Chelsio Communications

Accelerate

Chelsio DPDK Driver for Linux

Installation and User's Guide

Chelsio Communications (Headquarters)
209 North Fair Oaks Avenue,
Sunnyvale, CA 94085
U.S.A

www.chelsio.com

Tel: 408.962.3600
Fax: 408.962.3661

Chelsio (India) Private Limited
Subramanya Arcade, Floor 3, Tower B
No. 12, Bannerghatta Road,
Bangalore-560029
Karnataka,
India

Tel: +91-80-4039-6800
Fax: +91-80-4039-6807

Chelsio KK (Japan)
Yamato Building 8F,
5-27-3 Sendagaya,
Shibuya-ku,
Tokyo 151-0051,
Japan

Sales
For all sales inquiries please send email to sales@chelsio.com

Support
For all support related questions please send email to support@chelsio.com

Document History

Version	Revision Date
1.0.0	07/08/2015
1.0.1	10/26/2015
1.0.2	04/01/2016
1.0.3	07/12/2016
1.0.4	12/30/2016
1.0.5	01/30/2017
1.0.6	02/24/2017
1.0.7	05/04/2017
1.0.8	05/23/2017

TABLE OF CONTENTS

1. Introduction

Thank you for choosing Chelsio Unified Wire adapters. These high speed, single chip, single firmware cards provide enterprises and data centers with high performance solutions for various Network and Storage related requirements.

The **Terminator** series is Chelsio's next generation of highly integrated, hyper-virtualized 1/10/25/40/50/100GbE controllers. The adapters are built around a programmable protocol-processing engine, with full offload of a complete Unified Wire solution comprising NIC, TOE, iWARP RDMA, iSCSI, FCoE and NAT support. It scales to true 100Gb line rate operation from a single TCP connection to thousands of connections, and allows simultaneous low latency and high bandwidth operation thanks to multiple physical channels through the ASIC.

Ideal for all data, storage and high performance clustering applications, the Unified Wire adapters enable a unified fabric over a single wire by simultaneously running all unmodified IP sockets, Fibre Channel and InfiniBand applications over Ethernet at line rate.

Designed for deployment in virtualized data centers, cloud service installations and high performance computing environments, Chelsio adapters bring a new level of performance metrics and functional capabilities to the computer networking industry.

This document describes the installation, use and maintenance of the DPDK driver and its various components.

1.1. Features

Chelsio Data Plane Development Kit (DPDK) driver package is a collection of data plane libraries and NIC drivers optimized for running in the Linux user space to boost packet processing.

The driver has support for:

- Multiple queues for Tx and Rx
- Receive Side Scaling (RSS)
- VLAN filtering
- Checksum offload
- Promiscuous mode
- All multicast mode
- Port hardware statistics
- Jumbo frames (only for UIO)
- Classification and Filtering (only for UIO)

Note *Chelsio DPDK driver is built using open-source DPDK driver v17.02 with added support for Chelsio adapters.*

1.2. Hardware Requirements

The following are the currently shipping Chelsio adapters that are compatible with DPDK driver:

- T62100-CR
- T62100-LP-CR
- T62100-SO-CR
- T6425-CR
- T6225-CR
- T6225-LL-CR
- T6225-SO-CR
- T580-CR
- T580-LP-CR
- T580-SO-CR
- T540-CR
- T520-CR
- T520-LL-CR
- T520-SO-CR
- T520-BT

1.3. Software Requirements

The Chelsio DPDK driver has been developed to run on 64-bit Linux platforms. Following is the list of supported distributions:

Linux Distribution	Driver/Software
RHEL 7.3, 3.10.0-514.el7	UIO, VFIO
RHEL 7.2, 3.10.0-327.el7	UIO, VFIO
RHEL 7.3, 3.10.0-514.el7.ppc64le (POWER8 LE)	VFIO
RHEL 7.2, 3.10.0-327.el7.ppc64le (POWER8 LE)	VFIO

Other kernel versions have not been tested and are not guaranteed to work.

Note *flex, bison, byacc, patch, patchutils, autoconf, automake and rpm-build packages must be present in the machine (required for libpcap installation).*

1.4. Package Contents

The Chelsio DPDK driver package consists of the following files/directories:

- **Makefile:** Makefile for building and installing Chelsio DPDK driver and tools.
- **EULA**: Chelsio's End User License Agreement.
- **src**: Source code for Chelsio DPDK driver, Kernel driver and Tools.
- **scripts**: Contains *res_hugepages.sh* shell script used to reserve Huge Page memory.
- **docs**: Support documents - README, Release Notes and User's Guide (this document) for the driver.

2. Hardware Installation

Follow these steps to install Chelsio Adapter in your system:

i. Shutdown/power off your system.
ii. Power off all remaining peripherals attached to your system.
iii. Unpack the Chelsio adapter and place it on an anti-static surface.
iv. Remove the system case cover according to the system manufacturer's instructions.
v. Remove the PCI filler plate from the slot where you will install the Ethernet adapter.
vi. For maximum performance, it is highly recommended to install the adapter into a PCIE x8/x16 slot.
vii. Holding the Chelsio adapter by the edges, align the edge connector with the PCI connector on the motherboard. Apply even pressure on both edges until the card is firmly seated. It may be necessary to remove the SFP (transceiver) modules prior to inserting the adapter.
viii. Secure the Chelsio adapter with a screw, or other securing mechanism, as described by the system manufacturer's instructions. Replace the case cover.
ix. After securing the card, ensure that the card is still fully seated in the PCIE x8/x16 slot as sometimes the process of securing the card causes the card to become unseated.
x. Connect a fiber/twinax cable, multi-mode for short range (SR) optics or single-mode for long range (LR) optics, to the Ethernet adapter or regular Ethernet cable for the 1Gb Ethernet adapter.
xi. Power on your system.
xii. Run *update-pciids* command to download the current version of PCI ID list

```
[root@host~]# update-pciids
%   Total   %  Received  %  Xferd  Average Speed   Time    Time     Time  Current
                                    Dload  Upload Total   Spent    Left  Speed
100 198k  100  198k     0    0     491k      0 --:--:-- --:--:-- --:--:-- 626k
Done.
```

xiii. Verify if the adapter was installed successfully by using the *lspci* command:

```
[root@localhost ~]# lspci | grep -i Chelsio
02:00.0 Ethernet controller: Chelsio Communications Inc T520-CR Unified Wire Ethernet Controller
02:00.1 Ethernet controller: Chelsio Communications Inc T520-CR Unified Wire Ethernet Controller
02:00.2 Ethernet controller: Chelsio Communications Inc T520-CR Unified Wire Ethernet Controller
02:00.3 Ethernet controller: Chelsio Communications Inc T520-CR Unified Wire Ethernet Controller
02:00.4 Ethernet controller: Chelsio Communications Inc T520-CR Unified Wire Ethernet Controller
02:00.5 SCSI storage controller: Chelsio Communications Inc T520-CR Unified Wire Storage Controller
02:00.6 Fibre Channel: Chelsio Communications Inc T520-CR Unified Wire Storage Controller
03:00.0 Ethernet controller: Chelsio Communications Inc T6225-CR Unified Wire Ethernet Controller
03:00.1 Ethernet controller: Chelsio Communications Inc T6225-CR Unified Wire Ethernet Controller
03:00.2 Ethernet controller: Chelsio Communications Inc T6225-CR Unified Wire Ethernet Controller
03:00.3 Ethernet controller: Chelsio Communications Inc T6225-CR Unified Wire Ethernet Controller
03:00.4 Ethernet controller: Chelsio Communications Inc T6225-CR Unified Wire Ethernet Controller
03:00.5 SCSI storage controller: Chelsio Communications Inc T6225-CR Unified Wire Storage Controller
03:00.6 Fibre Channel: Chelsio Communications Inc T6225-CR Unified Wire Storage Controller
```

For Chelsio adapters, the physical functions are currently assigned as:

- Physical functions 0 - 3: for the SR-IOV functions of the adapter
- Physical function 4: for all NIC functions of the adapter
- Physical function 5: for iSCSI
- Physical function 6: for FCoE
- Physical function 7: Currently not assigned

xiv. Install and load the appropriate network driver

xv. Finally, verify if the card is discovered:

```
cxgb4 0000:03:00.4: Chelsio T6225-CR rev 0
cxgb4 0000:03:00.4: S/N: RE39160018, P/N: 11012096003
cxgb4 0000:03:00.4: Firmware version: 1.16.29.0
cxgb4 0000:03:00.4: Bootstrap version: 255.255.255.255
```

The above outputs indicate the hardware configuration of the adapters as well as the Serial numbers.

Note

Network device names for Chelsio's physical ports are assigned using the following convention: the port farthest from the motherboard will appear as the first Ethernet network interface. However, for T5 40G adapters, the association of physical Ethernet ports and their corresponding network device names is opposite. For this adapter, the port nearest to the motherboard will appear as the first network interface.

3. Software/Driver Installation

Important *Please ensure that any existing version of DPDK driver is uninstalled before proceeding.*

i. If you haven't done already, download the driver package *Chelsio-DPDK-x.x.x.x.tar.gz*.
ii. Extract the tar ball:

```
[root@host~]# tar zxvf Chelsio-DPDK-x.x.x.x.tar.gz
```

iii. Change your working directory to *Chelsio-DPDK -x.x.x.x* directory:

```
[root@host~]# cd Chelsio-DPDK-x.x.x.x
```

iv. Compile and install the driver using one of the following options:

- To install DPDK source, testpmd and Pktgen tool only:

```
[root@host~]# make dpdk_install
```

- To install DPDK source and all the tools provided in the DPDK repository:

```
[root@host~]# make install
```

Note *The above commands will*

- *install the driver with the following build configuration:*
  arch: x86_64/ppc_64
  machine: native/power8
  execenv: linuxapp
  toolchain: gcc
- *create a target environment directory "x86_64-native-linuxapp-gcc/ppc_64-power8-linuxapp-gcc" in the driver package directory.*

Note *The pktgen application has been modified to disable packet classification, to reflect correct Rx rate as packet classification is an expensive operation.*

3.1. Firmware Update

Firmware is installed on the system, typically in `/lib/firmware/cxgb4`, and the kernel mode driver (cxgb4) will auto-load the firmware if an update is required:

Load the kernel mode NIC driver (cxgb4):

```
[root@host~]# modprobe cxgb4
```

Verify the firmware version using *ethtool*:

```
[root@host~]# ethtool -i <iface>
```

3.2. Flashing Firmware Configuration File

i. If not done already, load the kernel mode NIC driver (cxgb4):

```
[root@host~]# modprobe cxgb4
```

ii. Flash the firmware configuration file (for T6 adapters):

```
[root@host~]# cxgbtool <iface> loadcfg /lib/firmware/cxgb4/t6-config.txt
```

Note *For T5 adapters, use t5-config.txt.*

4. Software/Driver Loading

Note *Before proceeding, unload all Chelsio inbox and outbox kernel mode drivers using the following commands:*

```
[root@host~]# rmmod cxgb4
[root@host~]# rmmod csiostor
```

The driver must be loaded by the root user. Any attempt to load as a regular user will fail.

UIO Support

Follow the steps mentioned below to load DPDK driver with UIO support:

i. Disable Intel VT-d in system BIOS.
ii. Turn off Intel iommu by adding below entry in Kernel grub/grub2 menu:

```
intel_iommu=off
```

iii. Reboot the system for changes to take effect.
iv. Load the UIO module:

```
[root@host~]# modprobe uio
[root@host~]# insmod Chelsio-DPDK-x.x.x.x/src/DPDK/x86_64-native-linuxapp-
gcc/kmod/igb_uio.ko
```

VFIO Support

Follow the steps mentioned below to load DPDK driver with VFIO support:

i. Enable Intel VT-d in system BIOS.
ii. Add the following entry to grub/grub2 menu:

```
intel_iommu=on vfio_iommu_type1.allow_unsafe_interrupts=1
```

E.g.:

```
kernel /vmlinuz-3.13.0-32-generic ro root=UUID=5149fae1-c52b-42a9-a48c-
b0a70937e8fb intel_iommu=on vfio_iommu_type1.allow_unsafe_interrupts=1
rd_NO_LUKS rd_NO_LVM LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
crashkernel=128M KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
```

iii. Reboot the system for changes to take effect.
iv. Load the VFIO module:

```
[root@host~]# modprobe vfio-pci
```

5. Software/Driver Configuration and Fine Tuning

5.1. Huge Pages

5.1.1. Using script

Run the *res_hugepages.sh* shell script (copied to */sbin* during installation):

```
[root@host~]# res_hugepages.sh
```

5.1.2. Manual

i. Mount *hugetlbfs*:

```
[root@host~]# mkdir -p /mnt/huge
[root@host~]# mount -t hugetlbfs nodev /mnt/huge
```

ii. Reserve Huge Page memory manually:

- x86_64

```
[root@host~]# echo 1024 >
/sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
```

- POWERPC64

```
[root@host~]# echo 512 > /sys/devices/system/node/node0/hugepages/hugepages-
16384kB/nr_hugepages
```

In case of dual socket machines, run the above command for other CPU nodes.

5.2. Binding network ports

Note *Please make sure that adapter's physical function 4 is used, since it is assigned for all NIC functions of the adapter.*

- Run the following command to bind network ports to DPDK environment, with UIO support:

```
[root@host~]# dpdk_nic_bind.py --bind=igb_uio <PCI-ID-PF4>
```

Now, verify using:

```
[root@host~]# dpdk_nic_bind.py -status
```

Example:

```
[root@localhost ~]# dpdk_nic_bind.py --bind=igb_uio 0000:03:00.4
[root@localhost ~]# dpdk_nic_bind.py --status

Network devices using DPDK-compatible driver
============================================
0000:03:00.4 'T6225-CR Unified Wire Ethernet Controller' drv=igb_uio unused=
```

- Run the following command to bind network ports to DPDK environment, with VFIO support:

```
[root@host~]# dpdk_nic_bind.py --bind=vfio-pci <PCI-ID-PF4>
```

Now, verify using:

```
[root@host~]# dpdk_nic_bind.py -status
```

Example:

```
[root@localhost ~]# dpdk_nic_bind.py --bind=vfio-pci 0000:03:00.4
[root@localhost ~]# dpdk_nic_bind.py --status

Network devices using DPDK-compatible driver
============================================
0000:03:00.4 'T6225-CR Unified Wire Ethernet Controller' drv=vfio-pci unused=
```

5.3. Unbinding network ports

i. Run the following command to unbind network ports from DPDK environment:

```
[root@host~]# dpdk_nic_bind.py -u <PCI-ID-PF4>
```

ii. Run the following command to verify:

```
[root@host~]# dpdk_nic_bind.py –status
```

5.4. Performance Tuning

In order to auto tune the system for best performance, Chelsio recommends:

- disabling virtualization, c-state technology, Intel I/O AT and SR-IOV in the BIOS settings.
- setting the Power Profile to Maximum Performance in BIOS settings.
- installing the adapter into a PCIe Gen3 x8/x16 slot.
- using traffic with multiple tuples. The 'range' option of DPDK-Pktgen app can be used to create traffic with different 4-tuple values, so as to take advantage of RSS and hence, allow the traffic to be distributed across different queues/CPUs in Rx direction.

6. Running DPDK Test Applications

6.1. Testpmd application

The **Testpmd** application is provided as a part of the Chelsio DPDK driver package, and can be used to test the DPDK in a packet forwarding mode and also to access NIC hardware features such as Flow Director.

6.1.1. Syntax

Execute the following command to test DPDK using the **Testpmd** application:

```
[root@host~]# testpmd -c <coremask> -n <channels> -- -i --nb-cores=<cores> --txq=<TxQueue> --rxq=<RxQueue> --max-pkt-len=<PacketSize>
```

Where,

-c <coremask>	: A hexadecimal bit mask of the cores to run on. Note that core numbering can change between platforms and should be determined beforehand.
-n <channels>	: Number of memory channels per processor socket.
-i	: Enable interactive mode.
--nb-cores=<cores>	: Number of forwarding cores.
--txq=<TxQueue>	: Number of Tx queues.
--rxq=<RxQueue>	: Number of Rx queues.
--max-pkt-len=<PacketSize>	: Enable Jumbo mode and specify the packet size allowed for forwarding.

Note *In case of POWERPC64, if testpmd is not initializing, reallocate the huge pages using the setup script "setup.sh" present in <driver-package>/src/DPDK/tools/ directory.*

6.1.2. Examples

i. To run traffic using single Tx and Rx queue:

```
[root@host~]# testpmd -c f -n 4 -- -i --nb-cores=2 --txq=1 --rxq=1
```

ii. To run traffic using 4 Tx and Rx queues:

```
[root@host~]# testpmd -c 3ff -n 4 -- -i --nb-cores=8 --txq=4 --rxq=4
```

6.1.3. Flow Control

Flow control pause TX/RX is disabled by default and can be enabled via *Testpmd* as follows:

```
testpmd> set flow_ctrl rx on tx on 0 0 0 0 mac_ctrl_frame_fwd off autoneg
off 0
testpmd> set flow_ctrl rx on tx on 0 0 0 0 mac_ctrl_frame_fwd off autoneg
off 1
```

To disable again, run:

```
testpmd> set flow_ctrl rx off tx off 0 0 0 0 mac_ctrl_frame_fwd off autoneg
on 0
testpmd> set flow_ctrl rx off tx off 0 0 0 0 mac_ctrl_frame_fwd off autoneg
on 1
```

6.1.4. RSS

RSS is enabled by default and can be disabled via *Testpmd* as follows:

```
testpmd> port config all rss none
```

To enable again, run:

```
testpmd> port config all rss [all|ip|tcp|udp|sctp|ether|none]
```

6.1.5. Jumbo Mode

There are multiple ways to enable sending and receiving of jumbo frames: the first method involves using the command prompt and the other two via testpmd application.

- **Command Prompt**

Run the following command at the prompt:

```
[root@host~]# testpmd -c ffff -n 3 -- -i --nb-cores=8 --txq=4 --rxq=4 --max-
pkt-len=9018
```

- **Testpmd**

i. This method involves using the mtu command, which changes the MTU of an individual port without having to stop the selected port. To configure each port individually, run the mtu command as follows:

```
testpmd> port config mtu 0 9000
testpmd> port config mtu 1 9000
```

ii. This method involves stopping all the ports first and then running *max-pkt-len* command to configure the MTU of all the ports with a single command.To configure all the ports at once, stop all the ports first and run the *max-pkt-len* command as follows:

```
testpmd> port stop all
testpmd> port config all max-pkt-len 9000
```

6.2. Pktgen application

Pktgen is a traffic generator application capable of displaying real time metrics for ports and can handle packets with UDP, TCP, ARP, ICMP, GRE, MPLS and Queue-in-Queue. The application can run command scripts to set up repeatable test cases.

Note *The pktgen application has been modified to disable packet classification, to reflect correct Rx rate as packet classification is an expensive operation.*

6.2.1. Syntax

Execute the following command to run **Pktgen** application. Observe unidirectional pkts/sec and throughput performance under tool output:

```
[root@host~]# pktgen -c <coremask> -J -n <channels> -- -T -P -m
"<[cores]>.<port_id>" -N
```

Where,

-c <coremask>	: A hexadecimal bit mask of the cores to run on. Note that core numbering can change between platforms and should be determined beforehand.
-n <channels>	: Number of memory channels per processor socket.
-T	: Enable pktgen themes.
-P	: Enable PROMISCUOUS mode on all ports.
-m <[cores]>.<port_id>	: Map CPU logical cores to Chelsio ports.
-N	: Enable NUMA support.
-J	: Enable Jumbo mode.

6.2.2. Examples

- **Running single Tx and Rx queue traffic**

i. Configure Pktgen tool to use single Tx and Rx queue, mapping core 1 to Rx and core 2 to Tx queue on port 0:

```
[root@host~]# pktgen -c f -n 4 -- -T -P -m "[1:2].0" -N
.
.
.
Pktgen created by Keith Wiles -- >>> Powered by IntelÂ® DPDK <<<
- Ports 0-3 of 4   ** Main Page **  Copyright (c) <2010-2015>, Wind River
Systems, Inc. All rights reserved. Powered by IntelÂ® DPDK
  Flags:Port    :   P-------------:0
```

```
Link State        :       <UP-10000-FD>                        ---TotalRate---
Pkts/s  Rx        :                   0                                    0
        Tx        :            14880593                             14880593
MBits/s Rx/Tx     :              0/9999                               0/9999
Broadcast         :                   0
Multicast         :                   0
  64 Bytes        :                   0
  65-127          :                   0
  128-255         :                   0
  256-511         :                   0
  512-1023        :                   0
  1024-1518       :                   0
Runts/Jumbos      :                 0/0
Errors Rx/Tx      :                 0/0
Total Rx Pkts     :                   0
      Tx Pkts     :           175605364
      Rx MBs      :                   0
      Tx MBs      :              118006
ARP/ICMP Pkts     :                 0/0
                  :
Tx Count/% Rate   :        Forever/100%
PktSize/Tx Burst:                64/32
Src/Dest Port     :           1234/5678
Pkt Type:VLAN ID:       IPv4/TCP:0001
Dst  IP Address   :         192.168.1.1
Src  IP Address   :      192.168.0.1/24
Dst MAC Address   :  00:00:00:00:00:00
Src MAC Address   :  00:07:43:29:3c:40
-- Pktgen Ver:2.8.5(DPDK-2.1.0) ------------------------------------------
```

ii. Set packet size to 64B on all ports

```
Pktgen> set all size 64
```

iii. Start Tx traffic on port 0

```
Pktgen> start 0
```

iv. Stop Tx traffic on port 0

```
Pktgen> stop 0
```

- **Running multiple Tx and Rx queue traffic**

i. Configure Pktgen tool to use 4 Tx and Rx queues, mapping cores 1-4 to Rx and 5-8 to Tx queues, on port 0:

```
[root@host~]# pktgen -c 3ff -n 4 -- -T -P -m "[1-4:5-8].0" -N
```

ii. Set packet size to 64B on all ports

```
Pktgen> set all size 64
```

iii. Start Tx traffic on port 0

```
Pktgen> start 0
```

iv. Stop Tx traffic on port 0

```
Pktgen> stop 0
```

- **Running Tx and Rx queue traffic in Jumbo mode**

i. Configure Pktgen tool to use 4 Tx and Rx queues, mapping cores 1-4 to Rx and 5-8 to Tx queues, on port 0 in Jumbo mode:

```
[root@host~]# pktgen -c 3ff -n 4 -- -T -J -P -m "[1-4:5-8].0" -N
```

ii. Set packet size to 9018B on all ports

```
Pktgen> set all size 9018
```

iii. Start Tx traffic on port 0

```
Pktgen> start 0
```

iv. Stop Tx traffic on port 0

```
Pktgen> stop 0
```

7. Classification and Filtering

Classification and Filtering feature enhances network security by controlling incoming traffic as they pass through network interface based on source and destination addresses, protocol, source and receiving ports, or the value of some status bits in the packet. This feature can be used in the ingress path to:

- Steer packets that meet ACL (Access Control List) accept criteria to a particular receive queue.
- Switch (proxy) packets that meet ACL accept criteria to an output port, with optional header rewrite.
- Drop packets that meet ACL accept criteria.

DPDK driver currently supports LE-TCAM (*maskfull*) filters which enable specifying masks to the accept criteria. The masks will allow specifying a match for a range of values. You can create up to 496 LE-TCAM filters.

Note *This feature is currently not supported and will be fully functional in the next release.*

7.1. Pre-requisites

To use the Classification and Filtering feature, DPDK driver should be loaded with UIO support. See Software/Driver Loading for step-by-step instructions.

7.2. Usage

7.2.1. Filter Modes

The Classification and Filtering feature is configured by specifying the filter modes in the firmware configuration file, *t5-config.txt*, located in */lib/firmware/cxgb4/*

The following filter modes are supported:

fcoe : Fibre Channel over Ethernet frames
port : Packet ingress physical port number
vnic_id : VF ID in MPS TCAM (*Currently not supported*) and outer VLAN ID
vlan : Inner VLAN ID
tos : Type of Service
protocol : IP protocol number (ICMP=1, TCP=6, UDP=17, etc)
ethertype : Layer 2 EtherType
macmatch : MAC index in MPS TCAM
mpshittype : MAC address "match type" (none,unicast,multicast,promiscuous,broadcast)
fragmentation : Fragmented IP packets

7.2.2. Supported Filter Combinations

The following combination is set by default in the firmware configuration file and packets will be matched accordingly:

```
filterMode =
fcoemask,srvrsram,fragmentation,mpshittype,protocol,vlan,port,fcoe
```

You can change the default filter mode to any one of the following combinations, based on your requirement. The firmware configuration file, then needs to re-flashed on to the adapter (See Flashing Firmware Configuration File sub-section).

Important *Using any other filter mode combination is strictly not supported.*

fragmentation,mpshittype,macmatch,ethertype,protocol,port
fragmentation,mpshittype,macmatch,ethertype,protocol,fcoe
fragmentation,mpshittype,macmatch,ethertype,tos,port
fragmentation,mpshittype,macmatch,ethertype,tos,fcoe
fragmentation,mpshittype,macmatch,ethertype,port,fcoe
fragmentation,mpshittype,macmatch,protocol,tos,port,fcoe
fragmentation,mpshittype,macmatch,protocol,vlan,fcoe
fragmentation,mpshittype,macmatch,protocol,vnic_id,fcoe
fragmentation,mpshittype,macmatch,tos,vlan,fcoe
fragmentation,mpshittype,macmatch,tos,vnic_id,fcoe
fragmentation,mpshittype,macmatch,vlan,port,fcoe
fragmentation,mpshittype,macmatch,vnic_id,port,fcoe
fragmentation,mpshittype,ethertype,protocol,tos,port,fcoe
fragmentation,mpshittype,ethertype,vlan,port
fragmentation,mpshittype,ethertype,vlan,fcoe
fragmentation,mpshittype,ethertype,vnic_id,port
fragmentation,mpshittype,ethertype,vnic_id,fcoe
fragmentation,mpshittype,protocol,tos,vlan,port
fragmentation,mpshittype,protocol,tos,vlan,fcoe
fragmentation,mpshittype,protocol,tos,vnic_id,port
fragmentation,mpshittype,protocol,tos,vnic_id,fcoe
fragmentation,mpshittype,protocol,vlan,port,fcoe
fragmentation,mpshittype,protocol,vnic_id,port,fcoe
fragmentation,mpshittype,tos,vlan,port,fcoe
fragmentation,mpshittype,tos,vnic_id,port,fcoe
fragmentation,mpshittype,vlan,vnic_id,fcoe
fragmentation,macmatch,ethertype,protocol,port,fcoe
fragmentation,macmatch,ethertype,tos,port,fcoe
fragmentation,macmatch,protocol,vlan,port,fcoe

fragmentation,macmatch,protocol,vnic_id,port,fcoe
fragmentation,macmatch,tos,vlan,port,fcoe
fragmentation,macmatch,tos,vnic_id,port,fcoe
fragmentation,ethertype,vlan,port,fcoe
fragmentation,ethertype,vnic_id,port,fcoe
fragmentation,protocol,tos,vlan,port,fcoe
fragmentation,protocol,tos,vnic_id,port,fcoe
fragmentation,vlan,vnic_id,port,fcoe
mpshittype,macmatch,ethertype,protocol,port,fcoe
mpshittype,macmatch,ethertype,tos,port,fcoe
mpshittype,macmatch,protocol,vlan,port
mpshittype,macmatch,protocol,vnic_id,port
mpshittype,macmatch,tos,vlan,port
mpshittype,macmatch,tos,vnic_id,port
mpshittype,ethertype,vlan,port,fcoe
mpshittype,ethertype,vnic_id,port,fcoe
mpshittype,protocol,tos,vlan,port,fcoe
mpshittype,protocol,tos,vnic_id,port,fcoe
mpshittype,vlan,vnic_id,port

7.2.3. Test CXGBE Filters Application

The test cxgbe filters application, *test_cxgbe_filters,* provides a command line interface to configure Chelsio NIC packet classification and filtering features available in hardware. You will be presented with a prompt, *cxgbe*, which can be then used to configure filtering features. Use the following syntax to run the application:

```
[root@host~]# test_cxgbe_filters -c <coremask> -n <channels> -- -i
```

Where,

-c <coremask> : A hexadecimal bit mask of the cores to run on. Note that core numbering can change between platforms and should be determined beforehand.
-n <channels> : Number of memory channels per processor socket.
-i : Enable interactive mode.

7.2.4. Syntax

To configure filtering features, use the *filter* command as given below. Please note that all the parameters mentioned below are mandatory:

```
cxgbe> filter <port_id> <filter_operation> <ip_version> mode <filter_mode>
<priority> ingress-port <iport> <iport_mask> fcoe <fcoe_bit> <fcoe_bit_mask>
mac-match-type <match_type> mac-index <macidx> <macidx_mask> ether
<ether_type> <ether_type_mask> vlan <inner_vlan> <inner_vlan_mask>
<outer_vlan> <outer_vlan_mask> ip <frag_bit> <frag_bit_mask> <tos>
<tos_mask> <proto> <proto_mask> <src_ip_address> <src_ip_mask>
<dst_ip_address> <dst_ip_mask> <src_port> <src_port_mask> <dst_port>
<dst_port_mask> <action> queue <queue_id> <egress_port_redirect>
<egress_port> <mac_addr_rewrite> <src_mac> <dst_mac> <vlan_operation>
<new_vlan> <nat_operation> <nat_src_ip> <nat_dst_ip> <nat_src_port>
<nat_dst_port> fd_id <fd_id_value>
```

***filter* command parameters**

Key	Valid Values	Description
port_id	0,1,2,3	Chelsio port ID
filter_operation	add,del	Filter rule operation *add*: Create filter rule *del*: Delete filter rule
ip_version	ipv4,ipv6	IP version
filter_mode	maskfull,maskless	Create filter rule on LE-TCAM or Hash region. *maskfull*: Create filter rule on LE-TCAM. region *maskless*: Create filter rule on Hash region. **Note**: *maskless* mode currently not supported.
priority	no-prio,prio	Priority over hash filter *no-prio*: No priority *prio*: Enable priority **Note**: *priority* currently not supported.
iport	0,1,2,3	Ingress port ID
iport_mask	0,0x7,0x6	Ingress port mask *0*: Don't care *0x7*: Exact match 0x6: Two ports starting with port specified for *iport*
fcoe_bit	0,1	FCoE based Filtering *0*: Disable *1*: Enable
fcoe_bit_mask	0,1	FCoE traffic mask *0*: Don't care *1*: Exact match

match_type	none,unicast,multicast, promiscuous,broadcast	Filtering based on traffic type. Setting to *none* will allow all traffic types.
macidx	0-n	Filtering based on MPS TCAM Idx-ID
macidx_mask	0,1	MAC index mask *0*: Don't care *1*: Exact match
ether_type	Valid EtherType	EtherType based filtering
ether_type_mask	0,0xFFFF,range*	EtherType mask *0*: Don't care *0xFFFF*: Exact match
inner_vlan	0-4095	Inner VLAN ID
inner_vlan_mask	0,0xFFFF,range*	Inner VLAN mask *0*: Don't care *0xFFFF*: Exact match
outer_vlan	0-4095	Outer VLAN ID
outer_vlan_mask	0,0xFFFF,range*	Outer VLAN mask *0*: Don't care *0xFFFF*: Exact match
ip_frag_bit	0,1	IP fragmentation bit *0*: Don't fragment *1*: Fragmented bit
ip_frag_bit_mask	0,1	IP fragmentation bit mask *0*: Don't care *1*: Exact match
ip_tos	Valid TOS value	IP Type of Service
ip_tos_mask	0,0xFFFF,range*	IP Type of Service mask when range is provided for *ip_tos*
ip_proto	Valid protocol number	IANA assigned Protocol number
ip_proto_mask	0,0xFFFF,range*	IP protocol mask
ip_src_addr	Valid IPv4/IPv6 address.	Source IP address
ip_src_mask	Valid subnet mask.	Subnet mask
ip_dst_addr	Valid IPv4/IPv6 address.	Destination IP address
ip_dst_mask	Valid subnet mask.	Subnet mask
ip_src_port	0-65535	Source Port Number
ip_src_port_mask	0,0xFFFF,range*	Source Port Mask
ip_dst_port	0-65535	Source Port Number
ip_dst_port_mask	0,0xFFFF,range*	Destination Port Mask
action	drop,fwd,switch	Filter action on Ingress packets *drop*: Ingress packets will be dropped. *fwd*: Ingress packets will be passed through set ingress queues. *switch*: Ingress packets will be routed to an output port with optional header rewrite.
queue_id	0-n	Rx queue ID
egress_port_redirect	port-none,port-redirect	Egress Port redirection *port-none*: No redirection

		port-redirect: Redirect to the egress port
egress_port	0-3	Egress port ID
mac_addr_rewrite	ether-none,mac-rewrite, mac-swap	MAC address rewrite *ether-none*: No rewrite or swap *mac-rewrite*: Rewrite MAC address based on the values provided for *src_mac* and *dst_mac*. *mac-swap*: Interchange the MAC addresses of source and destination. **Note**: *mac-swap* currently not supported.
src_mac	Valid MAC address	Source MAC address to be changed when *mac_addr_rewrite* is set to *mac-rewrite*.
dst_mac	Valid MAC address	Destination MAC address to be changed when *mac_addr_rewrite* is set to *mac-rewrite*.
vlan_operation	vlan-none,vlan-insert, vlan-rewrite,vlan-delete	VLAN ID operation *vlan-none*: No action *vlan-insert*: Insert VLAN ID specified in *new_vlan* *vlan-rewrite*: Change VLAN ID to the one specified in *new_vlan* *vlan-delete*: Delete VLAN ID
new_vlan	0-4095	VLAN ID. Applicable when *vlan_operation* is set to *vlan-insert* or *vlan-rewrite*.
nat_operation	nat-none, nat-rewrite	NAT operation *nat-none*: No NAT *nat-rewrite*: Rewrite IP address and port **Note:** *nat_operation* currently not supported.
nat_src_ip	Valid IP address	Source IP address that needs to be translated to.
nat_dst_ip	Valid IP address	Destination IP address that needs to be translated to.
nat_src_port	0-65535	Source port that needs to be translated to.
nat_dst_port	0-65535	Destination port that needs to be translated to.
fd_id_value	0-495	Filter ID **Note:** The rule with the lowest filter ID takes the highest precedence.

* To calculate the range of possible values, please use the U32 Port Masks Calculator at http://blog.of.geek.nz/2012/07/22/u32-port-masks-calculator/

7.2.5. Creating Filter Rules

To create filter rules, use the *add* operation as given below. Please note that all the parameters mentioned are mandatory:

```
cxgbe> filter <port_id> add <ip_version> mode <filter_mode> <priority>
ingress-port <iport> <iport_mask> fcoe <fcoe_bit> <fcoe_bit_mask> mac-match-
type <match_type> mac-index <macidx> <macidx_mask> ether <ether_type>
<ether_type_mask> vlan <inner_vlan> <inner_vlan_mask> <outer_vlan>
<outer_vlan_mask> ip <frag_bit> <frag_bit_mask> <tos> <tos_mask> <proto>
<proto_mask> <src_ip_address> <src_ip_mask> <dst_ip_address> <dst_ip_mask>
<src_port> <src_port_mask> <dst_port> <dst_port_mask> <action> queue
<queue_id> <egress_port_redirect> <egress_port> <mac_addr_rewrite> <src_mac>
<dst_mac> <vlan_operation> <new_vlan> <nat_operation> <nat_src_ip>
<nat_dst_ip> <nat_src_port> <nat_dst_port> fd_id <fd_id_value>
```

Note *Packets that don't meet the filter accept criteria will be forwarded to the DPDK stack.*

Examples

- ***fcoe* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 1 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0
fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch FCoE packets from port 0 to port 1.

- ***port* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch packets received on port 0 to port 1.

- ***vlan* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 65 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch packets only if iVLAN = 65.

- ***tos* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0x30 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch packets only if TOS = 0x30.

- ***protocol* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 6 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch only TCP packets (proto = 6).

- ***ethertype* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type promiscuous mac-index 0 0 ether 0x8906 0 vlan 0 0 0 0 ip 0 0 0 0
0 0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0
0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch packets with EtherType = 0x8906.

- ***macmatch* (mac-index) filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type none mac-index 1 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch packets with MAC index = 1 in MPS TCAM table.

- ***mpshittype* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type unicast mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch only unicast packets.

- ***fragmentation* filter mode**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 1 1 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0
fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will switch only fragmented packets.

- ***drop* action**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type unicast mac-index 0 0 ether 0 0
vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 drop queue 0 port-none 1 mac-rewrite 00:00:00:00:00:00
00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will drop all packets received on port 0.

- ***fwd (forward)* action**

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type unicast mac-index 0 0 ether 0 0
vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 fwd queue 0 port-none 1 mac-rewrite 00:00:00:00:00:00
00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
```

The above filter rule will forward packets to the application using queue 0.

7.2.6. Listing Filter Rules

To list the filters set on a port, run the following command:

```
cxgbe> debug filters <port_id>
```

Example:

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0 fcoe 1 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0
fd_id 0
filter inserted at fd_id: 0
cxgbe> debug filters 0

filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                  Hits FCoE Port     vld:iVLAN  Prot MPS Frag                    LIP                     FIP     LPORT     FPORT Action
   0                     0  1/1  0/0 0:0000/0:0000 00/00 0/0  0/0          00000000/00000000       00000000/00000000 0000/0000 0000/0000 Switch
: port=1, dmac=00:00:00:00:00:00, l2tidx=0, smac=00:00:00:00:00:00, smtidx=0

Done...
```

7.2.7. Hit Counters

To verify if the filter rule set is honoured, list the filter rule and observe the hit counters (*Hits* parameter) incrementing.

```
cxgbe> debug filters 0

filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                  Hits FCoE Port     vld:iVLAN  Prot MPS Frag                    LIP                     FIP     LPORT     FPORT Action
   0              91304954 0/0  0/7 0:0000/0:0000 00/00 0/7  0/0          00000000/00000000       00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:07:43:29:1c:48, l2tidx=0, smac=00:07:43:29:15:48, smtidx=0

Done...
```

7.2.8. Removing Filter Rules

To remove a filter rule, use the *del* operation as given below. Please note that all the parameters mentioned are mandatory:

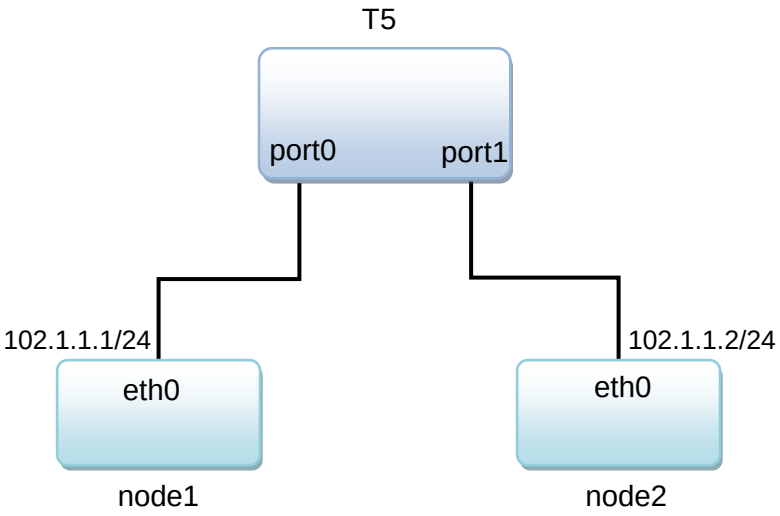
```
cxgbe> filter <port_id> del <ip_version> mode <filter_mode> <priority>
ingress-port <iport> <iport_mask> fcoe <fcoe_bit> <fcoe_bit_mask>
mac-match-type <match_type> mac-index <macidx> <macidx_mask> ether
<ether_type> <ether_type_mask> vlan <inner_vlan> <inner_vlan_mask>
<outer_vlan> <outer_vlan_mask> ip <frag_bit> <frag_bit_mask> <tos>
<tos_mask> <proto> <proto_mask> <src_ip_address> <src_ip_mask>
<dst_ip_address> <dst_ip_mask> <src_port> <src_port_mask> <dst_port>
<dst_port_mask> <action> queue <queue_id> <egress_port_redirect>
<egress_port> <mac_addr_rewrite> <src_mac> <dst_mac> <vlan_operation>
<new_vlan> <nat_operation> <nat_src_ip> <nat_dst_ip> <nat_src_port>
<nat_dst_port> fd_id <fd_id_value>
```

Example:

```
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx               Hits FCoE Port     vld:iVLAN  Prot MPS Frag                    LIP                     FIP     LPORT     FPORT Action
   0           50877269  0/0  0/7 0:0000/0:0000 00/00 0/7  0/0          00000000/00000000          00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:07:43:29:1c:48, l2tidx=0, smac=00:07:43:29:15:48, smtidx=0
   1            7915731  0/0  1/7 0:0000/0:0000 00/00 0/7  0/0          00000000/00000000          00000000/00000000 0000/0000 0000/0000 Switch: port=0,
 dmac=00:07:43:29:1c:40, l2tidx=1, smac=00:07:43:29:15:40, smtidx=1

Done...
cxgbe> filter 0 del ipv4 mode maskfull no-prio ingress-port 1 0x7 fcoe 0 0 mac-match-type unicast mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0 0 0.0.0.0
0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 0 mac-rewrite 00:07:43:29:15:40 00:07:43:29:1C:40 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd_id 1

cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx               Hits FCoE Port     vld:iVLAN  Prot MPS Frag                    LIP                     FIP     LPORT     FPORT Action
   0           91304954  0/0  0/7 0:0000/0:0000 00/00 0/7  0/0          00000000/00000000          00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:07:43:29:1c:48, l2tidx=0, smac=00:07:43:29:15:48, smtidx=0

Done...
```

7.3. Layer 2 example

Here's an example on how to achieve L2 routing functionality:



- **Configure IP addresses on node1 and node2**

node1:

```
[root@host~]# ifconfig eth0 102.1.1.1/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:04:7D:50
          inet addr:102.1.1.1  Bcast:102.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::207:43ff:fe04:7d50/64 Scope:Link
```

node2:

```
[root@host~]# ifconfig eth0 102.1.1.2/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:12:D4:88
          inet addr:102.1.1.2  Bcast:102.1.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:112:d488/64 Scope:Link
```

- **Follow these steps on machine with T5 adapter**

i. Create filter rule to switch all packets from port0 to port1:

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0
0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0
0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                  Hits FCoE Port     vld:iVLAN  Prot MPS Frag                LIP                   FIP     LPORT     FPORT Action
   0                     1  0/0  0/7 0:0000/0:0000 00/00 0/0  0/0    00000000/00000000     00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:00:00:00:00:00, l2tidx=0, smac=00:00:00:00:00:00, smtidx=0

Done...
```

ii. Create another filter rule to switch all packets from port1 to port0:

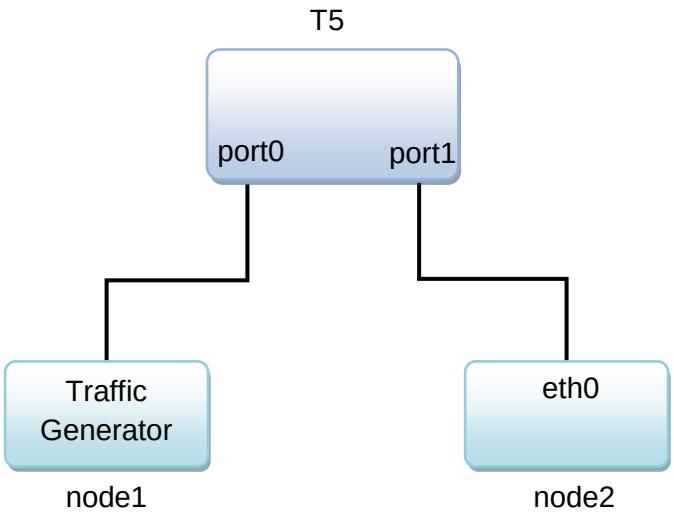
```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 1 0x7 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0
0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 0 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.
0.0.0.0 0 0 fd_id 1
filter inserted at fd_id: 1
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                  Hits FCoE Port     vld:iVLAN  Prot MPS Frag                LIP                   FIP     LPORT     FPORT Action
   0                     1  0/0  0/7 0:0000/0:0000 00/00 0/0  0/0    00000000/00000000     00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:00:00:00:00:00, l2tidx=0, smac=00:00:00:00:00:00, smtidx=0
   1                     0  0/0  1/7 0:0000/0:0000 00/00 0/0  0/0    00000000/00000000     00000000/00000000 0000/0000 0000/0000 Switch: port=0,
 dmac=00:00:00:00:00:00, l2tidx=1, smac=00:00:00:00:00:00, smtidx=0

Done...
```

iii. Generate traffic from node1 to node2 and vice versa. Observe that traffic is forwarded successfully:

```
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                  Hits FCoE Port     vld:iVLAN  Prot MPS Frag                LIP                   FIP     LPORT     FPORT Action
   0               3160185  0/0  0/7 0:0000/0:0000 00/00 0/0  0/0    00000000/00000000     00000000/00000000 0000/0000 0000/0000 Switch: port=1,
 dmac=00:00:00:00:00:00, l2tidx=0, smac=00:00:00:00:00:00, smtidx=0
   1               1435987  0/0  1/7 0:0000/0:0000 00/00 0/0  0/0    00000000/00000000     00000000/00000000 0000/0000 0000/0000 Switch: port=0,
 dmac=00:00:00:00:00:00, l2tidx=1, smac=00:00:00:00:00:00, smtidx=0

Done...
```

7.4. Forwarding multi-cast traffic

Here's an example on how to forward multi-cast traffic using *macmatch* and *mpshittype* filter modes:



Follow these steps on machine with T5 adapter:

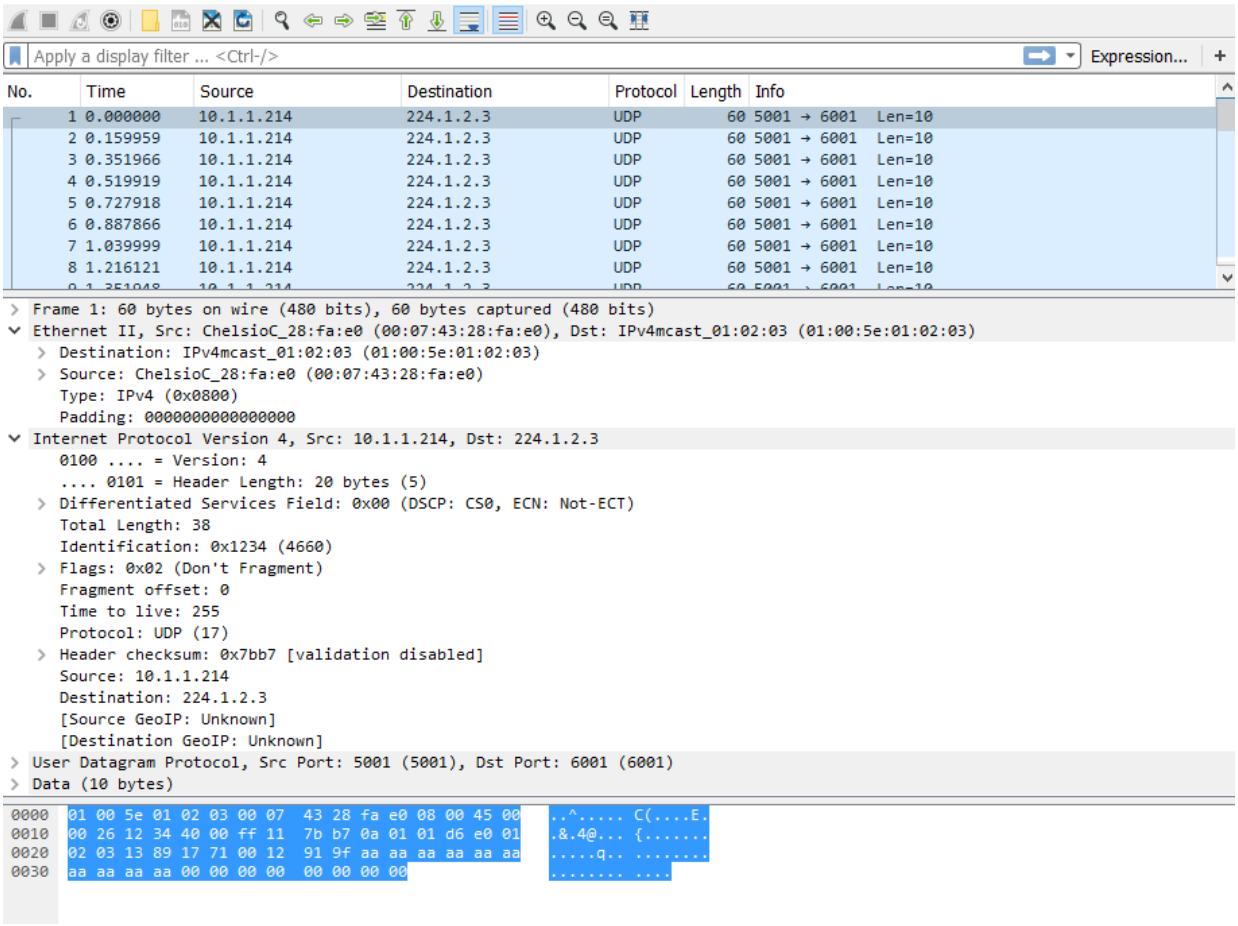
i. Add the multi-cast MAC address to MPS TCAM table and note the corresponding index:

```
cxgbe> mcast_add 01:00:5e:01:02:03 0
cxgbe> debug mps_tcam 0


mps_tcam:

Idx  Ethernet address     Mask     Vld Ports PF  VF              Replication             P0 P1 P2 P3  ML
  0 01:80:c2:00:00:0e ffffffffffff  Y   0x3   7 104 00000300 00000000 00000000 00000000   0  0  0  0 0
  1 00:07:43:29:15:40 ffffffffffff  Y   0x1   4  68                                       0  0  0  0 0
  2 00:07:43:29:15:48 ffffffffffff  Y   0x2   4  69                                       0  0  0  0 0
  3 01:00:5e:01:02:03 ffffffffffff  Y   0x1   4  68                                       0  0  0  0 0
  4         -
  5         -
  6         -
  7         -
  8         -
  9         -
 10         -
```

ii. Create a filter rule to switch packets using the index of multi-cast mac address and mac-match-type as multicast, from port0 to port1:

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type multicast mac-index 3 1 ether 0 0 vlan 0 0 0 0 ip 0 0 0 0 0 0
0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 switch queue 0 port-redirect 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0
0.0.0.0 0 0 fd_id 0
filter inserted at fd_id: 0
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                Hits Port  Prot   EthType  MACIdx MPS Frag              LIP                     FIP     LPORT     FPORT Action
   0             1998317  0/7 00/00 0000/0000 003/001 2/7  0/0     00000000/00000000       00000000/00000000 0000/0000 0000/0000 Switch: port=1,
dmac=00:00:00:00:00:00, l2tidx=0, smac=00:00:00:00:00:00, smtidx=0

Done...
```

iii. Generate traffic using the traffic generator with the multi-cast mac address as the destination mac address. Observe that node2 receives the multi-cast traffic:



7.5. Filtering using outer VLAN

The following is an example of filtering traffic based on outer VLAN ID

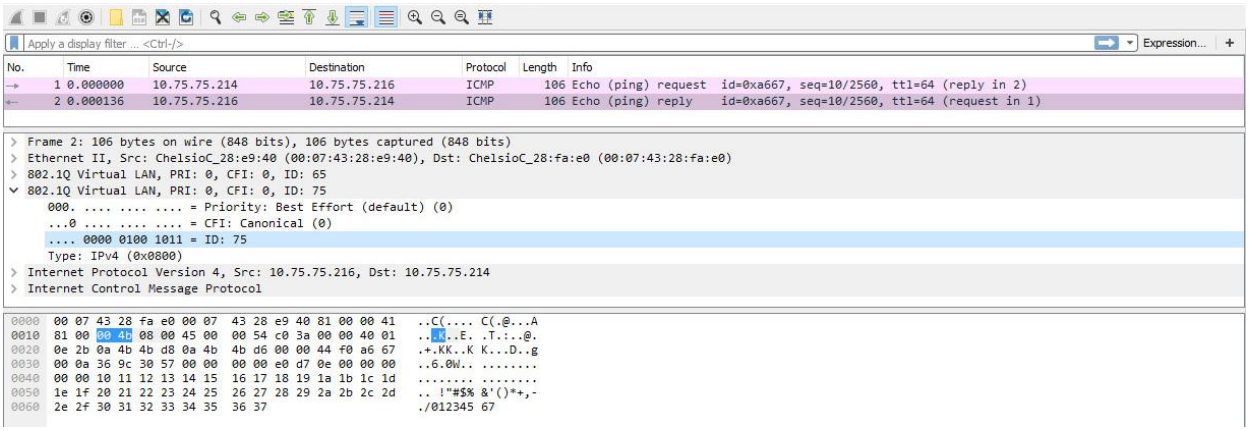
i. In the firmware configuration file, set *register*, *filterMode* and *filterMask* as follows:

```
reg[tp_pio: 0x141] = 0x0/0x800

# TP_VLAN_PRI_MAP to select filter tuples and enable ServerSram
# filter control: compact, fcoemask
# server sram   : srvrsram
# filter tuples : fragmentation, mpshittype, macmatch, ethertype,
#         protocol, tos, vlan, vnic_id, port, fcoe
# valid filterModes are described the Terminator 5 Data Book
filterMode = fragmentation, vlan, vnic_id, port, fcoe

# filter tuples enforced in LE active region (equal to or subset of filterMode)
filterMask = fragmentation, vlan, vnic_id, port, fcoe
```

ii. Create filter to drop packets with outer VLAN ID = 75

```
cxgbe> filter 0 add ipv4 mode maskfull no-prio ingress-port 0 0x7 fcoe 0 0 mac-match-type none mac-index 0 0 ether 0 0 vlan 0 0 75 0 ip 0 0 0 0 0 0 0.0.0.0
0.0.0.0 0.0.0.0 0.0.0.0 0 0 0 0 drop queue 0 port-none 1 mac-rewrite 00:00:00:00:00:00 00:00:00:00:00:00 vlan-none 0 nat-none 0.0.0.0 0.0.0.0 0 0 fd id 0
filter inserted at fd_id: 0
cxgbe> debug filters 0


filters:

[[Legend: '!' => locked; '+' => pending set; '-' => pending clear]]
 Idx                 Hits FCoE Port     vld:oVLAN     vld:iVLAN Frag                          LIP                          FIP     LPORT     FPORT Acti
on
   0                    0  0/0  0/7 1:004b/1:ffff 0:0000/0:0000  0/0               00000000/00000000            00000000/00000000 0000/0000 0000/0000 Dr
op

Done...
```

iii. Transmit packets from peer machine with outer VLAN as 75. Observe that packets will be dropped by the DPDK machine.



8. Software/Driver Unloading

UIO Support

Run the following commands to unload DPDK driver with UIO support:

```
[root@host~]# rmmod igb_uio
[root@host~]# rmmod uio
```

If unloading *uio* module reports an error, unload the following dependent modules and try again:

```
[root@host~]# rmmod bnx2fc
[root@host~]# rmmod bnx2i
[root@host~]# rmmod cnic
```

VFIO Support

Run the following command to unload DPDK driver with VFIO support:

```
[root@host~]# rmmod vfio-pci
```

9. Software/Driver Uninstallation

i. Change your working directory to Chelsio-DPDK-x.x.x.x directory:

```
[root@host~]# cd Chelsio-DPDK-x.x.x.x
```

ii. Uninstall the DPDK driver using the following command:

```
[root@host~]# make uninstall
```

10. Software/Driver Update

For any distribution specific problems, please check README and Release Notes included in the release for possible workaround.

Please visit Chelsio support web site http://service.chelsio.com/ for regular updates on various software/drivers. You can also subscribe to our newsletter for the latest software updates.

11. Appendix

11.1. Chelsio End-User License Agreement (EULA)

Installation and use of the driver/software implies acceptance of the terms in the Chelsio End-User License Agreement (EULA).

IMPORTANT: PLEASE READ THIS SOFTWARE LICENSE CAREFULLY BEFORE DOWNLOADING OR OTHERWISE USING THE SOFTWARE OR ANY ASSOCIATED DOCUMENTATION OR OTHER MATERIALS (COLLECTIVELY, THE "SOFTWARE"). BY CLICKING ON THE "OK" OR "ACCEPT" BUTTON YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, CLICK THE "DO NOT ACCEPT" BUTTON TO TERMINATE THE INSTALLATION PROCESS.

1. License. Chelsio Communications, Inc. ("Chelsio") hereby grants you, the Licensee, and you hereby accept, a limited, non-exclusive, non-transferable license to install and use the Software with one or more Chelsio network adapters on a single server computer for use in communicating with one or more other computers over a network. You may also make one copy of the Software in machine readable form solely for back-up purposes, provided you reproduce Chelsio's copyright notice and any proprietary legends included with the Software or as otherwise required by Chelsio.

2. Restrictions. This license granted hereunder does not constitute a sale of the Software or any copy thereof. Except as expressly permitted under this Agreement, you may not:

(i) reproduce, modify, adapt, translate, rent, lease, loan, resell, distribute, or create derivative works of or based upon, the Software or any part thereof; or

(ii) make available the Software, or any portion thereof, in any form, on the Internet. The Software contains trade secrets and, in order to protect them, you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human-perceivable form. You assume full responsibility for the use of the Software and agree to use the Software legally and responsibly.

3. Ownership of Software. As Licensee, you own only the media upon which the Software is recorded or fixed, but Chelsio retains all right, title and interest in and to the Software and all subsequent copies of the Software, regardless of the form or media in or on which the Software may be embedded.

4. Confidentiality. You agree to maintain the Software in confidence and not to disclose the Software, or any information or materials related thereto, to any third party without the express written consent of Chelsio. You further agree to take all reasonable precautions to limit access of the Software only to those of your employees who reasonably require such access to perform their employment obligations and who are bound by confidentiality agreements with you.

5. Term. This license is effective in perpetuity, unless terminated earlier. You may terminate the license at any time by destroying the Software (including the related documentation), together with all copies or modifications in any form. Chelsio may terminate this license, and this license shall be deemed to have automatically terminated, if you fail to comply with any term or condition of this Agreement. Upon

any termination, including termination by you, you must destroy the Software (including the related documentation), together with all copies or modifications in any form.

6. Limited Warranty. If Chelsio furnishes the Software to you on media, Chelsio warrants only that the media upon which the Software is furnished will be free from defects in material or workmanship under normal use and service for a period of thirty (30) days from the date of delivery to you.

CHELSIO DOES NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE OR ANY PART THEREOF. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, CHELSIO MAKES NO OTHER WARRANTIES, EXPRESS OR IMPLIED, AND HEREBY DISCLAIMS ALL OTHER WARRANTIES, INCLUDING, BUT NOT LIMITED TO, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow the exclusion of implied warranties or limitations on how long an implied warranty may last, so the above limitations may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

7. Remedy for Breach of Warranty. The sole and exclusive liability of Chelsio and its distributors, and your sole and exclusive remedy, for a breach of the above warranty, shall be the replacement of any media furnished by Chelsio not meeting the above limited warranty and which is returned to Chelsio. If Chelsio or its distributor is unable to deliver replacement media which is free from defects in materials or workmanship, you may terminate this Agreement by returning the Software.

8. Limitation of Liability. IN NO EVENT SHALL CHELSIO HAVE ANY LIABILITY TO YOU OR ANY THIRD PARTY FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, HOWEVER CAUSED, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE LICENSE OR USE OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR LOSS OF ANTICIPATED PROFITS, EVEN IF CHELSIO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL CHELSIO'S LIABILITY ARISING OUT OF OR RELATED TO THE LICENSE OR USE OF THE SOFTWARE EXCEED THE AMOUNTS PAID BY YOU FOR THE LICENSE GRANTED HEREUNDER. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

9. High Risk Activities. The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as online equipment control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage. Chelsio specifically disclaims any express or implied warranty of fitness for any high risk uses listed above.

10. Export. You acknowledge that the Software is of U.S. origin and subject to U.S. export jurisdiction. You acknowledge that the laws and regulations of the United States and other countries may restrict the export and re-export of the Software. You agree that you will not export or re-export the Software or documentation in any form in violation of applicable United States and foreign law. You agree to comply with all applicable international and national laws that apply to the Software, including the U.S.

Export Administration Regulations, as well as end-user, end-use, and destination restrictions issued by U.S. and other governments.

11. Government Restricted Rights. The Software is subject to restricted rights as follows. If the Software is acquired under the terms of a GSA contract: use, reproduction or disclosure is subject to the restrictions set forth in the applicable ADP Schedule contract. If the Software is acquired under the terms of a DoD or civilian agency contract, use, duplication or disclosure by the Government is subject to the restrictions of this Agreement in accordance with 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors and 49 C.F.R. 227.7202-1 of the DoD FAR Supplement and its successors.

12. General. You acknowledge that you have read this Agreement, understand it, and that by using the Software you agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between Chelsio and you, and supersedes any proposal or prior agreement, oral or written, and any other communication between Chelsio and you relating to the subject matter of this Agreement. No additional or any different terms will be enforceable against Chelsio unless Chelsio gives its express consent, including an express waiver of the terms of this Agreement, in writing signed by an officer of Chelsio. This Agreement shall be governed by California law, except as to copyright matters, which are covered by Federal law. You hereby irrevocably submit to the personal jurisdiction of, and irrevocably waive objection to the laying of venue (including a waiver of any argument of forum non conveniens or other principles of like effect) in, the state and federal courts located in Santa Clara County, California, for the purposes of any litigation undertaken in connection with this Agreement. Should any provision of this Agreement be declared unenforceable in any jurisdiction, then such provision shall be deemed severable from this Agreement and shall not affect the remainder hereof. All rights in the Software not specifically granted in this Agreement are reserved by Chelsio. You may not assign or transfer this Agreement (by merger, operation of law or in any other manner) without the prior written consent of Chelsio and any attempt to do so without such consent shall be void and shall constitute a material breach of this Agreement.

Should you have any questions concerning this Agreement, you may contact Chelsio by writing to:

Chelsio Communications, Inc.
209 North Fair Oaks Avenue,
Sunnyvale, CA 94085