



# **Cisco CSR 1000V**

## **Packet Flow Troubleshooting Guide**

March 21, 2016

**Cisco Systems, Inc.**

[www.cisco.com](http://www.cisco.com)

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The following information is for FCC compliance of Class A devices: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

The following information is for FCC compliance of Class B devices: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If the equipment causes interference to radio or television reception, which can be determined by turning the equipment off and on, users are encouraged to try to correct the interference by using one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Modifications to this product not authorized by Cisco could void the FCC approval and negate your authority to operate the product.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

© 2016 Cisco Systems, Inc. All rights reserved.

# Contents

1	Introduction.....	1
2	Product Overview.....	1
2.1	I/O Models .....	3
2.1.1	Virtual Switch .....	4
2.1.2	PCI Passthrough .....	4
2.1.3	SR-IOV .....	5
2.2	Host System Configuration.....	5
2.2.1	CPU Specifications.....	5
2.2.2	Multi Socket Host.....	5
2.3	BIOS Settings.....	5
2.3.1	Power Saving Settings.....	6
2.3.2	Intel SpeedStep .....	6
2.3.3	Hyper-threading.....	6
3	Packet Flow in CSR VM.....	6
4	Troubleshooting for Packet Drops in CSR 1000V .....	7
4.1	CSR 1000V Overview Information .....	7
4.2	Software License.....	8
4.3	Interface Speed.....	9
4.4	Interface MTU.....	9
4.5	Supported NICs .....	10
4.6	CPU and Memory Usage.....	10
4.7	Packet Trace.....	14
4.8	Embedded Packet Capture .....	15
5	CSR Packet Flow in VMware ESXi .....	15
5.1	NIC Information .....	15
5.1.1	NIC MTU .....	15
5.2	ESXi I/O Models .....	16
5.2.1	Using vSwitch .....	16
5.2.2	Passthrough (VMDirectPath I/O) Mode.....	24
5.2.3	Single Root I/O Virtualization (SR-IOV) Mode .....	27
6	CSR Packet Flow in KVM.....	29
6.1	KVM I/O Models .....	30
6.1.1	Using Linux Bridge.....	30
6.1.2	Passthrough Mode.....	35
6.1.3	Single Root I/O Virtualization (SR-IOV) Mode .....	35
6.2	NIC Information .....	35
6.2.1	NIC MTU .....	35
6.3	CSR VM Tuning.....	36

6.3.1	CPU Performance State Control .....	36
6.3.2	vCPU Pinning .....	37
6.3.3	Enabling vhost-net .....	39
6.3.4	NIC Offloading .....	40
7	CSR Packet Flow in Citrix XenServer .....	40
7.1	NIC Information .....	40
7.1.1	NIC MTU .....	40
7.2	XenServer I/O Models .....	41
7.3	Using Bridge .....	41
7.3.1	CSR VM Tuning .....	46
7.4	Passthrough Mode .....	48
7.5	Single Root I/O Virtualization (SR-IOV) Mode .....	49
8	Acronyms .....	50
9	References .....	50

# 1 Introduction

This document provides information about troubleshooting common packet flow issues in the Cisco CSR 1000V platform. The goal of this document is to help identify and fix some of the common issues affecting the packet flow as well as perform more extensive troubleshooting. A general introduction of the CSR 1000V architecture is also provided.

## 2 Product Overview

The Cisco CSR 1000V is the Cisco Cloud Services Router. It is a software router that an enterprise or a service provider can deploy as a virtual machine on a hypervisor. It is based on the Cisco IOS-XE software and shares the feature set with the ASR1000 and ISR4000 series router platforms.

The CSR 1000V is supported on the following hypervisors:

- VMware ESXi
- Red Hat KVM
- Ubuntu KVM
- Citrix XenServer
- Microsoft Hyper-V

CSR is also supported on public cloud environments like Amazon AWS and Microsoft Azure.

The CSR 1000V supports a rich set of features, including the following:

<b>Basic networking</b>	BGP, OSPF, EIGRP, Routing Information Protocol (RIP), Intermediate System-to-Intermediate System (IS-IS), IPv6, GRE, VRF-Lite, NTP, and QoS
<b>Advanced networking</b>	Layer 2 Tunneling Protocol Version 3 (L2TPv3), BFD, MPLS, VRF, VXLAN and Lawful Intercept
<b>Multicast</b>	Internet Group Management Protocol (IGMP) and Protocol Independent Multicast (PIM)
<b>Addressing</b>	802.1Q VLAN, EVC, NAT, DHCP, and DNS
<b>Security</b>	ACL, AAA, RADIUS, and TACACS+, Zone Based Firewall, IPSec VPN, Easy VPN, DMVPN, and FlexVPN
<b>Application experience</b>	WCCPv2, AppNAV, Network-Based Application Recognition Version 2 (NBAR2), AVC, and IP SLA
<b>Hybrid cloud connectivity</b>	LISP, OTV, VPLS, and EoMPLS
<b>Subscriber management</b>	PTA, LNS, and ISG
<b>High availability</b>	HSRP, VRRP, GLBP, Box-to-box high-availability for ZBFW and NAT
<b>Management</b>	Cisco IOS XE CLI, SSH, Flexible NetFlow, SNMP, EEM, and NETCONF

**Note:** Support for the above listed features may vary depending on the hypervisor and the deployment environment.

Figure 1 shows a high level overview of the CSR1000V Architecture. It consists of four software complexes: Control Plane, Data Plane, Service Plane and Management Plane.

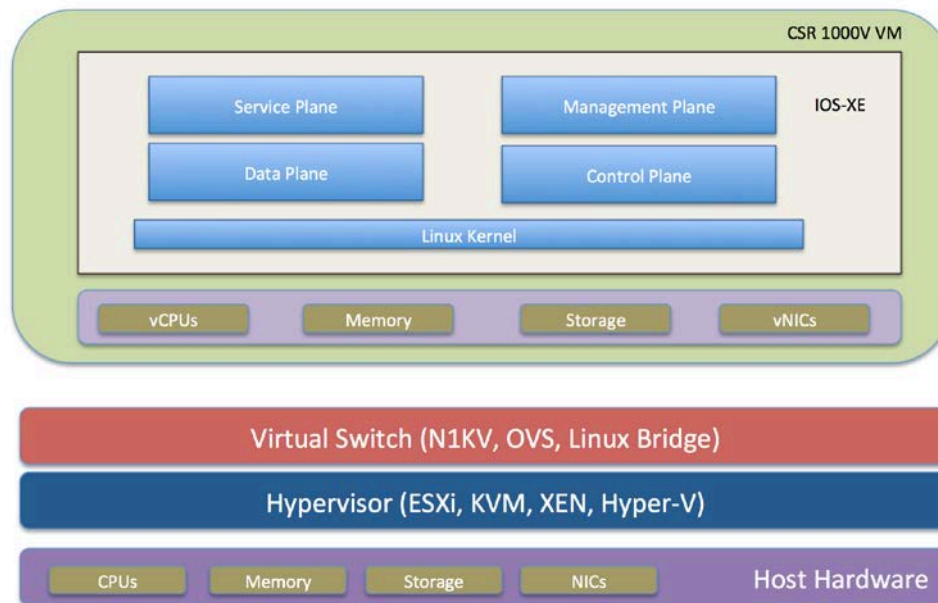


Figure 1: CSR Architecture

The CSR 1000V uses the Cisco IOS-XE operating system which is also used on other Cisco routing and switching products. The IOS-XE operating system is based on the Linux kernel. The CSR 1000V control plane consists of several user space applications, including IOSd and other control plane infrastructure processes. The IOSd user space application provides the familiar IOS CLI user interface and runs the L2/L3 feature and services control plane.

The CSR data plane consists of a single user-space multi-threaded Linux application. The data plane processes all the packets entering and leaving the system. An efficient near-zero copy software infrastructure is used to transfer the packets between the vNIC interface and the user space data path application. The data plane applies all the configured features and services on the received packets. If the received packet is a control packet, then the data path punts the packet to the control plane. The control packets generated by the control plane processes are injected into the data path process for transmission to the vNIC interfaces.

The management plane contains applications that provide the RESTful API access to the system and it also runs applications that integrate the CSR 1000V with the Cisco Prime Network Services Controller (PNSC).

The service plane hosts third party applications and services within the CSR 1000V either in a container or a VM.

The CSR 1000V supports the following virtual machine configuration:

- CPU: 1, 2, 4 and 8 virtual CPUs (depending on the throughput and feature set)
- Memory: 4GB to 16 GB (depending on the throughput and feature set)
- Disk space: 8 GB
- Ethernet network interfaces: One or more virtual network interface cards (vNICs), up to the maximum allowed by the hypervisor

During system boot up, the CSR 1000V allocates the VM vCPUs to the control, data and service plane processes. The allocation depends on the number of vCPUs assigned to the virtual machine. In a multi-vCPU CSR configuration, a single vCPU is allocated to the IOS-XE control, management and service plane processes. The remaining vCPUs are allocated to the data plane process. In a single vCPU CSR configuration, the vCPU is shared between the software complexes.

## 2.1 I/O Models

In a virtualized environment, a physical NIC interface can be connected to a VM in several different ways that are listed below:

- Virtual Switch
- PCI pass-through or VMDirectpath I/O
- SR-IOV

The throughput and latency of packets through a CSR 1000V VM depends on how the physical NIC interface is connected to the CSR VM. The CSR 1000V supports different I/O models. Figure 2 shows various configurations in which Ethernet network interfaces can be connected to a CSR1000V VM. These models are described in the following sections.

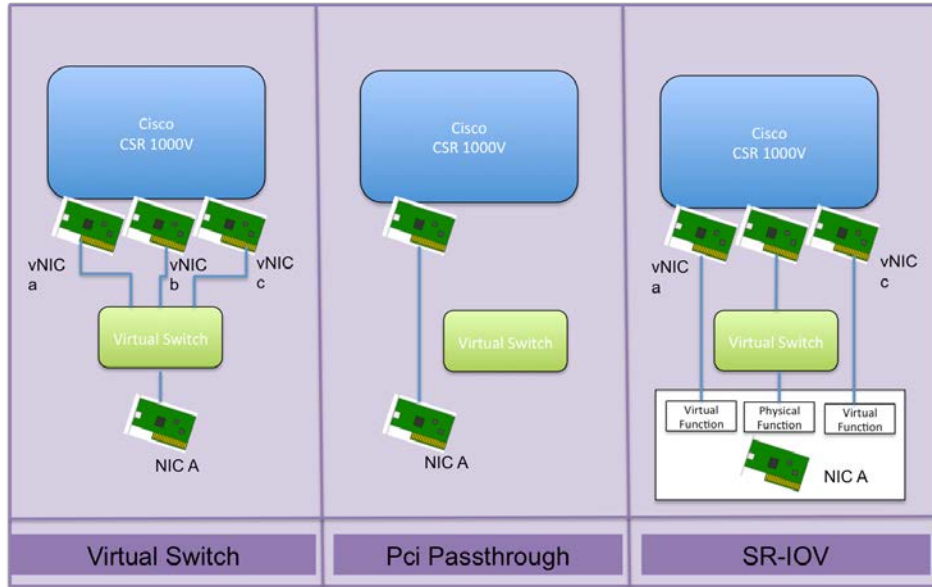


Figure 2: CSR 1000V Network Card Configurations

### 2.1.1 Virtual Switch

In a virtualized environment, the virtual interfaces (vNIC) of a virtual machine are usually connected to a software virtual switch in the hypervisor. The CSR 1000V virtual interfaces can be connected to different types of virtual switches. The type of vSwitch supported depends on the hypervisor and it is listed below.

Hypervisor	Supported Virtual Switch
VMware ESXi	VMware Distributed vSwitch, Cisco Nexus 1000V
Redhat KVM	Open vSwitch (OVS), Linux Bridge, Cisco Nexus 1000V
Ubuntu KVM	Open vSwitch (OVS), Linux Bridge, Cisco Nexus 1000V
Citrix Xen	Open vSwitch (OVS), Linux Bridge, Cisco Nexus 1000V
Microsoft Hyper-V	Cisco Nexus 1000V, Virtual Switch

Virtual Switch is easy to configure and can support a significant number of ports per switch. It also allows the use of virtualization features such as vMotion and Fault Tolerance. Unfortunately the vSwitch has a fixed aggregate performance capability and this makes the vSwitch a serious performance bottleneck for CSR 1000V.

### 2.1.2 PCI Passthrough

A method to improve the throughput and reduce the latency is to pass the NIC hardware, previously controlled by host, directly to the virtual machine in PCI-e pass through mode. In this mode, the hypervisor is bypassed and the traffic flows directly between the virtual machine and the physical NIC device leading to increased throughput and reduced latency.



The CSR 1000V supports the Intel 1Gig and 10Gig NICs in PCI-e passthrough mode. If the Cisco UCS server hardware is used as the host, then CSR1000V VM can use the Cisco VMFEX technology with the Cisco Virtual Interface Card (VIC).

One of the limitations of this approach is that the number of available physical NIC devices limits the number of vNIC interfaces (and therefore VMs) Also, some of the virtualization features like vMotion cannot be used with this method.

### 2.1.3 SR-IOV

Single Root I/O Virtualization (SR-IOV) is another method to improve the throughput and reduce latency through a VM. This solution scales much better than PCI passthrough as most SR-IOV capable NICs can support relatively large numbers of virtual functions, each of which can be assigned to a CSR1000V vNIC. To use SR-IOV, the host processor should support either Intel VT-d or AMD IOMMU technology and the physical NIC hardware should support SR-IOV. CSR 1000V currently supports the Intel IXGBE SR-IOV NIC. Some of the virtualization features like vMotion cannot be used with SR-IOV.

The SR-IOV NICs have a few limitations. The number of supported VLANs is limited. For example, the Intel 82599 EB/ES NIC supports only 64 VLANs across all the virtual functions in SR-IOV mode. The number of multicast groups supported across all the virtual functions is also limited.

## 2.2 Host System Configuration

The host hardware configuration significantly impacts the CSR 1000V forwarding performance. Some of the configuration parameters are described in this section.

### 2.2.1 CPU Specifications

The CPU microarchitectures and the core clock speed differ from server to server. These factors affect the performance of a VM. The CSR 1000V throughput depends on the underlying host CPU processor core speed.

### 2.2.2 Multi Socket Host

Most modern host servers have multiple CPU sockets (NUMA nodes), generally two, and each socket has certain dedicated resources such as memory, PCIe slots and multi-level cache resources per-core. Running an application across multiple sockets on such hosts can result in reduced performance as a result of:

- Poor cache utilization
- Increased bus/QPI access
- Memory access across NUMA nodes

To enhance performance make sure that the CSR 1000V VM runs on the same socket (NUMA node) that has the PCIe slot for its physical NIC and also allocate CPUs to the CSR VM from that same socket.

## 2.3 BIOS Settings

The CSR 1000V performance can be impacted by some of the BIOS settings in the host.

### 2.3.1 Power Saving Settings

The processor C-states are processor idle states intended to reduce power consumption. When this setting is enabled, the processor may go to sleep state to conserve power. This can negatively impact the CSR forwarding performance as the CSR VM may not get enough CPU resources to run and should be disabled. The C-state should be set to C-0 (no C-state) in the BIOS.

### 2.3.2 Intel SpeedStep

Another processor setting that can impact the performance is the **Intel SpeedStep**. This technology is used to reduce the processor power consumption by dynamically changing the processor clock speed. When the CPU load is decreased the processor speed is scaled down and increased when the processor load goes up. This dynamic change in the processor speed results in unpredictable CSR 1000V VM performance and packet loss under traffic burst conditions. This setting should be disabled in the BIOS.

### 2.3.3 Hyper-threading

Hyper-threading technology is intended to provide better utilization of physical cores. On enabling hyper-threading, each physical core appears to the operating system as two logical cores. As a result of hyper-threading, system maintains two copies of architectural state but shares cache and execution resources. Applications that require extensive CPU resources can experience degradation due to CPU contention and poor cache utilization. For better CSR performance, hyper-threading should be disabled in BIOS.

## 3 Packet Flow in CSR VM

Understanding the path of a packet in CSR VM can help in narrowing down the point of packet drop. The following Figure 3 shows the path of a packet through the various components in the CSR forwarding path.

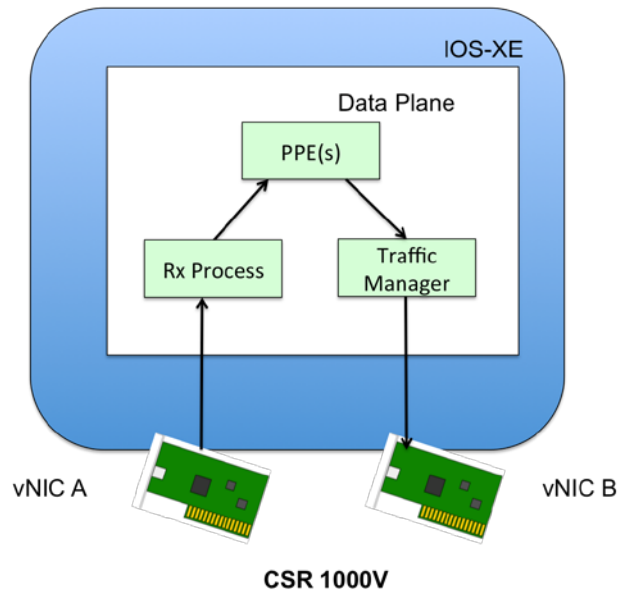


Figure 3: Packet Flow in CSR

The data plane consists of three primary components: Rx process, Packet Processing Engine (PPE) and the Traffic Manager. Depending on the number of vCPUs allocated to the CSR VM, these can run either as separate threads or combined into a single thread of the forwarding process.

Figure 3 shows a packet received at **vNIC A** being forwarded to the Rx processing component of the forwarding plane. The Rx component selects a PPE thread and sends the packet to the PPE for processing. If the received packet is a control packet, it is punted to the control plane for further processing. If it is a data packet, then the forwarding features like ACL, Firewall, etc. are applied. After the features are applied and the output interface is determined from forwarding lookups, the packet is sent to the traffic manager. The Traffic Manager is Hierarchical Queuing Framework (HQF) thread and manages the queuing and scheduling of packets. The traffic manager is responsible for forwarding the egress frames to the virtual interface **vNIC B** for transmission. This example assumes two separate vNICs, but **vNIC A** and **vNIC B** can also be same if packets are going in and out on the same interface.

## 4 Troubleshooting for Packet Drops in CSR 1000V

This section describes various factors to look for while troubleshooting CSR 1000V.

### 4.1 CSR 1000V Overview Information

The first step is to check configuration of the installed CSR VM. This can be found by executing the following command:

```
# show platform software system all
```

```

Router#sh pla so sys all
Processor Details
=====
Number of Processors : 2
Processor : 1 - 2
vendor_id : GenuineIntel
cpu MHz : 2300.000
cache size : 20480 KB
Crypto Supported : Yes
model name : Intel(R) Xeon(R) CPU E5-2470 0 @ 2.30GHz

Memory Details
=====
Physical Memory : 8388608KB

VNIC Details
=====
Name                Mac Address          Status Platform MTU
GigabitEthernet1    0050.569c.7195        UP      9000
GigabitEthernet2    0050.569c.3817        UP      9000

Hypervisor Details
=====
Manufacturer: VMware, Inc.
Product Name: VMware Virtual Platform
Serial Number: VMware-42 1c b5 eb 46 c5 b0 bd-16 e5 24 ed 43 f1 f9 37
UUID: 421CB5EB-46C5-B0BD-16E5-24ED43F1F937

```

This command displays the number of vCPUs, memory and number of interfaces configured for that CSR. Besides this, the command also displays processor and hypervisor details. Make sure that the VM configuration is supported as mentioned in Section 2 “Product Overview”.

If the command displays unsupported configuration for example, number of processors as 3, then shutdown the VM and modify the CSR 1000V VM configuration to change the vCPUs to a supported number. After modifying the configuration, boot the CSR VM for the change to take effect.

## 4.2 Software License

The CSR 1000V throughput is limited by the installed throughput license. Without any license, the CSR 1000V throughput is limited to 100 kb/s. The current throughput level can be listed using the following command:

```
# show platform hardware throughput level
```

```

Router#show platform hardware throughput level
The current throughput level is 100 kb/s

```

To get the desired throughput level, proper throughput license should be purchased and installed. The following command can be used to display information about the currently installed licenses.

```
# show license detail
```

```
Router# show license detail
Index: 1 Feature: sec_100M Version: 1.0
License Type: Permanent
License State: Active, In Use
License Count: Non-Counted
License Priority: Medium
Store Index: 0
Store Name: Primary License Storage
```

For a properly installed and activated license, the “License State” will be set to “Active, In Use”.

It is important to understand the license throughput in CSR 1000V. For example, if a 50 Mbps license is installed, then a maximum of 25 Mbps of bidirectional traffic is possible through CSR. The Cisco CSR 1000V includes a license shaper that restricts the total bandwidth of the router’s all interfaces, not on a single interface. The license shaper does not distinguish between different types of traffic. If the aggregate packet rate on all the interfaces through the CSR is larger than the installed license, the packets get discarded.

## 4.3 Interface Speed

By default, the CSR 1000V vNIC interface (Gigabit Ethernet) speed is set to 1000Mb/s (1Gbps). This can be displayed using the “show interface” command:

**#show interfaces**

```
Router#show interfaces
GigabitEthernet4 is up, line protocol is up
MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
```

When a vNIC interface is connected to a vSwitch, the speed can be set to either 1Gbps or 10Gbps. The interface speed can be changed using the “speed” IOS interface configuration command. Note that if the physical NIC hardware in the host can handle only 1Gb/s traffic, then one should not change the CSR vNIC interface speed (as the interface hardware will be oversubscribed).

## 4.4 Interface MTU

The Cisco CSR 1000V can support a maximum transmission unit (MTU) in the range of 1500 to 9216 bytes. However, the maximum MTU configured on the Cisco CSR1000V cannot exceed the maximum MTU value supported on the hypervisor and the physical NIC. Currently enabled MTU size on an interface can be displayed using “show interface” command mentioned above in section “Interface Speed”. The MTU size can be changed using the “mtu” IOS interface configuration command. To verify the MTU on the physical NIC refer to Section “NIC MTU” under the corresponding VM’s hypervisor section.

## 4.5 Supported NICs

The CSR 1000V officially supports the following vNICs:

Configuration	Supported Driver(s)
ESXi vSwitch	vmxnet3
KVM Linux bridge	virtio
Xen Linux bridge	vif
PCI passthrough	igb, ixgbe
SR-IOV	igbvf, ixgbev

The type of vNIC currently attached to a CSR VM can be displayed using the following show command:

```
# show platform software vnic-if interface-mapping
```

```
Router#show platform software vnic-if interface-mapping
```

Interface Name	Driver Name	Mac Addr
GigabitEthernet3	vmxnet3	0050.569c.126e
GigabitEthernet2	vmxnet3	000c.2927.56f6
GigabitEthernet1	vmxnet3	0050.569c.7a49

Make sure that the “Driver Name” displayed by this command match one of the supported vNIC types in the above table.

If the command displays an unsupported vNIC type (e.g. e1000), then modify the CSR 1000V VM configuration in the hypervisor to change the vNIC type to a supported type. After modifying the vNIC type, reboot the CSR VM for the change to take effect.

## 4.6 CPU and Memory Usage

The next step is to look at the overall CPU and memory utilization within the CSR VM. The following command can be used to display the overall health status of a CSR VM:

```
# show platform software status control-processor brief
```

```
Router#show platform software status control-processor brief
```

Load Average

Slot	Status	1-Min	5-Min	15-Min
RP0	Healthy	2.10	1.35	0.59

Memory (kB)

Slot	Status	Total	Used(Pct)	Free(Pct)	Committed(Pct)
RP0	Healthy	8117052	3226704(40%)	4890348(60%)	3711452(46%)

CPU Utilization

Slot	CPU	User	System	Nice	Idle	IRQ	SIRQ	IOwait
RP0	0	1.93	0.90	0.00	96.36	0.00	0.80	0.00
	1	95.01	4.98	0.00	0.00	0.00	0.00	0.00

This command displays the average CPU load, memory usage and CPU utilization on all the vCPUs configured in CSR. Packet drops can occur in CSR if any of the CPU or memory usage approaches 100% utilization. Make sure that the value displayed under the “Status” column is “Healthy”.

When traffic is flowing through the CSR 1000V data plane, packet drops can happen at any of the components shown in Figure 3. For detailed investigation, it is necessary to understand where the packet drop is happening.

The first step is to make sure that the packets are properly received and transmitted by the CSR vNIC interface. The IOS “show interface” command displays the packet statistics for a vNIC interface.

**# show interface <interface name>**

```
5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
 197076 packets input, 23427142 bytes, 0 no buffer
 Received 0 broadcasts (0 IP multicasts)
 0 runs, 0 giants, 0 throttles
 0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
 0 watchdog, 0 multicast, 0 pause input
```

In this output, non-zero error counters indicates packet drops at the virtual interface. If the CSR vNIC driver cannot forward the received packets fast enough to the data plane, then the packets will be dropped. To minimize this, the vNIC queue size in the host can be increased. The command to increase the queue size depends on the hypervisor and is described in later sections.

The IOS “show interface controller” command displays additional vNIC driver packet statistics.

**# show interface <interface name> controller**

```
GigabitEthernet1 - Gi1 is mapped to eth0 on VXE
  DPIO Rx Drop 0 Packets 197076
  Driver Rx Stops 0 DPIO Rx Congestion Drop 0
Detailed interface statistics:
 TSO pkts tx 0
 TSO bytes tx 0
 ucast pkts tx 11
 ucast bytes tx 660
 mcast pkts tx 148
 mcast bytes tx 11396
 bcast pkts tx 74
 bcast bytes tx 4440
 pkts tx err 0
 pkts tx discard 0
 drv dropped tx total 0
   too many frags 0
   giant hdr 0
   hdr err 0
```

This command displays packet statistics such as packets received, packets dropped and other errors. The detailed interface statistics are hypervisor specific and not available with all NICs. If errors are seen, these indicate that CSR data plane is overloaded and is not able to process the received packets.

The next place to check for packet statistics is the CSR data plane. Use the following command to confirm that the data plane received the packets from a particular vNIC interface.

**# show interface <interface name> stats**

```
Router#show interfaces GigabitEthernet 1 stats
GigabitEthernet1
  Switching path    Pkts In   Chars In   Pkts Out   Chars Out
  Processor         53913    3352905    240        1691
  Route cache       0         0          0          0
Distributed cache  197103   23429772   240        16916
  Total            251016   26782677   480        33832
```

To check whether the data plane has dropped any packets, use the following show command.

**# show platform hardware qfp active statistics drop**

```
Router#show platform hardware qfp active statistics drop
-----
Global Drop Stats          Packets          Octets
-----
Disabled                   131              31260
Ipv4NoRoute                1442             209548
UnconfiguredIpv6Fia        31439            5670464
```

This command will display the packet drop counters across all the interfaces in the CSR forwarding plane.

The following table explains some of the common drops seen:

No.	Drop Type	Possible Reason
1	BadUidbSubIdx	This means that a received packet could not be mapped to a known or configured interface. Sometimes this could be a race condition where a packet could be received before the configuration is applied to data plane.
2	BqsOor	This drop occurs when BQS is out of packet memory that happens due to congestion at BQS.
3	Disabled	This means that packets are being received on an interface that is not in up state. For this run “ <i>show ip interface brief</i> ” to check the interface status and it shows as down, bring up the interface.
4	Ipv4NoAdj	This means ARP is not resolved on that interface.
5	Ipv4NoRoute	This indicates that destination IP address is not there in routing table. Check if the IP address assigned to the interface is correct.



6	TailDrop	This drop is generally due to the packets getting dropped at the egress interface. This can be an indication of sending traffic beyond license. One of the reasons for this is oversubscription. Another reason can be overload by packet fragmentation. This can happen when packets are fragmented due to the MTU size. In such cases even if the ingress interface is less than the wire rate, wire rate can be exceeded at the egress interface and hence dropped. Sometimes too many punt packets can also cause the TailDrop of QFP drop statistics to increase.
7	UnconfiguredIpv4Fia	This means that IP packet is being received and IP v4 address is not set on that interface.
8	UnconfiguredIpv6Fia	This means that IP packet is being received and IP v6 address is not set on that interface.

The CSR data plane may not be able to process all the received traffic if the data plane is overloaded. The following command displays the current data path utilization:

**#show platform hardware qfp active datapath utilization summary**

```
Router#show platform hardware qfp active datapath utilization summary
  CPP 0:
Input:      Total (pps)      5 secs      1 min      5 min      60 min
              (bps)      2          2          2          2
              544        992        992        992
Output:      Total (pps)      2          2          2          2
              (bps)      3712       7952       7952       7952
Processing:  Load (pct)      1          1          1          1
```

This command displays the CPU Utilization of forwarding plane in CSR. The *Processing Load* field provides percentage CPU load. If it is close to 100%, either reduce the load on CSR by distributing traffic to some other CSR VMs or increase the CSR's processing capability by adding more vCPUs to the CSR VM.

The next place to check for packet drops and errors is the Rx process in the data plane. The following command displays the packet statistics from the Rx process:

**# show platform hardware qfp active datapath infrastructure sw-nic**

```
znm 8c454040 device Gil (GigabitEthernet1)
Rx: pkts 197093 bytes 23428178 xoff
  Ring read 800811 empty 610904 rx_avail 0
  revents 0 len err 0 credit err 0 audit 0
Tx: pkts 238 bytes 16796 send 0 forced-txsync 238
  fill 0 poll 0 thd_poll 0
  full 0 lowater 0 hiwater 0
  avail 2046 batch 238 tx_batch_sz 0 sendnow 0 im_alloc_err 0
```

In addition to the commands described above, there are many feature related IOS show commands that can provide additional information. For example, the "show ip traffic" IOS command displays the IP

protocol statistics. For a complete list of feature related commands, refer to the Cisco IOS-XE documentation.

The commands described above displays both the input/ingress and output/egress packet statistics. These can be used to debug both the input and output packet flow through the CSR data plane.

## 4.7 Packet Trace

CSR and Cisco IOSXE platforms in general provide packet trace tool (a.k.a. pactrac) for troubleshooting, debugging and gaining deeper understanding about packet flow. Packet trace uses conditional debug infra feature that determines packets to be inspected.

- Interface – ingress/egress
- IP address – ipv4/6, mask, ACL

The following explains the steps involved in this executing this tool.

1. Enable packet trace  

```
# debug platform packet-trace enable
```
2. How much data do we want to collect?  

```
# debug platform packet-trace packet num_pkts [circular | fia-trace | summary-only]
```
3. What do we want to trace?  

```
# debug platform condition [ipv4 | ipv6 | interface | ingress | egress]
```
4. Start the trace  

```
# debug platform condition start
```
5. Stop the trace  

```
# debug platform condition stop
```
6. View trace stats  

```
# show platform packet-trace statistics
```
7. View trace summary  

```
# show platform packet-trace summary
```
8. View packet information  

```
# show platform packet-trace packet packet_number
```
9. Turn off packet tracing  

```
# no debug platform packet-trace...
# no debug platform condition...
or
# clear platform condition all
```

For a detailed understanding refer to the Cisco IOS-XE “Embedded Packet Capture Configuration Guide” document.

## 4.8 Embedded Packet Capture

Embedded Packet Capture (EPC) is another tool that is available for troubleshooting packets in Cisco IOSXE platforms. This tool helps to copy and store packets in the router's memory. EPC can be used to capture transit packets as well as punt packets. The output packets are available in pcap format.

The following explains in brief the steps involved in this executing this tool.

1. Configuring buffer  
`# monitor capture buffer <buffer name> {options}`
2. Configuring capture point  
`# monitor capture point <protocol> <capture point name> <interface>  
 <direction>`
3. Associating the buffer and the capture point;  
`# monitor capture point <capture point name> <buffer name>`
4. Capture start  
`# monitor capture start <all | buffer name>`
5. Monitor capture status  
`# show monitor capture pointm <all / capture point>`
6. Capture stop  
`# monitor capture stop <all | buffer name>`
7. Dumping the captured packets  
`# show monitor capture buffer <all / merged / buffer name>`
8. Cleaning up the buffer  
`# monitor capture buffer <buffer name> clear`
9. Disabling capture point  
`# monitor capture point disassociate <capture point>`

For a detailed understanding refer to the Cisco IOS-XE “Embedded Packet Capture Configuration Guide” document.

## 5 CSR Packet Flow in VMware ESXi

This section describes the packet flow through a CSR virtual machine in a VMware ESXi hypervisor environment. The ESXi commands to display the packet statistics along the packet path are also described in this section.

### 5.1 NIC Information

#### 5.1.1 NIC MTU

The following command lists all NICs configured on a host.

```
# esxcli network nic list
```

```
# esxcli network nic list
Name      PCI Device      Driver  Link  Speed  Duplex  MAC Address      MTU
Description
-----
vmnic0    0000:009:00.0    igb     Down   0      Half    f8:72:ea:f6:ff:48 1500
Intel Corporation I350 Gigabit Network Connection
vmnic1    0000:009:00.1    igb     Down   0      Half    f8:72:ea:f6:ff:49 1500
Intel Corporation I350 Gigabit Network Connection
vmnic2    0000:003:00.0    ixgbe   Up     10000  Full    90:e2:ba:29:f3:d0 1500
Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection
vmnic3    0000:003:00.1    ixgbe   Up     10000  Full    90:e2:ba:29:f3:d1 1500
Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection
vmnic4    0000:083:00.0    bnx2    Down   0      Half    00:0a:f7:29:6a:88 1500
Broadcom Corporation Broadcom NetXtreme II BCM5709 1000Base-T
```

Besides this, the command also displays information such as NIC driver, Speed, MAC Address and MTU size. MTU field will be help to identify if NIC is enabled to support jumbo size packets or not.

## 5.2 ESXi I/O Models

In an ESXi environment, the virtual NIC interfaces in a CSR virtual machine are usually connected to a VMware virtual switch. Other I/O models like PCIe passthrough and SR-IOV are also supported.

### 5.2.1 Using vSwitch

An example of packet forwarding from one interface to another interface through a CSR VM in ESXi is shown in Figure 4.

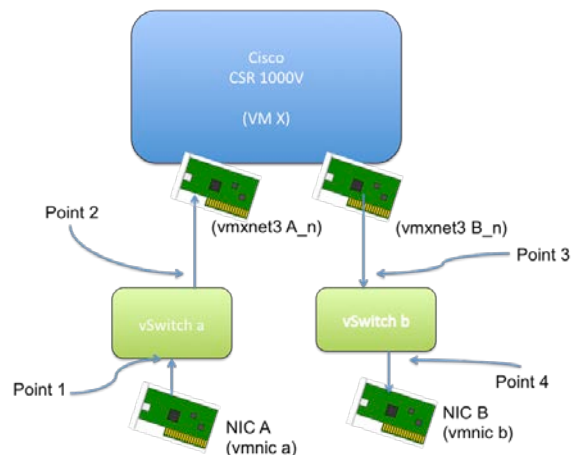


Figure 4: CSR VM in ESXi (Using vSwitch)

In this figure, two physical NIC interfaces are shown. These are connected to two separate virtual switches. The CSR VM has two virtual interfaces and they are connected to the virtual switches. The physical and virtual interfaces can be connected in other ways also. Both the physical NICs can be connected to a single virtual switch and the CSR virtual interfaces can be connected to the single vSwitch.

When a packet is received from the physical NIC, the vSwitch forwards the frame based on the destination MAC address. The CSR VM receives the packet through the virtual interface. After applying the configured features, the CSR VM forwards the packet. The vSwitch receives the packet and based on the destination MAC address forwards the frame. The packet can either go to another VM or it can be forwarded to the physical interface.

Note that the VMware vSwitch is not a learning switch. The VM vNIC MAC addresses are programmed by the ESXi hypervisor in the vSwitch. For security reasons, the vSwitch will drop packets with unknown destination MAC addresses. Only if the destination MAC address in the packet matches that of a VM vNIC interface, the packet will be forwarded to the VM (unless the vNIC interface is configured in promiscuous mode).

The packet statistics can be displayed at various points in the path of a packet into and out of a VM vNIC interface. The statistics can be obtained either using the CLI commands on the ESXi host shell or using the vSphere GUI interface. These commands are described below.

The following sections describe several VMware ESXi commands to display traffic flow related information. Refer to the official VMware documentation for more information about these commands.

The first step is to find the vNIC interfaces of a CSR VM on the ESXi host. This can be obtained by running the following command:

```
# esxcli network vm list
```

This command displays the World ID, name, number of ports and networks for each virtual machine on the ESXi host system.

```
# esxcli network vm list
World ID   Name           Num Ports   Networks
-----
1477266   csrTest1       1   VM Network
1653025   ubuntu         2   VM Network, VM Network
9597914   windowsVM      4   VM Network, VM Network, VM Network
12096180  firewallCSR    3   VM Network, VM Network 2, VM Network
```

In the above output, the World ID of the *firewallCSR* CSR VM is 12096180. It has 3 vNIC interfaces and they are connected to the *VM Network* and *VM Network 2* virtual switches.

Now using the world id of the CSR VM found from the above command, run the following command:

```
# esxcli network vm port list -w <worldID>
```

This command displays the port information such as port ID, vSwitch name and the uplink physical NIC (referred to as vmnic in VMware terminology).

```
# esxcli network vm port list -w 1653025
Port ID: 33554509
vSwitch: vSwitch0
Portgroup: VM Network
DVPort ID:
MAC Address: 00:50:56:9c:48:23
IP Address: 0.0.0.0
Team Uplink: vmnic7
Uplink Port ID: 33554434
Active Filters:

Port ID: 33554510
vSwitch: vSwitch0
Portgroup: VM Network
DVPort ID:
MAC Address: 00:50:56:9c:64:a3
IP Address: 0.0.0.0
Team Uplink: vmnic7
Uplink Port ID: 33554434
Active Filters:
```

These same details can also be obtained from vCenter as shown in Figure 5. They are present at:  
Hosts > Manage > Networking > Virtual switches

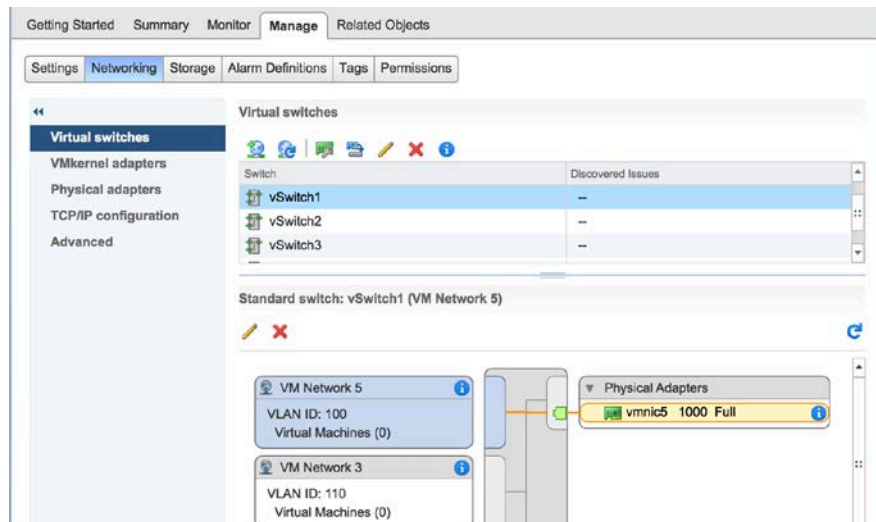


Figure 5: Networking Details of ESXi Host using vCenter

Once the vmnic of the VM is found, execute the following command to see the packet statistics for that vmnic (physical NIC):

```
# esxcli network nic stats get -n <vmnic>
```

This command displays information such as packets received, bytes received, and packets dropped and received errors. This will help to identify if there are any drops happening at the NIC.

```
# esxcli network nic stats get -n vmnic7
NIC statistics for vmnic7
  Packets received: 80419393
  Packets sent: 27482589
  Bytes received: 94073700825
  Bytes sent: 5502105043
  Receive packets dropped: 0
  Transmit packets dropped: 0
  Total receive errors: 0
  Receive length errors: 0
  Receive over errors: 0
  Receive CRC errors: 0
  Receive frame errors: 0
  Receive FIFO errors: 0
  Receive missed errors: 0
  Total transmit errors: 0
  Transmit aborted errors: 0
  Transmit carrier errors: 0
  Transmit FIFO errors: 0
  Transmit heartbeat errors: 0
  Transmit window errors: 0
```

Error statistics will help in understanding the cause of these drops. If any **Receive Missed** errors are observed, this means that the NIC is dropping the received packets because the device driver is not fast enough to retrieve these received packets and post new buffers. Increasing the receive ring buffer size to 4K might help. This can be achieved using the following command:

```
# ethtool -G <vmnic> rx <size>
```

If this does not help, check the version of driver using the following command:

```
# ethtool -i <vmnic>
```

```
# ethtool -i vmnic1
driver: igb
version: 2.1.11.1
firmware-version: 1.6-3
bus-info: 0000:09:00.1
~ #
```

From the VMware support site, check whether an updated version of this driver is available. Sometimes upgrading the driver helps in resolving an issue.

These same details of packet statistics can also be obtained from the vCenter as shown in Figure 6. They are present at:

Hosts > Monitor > Performance > Advanced

Here using the chart options, one can select the chart metrics as Network, object type as VM's vmnic and other desired counters such as data received rate, data receive rate and packet transmit errors.

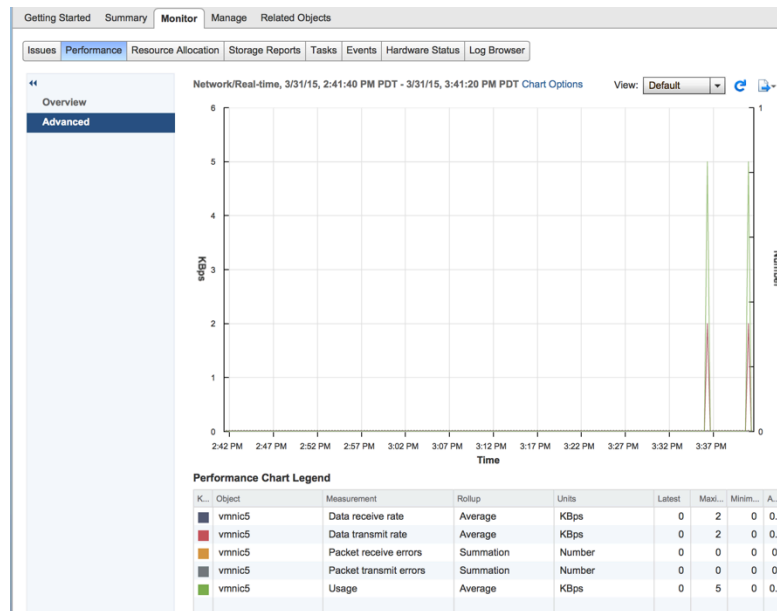


Figure 6: Network Statistic of NIC using vCenter

The next place to investigate for packet drops is in the vSwitch of this VM. Check if the packets are being received at vmnic to vSwitch ingress port, (Point 1 in Figure 4) and at vSwitch to vNIC ingress (Point 2 in Figure 4) by running the following command for both the port IDs (obtained earlier).

```
# esxcli network port stats get -p <portID>
```

```
# esxcli network port stats get -p 33554510
Packet statistics for port 33554510
Packets received: 1341721
Packets sent: 1590633
Bytes received: 281401865
Bytes sent: 294193942
Broadcast packets received: 1129705
Broadcast packets sent: 186641
Multicast packets received: 212006
Multicast packets sent: 1403992
Unicast packets received: 10
Unicast packets sent: 0
Receive packets dropped: 465
Transmit packets dropped: 0
```

This command displays detailed packet statistics such as packets received and receive packets dropped. If any receive packet drops are present, these are packets that are dropped between the virtual switch and the guest operating system driver when the virtual machine's network driver runs out of receive (RX) buffer memory.

One of the causes for the packet drops might be due to the high CPU Ready Time for the VM. The CPU Ready Time is the time that the VM waits in a ready-to-run state on one or more of the physical CPUs by the hypervisor (i.e. the VM is ready to run but the hypervisor is not able to schedule it on a physical processor). A high value is an indication of poor performance. This value can be found by running following command at host:

```
# esxtop
```



ID	GID	NAME	NWLD	%USED	%RUN	%SYS	%WAIT	%VMWAIT	%RDY	%IDLE	%OVRLP	%CSTP	%MLMTD
2810379	2810379	csrTest1	10	17.27	22.52	0.44	947.29	1.51	0.91	364.28	0.19	0.00	0.00
18600926	18600926	windowsVM	10	17.00	18.28	0.21	951.91	1.30	0.65	368.70	0.16	0.00	0.00
23455162	23455162	firewallCSR	10	14.78	20.98	0.16	949.14	1.70	0.66	366.07	0.08	0.00	0.00
3152322	3152322	ubuntu	6	9.51	8.60	0.00	573.81	0.00	0.10	89.72	0.00	0.00	0.00

A value greater than zero indicates that the CSR VM is not getting enough physical CPU resources to run. This may lead to packet drops. Reducing the number of virtual machines provisioned on the ESXi host may help in alleviating this condition.

These same details can also be obtained from vCenter as shown in Figure 6 below.

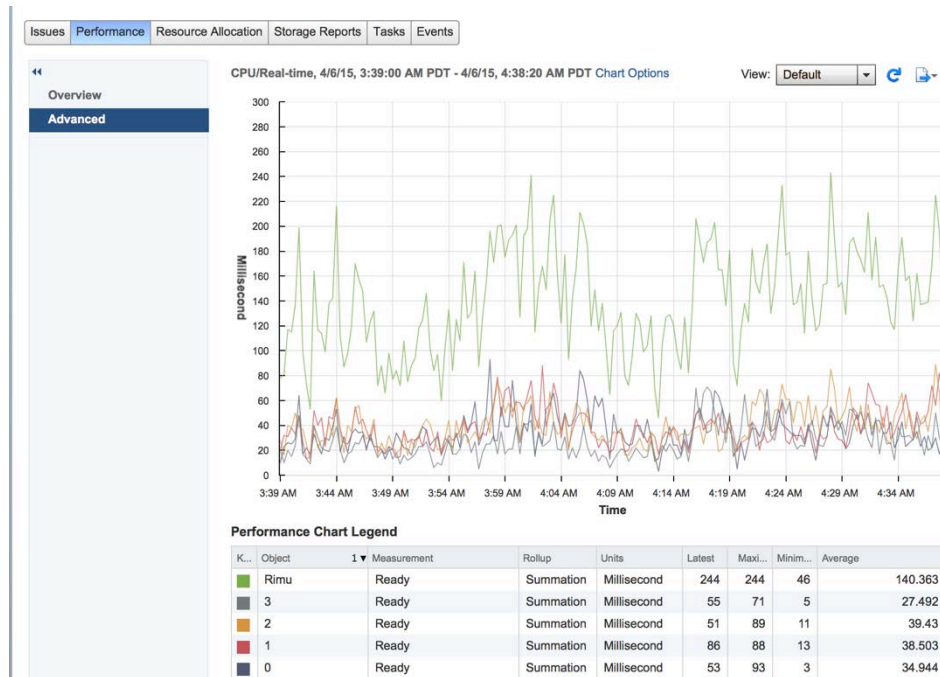


Figure 7: CPU Ready Time for VM using vCenter

They are present at:

Hosts > VM > Monitor > Performance > Advanced

Here, using the chart options, one can select chart metrics as CPU, object type as desired VM and other desired counter as Ready.

This chart displays CPU Ready time as a summation of milliseconds of CPU ready time for the sampling period (20 sec). To calculate the Ready Timer percentage, use the following formula:

$$\text{CPU Ready \%} = (\text{CPU Ready Summation in milliseconds} / (\text{Chart Default Update Interval in Seconds} * 1000 * \text{Number of vCPUs in VM})) * 100$$

Sometimes capturing packets can also provide insight on why packet drops/errors are happening.

Live capture of packets on a specific NIC can be done using the following command:

```
# pktcap-uw --uplink <vmnic>
```

```
# pktcap-uw --uplink vmnic7
The name of the uplink is vmnic7
No server port specified, select 63070 as the port
Output the packet info to console.
Local CID 2
Listen on port 63070
Accept...Vsock connection from port 1025 cid 2
14:39:44.53052[1] Captured at EtherswitchDispath point, TSO not enabled,
Checksum not offloaded and verified, length 66.
    Segment[0] ---- 66 bytes:
    0x0000:  000a f729 6a8e 0013 5f22 0f8a 0800 4500
    0x0010:  0034 274b 4000 3606 57f6 ab46 3b26 ac1b
    0x0020:  32fb b479 0016 4b3b 540f 0a47 eadb 8010
    0x0030:  003b alaf 0000 0101 080a ela8 66a4 12a8
    0x0040:  6b5d
14:39:44.53117[2] Captured at EtherswitchDispath point, TSO not enabled,
Checksum not offloaded and verified, length 66.
    Segment[0] ---- 66 bytes:
    0x0000:  000a f729 6a8e 0013 5f22 0f8a 0800 4500
    0x0010:  0034 274c 4000 3606 57f5 ab46 3b26 ac1b
```

Similarly to view the packets on a particular vSwitch port for a virtual machine, use the `--switchport` option as shown below:

```
# pktcap-uw --switchport <portID>
```

```
# pktcap-uw --switchport 33554437
The switch port id is 0x02000005
No server port specified, select 2954 as the port
Output the packet info to console.
Local CID 2
Listen on port 2954
Accept...Vsock connection from port 1027 cid 2
15:10:10.197433[1] Captured at PortInput point, TSO not enabled, Checksum
offloaded and not verified, length 66.
    Segment[0] ---- 66 bytes:
    0x0000:  0000 0c07 ac02 000c 2940 60b6 0800 4500
    0x0010:  0034 dad6 4000 4006 4598 ac1b 32fc 0a9a
    0x0020:  30a4 24e3 c146 509a 31fc a40b 61c4 8010
    0x0030:  0093 1a7c 0000 0101 050a a40b 61c3 a40b
    0x0040:  61c4
```

If packets capture needs to be saved in a file, execute the above `pktcap-uw` commands with `-o` option as shown in an example below.

```
# pktcap-uw --switchport <portID> -o <outputfile.pcap>
```

Now the next step is to check if the packets are being received by the virtual interface in the CSR VM. Refer to the Section “Packet Flow in CSR VM” of this document for details.

### 5.2.1.1 Configuring promiscuous mode on vSwitch

As described earlier, the VMware vSwitch is not a learning switch. For security reasons, only packets with destination MAC address matching that of a VM vNIC interface address are forwarded to the VM. To support some of the forwarding features (e.g. EoMPLS, EVC, etc.) the CSR VM needs to receive all the incoming packets irrespective of the destination MAC address or the VLAN. To do this, the vSwitch

should allow a VM to configure a vNIC interface in promiscuous mode. Promiscuous mode is disabled by default on the vSwitch. To turn it on, execute the following command:

```
# esxcli network vswitch standard policy security set -v <vSwitch> -p true
```

Using the following command verify that the mode has been enabled.

```
# esxcli network vswitch standard policy security get -v <vSwitch>
```

```
# esxcli network vswitch standard policy security get -v vSwitch1
  Allow Promiscuous: true
  Allow MAC Address Change: true
  Allow Forged Transmits: true
```

The above output shows **Allow Promiscuous** set as **true**.

The same can be achieved using vCenter as shown in Figure 8 below:

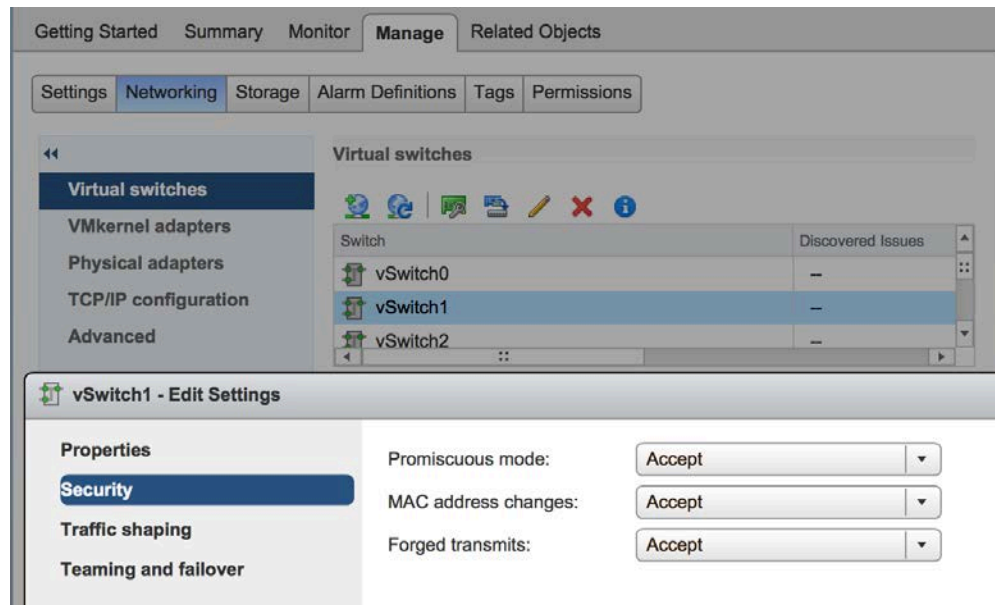


Figure 8: Setting Promiscuous mode of vSwitch using vCenter

They are present at:

Hosts > Manage > Networking > Virtual switches

Here, select the vSwitch and click **Edit Setting**. Under the Security tab set the promiscuous mode as **Accept**. By default it is set as **Reject**.

### 5.2.1.2 Configuring VLAN tagging on vSwitch

By default, a VM vNIC interface is not part of any VLANs. To support multiple VLANs on a CSR vNIC interface, the VLAN ID in the vSwitch should be set to a special value (4095). This makes the vSwitch to

pass all the packets to the CSR irrespective of the VLAN tag and without modifying the VLAN tags. This can be set using the following command:

```
# esxcli network vswitch standard portgroup set -v <vSwitch> -v 4095
```

The same can be achieved using vCenter as shown in Figure 9 below:

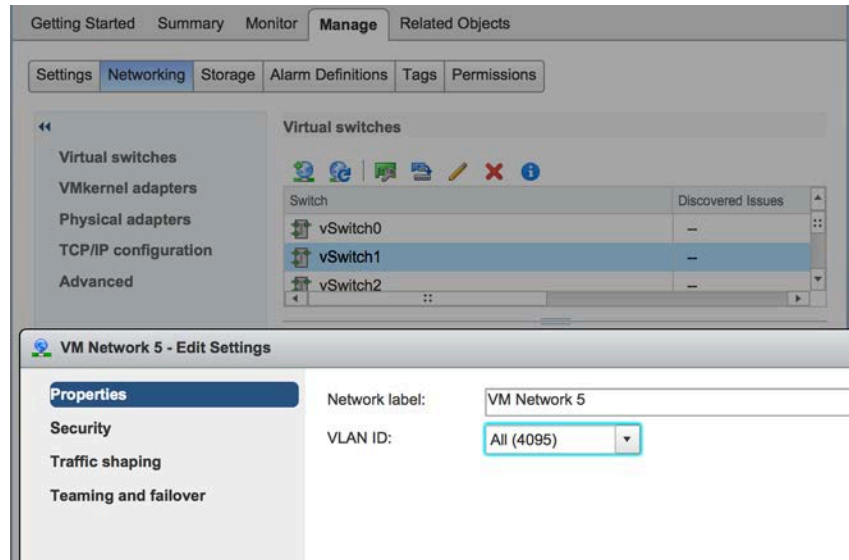


Figure 9: Setting VLAN ID of vSwitch

They are present at:

*Hosts > Manage > Networking > Virtual switches*

Here, select the vSwitch and click **Edit Setting**. Under the Properties tab make VLAN ID as **All (4095)**. By default it is set as **None (0)**.

## 5.2.2 Passthrough (VMDirectPath I/O) Mode

To increase throughput and to reduce latency, a physical NIC interface can be mapped directly to a CSR virtual machine. The Figure 10 below shows CSR VM on ESXi with NIC in passthrough mode.

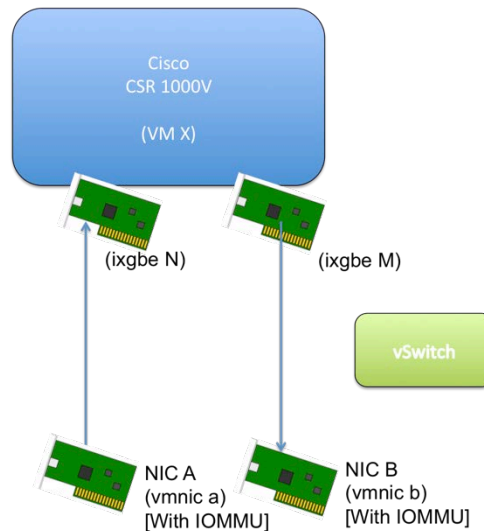


Figure 10: CSR VM in ESXi (Using NIC passthrough)

The NIC sends the received packets directly to the CSR VM bypassing the hypervisor vSwitch. For packet transmission from CSR, the CSR VM forwards the packets directly to the NIC. The packets do not traverse through vSwitch of ESXi as the physical function (PF) is directly mapped to the VM. For packet statistics related information refer Section 3 “Packet Flow in CSR VM” of this document.

Note that in this configuration, only one virtual machine can use a single physical interface. This limits the number of virtual machines that can be created on the ESXi host.

Configuring VMDirectPath I/O pass-through device on a CSR 1000V VM through vCenter is described below. It is a two-step process and CSR 1000V virtual machine should be powered off before starting the configuration. Step one is to enable passthrough for a device on ESXi host as shown in Figure 11 below.

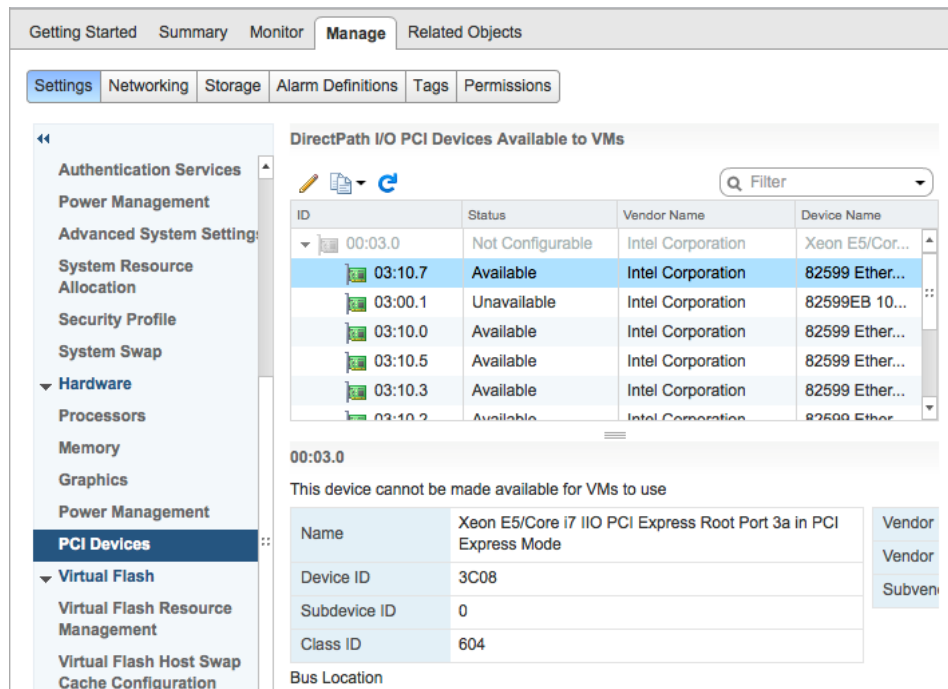


Figure 11: Enable PCI passthrough for a device on ESXi

Second step is to add that device on a CSR VM. This is done by going to the VM on vCenter GUI and selecting Virtual Hardware present at:

*Virtual Machine > Manage > Settings > VM Hardware*

Here, click **Edit** and from the **New device** drop-down menu select **PCI Device** and click **Add** as shown in Figure 12 below.

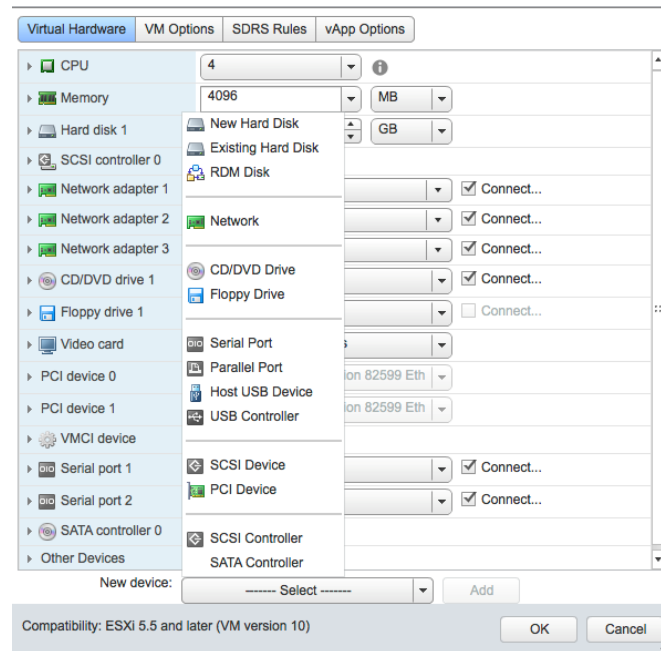


Figure 12: Configuring PCI Device on a CSR in ESXi

Also in the Memory section, set the **Limit** to **Unlimited**. After this, on powering on the CSR 1000V VM, a new interface will be shown in passthrough mode.

### 5.2.3 Single Root I/O Virtualization (SR-IOV) Mode

The Figure 13 below shows CSR VM on ESXi with NIC in SR-IOV mode:

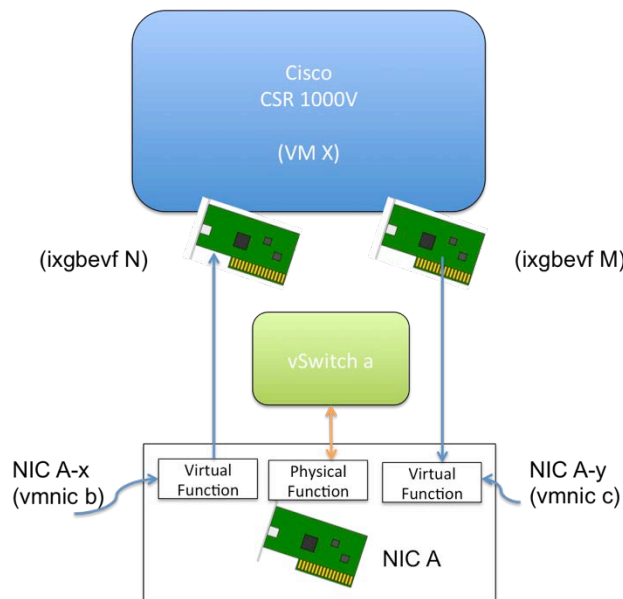


Figure 13: CSR VM in ESXi (SR-IOV mode)

Figure 13 shows packets being received on NIC A on which SR-IOV has been enabled. As a result the virtual functions (VFs) become active and appear as PCI devices. When a packet is received from a network on a PCI device (virtual function), it is forwarded to virtual NIC of CSR VM based on the

destination MAC address. Similarly, when a packet has to be transmitted from a CSR, it forwards the packet to its virtual interface. This virtual interface is mapped to a VF that transmits the packet out the network. Following command can be used to get information on SR-IOV enabled NICs.

```
# esxcli network sriovnic list
```

This command lists the SR-IOV Enabled NICs (PFs) currently installed and loaded on the host.

```
# esxcli network sriovnic vf list -n <vmnic>
```

This command lists vmnic information such as VF ID, its state, PCI Address and Owner WorldID if any.

Configuring SR-IOV device on a CSR 1000V VM through vCenter GUI is a two-step process and CSR 1000V virtual machine should be powered off before starting the configuration. Step one is to enable SR-IOV for a device on ESXi host as shown in Figure 13 below.

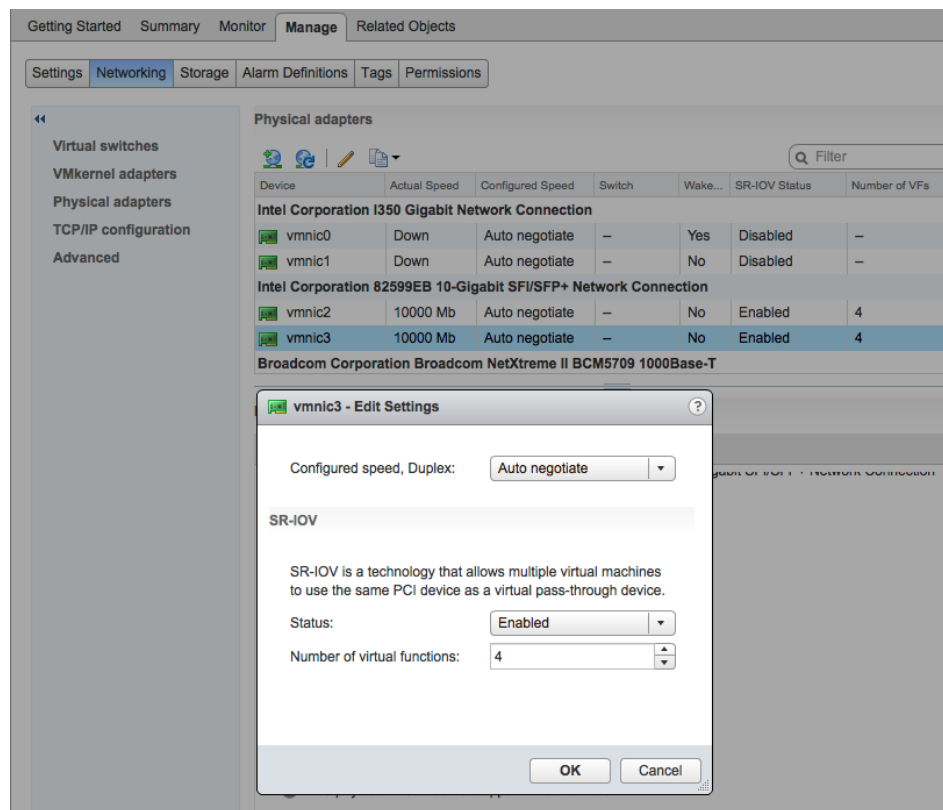


Figure 14: Enable SR-IOV on a physical adapter

This is done by going to the **Physical adapters** tab present at:

Host > Manage > Networking > Physical adapters



Select the physical adapter that must be enabled for SR-IOV and click Edit adapter settings. Then from the **SR-IOV tab** under Status drop-down menu select Enabled. Then in the Number of virtual functions section, add the number of virtual functions that need to be configured and click OK. Reboot the host.

The second step is to add a virtual function to the CSR VM. This is done by going to the VM on vCenter GUI and selecting Virtual Hardware present at:

Virtual Machine > Manage > Settings > VM Hardware

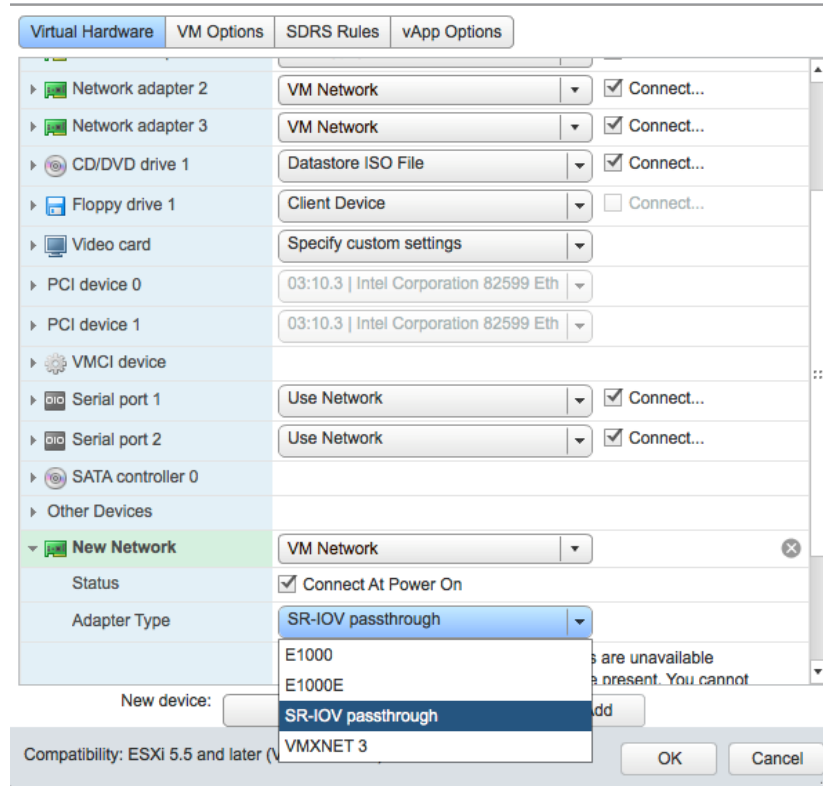


Figure 15: Adding SR-IOV device to CSR VM in ESXi

Click **Edit** and from the **New device** drop-down menu select **Network** and click **Add**. Then from **Adapter Type** drop-down menu select **SR-IOV passthrough** as shown in Figure X above. Also in the **Physical function** drop-down menu, select the physical adapter that was enabled and in the Memory section, set the **Limit** to **Unlimited**.

For packet statistics related information refer to Section 3 of this document, “Packet Flow in CSR VM.”

## 6 CSR Packet Flow in KVM

This section describes the packet flow through a CSR virtual machine in a Linux/KVM hypervisor environment. The CSR 1000V is officially supported in the RedHat Enterprise Linux (RHEL) and Ubuntu Linux/KVM environments. A CSR 1000V VM instance can be created using many different tools like virt-manager, virt-install, qemu-kvm, virsh in these environments. The CSR VM instance can also be created

using an orchestrator like OpenStack or oVirt. The type of tools supported for managing virtual machines may differ between these two environments.

## 6.1 KVM I/O Models

In a Linux/KVM environment, the virtual NIC interfaces in a CSR virtual machine can be connected using a Linux bridge or an Open Virtual Switch (OVS) or a Cisco Nexus 1000v vSwitch or Macvtap. Similar to VMware ESXi, other I/O models like PCIe passthrough and SR-IOV are also supported.

### 6.1.1 Using Linux Bridge

When using a Linux bridge to connect the CSR vNIC interfaces, only the “virtio” vNIC type is supported. Other vNIC types are not supported. When creating a CSR VM instance, make sure that the device model for the vNIC interfaces is “virtio”.

The vNIC device model type can be displayed using the following IOS-XE command:

```
# show platform software interface vnic-if interface-mapping
```

If the displayed type is not “virtio”, then shutdown the CSR VM, change the type to “virtio” and then restart the CSR VM.

In a KVM environment, many of the tools refer to the virtual machines as domains. The domain name of the virtual machines can be obtained using the following command:

```
#virsh list --all
```

```
# virsh list --all
Id      Name                                State
-----
3       csrVM                               running
9       firewallCSR                         running
12      vpp                                 running
-       oirVM                              shut off
-       csr-vpp                            shut off
```

In the above output, the domain names are listed under the “Name” column.

Further information about the domains can be obtained by passing this name to other commands. For example, the following command can be used to list all the virtual interfaces of a VM:

```
#virsh domiflist <vmname>
```

```
# virsh domiflist firewallCSR
Interface  Type      Source    Model    MAC
-----
vnet3      bridge    br2       virtio    52:54:00:1e:67:7f
vnet4      bridge    br2       virtio    52:54:00:70:e2:f1
```

In the above output, the vNIC device model type is listed under the “Model” column.

When debugging packet drop issues, the first place to check for packet statistics is the physical NIC. The `ethtool` Linux command can be used to display the packet statistics.

```
# sudo ethtool -S <ethN> | grep error
```

In the output, look for any packet errors. Make sure that the incoming packets are reaching the Linux Bridge without any errors.

Next step is to check whether any received packets are dropped by the NIC. This can be done using again the `ethtool`:

```
# sudo ethtool -S <ethN> | grep drop
```

Look for any drops in the incoming packets. If drops are observed, it may be due to a small receive queue size. Current queue size can be found using the following command:

```
# sudo ethtool -g <ethN>
```

```
# sudo ethtool -G eth5 rx 512
# sudo ethtool -g eth5
Ring parameters for eth5:
Pre-set maximums:
RX:                4096
RX Mini:           0
RX Jumbo:          0
TX:                4096
Current hardware settings:
RX:                512
RX Mini:           0
RX Jumbo:          0
TX:                256
-g eth5
Ring parameters for eth5:
Pre-set maximums:
RX:                4096
RX Mini:           0
RX Jumbo:          0
TX:                4096
Current hardware settings:
RX:                256
RX Mini:           0
RX Jumbo:          0
```

```
TX:                256
```

Interface	Type	Source	Model	MAC
vnet3	bridge	br2	virtio	52:54:00:1e:67:7f
vnet4	bridge	br2	virtio	52:54:00:70:e2:f1

```
# sudo ethtool -g eth5
Ring parameters for eth5:
Pre-set maximums:
RX:                4096
RX Mini:           0
RX Jumbo:          0
TX:                4096
Current hardware settings:
RX:                256
RX Mini:           0
RX Jumbo:          0
TX:                256
```

Interface	Type	Source	Model	MAC
vnet3	bridge	br2	virtio	52:54:00:1e:67:7f
vnet4	bridge	br2	virtio	52:54:00:70:e2:f1

The above example shows current receive queue size as 256 for the eth5 interface. Try increasing the receive queue size of the NIC using the following command:

```
# sudo ethtool -g <ethN> rx <queue_size>
```

In the above example receive queue size of eth5 has been changed from 256 to 512.

If drops are still happening at the physical NIC level, please check interrupts, software IRQ and core usage as described below.

For interrupts use the following command:

```
# cat /proc/interrupts | egrep -i "cpu | <ethN>"
```

This command will display the interrupts on that interface and the core processing those interrupts. Make sure that the core processing the interrupts is not overloaded.

Now for the listed cores find usage through the following command:

```
# mpstat -P <core #> <time>
```

```
# mpstat -P 5 1
12:23:39 PM CPU      %usr   %nice    %sys     %iowait  %irq   %soft  %steal   %guest
      %gnice   %idle
12:23:40 PM   0      0.00    0.00    0.00     0.00     0.00    0.00    0.00    0.00
0.00         100.00
```

If core usage is 100%( irq and soft) or idle is 0%, means packet is getting dropped due to interrupt/software IRQ drop. This might be due to multiple interrupts for interface coming on the same

core. Try distributing the load to different cores. For this check if some other tasks are running on that core (top –H then press F and G on RHEL), move the tasks to other cores.

If packets are still getting dropped, it might be due the fact that core processing received packet interrupts is loaded and not able to handle all the interrupts. Try enabling Receive-Side Scaling (RSS), i.e. multiple receive and transmit queues on that physical interface. These multiple queues get individually mapped to different processors available on the host. When packets are received on an interface, interrupts are generated for each queue that are then sent to the mapped processor. This helps in load balancing by dividing the interrupt handing on multiple CPUs instead of just one that in turn can reduce packet drops. For example on Intel NIC, if driver is running on 4 physical interfaces, use the following command to enable two queues on each interface:

```
# modprobe <driver name> RSS=2,2,2,2
```

For intermediate drop, run:

```
# sar -n EDEV 1 10
```

```
# sar -n EDEV 1 10
Average: IFACE rxerr/s txerr/s  coll/s rxdrop/s  txdrop/s txcarr/s  rxfram/s
rxfifo/s  txfifo/s
Average: br7    0.00      0.00    0.00    0.00      0.00      0.00      0.00
0.00      0.00
```

This command displays network device failure statistics for every 1 second for a total of 10 times. Check for rxdrop/txdrop and see which entity is dropping packets. If there are multiple hops between physical interface and tap interface, intermediate interfaces or the bridge may be dropping the packets.

For intermediate interfaces, check configured Receive Packet Steering (RPS) setting. RPS is similar to RSS in that it is used to direct packets to specific CPUs for processing. However, RPS is implemented at the software level.

To find number of queues on an intermediate interface, execute the following command:

```
# ls -l /sys/class/net/<intermediate_intfX>/queues/
```

Then to distribute packet-processing load to multiple CPUs, configure rps\_cpus, a bitmap, on that interface's queue.

```
# echo <rps_cpus_bitmap> > /sys/class/net/<intermediate_intfX>/queues/<rx-queue_no>/rps_cpus
```

For example if core 2 and 3 are to be set for softIRQ handling in system that has 16 cores, rps\_cpus\_bitmap is 000c(0000 0000 0000 1100).

If any interface is still dropping packets, increase transmit queue length for that interface as follows:

```
# sudo ifconfig <intfN> txqueuelen <size>
```

```
# sudo ifconfig eth4 txqueuelen 4096
```

The default length of Tx queue for “eth” interface is 1000 that is small for handling packets with high latency and/or a high throughput. Increasing the queue length to 4096 or greater might give better performance.

The next place to investigate for packet drops is in the ovs-bridge connected to this VM. For this, execute the following command:

```
# ovs-dpctl show
```

```
# system@ovs-system:
    lookups: hit:14 missed:10 lost:0
    flows: 0
```

In the above output, check whether the missed, lost and flow counters are increasing significantly. If the counters are increasing by a large numbers, try updating OVS to version 2.1 or later for supporting megaflows.

Executing the following command can check OVS version:

```
# ovs-vsctl --version
```

```
# ovs-vsctl --version
ovs-vsctl (Open vSwitch) 2.0.2
Compiled May 13 2015 18:49:53system@ovs-system:
    lookups: hit:14 missed:10 lost:0
    flows: 0
```

Megaflows feature allows the OVS to maintain a set of flow entries for established traffic by specifying only those fields that actually affect flow and wildcarding all other fields. This boosts performance as it reduces packet path discovery for known paths.

Executing the following command can enable Megaflows on OVS:

```
# ovs-appctl dpif/disable-megaflows
```

The next step is to check whether any packets are being dropped by the virtual interface in the CSR VM. The IOS “show interface” command can be used to display these packet statistics for the virtual interface. Refer to the Section “Packet Flow in CSR VM” of this document for details. If the drops are happening at the virtual interface, then it means that the CSR forwarding path is not able to keep up with the incoming traffic rate. One of the possible solutions for this is to increase the virtual interface “vnet” transmit queue length.

```
# sudo ifconfig <vnetN> txqueuelen <size>
```

```
# sudo ifconfig vnet1 txqueuelen 4096
```

The default length of Tx queue for “vnet” interface is 500 that is small for handling packets with high latency and/or a high throughput. Increasing the queue length to 4096 or greater gives better performance. One downside in increasing the interface transmit queue length is that the latency for priority packets will increase.

## 6.1.2 Passthrough Mode

In the passthrough mode, physical NIC interface is directly mapped to a CSR virtual machine. The NIC sends the received packets directly to the CSR VM bypassing the hypervisor bridge. During packet transmission from CSR, the CSR VM forwards the packets directly to the NIC. No packet related statistics are available at the host in this configuration. For packet statistics related information in CSR VM refer Section 3 “Packet Flow in CSR VM” of this document.

## 6.1.3 Single Root I/O Virtualization (SR-IOV) Mode

In the SR-IOV mode, a single PCI device appears as multiple PCI devices called as Virtual Functions (VF) on the host’s physical PCI bus. KVM assigns these VFs directly to a CSR virtual machine. The NIC sends the received packets on a VF directly to the CSR VM bypassing the hypervisor’s bridge. During packet transmission from CSR, the CSR VM forwards the packets directly to the NIC’s VF.

To gather packet statistics on a VF execute the following command:

```
# ethtool -S <ethN>
```

```
# ethtool -S eth2
NIC statistics:
  rx_packets: 59735963
  tx_packets: 81925607
  rx_bytes: 28172839298
  tx_bytes: 92325425706
  rx_broadcast: 551861
  tx_broadcast: 2363
  rx_multicast: 3873461
  tx_multicast: 125998
  :
  :
  :
  rx_queue_7_packets: 17314505
  rx_queue_7_bytes: 1993590965
  rx_queue_7_drops: 0
  rx_queue_7_csum_err: 0
  rx_queue_7_alloc_failed: 0
```

It will provide information such as packet drops and packet errors (if any).

## 6.2 NIC Information

### 6.2.1 NIC MTU

The following command will provide information about all the NICs (referred to as ethN), configured on a host.

```
# ifconfig
```

```
# ifconfig
eth1      Link encap:Ethernet  HWaddr 74:26:ac:17:36:d0
          inet6 addr: fe80::7626:acff:fe17:36d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:9877 (9.8 KB)

eth2      Link encap:Ethernet  HWaddr 18:e7:28:2f:00:56
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3498396 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5739439 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:615946723 (615.9 MB)  TX bytes:6954789195 (6.9 GB)
          Memory:ddc00000-ddd00000

virbr0    Link encap:Ethernet  HWaddr fe:54:00:db:c0:35
          inet addr:192.168.122.1 Bcast:192.168.122.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:293 errors:0 dropped:166 overruns:0 frame:0
          TX packets:154 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:34354 (34.3 KB)  TX bytes:26328 (26.3 KB)

vnet0     Link encap:Ethernet  HWaddr fe:54:00:db:c0:35
          inet6 addr: fe80::fc54:ff:fedb:c035/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:267 errors:0 dropped:0 overruns:0 frame:0
          TX packets:46728 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:34923 (34.9 KB)  TX bytes:2451043 (2.4 MB)
```

Besides this, the command also displays information such as MAC Address, received packets, transmitted packets and MTU size. MTU field will be help to identify if NIC is enabled to support jumbo size packets or not.

To enable jumbo packet support on a NIC execute the following command:

```
# ifconfig <ethN> mtu <mtuSize>
```

```
# ifconfig eth2 mtu 5000
# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CC
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:5000  Metric:1
          RX packets:3245478 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1999017 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:182752709 (174.2 MiB)  TX bytes:405883309 (387.0 MiB)
```

## 6.3 CSR VM Tuning

### 6.3.1 CPU Performance State Control

The processor P-states are processor operational states intended to reduce the processor operating frequency and voltage. When P-state is high, the processor runs at low frequency and low voltage. This can negatively impact the CSR forwarding performance, as the CSR VM may not get enough CPU



resources to run. To get the maximum throughput, the CPU should be running at maximum frequency. To verify CPU frequency, execute the following:

```
# grep -E '^model name|^cpu MHz' /proc/cpuinfo
```

```
# grep -E '^model name|^cpu MHz' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7 CPU           920  @ 2.67GHz cpu MHz
: 1600.000
```

In the above example, command displays that the CPU maximum speed is 2.67 GHz but it is running at a speed of 1600 MHz. Use the below script to change the scaling governor setting to performance:

```
# for CPUFREQ in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do [ -f $CPUFREQ ] || continue; echo -n performance > $CPUFREQ; done
```

Verify that the changes have been applied:

```
# grep -E '^model name|^cpu MHz' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7 CPU           920  @ 2.67GHz cpu MHz
: 2670.000
```

The command displays the CPU speed as 2670 MHz. To make the change permanent on the host across system reboot, for Ubuntu, modify the /etc/init.d/ondemand script:

Change this: **echo -n ondemand > \$CPUFREQ**

To this: **echo -n performance > \$CPUFREQ**

Also stop any running daemons such as 'cpuspeed', 'cpufreqd', 'powerd' that control the CPU stepping. For this execute the following command on Ubuntu:

```
# service cpuspeed stop
```

On RHEL 6.6, run:

```
# chkconfig cpuspeed off --level 1
```

## 6.3.2 vCPU Pinning

In a KVM/Linux environment, the standard Linux scheduler is used to schedule virtual machines and the vCPUs in VMs. Each vCPU in a VM is treated similar to a Linux process/thread and is scheduled independently. The Linux scheduler is not optimized for VM scheduling and doesn't understand the relationship between the multiple vCPUs in a virtual machine. The Linux scheduler can schedule the vCPU of a CSR VM across the different cores in the system. This will negatively impact the CSR forwarding performance as the context switching thrashes the processor instruction and data caches. To overcome this, the vCPUs should be manually pinned to the underlying physical cores as explained below. When pinning vCPUs to physical cores, the processor NUMA node configuration should also be taken into consideration. The vCPUs should be pinned to the processor cores from the same NUMA node.

Assigning the physical cores from the same NUMA node as the physical NICs helps in increased throughput. To find NUMA node corresponding to a NIC, first step is to find pci bus-info of that NIC. This can be obtained by running the following command on host:

```
# ethtool -i <ethN>
```

```
# ethtool -i eth1
driver: ixgbe
version: 3.15.1-k
firmware-version: 0x61b50001
bus-info: 0000:06:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

The above example shows 0000:06:00.0 as pci bus info as 0000:06:00.0 for the eth1 NIC interface. To find the NUMA node associated with this bus, use the following command:

```
# cat /sys/bus/pci/devices/<pci_bus_info>/numa_node
```

```
# cat /sys/bus/pci/devices/0000:06:00.0/numa_node
0
```

The above example shows NUMA node as 0 for the eth1 interface. Now find all the associated CPU cores within that NUMA node.

```
# numactl -H
```

```
# numactl -H
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11
node 0 size: 64162 MB
node 0 free: 15620 MB
node 1 cpus: 12 13 14 15 16 17 18 19 20 21 22 23
node 1 size: 64510 MB
node 1 free: 19620 MB
```

In above example, NUMA node 0 has 0-11 cores and NUMA node 1 has 12-23 cores.

Finally pin the CSR VM's vCPUs to the physical cores on the same NUMA node as the NIC interface using the following command:

```
# sudo virsh vcpupin <vmname> <VCPU#> <PhyCPU#>
```

```
# virsh vcpupin firewallCSR 0 3
# virsh vcpupin firewallCSR 1 4
# virsh vcpupin firewallCSR 2 5
# virsh vcpupin firewallCSR 3 6
```

The above example shows vCPUs 0,1, 2 and 3 of VM named "firewallCSR" have been pinned to physical CPUs 3,4,5 and 6 respectively.

### 6.3.3 Enabling vhost-net

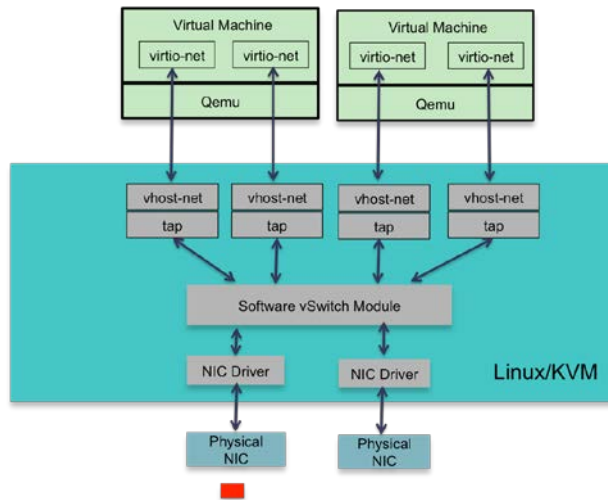


Figure 16 : CSR VM in KVM with vhost\_net

The vhost-net is a Linux kernel thread that optimizes the packet transfer into and out of a VM. Normally the QEMU userspace process emulates I/O accesses from the guest. Without vhost-net, there is context switching and packet copying between host, VM, and QEMU. vhost\_net driver emulates network device code into the kernel, that avoids system calls from the user-space. It moves virtio descriptors to SKBs conversion and back from the QEMU userspace to a kernel driver, improving the performance.

When creating the CSR VM instance, make sure to enable the vhost-net option. The vhost-net is enabled by passing the "vhost=on" command-line option to the network device.

To stabilize the throughput, pin the vhost-net thread to a specific core on the same NUMA node as the CSR VM. For this, first get the CSR VMs PID.

```
# ps -eaf | grep <vmname>
```

In above example, VM named "firewallCSR" has PID as 31230.

Then find the associated vhost for that CSR VM.

```
# ps -e | grep vhost | grep <vmid>
```

```
# ps -ef | grep vhost | grep 31230
root 31232 2 0 Feb19 ? 00:00:06 [vhost-31230]
root 31233 2 12 Feb19 ? 02:23:21 [vhost-31230]
root 31234 2 17 Feb19 ? 03:21:30 [vhost-31230]
root 31235 2 21 Feb19 ? 03:54:42 [vhost-31230]
```

The vhost-net thread is created for each interface. In the above example "firewallCSR" with PID 31230 has 4 vhost-net threads. Finally pin the vhost thread to a physical core on the same NUMA node as the CSR VM.

```
# sudo taskset -pc <PhyCPU#> <vhostPID>
```

```
# sudo taskset -pc 4 31233
# sudo taskset -pc 5 31234
```

If there are not enough available physical cores, then share the CPUs between the vhost-net and vCPU. But make sure to give vhost-net higher priority than vCPU using Linux “nice” command.

### 6.3.4 NIC Offloading

Certain NIC HW offloading features (e.g. LRO/TSO/GSO) are enabled by default that impact packet-forwarding performance. Since a router doesn't need to support Large Segment Offload (LRO)/TCP Segmentation Offload (TSO)/Generic Segmentation Offload (GSO) it has been observed that there is a performance boost if these offloading features are disabled. NIC offloading can be disabled using ethtool.

```
# sudo ethtool -K <vnetN> tso off gso off gro off
```

## 7 CSR Packet Flow in Citrix XenServer

This section describes the packet flow through a CSR virtual machine in a Citrix XenServer hypervisor environment. The XenServer commands to display the packet statistics along the packet path are also described in this section.

### 7.1 NIC Information

#### 7.1.1 NIC MTU

The following command will provide information about all the NICs (referred to as ethN), configured on a host.

```
# ifconfig
```

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CA
          UP BROADCAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

eth2      Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CC
          UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
          RX packets:3203598 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1958336 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:177187730 (168.9 MiB) TX bytes:401176914 (382.5 MiB)

vif5.0    Link encap:Ethernet  HWaddr FE:FF:FF:FF:FF:FF
          UP BROADCAST RUNNING NOARP PROMISC MTU:1500 Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:28 errors:0 dropped:5 overruns:0 carrier:0
          collisions:0 txqueuelen:32
          RX bytes:1759 (1.7 KiB) TX bytes:2041 (1.9 KiB)
```

```
vif5.1    Link encap:Ethernet  HWaddr FE:FF:FF:FF:FF:FF
          UP BROADCAST RUNNING NOARP PROMISC  MTU:1500  Metric:1
          RX packets:17159 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52109 errors:0 dropped:803 overruns:0 carrier:0
          collisions:0 txqueuelen:32
          RX bytes:789739 (771.2 KiB)  TX bytes:4357500 (4.1 MiB)
```

Besides this, the command also displays information such as MAC Address, received packets, transmitted packets and MTU size. MTU field will be help to identify if NIC is enabled to support jumbo size packets or not.

To enable jumbo packet support on a NIC execute the following command:

```
# ifconfig <ethN> mtu <mtuSize>
```

```
# ifconfig eth2 mtu 5000
# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CC
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:5000  Metric:1
          RX packets:3245478 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1999017 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:182752709 (174.2 MiB)  TX bytes:405883309 (387.0 MiB)
```

## 7.2 XenServer I/O Models

In a Xen environment, the virtual NIC interfaces in a CSR virtual machine are usually connected to a bridge. Other I/O models like PCIe passthrough and SR-IOV are also supported.

## 7.3 Using Bridge

An example of packet forwarding from one interface to another interface on CSR VM in XenServer is shown in Figure 16.

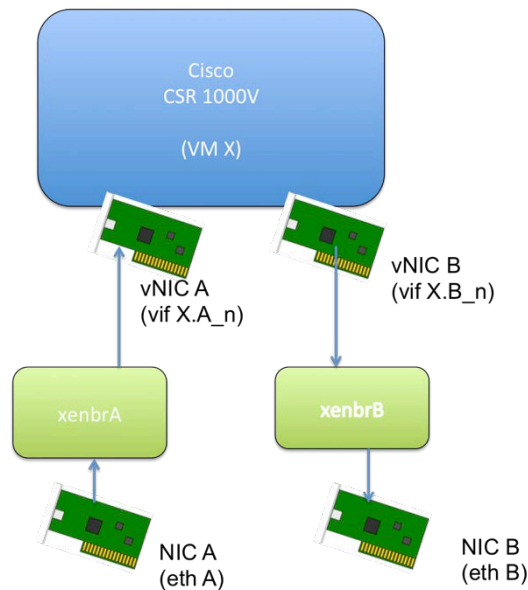


Figure 17: CSR VM in XenServer (Using xenbr)

Figure 16 shows packets being received on NIC A, forwarded by Xen hypervisor to CSR VM and then CSR routing these packets on NIC B. This example assumes Xen is using Openvswitch Bridge, xenbr.

When a packet is received from a network on a NIC, it is forwarded to the xenbr connected to that NIC and from xenbr the packet flows to virtual NIC. Similarly, when a packet has to be transmitted from a CSR, it forwards the packet to its virtual interface. This virtual interface is mapped to a xenbr that then sends out the packet on a physical NIC.

The packet statistics related information could be found out at various points in the path of a packet in and out of the CSRVM in Xen hypervisor.

To find details of networking interfaces for the VM, run the following command on the XenServer host:

```
# xl list
```

```
# xl list
Na          ID    Mem VCPUs    State    Time(s)
Domain-0    0    4082     4    r-----   129.6
CSR-DHCP-Server  1    3843     2    r-----   35.2
CSR-client  2    4095     1    r-----   17.6
```

This command returns for each virtual machine on the host, name, domain id, memory, virtual CPUs, state and total run time of the domain. Note the domain id of the CSR VM that is being investigated.

Now run the following command to get the NIC and bridge for that VM:

```
# brctl show
```

```
# brctl show
bridge name      bridge id                STP enabled  interfaces
xenbr0           0000.88f031b3edca       no           eth0
xenbr1           0000.88f031b3edcb       no           eth1
xenbr10          0000.7426aca0e021       no           eth10
xenbr11          0000.7426aca0e022       no           eth11
xenbr2           0000.88f031b3edcc       no           eth2
                                           vif1.0
                                           vif3.2
xenbr3           0000.88f031b3edcd       no           eth3
xenbr4           0000.a0369f3de8a0       no           eth4
xenbr5           0000.a0369f3de8a1       no           eth5
                                           vif3.0
xenbr6           0000.a0369f3de8a2       no           eth6
                                           vif1.1
xenbr7           0000.a0369f3de8a3       no           eth7
                                           vif1.2
                                           vif3.1
xenbr8           0000.90e2ba78b7f4       no           eth8
xenbr9           0000.90e2ba78b7f5       no           eth9
                                           vif1.3
                                           vif2.3
```

This command provides details such as bridge, physical interface and virtual interface for that domain id (VM). Xen uses bridging Dom0 to allow all domains to appear on the network. Physical NICs are available as ethN and virtual interfaces are named as vifDOMID.DEVID where DOMID is guest domain id and DEVID is index of device.

These same details can also be obtained from XenCenter as shown in Figure 17. They are present at:  
Hosts > > Networking

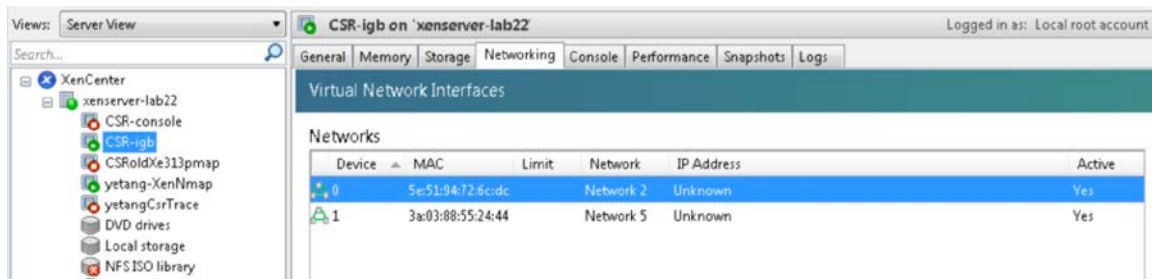


Figure 18: Networking Details of CSR VM Using XenCenter

Once the NIC and bridge id of the VM is known, execute the following command to see packet statistics for that NIC:

```
# ifconfig <ethN>
```

```
# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CC
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:2070366 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1236452 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2970958008 (2.7 GiB)  TX bytes:238994382 (227.9 MiB)
```

To get info on the bridge, run the following command:

```
# ifconfig <xenbrN>
```

```
# ifconfig xenbr2
xenbr2    Link encap:Ethernet  HWaddr 88:F0:31:B3:ED:CC
          inet addr:172.25.211.123  Bcast:172.25.211.127  Mask:255.255.255.128
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:2073917 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1150487 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2971378834 (2.7 GiB)  TX bytes:234420619 (223.5 MiB)
```

These commands provide info as packet received, packets dropped and packet errors. Further data into reason for errors can be found by executing following command:

```
# ethtool -S <ethN>
```

```
# ethtool -S eth2
NIC statistics:
  rx_packets: 2073823
  tx_packets: 1240240
  rx_bytes: 2979711454
  tx_bytes: 245524521
  rx_broadcast: 2858
  tx_broadcast: 3223
  rx_multicast: 8451
  tx_multicast: 3
  multicast: 8451
  collisions: 0
  rx_crc_errors: 0
  rx_no_buffer_count: 0
  rx_missed_errors: 0
  tx_aborted_errors: 0
  tx_carrier_errors: 0
  tx_window_errors: 0
  tx_abort_late_coll: 0
  tx_deferred_ok: 0
  tx_single_coll_ok: 0
  tx_multi_coll_ok: 0
  tx_timeout_count: 0
  rx_long_length_errors: 0
  rx_short_length_errors: 0
  rx_align_errors: 0
  tx_tcp_seg_good: 9669
  tx_tcp_seg_failed: 0
  rx_flow_control_xon: 0
  rx_flow_control_xoff: 0
  tx_flow_control_xon: 0
  tx_flow_control_xoff: 0
  rx_long_byte_count: 2979711454
  tx_dma_out_of_sync: 0
  lro_aggregated: 0
  lro_flushed: 0
  lro_recycled: 0
  tx_smbus: 0
  rx_smbus: 0
  dropped_smbus: 0
```



```

os2bmc_rx_by_bmc: 0
os2bmc_tx_by_bmc: 0
os2bmc_tx_by_host: 0
os2bmc_rx_by_host: 0
rx_errors: 0
tx_errors: 0
tx_dropped: 0
rx_length_errors: 0
rx_over_errors: 0
rx_frame_errors: 0
rx_fifo_errors: 0
tx_fifo_errors: 0
tx_heartbeat_errors: 0
tx_queue_0_packets: 1240240
tx_queue_0_bytes: 240560447
tx_queue_0_restart: 0
rx_queue_0_packets: 2073823
rx_queue_0_bytes: 2971416162
rx_queue_0_drops: 0
rx_queue_0_csum_err: 0
rx_queue_0_alloc_failed: 0

```

To get information on the virtual interface of that CSR, run the following command:

```
# ifconfig <vif X.a>
```

```

# ifconfig vif1.0
vif1.0    Link encap:Ethernet  HWaddr 74:26:AC:A0:E0:21
          UP BROADCAST RUNNING NOARP PROMISC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:480 errors:0 dropped:13536 overruns:0 carrier:0
          collisions:0 txqueuelen:32
          RX bytes:0 (0.0 b)  TX bytes:45240 (44.1 KiB)

```

This command displays information such as MAC Address, received packets, received packets dropped, transmitted packets, transmitted packets dropped and MTU size. MTU field helps to identify if virtual interface is enabled to support jumbo size packets or not.

To enable jumbo packet support on a “vif” execute the following command on XenServer:

```
# ifconfig <vif X.a> mtu <mtuSize>
```

```

# ifconfig vif1.1 mtu 5000
# ifconfig vif1.1
vif1.1    Link encap:Ethernet  HWaddr 74:26:AC:A0:E0:21
          UP BROADCAST RUNNING NOARP PROMISC  MTU:5000  Metric:1
          RX packets:39 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:32
          RX bytes:2440 (2.3 KiB)  TX bytes:0 (0.0 b)

```

Packet related information for all the configured virtual interfaces on all the VMs could also be obtained by executing the following command:

```
# xentop
```

```
# xentop - 05:50:09 Xen 4.1.5
6 domains: 2 running, 4 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 134004196k total, 25557704k used, 108446492k free CPUs: 20 @ 3000MHz
NAME STATE CPU(sec) CPU(%) MEM(k) MEM(%) MAXMEM(k) MAXMEM(%)
VCPUS NETS NETTX(k) NETRX(k) VBDS VBD_OO VBD
_RD VBD_WR VBD_RSECT VBD_WSECT SSID
CSR-client --b--- 68557 19.3 3936224 2.9 4232192 3.2
2 1 0 0 2 0 61
679 71058 1032510 661736 0
VCPUs(sec): 0: 27193s 1: 41364s
Net0 RX: 0bytes 0pkts 0err 0drop TX:
0bytes 0pkts 0err 564drop
CSR-DHCP-S --b--- 68602 20.0 3936224 2.9 4232192 3.2
2 4 61 104 2 0 58
433 66931 1029092 1233968 0
VCPUs(sec): 0: 27347s 1: 41255s
Net0 RX: 0bytes 0pkts 0err 0drop TX:
62090bytes 721pkts 0err 1576848drop
Net1 RX: 35515bytes 564pkts 0err 0drop TX:
0bytes 0pkts 0err 0drop
Net2 RX: 35515bytes 564pkts 0err 0drop TX:
480bytes 8pkts 0err 0drop
Net3 RX: 35515bytes 564pkts 0err 0drop TX:
0bytes 0pkts 0err 0drop
```

## 7.3.1 CSR VM Tuning

### 7.3.1.1 Vif TX Queue Length

The "vif" interface TX queue length that transmits packets to the CSR VM has a default length of 32 that is too low to handle the large, varying CSR packet forwarding latencies. Changing the default to 4096 or greater yields high performance gain for 0.01% drop rates. This can be achieved using the following command:

```
# ifconfig <vif X.a> txqueuelen <size>
```

```
# ifconfig vif2.3 txqueuelen 4096
```

### 7.3.1.2 vCPU Pinning

Xen allows specific vCPU's to be explicitly mapped to a given physical core that helps in performance by maximizing both instruction and data cache hits and minimizing vCPU context-switching overheads. It is recommended to separate XenServer Dom0 to a dedicated pool of physical cores that are different from the CSR VM's physical core resources. Also assigning CSR VM's the same CPU node as the Dom0 vCPUs helps in increased throughput. To do this, first step is to get the physical CPU core topology on Xen host. This can be obtained by running the following command:

```
# xenpm get-cpu-topology
```

```
# xenpm get-cpu-topology
CPU core socket node
CPU0 0 0 0
CPU1 1 0 0
CPU2 2 0 0
CPU3 3 0 0
CPU4 4 0 0
```

CPU5	8	0	0
CPU6	9	0	0
CPU7	10	0	0
CPU8	11	0	0
CPU9	12	0	0
CPU10	0	1	1
CPU11	1	1	1
CPU12	2	1	1
CPU13	3	1	1
CPU14	4	1	1
CPU15	8	1	1
CPU16	9	1	1
CPU17	10	1	1
CPU18	11	1	1
CPU19	12	1	1

The above example shows that Xen host is a two-socket machine with 12 cores per socket. It also has two NUMA nodes.

The domain to CPU affinity can be obtained using the following command:

```
# xl vcpu-list
```

```
# xl vcpu-list
```

Name	ID	VCPU	CPU	State	Time(s)	CPU Affinity
Domain-0	0	0	0	-b-	818.8	0
Domain-0	0	1	1	-b-	941.3	1
Domain-0	0	2	2	r--	734.2	2
Domain-0	0	3	3	-b-	775.4	3
CSR-DHCP-Server	1	0	4	-b-	2038.0	4-19
CSR-DHCP-Server	1	1	14	-b-	2950.8	4-19
CSR-client	2	0	9	-b-	1985.6	4-19
CSR-client	2	1	7	-b-	2927.9	4-19

vCPU pinning can be done using the following command:

```
# xl -v vcpu-pin <Domain-ID> <VCPU#> <PhyCPU#>
```

```
# xl -v vcpu-pin 1 0 6
# xl -v vcpu-pin 1 1 7
# xl vcpu-list
```

Name	ID	VCPU	CPU	State	Time(s)	CPU Affinity
Domain-0	0	0	0	-b-	826.3	0
Domain-0	0	1	1	-b-	949.8	1
Domain-0	0	2	2	r--	741.7	2
Domain-0	0	3	3	-b-	783.2	3
CSR-DHCP-Server	1	0	6	-b-	2060.3	6
CSR-DHCP-Server	1	1	7	-b-	2984.2	7
CSR-client	2	0	9	-b-	2007.2	4-19
CSR-client	2	1	5	-b-	2960.9	4-19

### 7.3.1.3 NIC Offloading

Certain NIC HW Offloading features are enabled by default that impact packet forwarding performance. It has been observed in XenServer that there is a performance boost if certain offloading features are disabled. NIC offloading can be disabled using `ethtool`.

```
# ethtool -K ethN tso off gso off gro off
# ethtool -K xenbrN sg off tso off
```

## 7.4 Passthrough Mode

To increase throughput and to reduce latency, a physical NIC interface can be mapped directly to a CSR virtual machine. The Figure 18 below shows CSR VM on XenServer with NIC in passthrough mode:

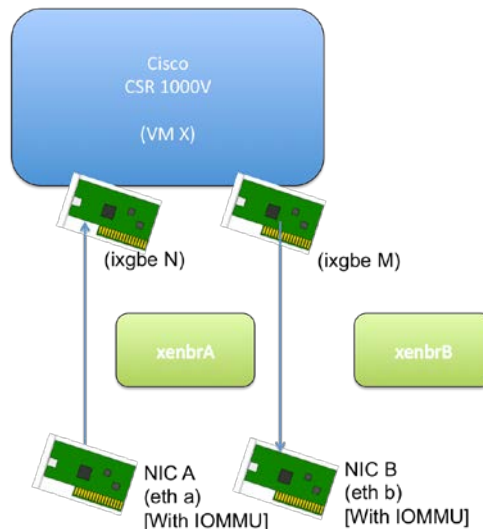


Figure 19: CSR VM in XenServer (Using NIC passthrough)

The NIC sends the received packets directly to the CSR VM bypassing the hypervisor bridge. For packet transmission from CSR, the CSR VM forwards the packets directly to the NIC. The packets do not traverse through bridge of XenServer as the physical function (PF) is directly mapped to the VM. For packet statistics related information refer Section 3 “Packet Flow in CSR VM” of this document.

Note that in this configuration, only one virtual machine can use a single physical interface. This limits the number of virtual machines that can be created on the XenServer host.

To configure a device for passthrough on XenServer first identify the BDF of the device. Executing the following command can do this:

```
# lspci | grep Ethernet
```

```
# lspci | grep Ethernet
02:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
02:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
02:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
02:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
04:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
04:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
04:00.2 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
04:00.3 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
04:10.2 Ethernet controller: Intel Corporation I350 Ethernet Controller Virtual Function (rev 01)
04:10.6 Ethernet controller: Intel Corporation I350 Ethernet Controller Virtual Function (rev 01)
06:00.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection (rev 01)
06:00.1 Ethernet controller: Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection (rev 01)
0d:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2)
0e:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2)
```

Next identify the UUID of the VM by running the following command:

```
# xe vm-list
```

```
# xe vm-list
uuid ( RO) : 2d6a5f12-ba0b-4691-8921-6c99df7f138e
  name-label ( RW): Control domain on host: xenserver-lab22
  power-state ( RO): running

uuid ( RO) : 76364ac2-f407-3735-9fba-c5a4dc71840e
  name-label ( RW): CSR-client
  power-state ( RO): running

uuid ( RO) : dadac879-ad80-9354-2af6-43b6653c1577
  name-label ( RW): CSR-DHCP-Server
  power-state ( RO): running
```

Finally, to configure a NIC as passthrough on the CSR VM, execute the following:

```
# xe vm-param-set other-config:pci=<NIC BDF> uuid=<VMs UUID>
```

```
# xe vm-param-set other-config:pci=0/0000:06:00.0 uuid=dadac879-ad80-9354-2af6-43b6653c1577
```

In passthrough mode, physical NIC interface is directly mapped to a CSR virtual machine. The NIC sends the received packets directly to the CSR VM bypassing the hypervisor bridge. During packet transmission from CSR, the CSR VM forwards the packets directly to the NIC. No packet related statistics are available at host in this configuration. For packet statistics related information in CSR VM refer Section 3 “Packet Flow in CSR VM” of this document.

## 7.5 Single Root I/O Virtualization (SR-IOV) Mode

Ethtool stat can be used to measure stats.

## 8 Acronyms

- **ASR:** Aggregation Services Router
- **BQS:** Buffer, Queuing and Scheduling
- **CLI:** Command Line Interface
- **CSR:** Cisco Cloud Services Router
- **HQF:** Hierarchical Queuing Framework
- **KVM:** Kernel Virtual Machine
- **IOS:** Cisco Internetwork Operating System
- **ISR:** Integrated Services Router
- **MTU:** Maximum Transmission Unit
- **PCIe:** Peripheral Component Interface Express
- **PF:** Physical Function (SR-IOV)
- **PPE:** Packet Processing Element
- **QFP:** Quantum Flow Processor
- **SR-IOV:** Single Root IO Virtualization
- **vCPU:** Virtual CPU
- **VF:** Virtual Function (SR-IOV)
- **VM:** Virtual Machine
- **vNIC:** Virtual Network Interface Card
- **vSwitch:** Virtual Switch
- **VT:** Intel Virtualization Technology

## 9 References

1. The CPU Scheduler in VMware vSphere® 5.1, Performance Study.  
<https://www.vmware.com/files/pdf/techpaper/VMware-vSphere-CPU-Sched-Perf.pdf>
2. VMware Virtual Networking Concepts.  
[http://www.vmware.com/files/pdf/virtual\\_networking\\_concepts.pdf](http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf)
3. Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs.  
<http://www.vmware.com/files/pdf/techpaper/VMW-Tuning-Latency-Sensitive-Workloads.pdf>
4. Performance Best Practices for VMware vSphere® 5.5.  
[https://www.vmware.com/pdf/Perf\\_Best\\_Practices\\_vSphere5.5.pdf](https://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.5.pdf)
5. VMware vSphere® Distributed Switch Best Practices.  
<http://www.vmware.com/files/pdf/techpaper/vsphere-distributed-switch-best-practices.pdf>
6. Network Function Virtualization: Packet Processing Performance of Virtualized Platforms with Linux\* and Intel® Architecture®. [https://networkbuilders.intel.com/.../network\\_builders\\_RA\\_NFV.pdf](https://networkbuilders.intel.com/.../network_builders_RA_NFV.pdf)