Configuring QoS Features with Intel Flexible Port Partitioning

Configuring teaming/bonding VLANs and Rate Limiting with Intel SR-IOV Technology

Intel® LAN Access Division

Revision 1.0

June 2012

Legal

Revisions

Date	Revision	Description
June 2012	1.0	Initial Release

Contents

1 Introduction

In the previous whitepaper, An Introduction to Intel Flexible Port Partitioning Using SR-IOV Technology, we introduced the concept of Intel Flexible Port Partitioning or FPP. FPP simply stated, provides a mechanism by which you may partition a physical Ethernet port into multiple virtual ports known as Virtual Functions (VFs).

This partitioning is done using a standards based approach utilizing the inherent flexibility within the SR-IOV specification. In addition, using Intel Flexible Port Partitioning provides an inherent Quality of Service (QoS); due to the round-robin scheduler within the hardware that services the underlying SR-IOV Virtual Functions (VFs).

This paper provides details and examples on how to configure additional QoS capabilities and features such as teaming, VLANs and rate limiting of individual VFs.

2 Bonding, Rate Limiting & VLANs

Before we discuss the practical tests and results of using the different types of bonding and configuring of VLANs and rate limiting, let's begin with how you configure these things when using Flexible Port Partitioning on Intel® Ethernet Controllers and Adapters.

2.1 Bonding

The bonding driver does not know nor care whether or not an Eth device is a using Physical Function (PF) or a Virtual Function (VF). It treats them exactly the same.

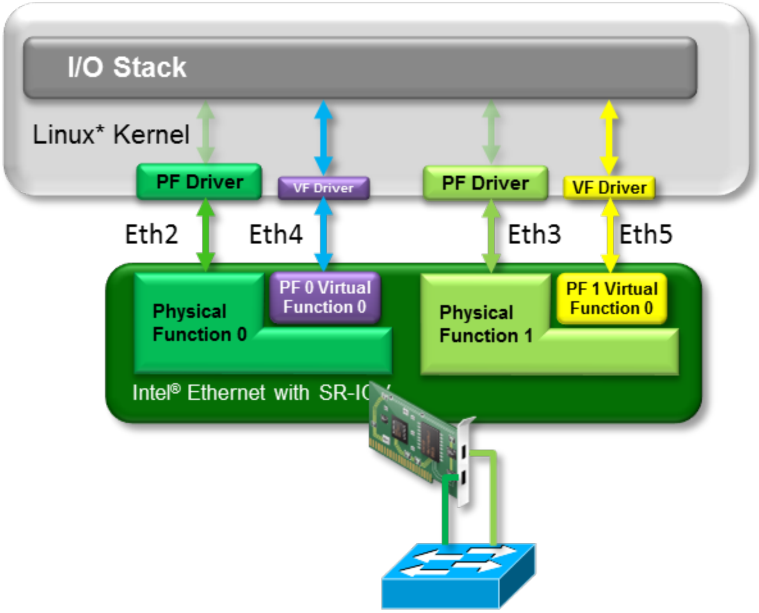

Figure 1 Eth Devices from PF's and VF's

See Figure 1. To the Operating System, both PF's and VF's come up as standard Eth devices. In the figure, we have two PF's (Eth2 and Eth3), each with a single VF (Eth4 and Eth5).

Since the Linux bonding driver simply uses Eth devices, you treat Eth devices that underneath are VF's just as you would PF's. For example, to create a simple Mode 0 bond on two PF's as described in Figure 1, do something like the following:

```
#modprobe bonding miimon=100
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth2 eth3
```

To do the same thing for the VF's, use the same commands. Just specify different Eth devices:

```
#modprobe bonding miimon=100
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

Later in the document, we provide details on configuring different types of bonds using VF's. The key takeaway at this time is that the commands for creating bonds work exactly the same for VF's as standard PF's.

2.2 VLANs

The bonding driver simply uses Eth devices for its purposes. It doesn't know anything about the underlying hardware.

With Intel Ethernet devices, VLANs can be programmed in the hardware of the Ethernet

Controller. This is true of both PFs and VFs.

The iproute2 utility has had additions made to it to support SR-IOV. To assign a VLAN to a VF, you must specify which VF on the PF to configure.

See Figure 1. If you wanted to configure a VLAN of 1234 to Eth4, which is actually VF0 on PF0, the command is:

```
#ip link set eth2 vf 0 vlan 1234
```

Assigning a VLAN with a value of 0, removes the VLAN from the interface.

2.3 Rate Limiting

Most Intel Ethernet devices supporting SR-IOV also support the ability to rate limit any of the VF's. As with VLANs, this is a hardware specific feature, and you must specify the exact VF on a specific PF for the rate-limiting action.

See Figure 1. If you want to rate limit Eth5, which is actually VF 0 on PF 1, to a maximum transmit rate of 2.5Gbps, the command to do so is:

```
#ip link set eth3 vf 0 rate 2500
```

The value of the rate limit equates to Mbps and ranges from 1 to max. The max value is the maximum value for the Intel Ethernet device being used. If it is a 1GbE device, then the maximum value is 1000 (1000Mbps = 1Gbps). If it is a 10GbE device, then the maximum value is 10000 (10000Mbps = 10Gbps).

Assigning a rate limit of 0 removes the rate limiting from the VF.

2.4 View Settings

Sometimes it is useful to be able to view the settings that have been configured. The iproute2 utility, in addition to providing a mechanism to configure settings, also displays results.

We produced an example that creates 6 VF's on PF 0 (Eth2), assigns some VLANS to some, and does some rate limiting. Then to view what we did, we issued the following command:

```
#ip link show eth2
```

The command displays the following results:

```
   eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:1b:21:70:d6:e4 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 5a:e9:62:20:eb:92, tx rate 2500 (Mbps)
    vf 1 MAC ba:14:e0:34:ee:63
    vf 2 MAC e2:3b:b9:ef:2c:73, vlan 2345
    vf 3 MAC f6:e0:d2:15:61:47, tx rate 500 (Mbps)
    vf 4 MAC 22:9b:df:16:1c:7c, vlan 1122, tx rate 5500 (Mbps)
    vf 5 MAC 2a:d0:b6:c2:1a:43
```

Looking at the output, note that eth2 has 6 VF's, with the following settings:

- VF 0 – Rate limited to 2.5Gbps

- VF 1 – No rate limiting or VLAN tag
- VF 2 – Has a VLAN tag of 2345
- VF 3 – Rate limited to .5Gbps
- VF 4 – Has a VLAN of 1122 and rate limited to 5.5Gbps
- VF 5 – No rate limiting or VLAN tag

3 Testing and Configuration for Intel Flexible Port Partitioning and Teaming Modes

3.1 Testing Configurations

The system used for testing was equipped with an Intel X520® Ethernet Converged Network Adapter X520-DA2 and used Red Hat* Enterprise Linux 6.1 as the Operating System.

For the purpose of the testing, 2 VFs were created; one for each Physical Function (PF). The PF's were Eth2 and Eth3, and the VFs were Eth4 and Eth5.

3.1.1 Bare Metal Flexible Port Partitioning

The first test was done without any virtualization. We simply created VF's and used them in the kernel as described in the previous paper.

In the following scenario, the VF's were assigned as Eth4 (from PF0) and Eth5 (from PF1).

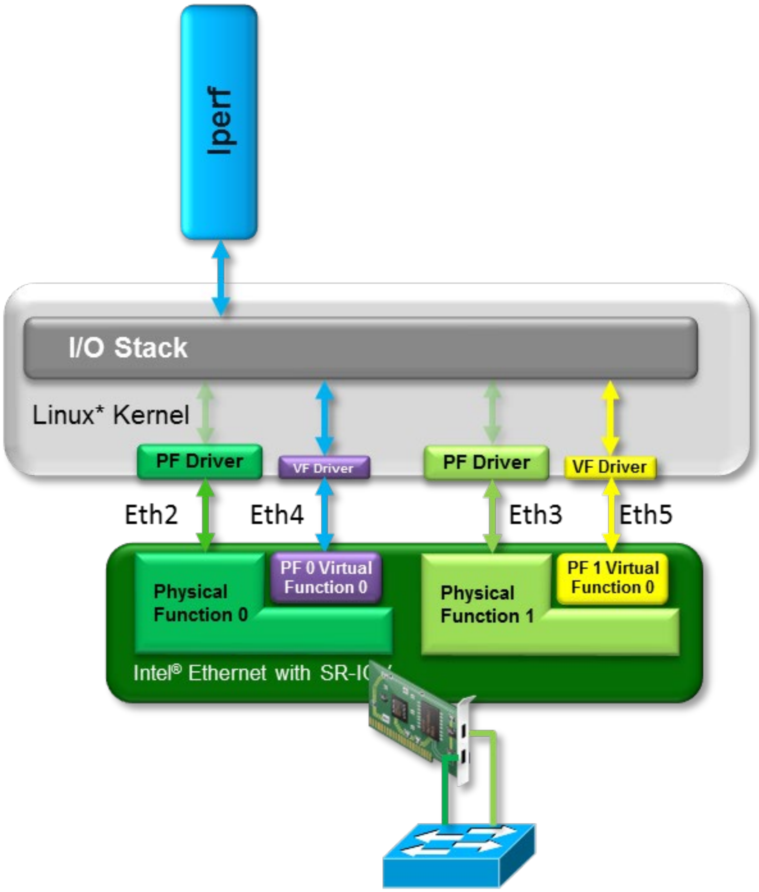

Figure 2 System Block Diagram

PF’s (Eth2 and Eth3) were not configured for these tests.

iPerf was used to test throughput on the VF’s.

3.1.1.1 VF's Assigned to VM's

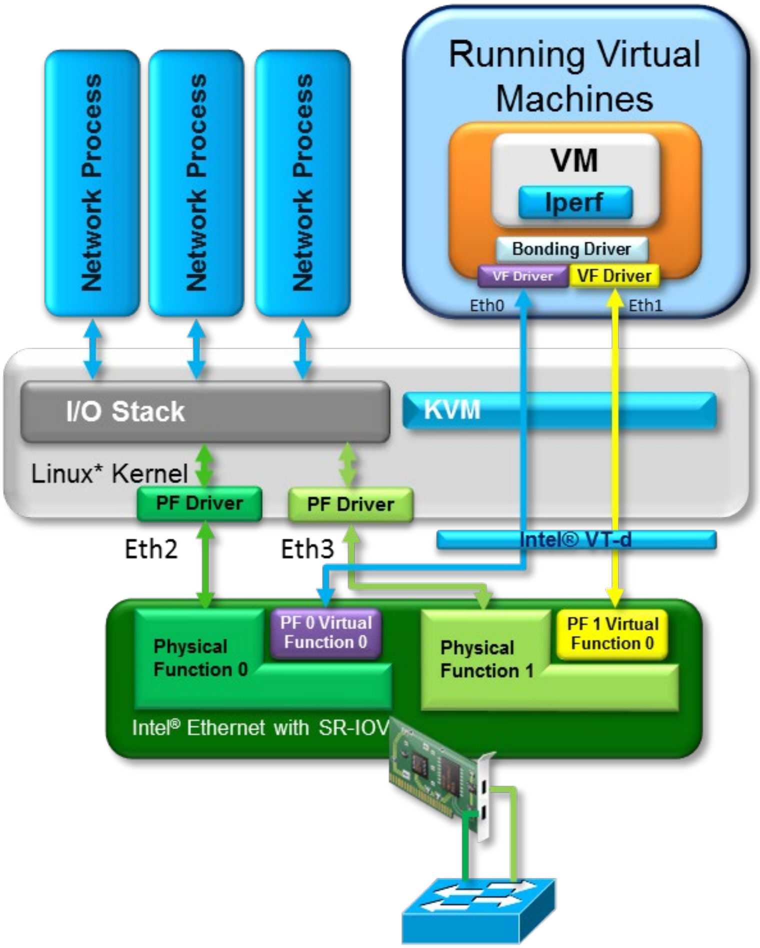

Figure 3 System Block Diagram – VF's to VM

PF's (Eth2 and Eth3) were not configured for these tests.

iPerf was used to test throughput on the VF's from within the VM.

4 Teaming/Bonding

This section discusses the teaming and bonding modes tested using Flexible Port Partitioning. Refer to the Linux Channel Bonding driver documentation for details on the various teaming modes.

4.1 Mode 0: Balance Round-Robin (balance-rr)

This mode provides load balancing as well as fault tolerance. It works by transmitting packets in sequential order from the 1st device to the last device in the team.

No switch configuration is required for this teaming mode.

4.1.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

Note that the PF's (Eth2 and Eth3) are not used. The bonding driver does not know or care that eth4 and eth5 are associated with VF's.

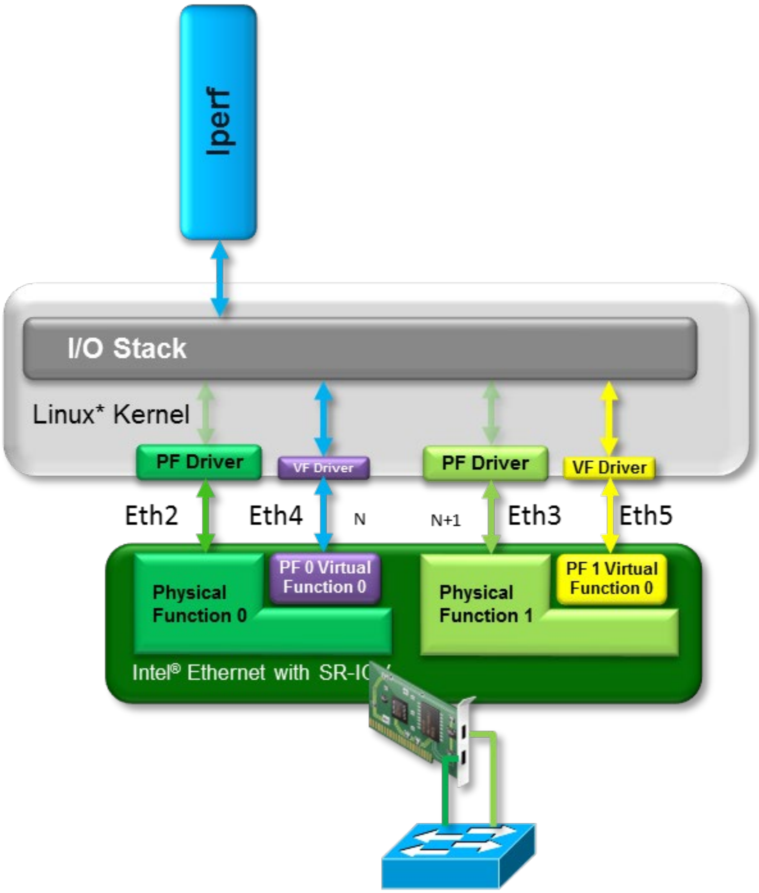

Figure 4 Mode 0 Bonding with VF's to Kernel Process

This test worked as expected, providing fault tolerance. When a cable from either PF was removed, connectivity was retained.

Note that the performance in this mode was around 20% less when compared to no bonding. We found that the throughput was 7.1 Gbps in this mode and jumped to 9.4Gbps when one of the cables was removed (resulting in only a single VF being utilized). At present, the issue is not fully investigated. Our assumption is that it is the software overhead of the round robin scheme.

4.1.2 VM Configuration

If you wish to do the same test from within a VM, to which the VF's were assigned, the configuration commands from within the VM are:

```
#modprobe bonding miimon=100
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth0 eth1
```

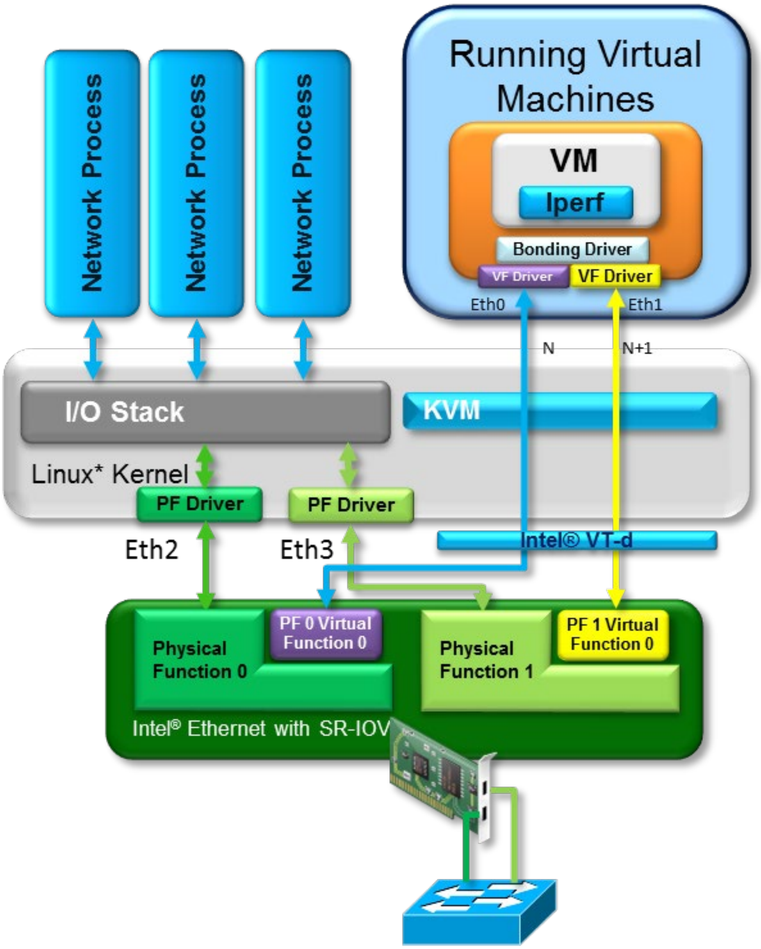

Figure 5 Mode 0 Bonding with VF's to a VM

4.2 Mode 1: Active-Backup

This mode provides fault tolerance. One device is Active while the other is the Backup or standby device. When the link goes down on the active device for a period of time, then all traffic is moved over to the other port. The MAC address is shared between the two ports.

No switch configuration is required for this teaming mode.

4.2.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100 mode=active-backup primary=eth4
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

Note that the PF's (Eth2 and Eth3) are not used. The bonding driver does not know or care that eth4 and eth5 are associated with VF's.

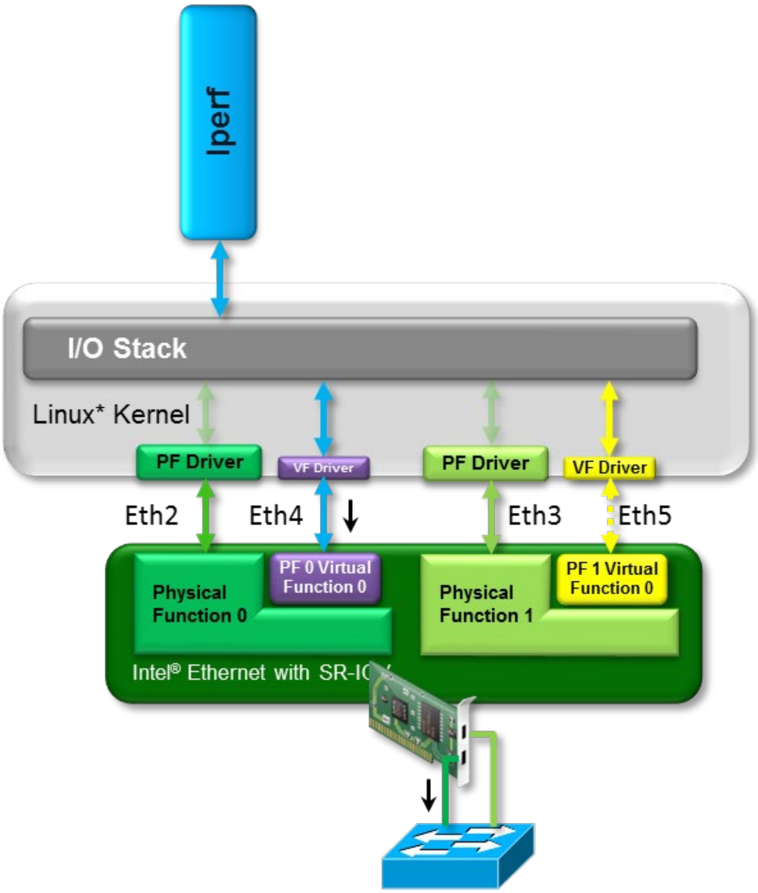

Figure 6 Mode 1 Bonding with VF's to Kernel Process

The results of this test were that it worked as expected, providing fault tolerance – when a cable from either PF was removed, connectivity was retained. Throughput for the testing was > 9Gbps.

4.2.2 VM Configuration

To do the same test from within a VM (to which the VF's are assigned), the configuration commands from within the VM are:

```
#modprobe bonding miimon=100 mode=active-backup primary=eth4
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth0 eth1
```

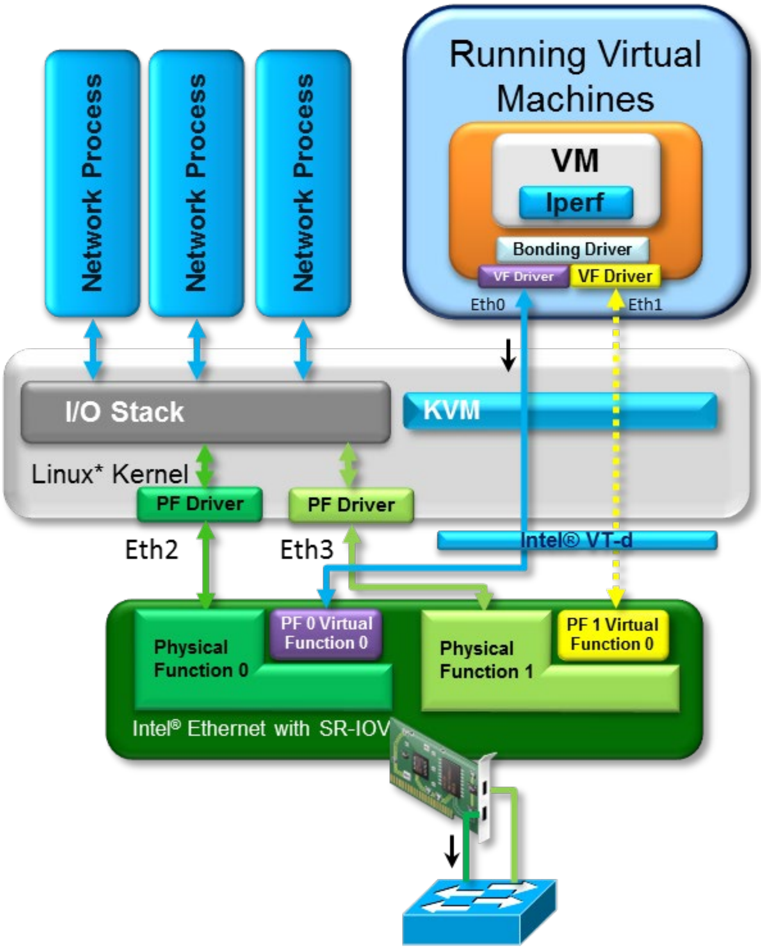

Figure 7 Mode 1 Bonding with VF's to a VM

4.3 Mode 2: Balance-XOR

This mode provides transmit load balancing and fault tolerance. Packets are transmitted to one of the devices in the bond based upon a HASH algorithm. The default algorithm is the SOURCE MAC address XOR'd with the DESTINATION MAC address modulo the number of devices in the bond.

No switch configuration is required.

4.3.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100 mode=xor xmit_hash_policy=layer3+4
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

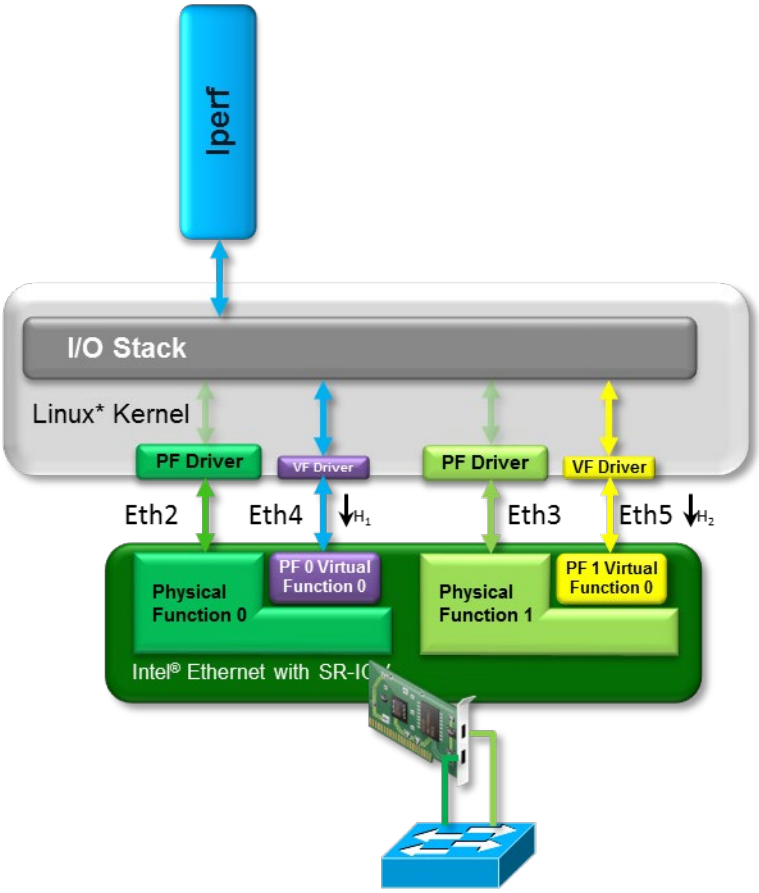

Figure 8 Mode 2 Bonding with VF's to Kernel Process

This test worked as expected, providing fault tolerance and load balancing. Throughput was

14.8Gbps.

4.3.2 VM Configuration

To do the same test from within a VM, to which the VF's are assigned, the configuration commands from within the VM are:

```
#modprobe bonding miimon=100 mode=xor xmit_hash_policy=layer3+4
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth0 eth1
```

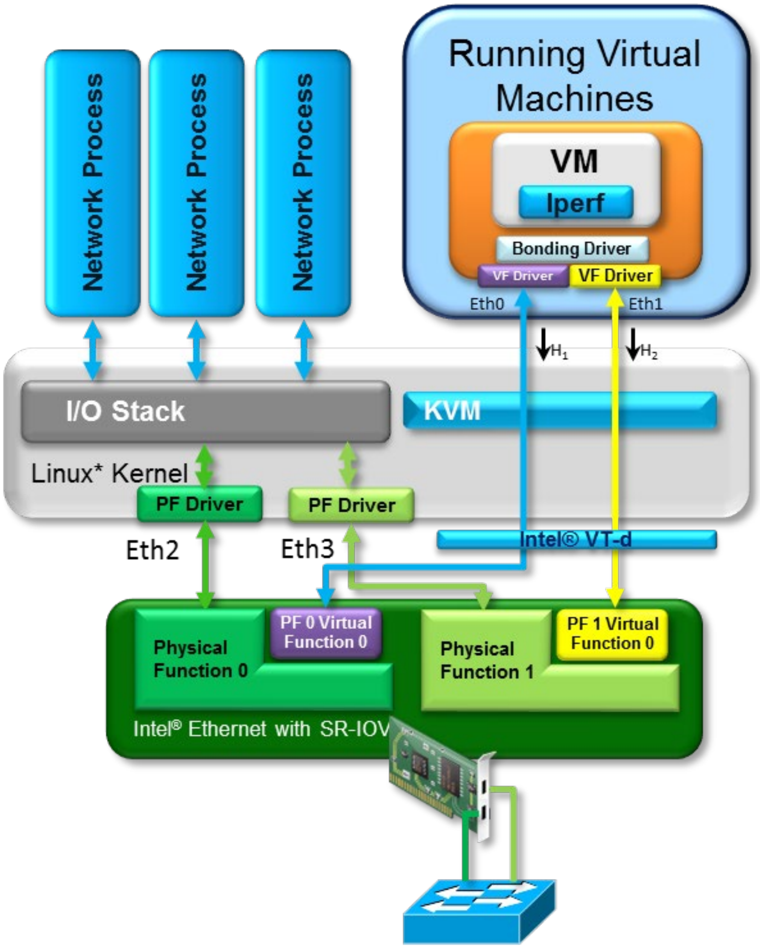

Figure 9 Mode 2 Bonding with VF's to a VM

4.4 Mode 3: Broadcast

This mode provides transmit and fault tolerance by transmitting all packets to all devices in the bond. It is a special purpose mode and not intended for high availability or link aggregation.

No switch configuration is required for this teaming mode.

4.4.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100 mode=broadcast
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

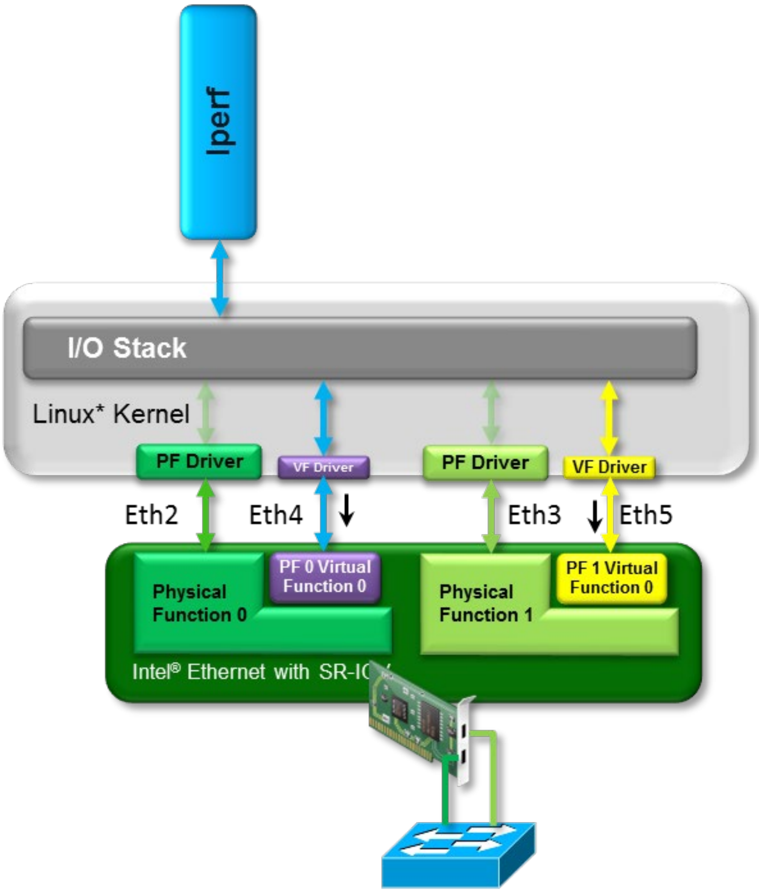

Figure 10 Mode 3 Bonding with VF's to Kernel Process

As mentioned previously, this is a special bonding mode. It has performance limitations, resulting in a throughput of < 2Gbps.

4.4.2 VM Configuration

To do the same test from within a VM (to which the VF's were assigned), the configuration commands from within the VM are:

```
#modprobe bonding miimon=100 mode=active-backup primary=eth4
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth0 eth1
```

4.5 Mode 4: 802.3ad (Dynamic link aggregation)

This mode creates a bond of devices that all share the same link speed and duplex, providing ling aggregation and fault tolerance. The algorithm of which device to transmit on is usually done via a HASH algorithm.

Most switches will require some type of configuration to enable 802.3ad mode.

4.5.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100 mode=802.3ad
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

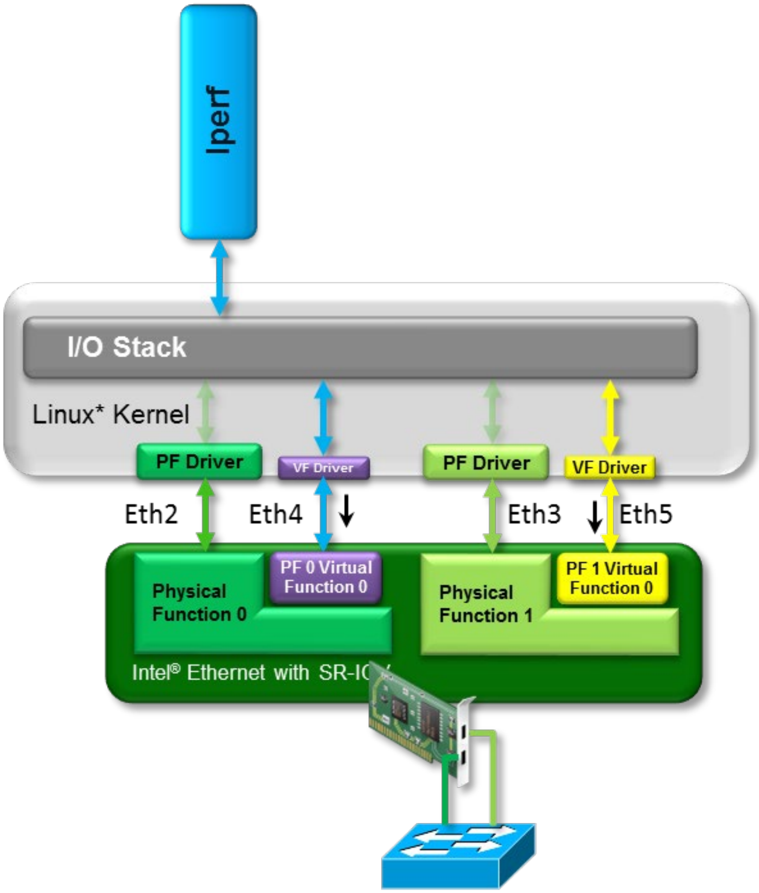

Figure 11 Mode 4 Bonding with VF's to Kernel Process

This test resulted in failure. The 2nd VF did not come up in the channel; the switch treated it as an individual port, rather than part of a bond.

Investigation revealed that the failure is due to the anti-spoofing capabilities in the Intel SR-IOV solution. An update to the iproute2 utility is under development to allow the disabling of the anti-spoofing.

4.6 Mode 6: Adaptive load balancing (balance-alb)

This mode provides load balancing as well as fault tolerance. When transmitting, it determines which device to transmit on based upon the current load and the link speed of the devices. In addition, it provides receive load balancing.

No switch configuration is required for this teaming mode.

4.6.1 Bare Metal FPP Testing

Configuration commands:

```
#modprobe bonding miimon=100 mode=balance-alb
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth4 eth5
```

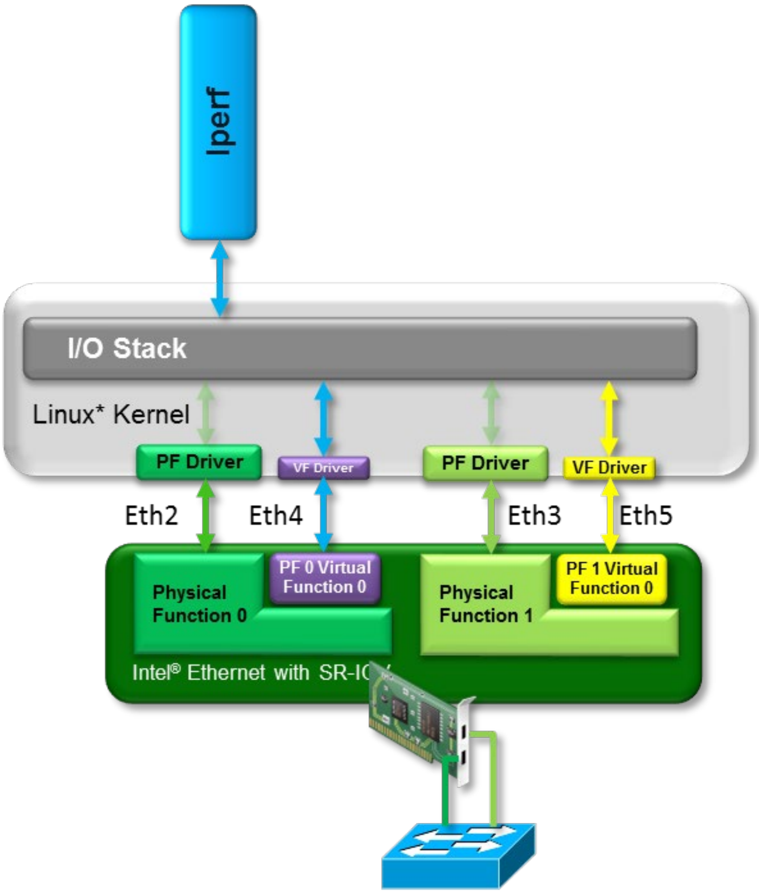

Figure 12 Mode 6 Bonding with VF’s to Kernel Process

4.6.2 VM Configuration

If you wished to do the same test from within a VM, to which the VF’s were assigned, the configuration commands from within the VM would be:

```
#modprobe bonding miimon=100 mode= balance-alb
#ifconfig bond0 192.168.22.11 netmask 255.255.255.0
#if enslave bond0 eth0 eth1
```

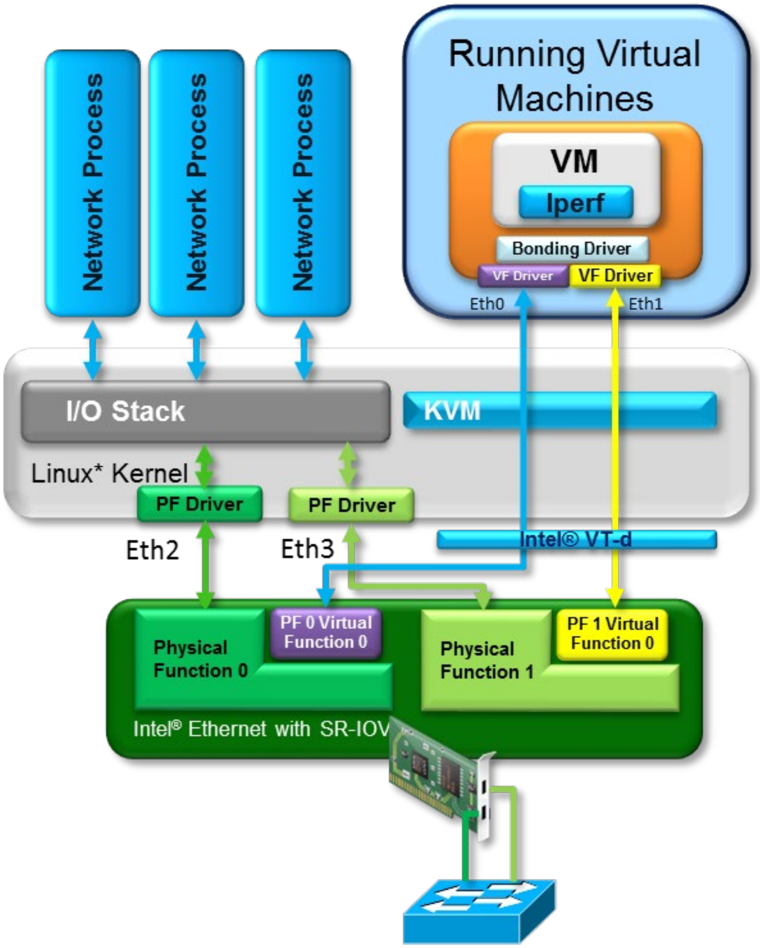

Figure 13 Mode 6 Bonding with VF's within a VM

4.7 PF Requirements

The testing performed did nothing specific with the PF's. Take care, however, if you are going to do any bonding with PFs as well as with VF's. As an example, if you were to configure the PF's for Mode 0 bonding (with one on standby and the other on active), ensure that the VF's are configured the same way.

5 VLANs

VLAN configuration differs from bonding in that the bonding driver is abstracted from the underlying hardware. It simply uses Eth devices that are associated with the hardware, PF's or VF's.

VLANs on the other hand are programmed into the hardware of the Ethernet device as a Layer-2 filter. What this means in the case of Intel Flexible Port Partitioning and VF's is that to configure a VLAN on a VF, you must specify the PF that owns the VF and the VF number.

5.1 Configuring Bare Metal VLAN for VF

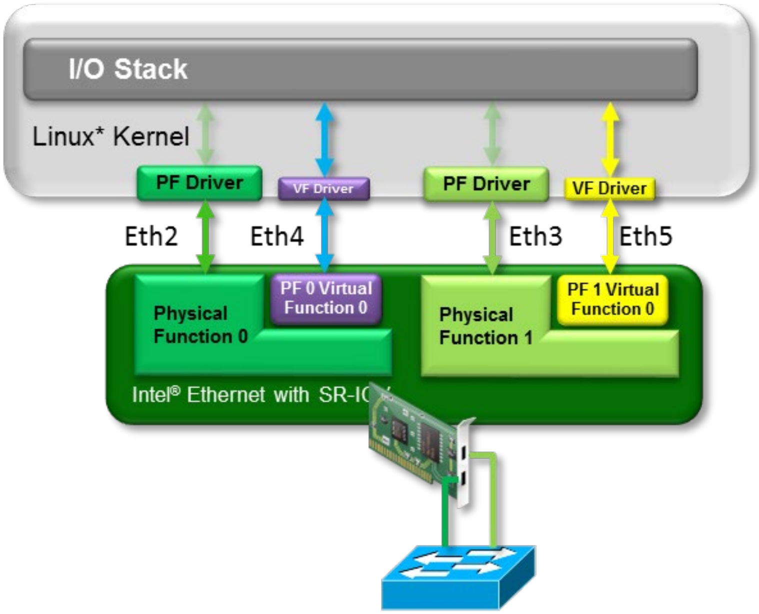

Figure 14 VLAN Configuration

Imagine a system setup like Figure 14. If you want to configure Eth2 with a VLAN of 1234 the mechanism to do is:

```
#ip link set eth2 vlan 1234
```

If you wish to configure Eth4 with the same VLAN, it's more complicated:

```
#ip link set eth2 vf 0 vlan 1234
```

Note that you must specify the VF# (in this case zero) and the Eth device associated with the PF where the VF resides (in this case Eth2).

5.2 VLAN for VF Assigned to a VM

Look at a different situation, where a VF is assigned to a VM. See Figure 15.

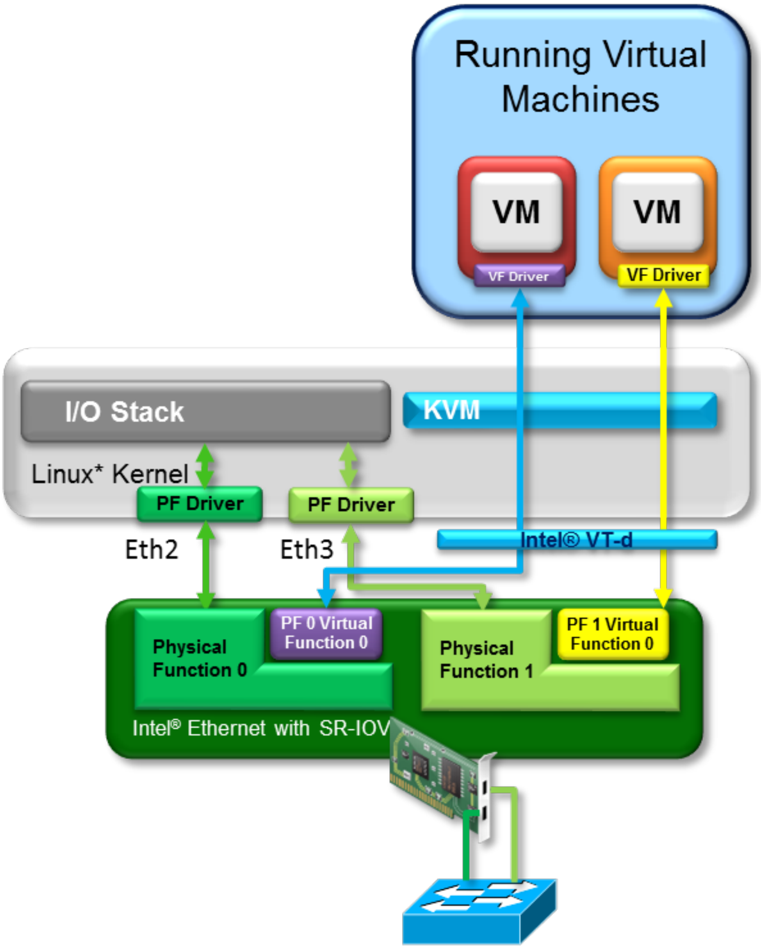

Figure 15 VLAN for VF in a VM

There are two VM's, each with a VF from a different PF. To configure both VM's with the same VLAN, do so from the kernel, not from within the VM. The commands to do so are:

```
#ip link set eth2 vf 0 vlan 1234
#ip link set eth3 vf 0 vlan 1234
```

For this, both PF's have a single VF. Each of those VF's were assigned to VM's. Now the VM's are both on the same VLAN.

5.3 Isolating VM to VM Traffic Using VLANs

Some configurations need VMs to be able to communicate with each other over VFs without ever having the traffic travel over the physical Ethernet cable. If the configuration required is similar to Figure 15, this cannot be accomplished by SR-IOV. This is because the VFs are on different PF's.

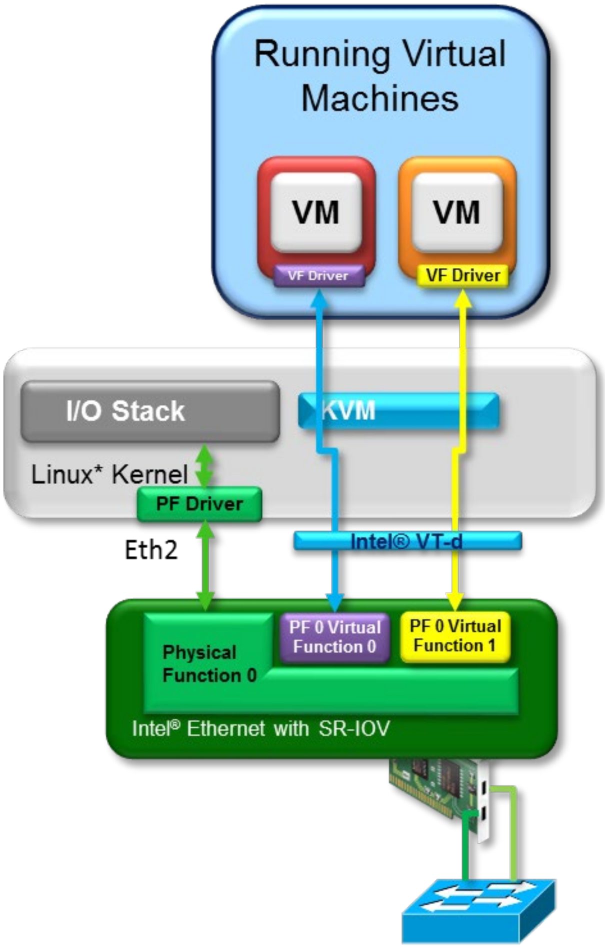

Figure 16 VM to VM Isolation

However, if the setup is similar to that in Figure 16, where the isolated VM's are using VF's that reside on the same PF; this is possible. The configuration of the VLAN tags for these is done from with the hypervisor:

```
#ip link set eth2 vf 0 vlan 1234
#ip link set eth2 vf 1 vlan 1234
```

Note how both commands indicate eth2 (the PF) and the vf#. Now the two VMs can communicate with each other and be isolated from all other traffic.

6 Rate Limiting

The Intel Flexible Port Partitioning solution provides a great way to partition up an Ethernet connection into smaller manageable pieces. This is done using the underlying PCI-SIG SR-IOV technology in the Intel® Ethernet Controllers that support SR-IOV.

Using FPP you can create a number of VF's and assign them different IP addresses and VLANs.

One feature is the ability to configure the rate at which any VF can transmit data. Rate limiting is flexible and can applied to any or all VFs independently, without requiring switches or a reboot to reconfigure.

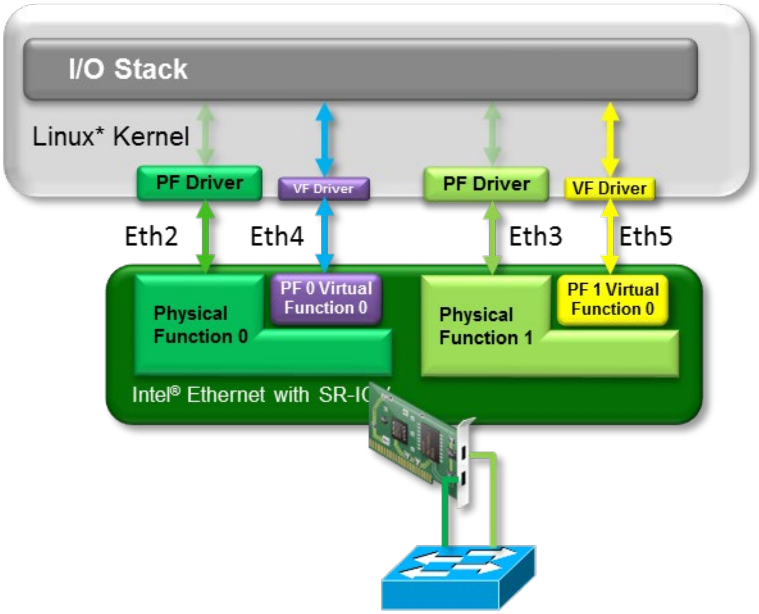

Figure 17 Example Configuration

See Figure 17. There are 4 Eth devices available. Two are VF's (Eth4 and Eth5). Assume that you want to assign your backup daemon to Eth5, on VLAN 4321 and you want to set the maximum transmit for backup data to be 2Gbps. The commands to do this are:

```
#ip link set eth3 vf 0 vlan 4321
#ip link set eth3 vf 0 rate 2000
```

Assuming the backup is associated with Eth5, backup traffic is now isolated via VLAN as well as limited to a maximum bandwidth of 2Gbps. As already mentioned, rate limiting is flexible. For example, if you wanted to let backup run at all times, but during peak operational times you want to limit the bandwidth even more (say down to 250Mbps), you can do so easily:

```
#ip link set eth3 vf 0 rate 250
```

Then in the middle of the night, when other uses of the network may be less, you can remove the limit on the backup by specifying a rate of 0:

```
#ip link set eth3 vf 0 rate 0
```

6.1 Combining the Technologies

Bonding, VLANs and rate limiting are all separate capabilities that work independently of each other. This means you can create VLANS for VF's, rate limit the VF's and bond them as well.

7 Final Thoughts

In the 1st paper on Intel Flexible Port Partitioning we discussed how you can partition up an Ethernet port into smaller partitions and how each of those partitions are serviced in a round-robin fashion, eliminating head-of-line-blocking and improving Quality Of Service.

This paper has detailed how the more traditional QoS features (such as teaming and VLANs) can be used with Intel Flexible Port Partitioning. With the addition of the ability to rate limit down to the VF level, we believe that Intel Flexible Port Partitioning is a robust and capable solution for Ethernet needs.

Care needs to be taken when configuring these advanced features. You must consider how your PF is configured when doing teaming. When teaming two VF's, make sure the VLANs and rate limiting match, etc.

7.1 How to Tell Which PF Owns a VF

SR-IOV is a cool and interesting technology. Its ecosystem continues to mature. However, there are still things that are a bit cumbersome. For example, determining which PF owns a particular VF.

The following script may be useful in figuring that out:

```
#!/bin/sh
if [ -z "$1" ]; then
  echo "usage: lsvf <etherdev> [vf]"
  exit 1
fi
if [ ! -d "/sys/class/net/$1" ]; then
  echo "lsvf: interface $1 not found"
  exit 2
fi
if [ -z "$2" ]; then
ls -ld /sys/class/net/"$1"/device/virt* | cut -f 11 -d ' ' | cut -b 4-
else
ls -ld /sys/class/net/"$1"/device/virtfn"$2" | cut -f 11 -d ' ' | cut -b 4-
```

```
fi
```

The working part of the script is :

```
ls -ld /sys/class/net/"$1"/device/virtfn"$2"
```

8 Additional Resources

An Introduction to Intel Flexible Port Partitioning using SR-IOV
http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/10-gbe-ethernet-flexible-port-partitioning-brief.pdf

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

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