwidth – so if there are 4 VF's on a physical port and all VF's are attempting to utilize all the available bandwidth, they will all receive 1/4 of the available bandwidth.

Look at Figure 5 a bit closer. Database traffic is split between VF's on Ports 3 and 4, both of which only have 2 VF's created. During normal operations (when there has not been a port failure) the Database traffic has at minimum 1.5Gbps traffic – 1 Gbps from Port 4, and at least 0.5Gbps from the VF on port 3. Other 3 traffic types (iSCSI, Backup and Management) are assigned to VF's as well.

So what does this get you? If we take a look back to Figure 1, there are eight ports in that example scenario, four of which are on standby, awaiting a failure. Contrast this with Figure 5 – the same processes are running, with the same QoS, however ½ of the number of ports are being used. Every port is being used – so there is not wasted bandwidth or port uplinks. Pretty interesting stuff!

# 7 Summary

---

Intel Flexible Port Partitioning (FPP) technology utilizes industry standard PCI SIG SR-IOV to efficiently divide your physical Ethernet device into multiple virtual devices. Intel Flexible Port Partitioning provides and instant Quality of Service by ensuring each process is assigned to a Virtual Function and is provided a fair share of available bandwidth.

Intel FPP requires an SR-IOV enabled platform, OS and an Intel SR-IOV capable Ethernet device. There are no requirements for special hardware outside of the system.

Intel FPP is available today on several Linux\* based Operating Systems. Intel continues to work with the Linux community to add additional features to the kernel.

Intel Flexible Port Partitioning uses SR-IOV Virtual Functions for kernel processes as well as for Virtual Machines.

# 8 What's Next

---

In the next whitepaper, we will discuss employing an additional QoS available as part of the Intel Flexible Partitioning solution – rate limiting for individual Virtual Functions.

We will also provide a discussion about fault tolerance using VF's from more than a single physical port as well as providing bandwidth aggregation by using the bonding driver to aggregate multiple VF's.

We hope you have found this introduction to Intel Flexible Port Partitioning of interest.

## 8.1 About the Authors

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 educational materials and answering questions. He has worked in nearly every facet of Server Manageability, from contributing to the IPMI specification to writing management software. Patrick splits his time between Manageability and I/O Virtualization. He frequently blogs about technologies at <http://communities.intel.com/community/wired>.

**Brian Johnson** is a Product Marketing Engineer for 10G Intel Ethernet Products at Intel Corporation. In this role, he is responsible for product definition, development and marketing of 10G silicon products along with virtualization, manageability and security technologies. Brian has over 15 years of experience in server product planning and marketing during which he has held various positions in strategic planning, product marketing, and product development.

Senior software engineer **Greg Rose** developed the first SR-IOV capable network devices for Intel 82576 Ethernet 1G devices and went on to enable SR-IOV features in Intel Ethernet 10G. Greg has almost 30 years of experience in the industry, developing SW for advanced graphics and communication devices. He wrote his first program on a University of Houston mainframe in 1973 at the age of 15 over a 300 baud modem.

## 9 Additional Resources

---

PCI-SIG Single Root I/O Virtualization 1.1 Specification:  
[http://www.pcisig.com/specifications/iov/single\\_root](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>

Intel® Flexible Port Partitioning using SR-IOV Explanation Video:  
<http://www.youtube.com/watch?v=bOMB9RsQfo4>

\* \* \*