Chelsio Communications

Accelerate

Chelsio T4 Unified Wire For Linux

Installation and User's Guide

Chelsio Communications (Headquarters)
370 San Aleso Ave.
Suite 100
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-560076
Karnataka,
India

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

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	12/08/2011
1.0.1	01/09/2012
1.0.2	01/27/2012
1.0.3	03/26/2012
1.0.4	04/12/2012
1.0.5	06/20/2012
1.0.6	08/17/2012

TABLE OF CONTENTS

I. Chelsio Unified Wire

1. Introduction

Thank you for choosing Chelsio T4 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 T4 adapters can fully offload TCP, UDP, iSCSI, iWARP and FCoE over a single Unified Wire. The adapters also fully support SR-IOV, EVB/VNTag, DCB, Traffic Management and Filtering.

Ideal for all data, storage and high performance clustering applications, the T4 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 T4 adapters bring a new level of performance metrics and functional capabilities to the computer networking industry.

The Chelsio T4 Unified Wire software comes in two formats: Source code and RPM package forms. Installing from source requires compiling the package to generate the necessary binaries. You can choose this method when you are using a custom-built kernel. You can also install the package using the interactive GUI installer. In other cases, download the RPM package specific to your operating system and follow the steps mentioned to install the package. Please note that the OFED software required to install Chelsio iWARP driver comes bundled in the RPM packages.

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

1.1. Features

The Chelsio Unified Wire Package uses a single command to install various drivers and utilities. It consists of the following software:

- **Network (NIC/TOE)**
- **Virtual Function Network (vNIC)**
- **iWARP (RDMA)**
- **iSCSI PDU Offload Target**
- **iSCSI PDU Offload Initiator**
- **FCoE full offload Initiator**
- **Offload Bonding driver**
- **Offload IPv6 driver**
- **Bypass driver**
- **UDP Segmentation Offload and Pacing**

- **Utility Tools(cop,cxgbtool,t4_perftune,sniffer & tracer)**
- **iWARP and WD-UDP libraries**

For detailed instructions on loading, unloading and configuring the drivers/tools please refer to their respective sections.

1.2. Hardware Requirements

The Chelsio T4 Unified Wire software supports Chelsio T4 Series of Unified Wire Adapters. To know more about the list of adapters supported by each driver, please refer to their respective sections.

1.3. Software Requirements

The Chelsio T4 Unified Wire software has been developed to run on 64-bit Linux based platforms. To know more about the complete list of operating systems supported by each driver, please refer to their respective sections.

1.4. Package Contents

1.4.1. Source Package

The Chelsio Unified Wire source package consists of the following:

- **install.py,dialog.py:** Python scripts needed for the GUI installer.
- **EULA:** Chelsio's End User License Agreement
- **install.log:** File containing installation summary.
- **docs:** The docs directory contains support documents - README, Release Notes and User's Guide (this document) for the software.
- **libs:** This directory is for libraries required to install the WD-UDP and iWARP drivers. The libibverbs library has implementation of RDMA verbs which will be used by iWARP applications for data transfers. The librdmacm library works as an RDMA connection manager. The libcxgb4 library works as an interface between the above mentioned generic libraries and Chelsio iWARP driver. The libcxgb4_sock library is a LD_PRELOAD-able library that accelerates UDP Socket communications transparently and without recompilation of the user application.
- **Makefile:** The Makefile for building and installing from the source.
- **sample_machinefile:** Sample file used during iWARP installation on cluster nodes.
- **scripts:** Directory containing scripts used by installer for iWARP driver installation on cluster nodes.
- **specs:** The packaging specification files required for building RPM packages.

- **src:** Source code for different drivers.
- **support:** This directory contains source files for the dialog utility.
- **tools:**
  - **ba_tools:** Management and configuration tools for bypass adapters.
  - **chsetup:** The chsetup tool loads NIC,TOE and iWARP drivers, and creates WD-UDP configuration file.
  - **chstatus:** This utility provides status information on any Chelsio NIC in the system.
  - **cop:** The cop tool compiles offload policies into a simple program form that can be loaded into the kernel and interpreted. These offload policies are used to determine the settings to be used for various connections. The connections to which the settings are applied are based on matching filter specifications. Please find more details on this tool in its manual page (run `man cop` command).
  - **cxgbtool:** The cxgbtool queries or sets various aspects of Chelsio network interface cards. It complements standard tools used to configure network settings and provides functionality not available through such tools. Please find more details on this tool in its manual page (run `man cxgbtool` command).

    Note *To use cxbtool for FCoE Initiator driver, use # cxgbtool stor –h*

  - **t4_perftune.sh:** This shell script is to tune the system for higher performance. It achieves it through modifying the IRQ-CPU binding. This script can also be used to change Tx coalescing settings.
  - **uname_r:** This tool provides the kernel version for desired target distribution. Use this option to build drivers for distribution kernel updates.
  - **wdload:** UDP acceleration tool.
  - **sniffer:** This directory contains sniffer tracing and filtering libraries. Refer to the **WD Sniffing and Tracing** section for more information.

1.4.2. RPM package

The Chelsio Unified Wire RPM package consists of the following:

- **docs:** The docs directory contains support documents i.e. README, Release Notes and User's Guide (this document) for the software.
- **DRIVER-RPMS:** RPM packages of Chelsio drivers.
- **OFED-RPMS:** OFED RPM packages required to install iWARP driver.
- **install.py:** Python script that installs the RPM package. Refer to the Software/Driver Installation section for more information.
- **uninstall.py:** Python script that uninstalls the RPM package. Refer to the Software/Driver Uninstallation section for more information.
- **EULA:** Chelsio's End User License Agreement

2. Hardware Installation

1. Shutdown and power off your system.
2. Power off all remaining peripherals attached to your system.
3. Unpack the Chelsio adapter and place it on an anti-static surface.
4. Remove the system case cover according to the system manufacturer's instructions.
5. Remove the PCI filler plate from the slot where you will install the 10Gb Ethernet adapter.
6. For maximum performance, it is highly recommended to install the adapter into a PCIE x8 slot.
7. 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.
8. Secure the Chelsio adapter with a screw, or other securing mechanism, as described by the system manufacturer's instructions. Replace the case cover.
9. After securing the card, ensure that the card is still fully seated in the PCIE x8 slot as sometimes the process of securing the card causes the card to become unseated.
10. Connect a fiber cable, multi-mode for short range (SR) optics or single-mode for long range (LR) optics, to the 10Gb Ethernet adapter or regular Ethernet cable for the 1Gb Ethernet adapter.
11. Power on your system.
12. Verify if the T4 adapter was installed successfully by using the following command:

```
[root@host]# lspci | grep –i Chelsio
03:00.0 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.1 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.2 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.3 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.4 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.5 SCSI storage controller: Chelsio Communications Inc T420-CR Unified
Wire Storage Controller
03:00.6 Fibre Channel: Chelsio Communications Inc T420-CR Unified Wire
Storage Controller
03:00.7 Ethernet controller: Chelsio Communications Inc Device 0000
```

For Chelsio T4 adapters, the physical functions are currently assigned as:

- Physical functions 0 - 3: for the SR-IOV functions of the T4
- Physical function 4: for all NIC functions of the card

- Physical function 5: for iSCSI
- Physical function 6: for FCoE
- Physical function 7: Currently not assigned

Once Unified Wire package is installed and loaded, examine the output of `dmesg` to see if the card is discovered

```
eth0: Chelsio T440-CR rev 2 10GBASE-SFP RNIC PCIe x8 MSI-X
0000:04:00.4: S/N: PT18111226, P/N: 110112140D0
```

This output indicates the hardware configuration of the card as well as the Serial number of the card. As observed by the x8, the card is properly installed in an x8 slot on the machine and the card is using MSI-X interrupts.

3. Software/Driver Installation

There are two main methods to install the Chelsio Unified Wire package: from source and RPM.
If you decide to use source, you can install the package using CLI or GUI mode.
If you decide to use RPM, you can install the package using Menu or CLI mode.

Irrespective of the method chosen for installation, the machine needs to be rebooted for changes to take effect.

3.1. Installing Chelsio Unified Wire from source

3.1.1. GUI mode (with Dialog utility)

i. Download the tarball ChelsioUwire-x.x.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

```
[root@host]# tar -zxvfm ChelsioUwire-x.x.x.x.tar.gz
```

iii. Run the following script to start the GUI installer:

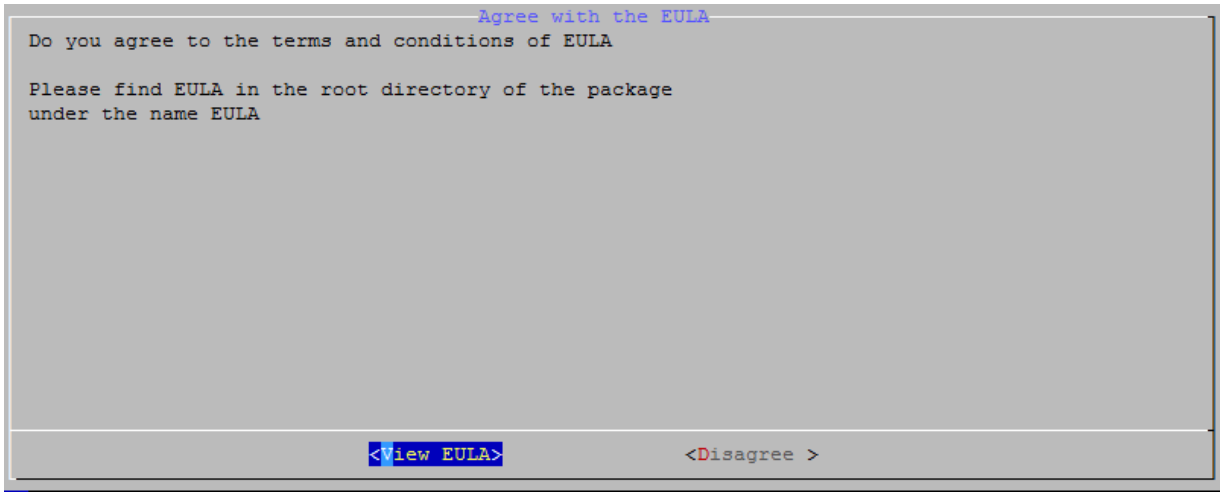
```
[root@host]# ./install.py
```

iv. If **Dialog** utility is present, you can skip to step (v). If not, press 'y' to install it when the installer prompts you for your input.

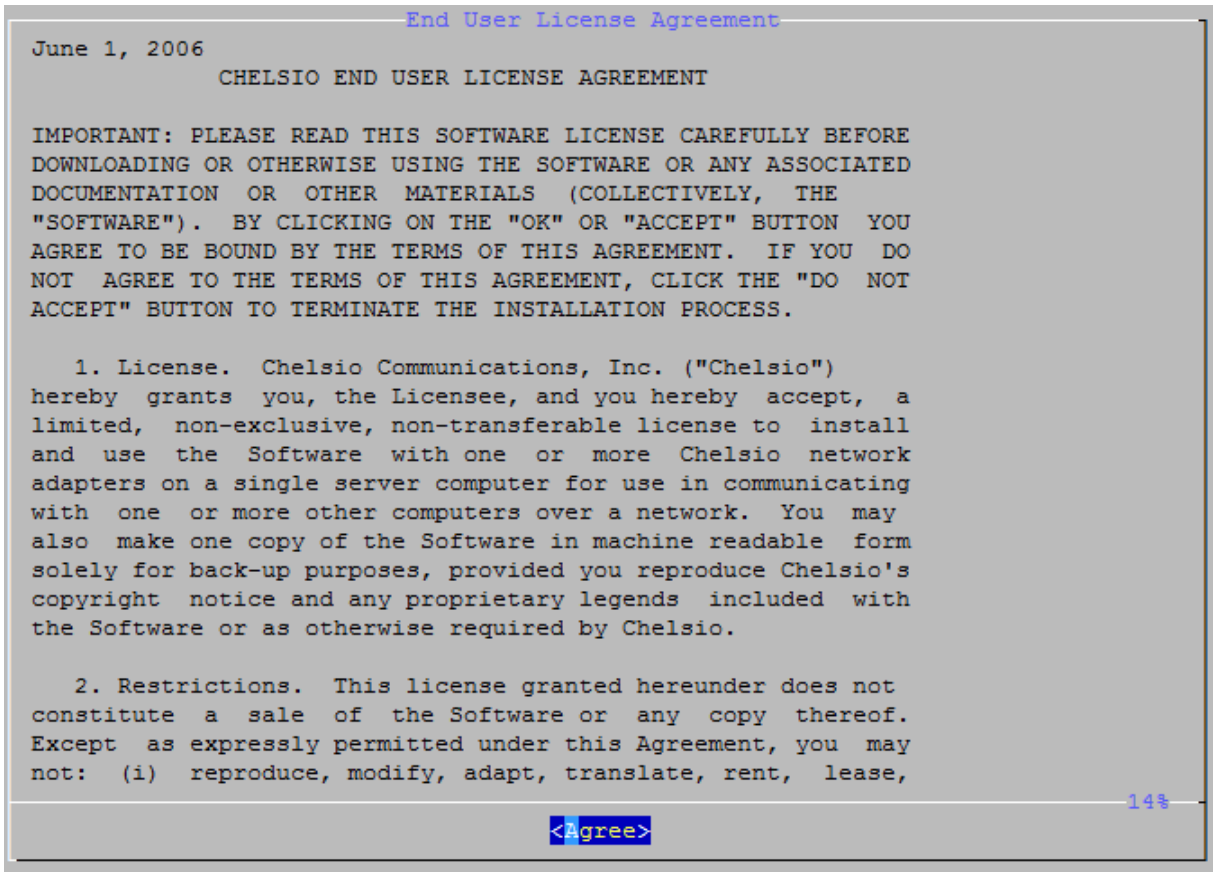
Note

- *If you decide to use RDMA feature, please refer to the **iWARP (RDMA)** section and follow the steps to install OFED package before installing the Chelsio Unified Wire Package*
- *If you want to install OFED with NFS-RDMA support, please refer "Setting up NFS-RDMA" in **iWARP (RDMA)** section before proceeding.*

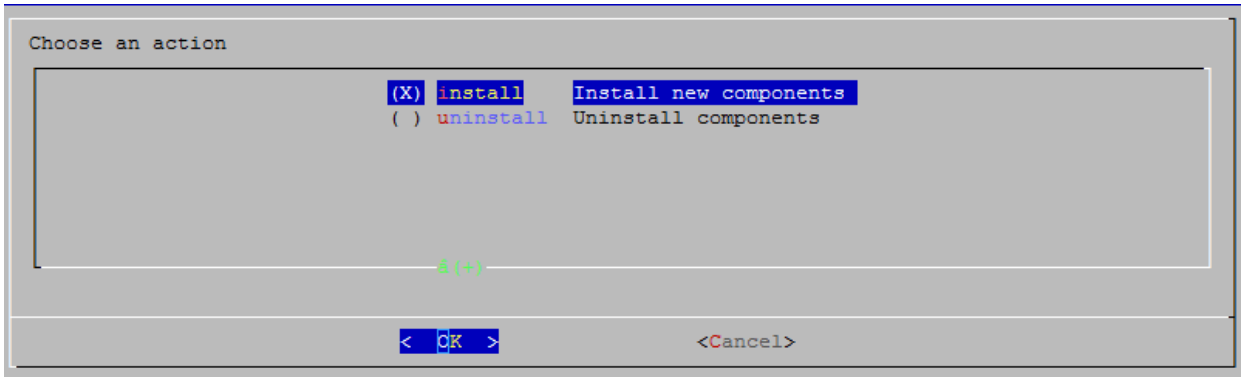
v. Select “View EULA” to read the Chelsio End User License Agreement:



vi. Select “Agree” to continue:

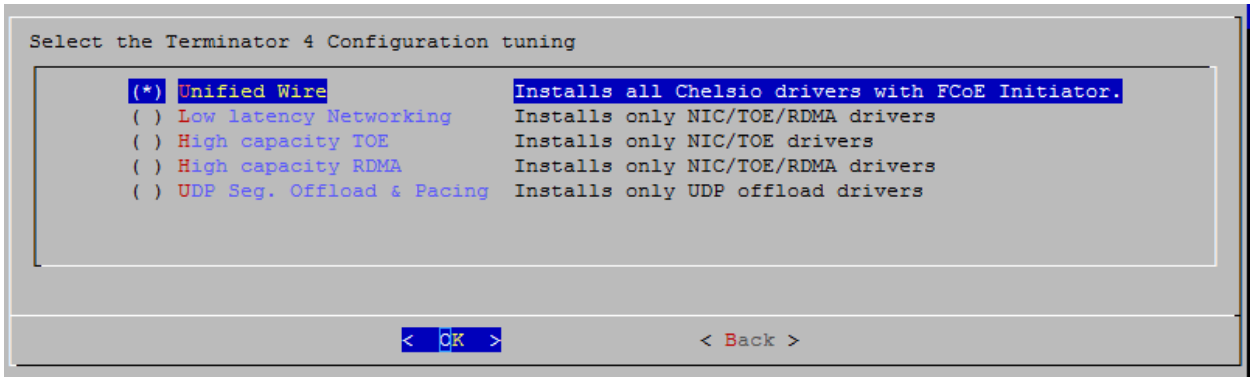


vii. Select “install” under “Choose an action”

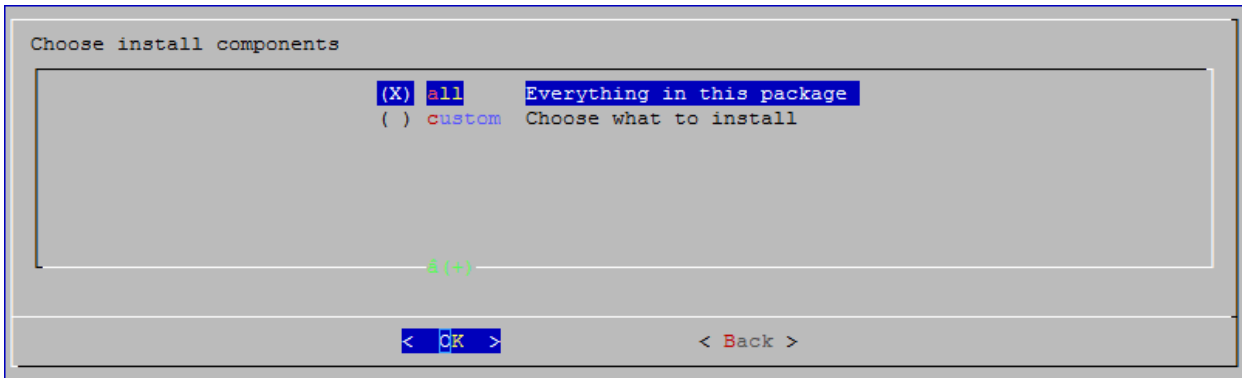


viii. Select the required T4 configuration tuning option:

a. *Unified Wire*: Configures T4 adapters to run multiple protocols like NIC/TOE, iWARP, iSCSI and FCoE Initiator simultaneously.
b. *Low latency Networking*: Configures T4 adapters to run NIC/TOE and iWARP traffic with low latency specially needed for financial applications.
c. *High capacity TOE*: Configures T4 adapters to establish a large number of TOE connections.
d. *High capacity RDMA*: Configures T4 adapters to establish a large number of RDMA connections.
e. *UDP Seg.Offload & Pacing*: Configures T4 adapters to establish a large number of UDP Segmentation Offload connections.

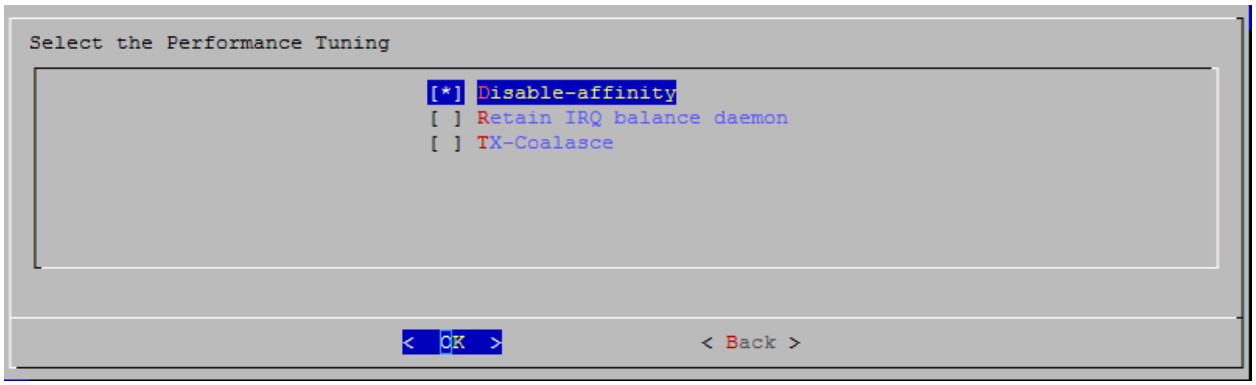


ix. Under “Choose install components”, select “all” to install all the related components for the option chosen in step (viii) or select “custom” to install specific components.



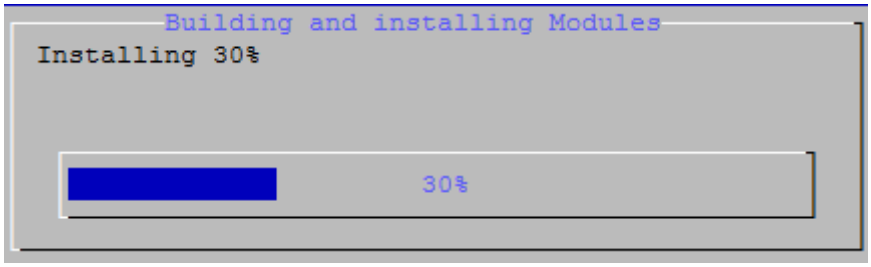
Important *To install Bypass driver, please select Unified Wire with FCoE Initiator in step (viii). Then select “Bypass” under “custom” option.*

x. Select the required performance tuning option.
   a. *Disable-affinity*: Disable binding IRQs to CPUs.
   b. *Retain IRQ balance daemon*: Bind IRQs to CPUs and do not disable IRQ balance daemon.
   c. *TX-Coalasce*: Write tx_coal=2 to modprobe.d/conf.

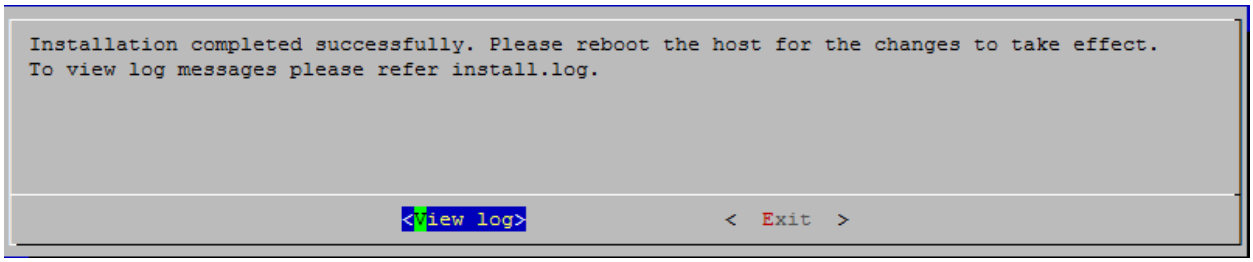


Note *For more information on the Performance tuning options, please refer to the **Software/Driver Configuration and Fine-tuning** section of the Network (NIC/TOE) chapter.*

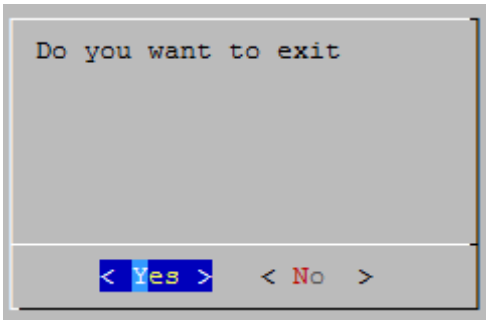
xi. The selected components will now be installed:



xii. After successful installation, select “View log” to view the installation summary or “Exit” to continue.



xiii. Select “Yes” to exit the installer or “No” to go back.



xiv. Reboot your machine for changes to take effect.

Note *Press Esc or Ctrl+C to exit the installer at any point of time.*

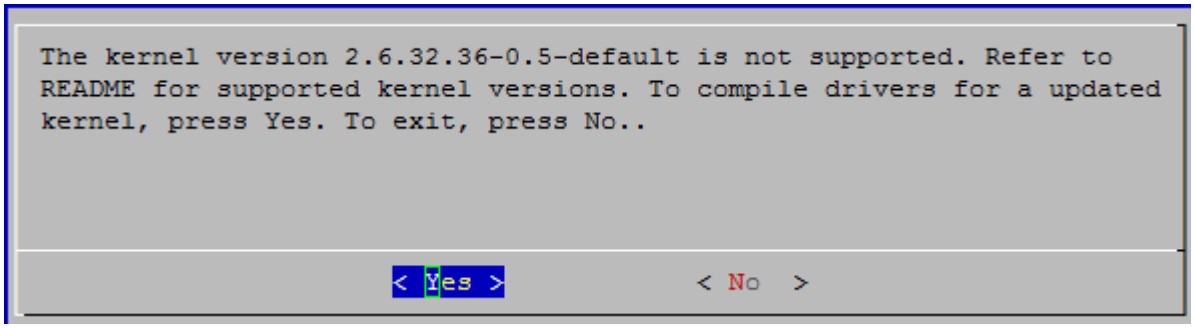
- **Installation on updated kernels**

If the kernel version on your Linux distribution is updated, follow the steps mentioned below to install the Unified Wire package:

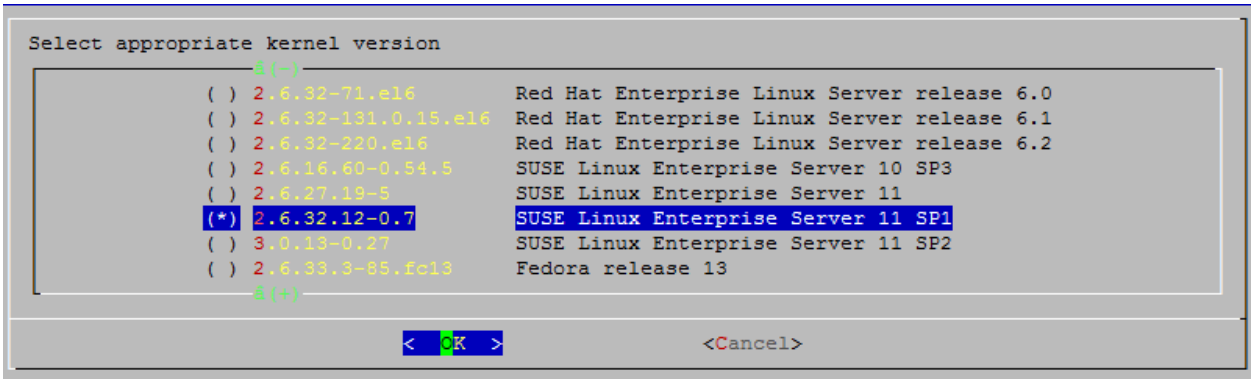
i. Run the following script to start the GUI installer:

```
[root@host]# ./install.py
```

ii. Select “Yes” to continue with the installation on the updated kernel or “No” to exit.



iii. Select the nearest supported kernel version from the list and select “OK”.



iv. Follow steps (v) to (xiv) mentioned in the previous section.

3.1.2. CLI mode (without Dialog utility)

If your system does not have **Dialog** or you choose not to install it, follow the steps mentioned below to install the Unified Wire package:

i. Download the tarball ChelsioUwire-x.x.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

```
[root@host]# tar -zxvfm ChelsioUwire-x.x.x.x.tar.gz
```

iii. Run the following script to start the installer:

```
[root@host]# ./install.py
```

iv. When the installer prompts you for your input, press 'n' to continue installation without the **Dialog** utility.

v. Press '1' to read the Chelsio End User License Agreement; Press 'q' to go back to installation menu. Press '2' to Accept the EULA and start the installation or '3' to Disagree and and exit from the installer.

vi. Enter the number corresponding to the T4 Configuration option in the Input field and press Enter. The selected components will now be installed.

vii. After successful installation you can press 1 to view the installation log. Press any other key to exit from the installer.

Important

- *To customize the installation, view the help by typing* `[root@host]#./install.py –h`
- *To install Bypass driver, run* `[root@host]# ./install.py -c bypass` *and follow steps (v) and (vi) mentioned above.*

viii. Reboot your machine for changes to take effect.

iWARP driver installation on Cluster nodes

Chelsio's Unified Wire package allows installing iWARP drivers on multiple Cluster nodes with a single command. Follow the procedure mentioned below:

i. Create a file (*machinefilename*) containing the IP addresses or hostnames of the nodes in the cluster. You can view the sample file, *sample_machinefile,* provided in the package to view the format in which the nodes have to be listed.

ii. Now, execute the following command:

```
[root@host] # ./install.py -C  -m <machinefilename>
```

This command will install the Unified Wire package with the default T4 configuration tuning option *Unified Wire (includes FCoE Initiator).*

Important *Please make sure that you have enabled password less authentication with ssh on the peer nodes for this feature to work.*

3.1.3. CLI mode

Note *If you decide to use RDMA feature, please refer to the **iWARP (RDMA) section** and follow the steps to install OFED package before installing the Chelsio Unified Wire Package.*

i. Download the tarball ChelsioUwire-x.x.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

```
[root@host]# tar -zxvfm ChelsioUwire-x.x.x.x.tar.gz
```

iii. Navigate to the 'ChelsioUwire-x.x.x.x' directory. Build the source using :

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

iv. Install the drivers, tools and libraries using the following command:

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

Important *Steps (iii) and (iv) mentioned above will NOT install UDP Segmentation Offload and Bypass drivers. These drivers will have to be installed manually. Please refer to section **3.1.4 CLI mode (individual drivers)** for instructions on installing them individually.*

v. The default T4 configuration tuning option is *Unified Wire (includes FCoE Initiator)*. The configuration tuning can be selected using the following commands:

```
[root@host]# make CONF=<T4 configuration>
[root@host]# make CONF=<T4 configuration> install
```

Note *To view the different T4 configuration tuning options, view the help by typing* `[root@host]#make help`

vi. Reboot your machine for changes to take effect.

- **Installation on updated kernels**

If the kernel version on your Linux distribution is updated, please execute the following command to install the Unified Wire package:

```
[root@host]# make UNAME_R=<kernel_version>
```

Where `kernel_version` is the nearest supported kernel version.

For example, if you want to install the package on a RHEL 6 distribution updated to 2.6.32-131.17.1.el6 kernel, run the following commands:

```
[root@host]# make UNAME_R=2.6.32-131.0.15.el6
[root@host]# make UNAME_R=2.6.32-131.0.15.el6 install
```

To view the list of the supported kernel versions, run the following command:

```
[root@host]# make list_kernels
```

Reboot your machine for changes to take effect.

3.1.4. CLI mode (individual drivers)

You can also choose to install drivers individually. Provided here are steps to build and install NIC, TOE, iWARP, Bypass and UDP Segmentation Offload drivers. To know about other drivers, access help by running `make help`.

- To build and install NIC driver without offload support :

```
[root@host]# make nic
[root@host]# make nic_install
```

- To build and install NIC driver with offload support and Offload drivers:

```
[root@host]# make toe
[root@host]# make toe_install
```

- To build and install iWARP driver against outbox OFED:

```
[root@host]# make iwarp
[root@host]# make iwarp_install
```

- To build and install all drivers except Bypass against outbox OFED:

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

- To build and install Bypass driver:

```
[root@host]# make bypass
[root@host]# make bypass_install
```

- To build and install UDP Segmentation Offload driver:

```
[root@host]# make udp_offload
[root@host]# make udp_offload_install
```

- The default T4 configuration tuning option is *Unified Wire (includes FCoE Initiator)*. The configuration tuning can be selected using the following commands:

```
[root@host]# make CONF=<T4 configuration> <Build Target>
[root@host]# make CONF=<T4 configuration> <Install Target>
```

Note *To view the different T4 configuration tuning options, view the help by typing* `[root@host]#make help`

3.2. Installing Chelsio Unified Wire from RPM

3.2.1. Menu Mode

i. Download the tarball specific to your operating system and architecture from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball:

E.g. For RHEL 6.0, untar using the following command:

```
[root@host]# tar -zxvfm ChelsioUwire-x.x.x.x-RHEL6.0_x86_64.tar.gz
```

iii. Navigate to ‘ChelsioUwire-x.x.x.x’ directory. Run the following command:

```
[root@host]# ./install.py
```

iv. Press “Enter” to read EULA or “q” to quit.
v. Press "y" to accept the EULA and continue or "q" to quit.
vi. Select the required T4 configuration tuning option as described below. Enter the corresponding number in the Input field and press Enter.
    a. *Unified Wire (includes FCoE Initiator)*: Configures T4 adapters to run multiple protocols like NIC/TOE, iWARP, iSCSI and FCoE Initiator simultaneously.
    b. *Low latency Networks*: Configures T4 adapters to run NIC/TOE and iWARP traffic with low latency specially needed for financial applications.
    c. *High capacity TOE*: Configures T4 adapters to establish a large number of TOE connections.
    d. *High capacity RDMA*: Configures T4 adapters to establish a large number of RDMA connections.
    e. *UDP Segmentation Offload & Pacing*: Configures T4 adapters to establish a large number of UDP Segmentation Offload connections.

Note *UDP Segmentation Offload & Pacing is not supported on all Linux distributions. Hence may not appear in the menu. Please refer to the Software Requirements section of UDP Segmentation Offload and Pacing to know about the supported*

vii. Select the Installation type as described below. Enter the corresponding number in the Input field and press Enter.
    a. Install all Chelsio drivers (including FCoE Initiator) built against inbox OFED
    b. Install all Chelsio drivers (including FCoE Initiator) built against OFED-1.5.4.1
    c. Install NIC and TOE drivers only
    d. Install bypass drivers and tools
    e. Install UDP segmentation offload capable NIC and TOE drivers only

Note *The Installation options may vary depending on the T4 Configuration tuning option selected.*

viii. The selected components will now be installed.
ix. Reboot your machine for changes to take effect.

Note

If the installation aborts with the message "Resolve the errors/dependencies manually and restart the installation", please go through the install.log to resolve errors/dependencies and then start the installation again.

3.2.2. CLI mode

i. Download the tarball specific to your operating system and architecture from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball:

E.g. For RHEL 6.0, untar using the following command:

```
[root@host]# tar -zxvfm ChelsioUwire-x.x.x.x-RHEL6.0_x86_64.tar.gz
```

iii. Navigate to the ChelsioUwire-x.x.x.x directory. Install the Unified Wire package using:

```
[root@host]# ./install.py -i <nic_toe/all/all_ofed/bypass/udpso>
```

nic_toe : NIC and TOE drivers only
all : all Chelsio drivers (including FCoE Initiator) built against inbox OFED
all_ofed : all Chelsio drivers (including FCoE Initiator) built against OFED-1.5.4.1.
bypass : bypass drivers and tools
udpso : UDP segmentation offload capable NIC and TOE drivers only

Note

The Installation options may vary depending on the Linux distribution.

iv. The default T4 configuration tuning option is *Unified Wire (includes FCoE Initiator)*. The configuration tuning can be selected using the following command:

```
[root@host]# ./install.py -i <Installation mode> -c <T4 configuration>
```

Note

To view the different T4 configuration tuning options, view the help by typing `[root@host]# ./install.py -h`

v. Reboot your machine for changes to take effect.

3.3. Firmware update

The T4 firmware is installed on the system, typically in `/lib/firmware/cxgb4`, and the driver will auto-load the firmware if an update is required. The kernel must be configured to enable userspace firmware loading support:

Device Drivers -> Generic Driver Options -> Userspace firmware loading support

The firmware version can be verified using *ethtool*:

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

4. Software/Driver Uninstallation

Similar to installation, the Chelsio Unified Wire package can be uninstalled using two main methods: from the source and RPM, based on the method used for installation. If you decide to use source, you can uninstall the package using CLI or GUI mode.

Note

- *If the Chelsio Unified Wire is installed using one method and you decide to reinstall the package using the other, please make sure that the package is first uninstalled using the method it was installed with.*

  E.g. If the package is installed using the source, uninstall it using the source only and not RPMs and vice versa.

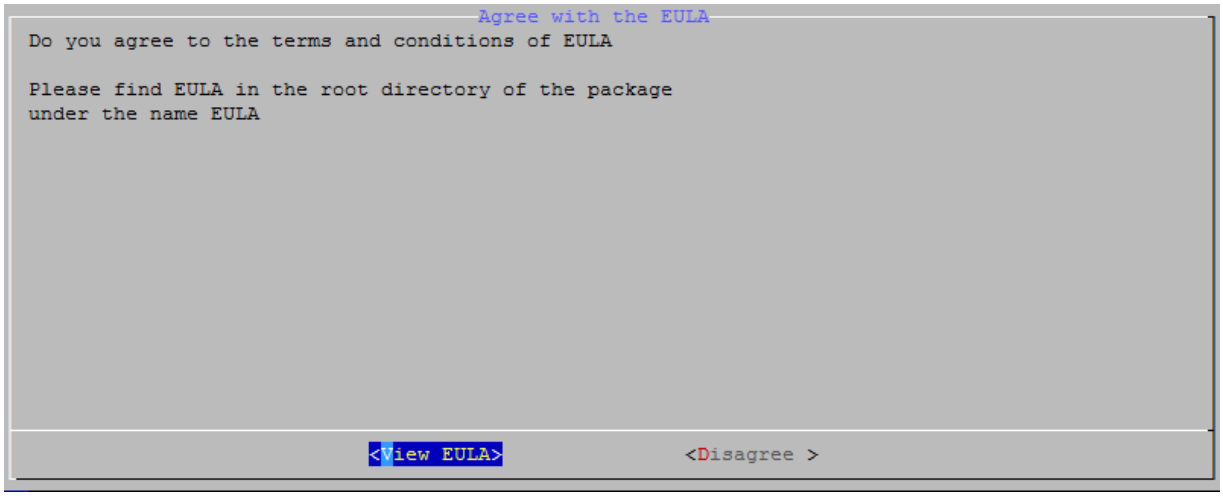
4.1. Uninstalling Chelsio Unified Wire from source

4.1.1. GUI mode (with Dialog utility)

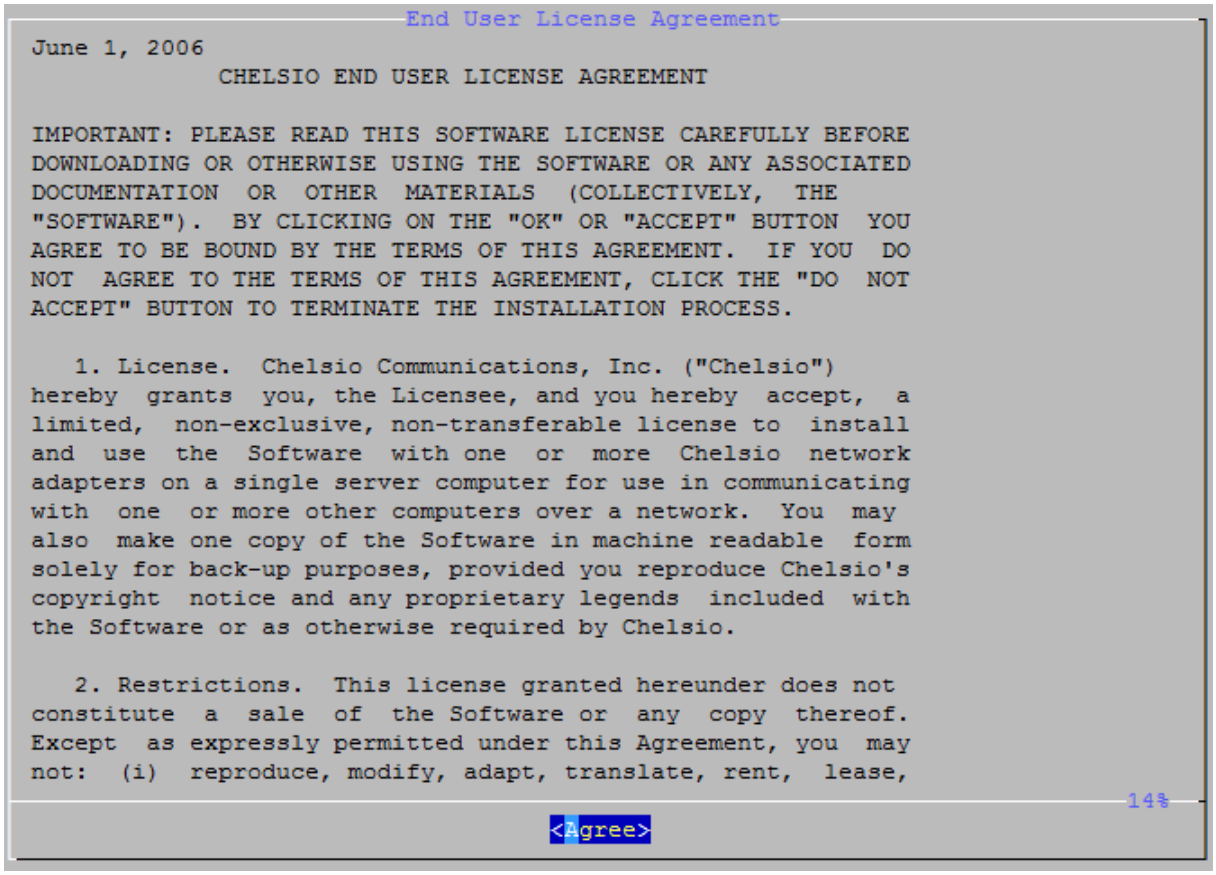
i. Run the following script to start the GUI installer:

```
[root@host]# ./install.py
```

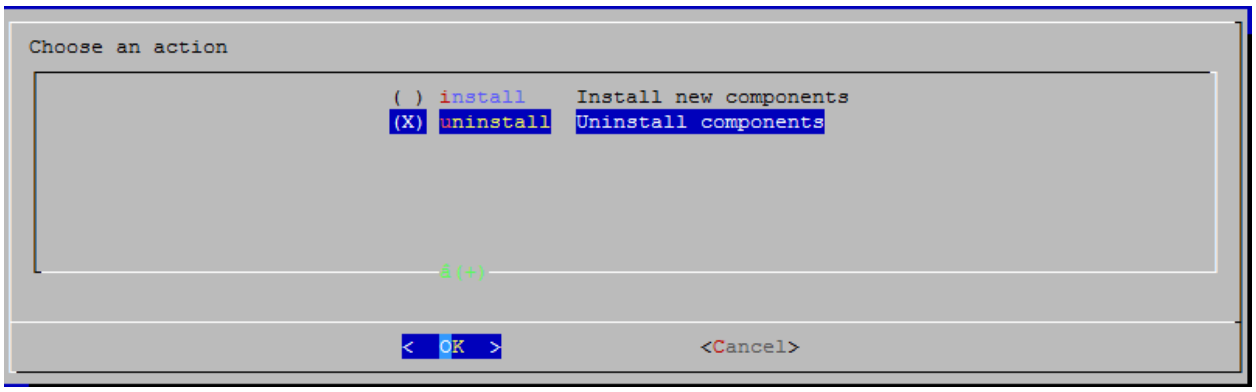
ii. Select “View EULA” to read the Chelsio End User License Agreement:



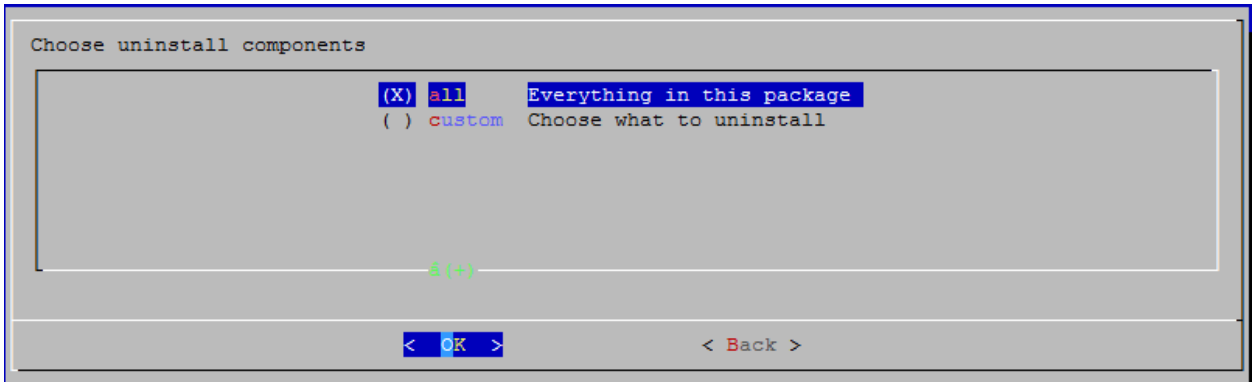
iii. Select “Agree” to continue:



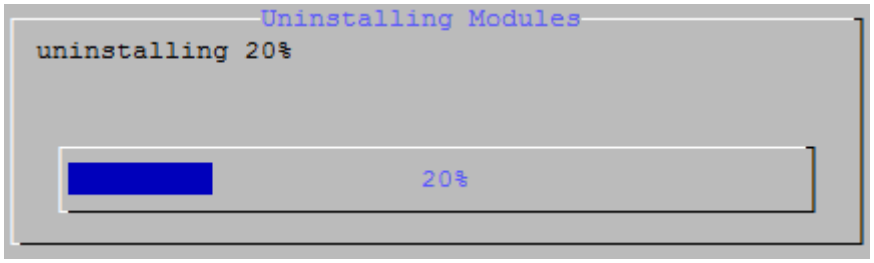
iv. Select “uninstall” , Under “Choose an action”



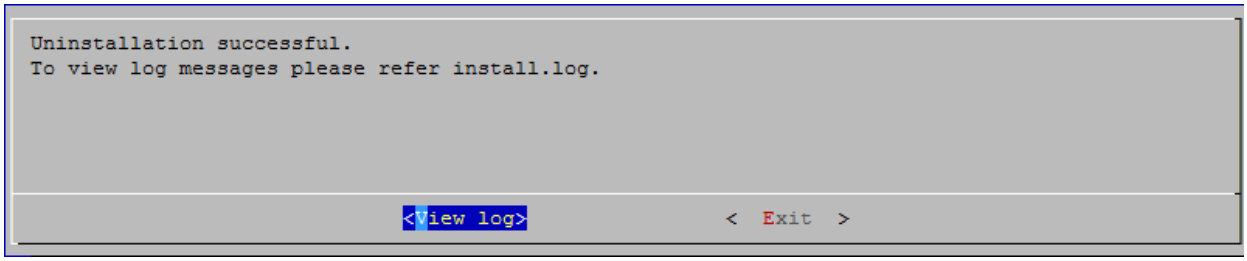
v. Select “all” to uninstall all the installed drivers, libraries and tools or select “custom” to remove specific components.



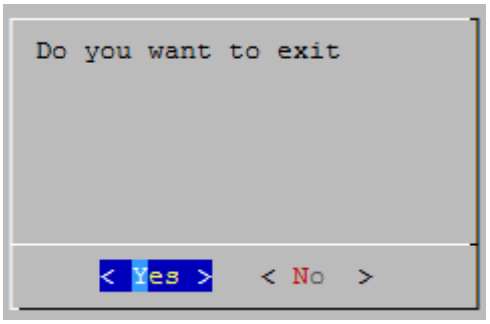
vi. The selected components will now be uninstalled.



vii. After successful uninstallation, select “View log” to view the summary or “Exit” to continue.



viii. Select “Yes” to exit the installer or “No” to go back.



Note *Press Esc or Ctrl+C to exit the installer at any point of time.*

4.1.2. CLI mode (without Dialog utility)

Run the following script with `-u` option to uninstall the Unified Wire Package:

```
[root@host]# ./install.py -u <target>
```

Note *View the help by typing* `[root@host]./install.py -h` for more information

4.1.3. CLI mode

Navigate to the 'ChelsioUwire-x.x.x.x' directory. Uninstall the Unified Wire package using the following command:

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

4.1.4. CLI mode (individual drivers)

You can also choose to uninstall drivers individually. Provided here are steps to uninstall NIC, TOE, iWARP, Bypass and UDP Segmentation Offload drivers. To know about other drivers, access help by running `make help`

- To uninstall NIC driver :

```
[root@host]# make nic_uninstall
```

- To uninstall offload drivers:

```
[root@host]# make toe_uninstall
```

- To uninstall iWARP drivers:

```
[root@host]# make iwarp_uninstall
```

- To uninstall Bypass driver:

```
[root@host]# make bypass_uninstall
```

- To uninstall UDP Segmentation Offload driver:

```
[root@host]# make udp_offload_uninstall
```

4.2. Uninstalling Chelsio Unified Wire from RPM

Navigate to the 'ChelsioUwire-x.x.x.x' directory. Run the following command:

```
[root@host]# ./uninstall.py <inbox/ofed>
```

inbox : for removing all Chelsio drivers.
ofed : for removing OFED and Chelsio drivers.

5. Configuring T4 interfaces

In order to test T4's features it is required to use two machines both with Chelsio's T4 network card installed. These two machines can be connected directly without a switch (back-to-back), or both connected to a 10Gb switch. The interfaces have to be declared and configured. The configuration files for network interfaces on Red Hat Enterprise Linux (RHEL) distributions are kept under `/etc/sysconfig/network-scripts`.

5.1. Configuring network-scripts

A typical interface network-script (e.g. eth0) on RHEL 6.0 looks like the following:

```
# file: /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
HWADDR=00:30:48:32:6A:AA
ONBOOT="yes"
NM_CONTROLLED="no"
BOOTPROTO="static"
IPADDR=10.192.167.111
NETMASK=255.255.240.0
```

Note *On earlier versions of RHEL the NETMASK attribute is named IPMASK. Make sure you are using the right attribute name.*

In the case of DHCP addressing the last two lines should be removed and `BOOTPROTO="static"` should be changed to `BOOTPROTO="dhcp"`

The `ifcfg-ethX` files have to be created manually. They are required for bringing the interfaces up and down and attribute the desired IP addresses.

5.2. Creating network-scripts

To spot the new interfaces, make sure the driver is unloaded first. To that point `ifconfig -a | grep HWaddr` should display all non T4 interfaces whose drivers are loaded, whether the interfaces are up or not.

```
[root@host]# ifconfig -a | grep HWaddr
eth0 Link encap:Ethernet HWaddr 00:30:48:32:6A:AA
```

Then load the driver using the `modprobe cxgb4` command (for the moment it does not make any difference whether we are using NIC-only or the TOE-enabling driver). The output of `ifconfig` should display the T4 interfaces now:

```
[root@host]# ifconfig -a | grep HWaddr
eth0 Link encap:Ethernet HWaddr 00:30:48:32:6A:AA
eth1 Link encap:Ethernet HWaddr 00:07:43:04:6B:E9
eth2 Link encap:Ethernet HWaddr 00:07:43:04:6B:F1
eth3 Link encap:Ethernet HWaddr 00:07:43:04:6B:F9
eth4 Link encap:Ethernet HWaddr 00:07:43:04:6C:01
```

For each T4 interface you can write a configuration file in `/etc/sysconfig/network-scripts`. The `ifcfg-eth1` could look like:

```
# file: /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE="eth1"
HWADDR=00:07:43:04:6B:E9
ONBOOT="no"
NM_CONTROLLED="no"
BOOTPROTO="static"
IPADDR=10.192.167.112
NETMASK=255.255.240.0
```

From now on, the `eth1` interface of the T4 adapter can be brought up and down through the `ifup eth1` and `ifdown eth1` commands respectively. Note that it is of course not compulsory to create a configuration file for every interface if you are not planning to use them all.

5.3. Checking Link

Once the network-scripts are created for the T4 interfaces you should check the link i.e. make sure it is actually connected to the network. First, bring up the interface you want to test using `ifup eth1`.

You should now be able to ping any other machine from your network provided it has ping response enabled.

6. 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.

II. Network (NIC/TOE)

1. Introduction

Chelsio's T4 series of Unified Wire Adapters provide extensive support for NIC operation, including all stateless offload mechanisms for both IPv4 and IPv6 (IP, TCP and UDP checksum offload, LSO - Large Send Offload aka TSO - TCP Segmentation Offload, and assist mechanisms for accelerating LRO - Large Receive Offload).

A high performance fully offloaded and fully featured TCP/IP stack meets or exceeds software implementations in RFC compliance. Chelsio's T4 engine provides unparalleled performance through a specialized data flow processor implementation and a host of features designed for high throughput and low latency in demanding conditions and networking environments, using standard size Ethernet frames.

TCP offload is fully implemented in the hardware, thus freeing the CPU from TCP/IP overhead. The freed CPU can be used for any computing needs. The TCP offload in turn removes network bottlenecks and enables applications to take full advantage of the networking capabilities.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio Network driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-SO-CR
- T420-CX
- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio Network driver runs on Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)*
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 10 SP3 kernel (SLES10SP3), 2.6.16.60-0.54.5(64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)*
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Ubuntu 12.04, 3.2.0-23
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35*
- Kernel.org linux-2.6.36*
- Kernel.org linux-2.6.37*
- Kernel.org linux-2.6.39
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

* Limited QA performed.

2. Software/Driver Loading

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

2.1. Loading in NIC mode (without full offload support)

To load the Network driver without full offload support, run the following command:

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

2.2. Loading in TOE mode (with full offload support)

To enable full offload support, run the following command:

```
[root@host]# modprobe t4_tom
```

Note *Offload support needs to be enabled upon each reboot of the system. This can be done manually as shown above.*

3. Software/Driver Unloading

3.1. Unloading the NIC driver

To unload the NIC driver, run the following command:

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

3.2. Unloading the TOE driver

Please reboot the system to unload the TOE driver.

4. Software/Driver Configuration and Fine-tuning

4.1. Instantiate Virtual Functions (SR-IOV)

To instantiate the Virtual functions, load the cxgb4 driver with `num_vf` parameter with a non-zero value. For example:

```
[root@host]# modprobe cxgb4 num_vf=1,0,0,0
```

The number(s) provided for `num_vf` parameter specifies the number of Virtual Functions to be instantiated per Physical Function. The Virtual Functions can be assigned to Virtual Machines (Guests). A maximum of 64 Virtual Functions can be instantiated with 16 Virtual Functions per Physical Function. Loading the cxgb4 driver with `num_vf` parameter loads the cxgb4vf module (the driver for Virtual Functions) in the host by default. Hence unload the cxgb4vf module (on the host) before assigning Virtual Functions to the Virtual Machines (Guest s), using the following command:

```
[root@host]# rmmod cxgb4vf
```

Note *To get familiar with physical and virtual function terminologies, please refer the PCI Express specification.*

4.2. Performance tuning

In order to auto tune the system and TOE devices for best performance, Chelsio recommends installing the **tools** which will copy `t4_perftune.sh` script to **/sbin** directory. Run the script by using the following command:

```
[root@host]# t4_perftune.sh
```

This script will configure RSS and enable Interrupt Coalescing.

- **Receiver Side Scaling (RSS)**

Receiver Side Scaling enables the receiving network traffic to scale with the available number of processors on a modern networked computer. RSS enables parallel receive processing and dynamically balances the load among multiple processors. Chelsio's T4 network controller fully supports Receiver Side Scaling for IPv4 and IPv6.

This script first determines the number of CPUs on the system and then each receiving queue is bound to an entry in the system interrupt table and assigned to a specific CPU. Thus, each receiving queue interrupts a specific CPU through a specific interrupt now. For example, on a 4-core system, `t4_perftune.sh` gives the following output:

```
[root@host]# t4_perftune.sh
...
Configuring Chelsio T4 devices ...
IRQ table length 4
Writing 1 in /proc/irq/36/smp_affinity
Writing 2 in /proc/irq/37/smp_affinity
Writing 4 in /proc/irq/38/smp_affinity
Writing 8 in /proc/irq/39/smp_affinity
eth6 now up and tuned
...
```

Because there are 4 CPUs on the system, 4 entries of interrupts are assigned. For other T4 network interfaces, you should see similar output message.

Now the receiving traffic is dynamically assigned to one of the system's CPUs through a T4 queue. This achieves a balanced usage among all the processors. This can be verified, for example, by using the **iperf** tool. First set up a server on the receiver host:

```
[root@receiver_host]# iperf –s
```

Then on the sender host, send data to the server using the iperf client mode. To emulate a moderate traffic workload, use *-P* option to request 20 TCP streams from the server:

```
[root@sender_host]# iperf -c receiver_host_name_or_IP -P 20
```

Then on the receiver host, look at interrupt rate at `/proc/interrupts`:

```
[root@receiver_host]# cat /proc/interrupts | grep eth6

Id      CPU0      CPU1      CPU2      CPU3       type              interface
36:   115229         0         0         1   PCI-MSI-edge     eth6 (queue 0)
37:        0    121083         1         0   PCI-MSI-edge     eth6 (queue 1)
38:        0         0    105423         1   PCI-MSI-edge     eth6 (queue 2)
39:        0         0         0    115724   PCI-MSI-edge     eth6 (queue 3)
```

Now interrupts from eth6 are evenly distributed among the 4 CPUs.

Without T4's RSS support, however, the interrupts caused by network traffic may be distributed unevenly over CPUs. For your information, the traffic produced by the same iperf commands gives the following output in `/proc/interrupts`.

```
[root@receiver_host]# cat /proc/interrupts | grep eth6

Id      CPU0      CPU1      CPU2      CPU3       type              interface
36:        0         9         0     17418  PCI-MSI-edge     eth6 (queue 0)
37:        0         0     21718      2063  PCI-MSI-edge     eth6 (queue 1)
38:        0         7    391519       222  PCI-MSI-edge     eth6 (queue 2)
39:        1         0        33     17798  PCI-MSI-edge     eth6 (queue 3)
```

Here there are 4 receiving queues from the T4's eth6 interface, but they are not bound to a specific CPU or interrupt entry. Queue 2 has caused a very large number of interrupts on CPU2 while CPU0 and CPU1 are barely used by any of the four queues. Enabling RSS is thus essential for best performance.

Note

Linux's `irqbalance` *may take charge of distributing interrupts among CPUs on a multiprocessor platform. However,* `irqbalance` *distributes interrupt requests from all hardware devices across processors. For a server with T4 network card constantly receiving large volume of data at 10Gbps, the network interrupt demands are significantly high. Under such circumstances, it is necessary to enable RSS to balance the network load across multiple processors and achieve the best performance.*

- **Interrupt Coalescing**

The idea behind Interrupt Coalescing (IC) is to avoid flooding the host CPUs with too many interrupts. Instead of throwing one interrupt per incoming packet, IC waits for 'n' packets to be available in the Rx queues and placed into the host memory through DMA operations before an interrupt is thrown, reducing the CPU load and thus improving latency. It can be changed using the following command:

```
[root@host]# ethtool –C ethX rx-frames n
```

Note *For more information, run the following command:*
`[root@host]# ethtool -h`

- **Large Receive Offload / Generic Receive Offload**

Large Receive Offload or Generic Receive Offload is a performance improvement feature at the receiving side. LRO/GRO aggregates the received packets that belong to same stream, and combines them to form a larger packet before pushing them to the receive host network stack. By doing this, rather than processing every small packet, the receiver CPU works on fewer packet headers but with same amount of data. This helps reduce the receive host CPU load and improve throughput in a 10Gb network environment where CPU can be the bottleneck.

LRO and GRO are only different names to refer to the same receiver packets aggregating feature. LRO and GRO actually differ in their implementation of the feature in the Linux kernel. The feature was first added into the Linux kernel in version 2.6.24 and named Large Receive Offload (LRO). However LRO only works for TCP and IPv4. As from kernel 2.6.29, a new protocol-independent implementation removing the limitation is added to Linux, and it is named Generic Receive Offload (GRO). The old LRO code is still available in the kernel sources but whenever both GRO and LRO are presented GRO is always the preferred one to use.

Please note that if your Linux system has IP forwarding enabled, i.e. acting as a bridge or router, the LRO needs to be disabled. This is due to a known kernel issue.

Chelsio's T4 card supports both hardware assisted GRO/LRO and Linux-based GRO/LRO. `t4_tom` is the kernel module that enables the hardware assisted GRO/LRO. If it is not already in the kernel module list, use the following command to insert it:

```
[root@host]# lsmod | grep t4_tom
[root@host]# modprobe t4_tom
[root@host]# lsmod | grep t4_tom
t4_tom 88378 0 [permanent]
toecore 21618 1 t4_tom
cxgb4 225342 1 t4_tom
```

Then T4's hardware GRO/LRO implementation is enabled.

If you would like to use the Linux GRO/LRO for any reason, first the `t4_tom` kernel module needs to be removed from kernel module list. Please note you might need to reboot your system.

After removing the `t4_tom` module, you can use `ethtool` to check the status of current GRO/LRO settings, for example:

```
[root@host]# ethtool -k eth6
Offload parameters for eth6:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
```

Now the `generic-receive-offload` option is on. This means GRO is enabled. Please note that there are two offload options here: `generic-receive-offload` and `large-receive-offload`. This is because on this Linux system (RHEL6.0), the kernel supports both GRO and LRO. As mentioned earlier, GRO is always the preferred option when both of them are present. On other systems LRO might be the only available option. Then `ethtool` could be used to switch LRO on and off as well.

When Linux's GRO is enabled, Chelsio's T4 driver provides two GRO-related statistics. They are displayed using the following command:

```
[root@host]# ethtool -S eth6
...
GROPackets : 0
GROMerged : 897723
...
```

`GROPackets` is the number of held packets. Those are candidate packets held by the kernel to be processed individually or to be merged to larger packets. This number is usually zero. `GROMerged` is the number of packets that merged to larger packets. Usually this number increases if there is any continuous traffic stream present.

`ethtool` can also be used to switch off the GRO/LRO options when necessary:

```
[root@host]# ethtool -K eth6 gro off
[root@host]# ethtool -k eth6
Offload parameters for eth6:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: off
large-receive-offload: off
```

The output above shows a disabled GRO.

4.3. Network Device Configuration

Please refer to the operating system documentation for administration and configuration of network devices.

Note

Some operating systems may attempt to auto-configure the detected hardware and some may not detect all ports on a multi-port adapter. If this happens, please refer to the operating system documentation for manually configuring the network device.

III. Virtual Function Network (vNIC)

1. Introduction

The ever increasing network infrastructure of IT enterprises has lead to a phenomenal increase in maintenance and operational costs. IT managers are forced to acquire more physical servers and other data center resources to satisfy storage and network demands. To solve the Network and I/O overhead, users are opting for server virtualization which consolidates I/O workloads onto lesser physical servers thus resulting in efficient, dynamic and economical data center environments. Other benefits of Virtualization include improved disaster recovery, server portability, cloud computing, Virtual Desktop Infrastructure (VDI), etc.

Chelsio's T4 Unified Wire family of Adapters deliver increased bandwidth, lower latency and lower power with virtualization features to maximize cloud scaling and utilization. The adapters support SFP+ and 10BASE-T media. The adapters also provide full support for PCI-SIG SR-IOV to improve I/O performance on a virtualized system. User can configure up to 128 Virtual and 8 Physical functions (with 4 PFs as SR-IOV capable) along with 336 virtual MAC addresses.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the Chelsio vNIC driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T422-CR
- T420-SO-CR
- T420-CX
- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio vNIC driver runs on Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)*
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 10 SP3 kernel (SLES10SP3), 2.6.16.60-0.54.5(64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)*
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)*
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35
- Kernel.org linux-2.6.36
- Kernel.org linux-2.6.37
- Kernel.org linux-2.6.39*
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

* Limited QA performed.

2. Software/Driver Loading

The vNIC driver must be loaded or unloaded on the Guest OS by the root user. Any attempt to load the driver as a regular user will fail.

2.1. Loading the driver

To load the driver, run the following command:

```
[root@host]# modprobe cxgb4vf
```

3. Software/Driver Unloading

3.1. Unloading the driver

To unload the driver, execute the following command:

```
[root@host]# rmmod cxgb4vf
```

4. Software/Driver Configuration and Fine-tuning

4.1. Instantiate Virtual Functions

To instantiate Chelsio Virtual Functions, please refer to the **Network (NIC/TOE)** section (click here)

IV. iWARP (RDMA)

1. Introduction

Chelsio's T4 engine implements a feature rich RDMA implementation which adheres to the IETF standards with optional markers and MPA CRC-32C.

The iWARP RDMA operation benefits from the virtualization, traffic management and QoS mechanisms provided by T4. It is possible to ACL process iWARP RDMA packets. It is also possible to rate control the iWARP traffic on a per-connection or per-class basis, and to give higher priority to QPs that implement distributed locking mechanisms. The iWARP operation also benefits from the high performance and low latency TCP implementation in the offload engine.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio iWARP driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-CX
- T404-BT

1.2. Software Requirements

To use the iWARP functionality with Chelsio adapters, user needs to install the iWARP drivers as well as the libcxgb4, libibverbs, and librdmacm libraries. Chelsio provides the iWARP drivers and libcxgb4 library as part of the driver package. The other libraries are provided as part of the Open Fabrics Enterprise Distribution (OFED) package.

The OFED package is composed of several software components and is generally intended to use on a computer cluster built with a iWARP network or Infiniband network.

Please visit http://www.openfabrics.org/downloads/OFED to download the appropriate OFED

Note *If you are planning to upgrade the OFED package on one member of the cluster, the upgrade needs to be installed on all the members.*

1.2.1. Linux Requirements

The Chelsio iWARP driver runs on Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1),2.6.32.12-0.7 (64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)*
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)*
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Ubuntu 12.04, 3.2.0-23
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35
- Kernel.org linux-2.6.36
- Kernel.org linux-2.6.37
- Kernel.org linux-2.6.39
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

*Limited QA performed

2. Software/Driver Loading

Before loading the iWARP driver, please install OFED software as mentioned below:

2.1. Installing OFED software

2.1.1. Installing OFED-1.5.4 software

i. Download OFED-1.5.4 package from the following location

http://www.openfabrics.org/downloads/OFED/ofed-1.5.4/OFED-1.5.4.tgz

ii. Untar the OFED package using following command:

```
[root@host]# tar -xf <path to OFED package>
```

iii. For iWARP driver to work additional library, libibverbs and librdmacm need to be built. For this navigate to OFED directory.

```
[root@host]# cd <path to OFED package directory>/OFED-X.Y.Z
```

iv. Run the following command to install all the packages from OFED

```
[root@host]# ./install.pl --all
```

v. If you are familiar with OFED installation you can run the following command to display the OFED Distribution Software Installation Menu:

```
[root@host]# ./install.pl
```

vi. Enter 2 and then 4 in the menu to initiate custom Installation. You can now choose to install various OFED drivers, libraries and applications.

vii. Reboot system for changes to take effect.

Note *The above mentioned installation steps can also be used to install OFED-1.5.3 package. Download the package from* http://www.openfabrics.org/downloads/OFED/ofed-1.5.3/OFED-1.5.3.tgz

2.1.2. Installing OFED-1.5.2 software

i. Download OFED-1.5.2 package from the following location

http://www.openfabrics.org/downloads/OFED/ofed-1.5.2/OFED-1.5.2.tgz

ii. Follow steps (ii) – (vii) mentioned in section 2.1.1. above before proceeding.
iii. Navigate to the share directory of OPENMPI as:

```
[root@host]#cd /usr/mpi/gcc/openmpi-1.4.3/share
```

iv. Navigate to OPENMPI in share directory:

```
[root@host]# cd openmpi
```

v. Now your present working directory should look like

```
/usr/mpi/gcc/openmpi-1.4.3/share/openmpi
```

vi. Open the file `mca-btl-openib-device-params.ini`
vii. Edit the file under the Chelsio section.
viii. If you are using a T4 adapter then make an entry into the `vendor_part_id` which contains a single entry as:

```
vendor_part_id = 0xa000
```

ix. Change it as:

```
vendor_part_id =
0xa000,0x4400,0x4401,0x4402,0x4403,0x4404,0x4405,0x4406,0x4407,0x4408,0x4409
,0x440a,0x440b,0x440c
```

x. Also make sure the entries for the `receive_queues` on both Chelsio T3 and Chelsio T4 are same.

Note *WD-UDP libraries are pre-compiled binaries against OFED-1.5.4 package*

2.2. Compiling and Loading iWARP driver

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

i. Change your current working directory to driver package directory & run the following command :

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

ii. To load the iWARP driver we need to load the NIC driver & core RDMA drivers first. Run the following commands:

```
[root@host]#modprobe cxgb4
[root@host]#modprobe iw_cxgb4
[root@host]#modprobe rdma_ucm
```

iii. Set Chelsio driver option for MPI connection changes. Run the following command on all systems:

```
[root@host]# echo 1 > /sys/module/iw_cxgb4/parameters/peer2peer
```

3. Software/Driver Unloading

To unload the iWARP driver, run the following command:

```
[root@host]#rmmod iw_cxgb4
```

4. Software/Driver Configuration and Fine-tuning

4.1. Testing connectivity with *ping* and *rping*

Load the NIC, iWARP & core RDMA modules as mentioned in **Software/Driver Loading** section. After which, you will see one or two ethernet interfaces for the T4 device. Configure them with an appropriate ip address, netmask, etc. You can use the Linux *ping* command to test basic connectivity via the T4 interface. To test RDMA, use the *rping* command that is included in the librdmacm-utils RPM:

Run the following command on the server machine:

```
[root@host]# rping -s -a server_ip_addr -p 9999
```

Run the following command on the client machine:

```
[root@host]# rping -c –Vv -C10 -a server_ip_addr -p 9999
```

You should see ping data like this on the client:

```
ping data: rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
ping data: rdma-ping-1: BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
ping data: rdma-ping-2: CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
ping data: rdma-ping-3: DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu
ping data: rdma-ping-4: EFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
ping data: rdma-ping-5: FGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvw
ping data: rdma-ping-6: GHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
ping data: rdma-ping-7: HIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
ping data: rdma-ping-8: IJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
ping data: rdma-ping-9: JKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyzA
client DISCONNECT EVENT...
#
```

4.2. Enabling various MPIs

4.2.1. DAPL Library configuration for Intel MPI and Platform MPI

You must set the iWARP (iw_cxgb4) module option `peer2peer=1` on all systems. This can be done by writing to the `/sys/module/` file system during boot.

E.g.: For RHEL 5 and SLES platforms use following commands:

```
[root@host]# echo 1 > /sys/module/iw_cxgb4/parameters/peer2peer
```

OR

You can add the following line to `/etc/modprobe.conf` to set the option at module load time:

```
options iw_cxgb4 peer2peer=1
```

To run Intel MPI over RDMA interface, DAPL 2.0 should be set up as follows:

Enable the Chelsio device by adding an entry at the beginning of the /`etc/dat.conf` file for the Chelsio interface. For instance, if your Chelsio interface name is eth2, then the following line adds a DAT version 2.0 device named “chelsio2" for that interface:

```
chelsio2 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth2 0" ""
```

4.2.2. Setting shell for Remote Login

User needs to set up authentication on the user account on all systems in the cluster to allow user to remotely logon or executing commands without password.

Quick steps to set up user authentication:

i. Change to user home directory

```
[root@host]# cd
```

ii. Generate authentication key

```
[root@host]# ssh-keygen -t rsa
```

iii. Press enter upon prompting to accept default setup and empty password phrase
iv. Create authorization file

```
[root@host]# cd .ssh
[root@host]# cat *.pub > authorized_keys
[root@host]# chmod 600 authorized_keys
```

v. Copy directory .ssh to all systems in the cluster

```
[root@host]# cd
[root@host]# scp -r /root/.ssh remotehostname-or-ipaddress:
```

4.2.3. Configuration of various MPIs (Installation and Setup)

- **Intel-MPI**

i. Download latest Intel MPI from the Intel website
ii. Copy the license file (.lic file) into `l_mpi_p_x.y.z` directory
iii. Create machines.LINUX (list of node names) in `l_mpi_p_x.y.z`
iv. Select advanced options during installation and register the MPI.
v. Install software on every node.

```
[root@host]#./install.py`
```

vi. Set IntelMPI with mpi-selector (do this on all nodes).

```
[root@host]# mpi-selector --register intelmpi --source-dir
/opt/intel/impi/3.1/bin/
[root@host]# mpi-selector --set intelmpi
```

vii. Edit `.bashrc` and add these lines:

```
export RSH=ssh
export DAPL_MAX_INLINE=64
export I_MPI_DEVICE=rdssm:chelsio
export MPIEXEC_TIMEOUT=180
export MPI_BIT_MODE=64
```

viii. Logout & log back in.
ix. Populate `mpd.hosts` with node names.

Note

- *The hosts in this file should be Chelsio interface IP addresses.*
- `I_MPI_DEVICE=rdssm:chelsio` *assumes you have an entry in* `/etc/dat.conf` *named* `chelsio`*.*
- `MPIEXEC_TIMEOUT` *value might be required to increase if heavy traffic is going across the systems.*

x. Contact Intel for obtaining their MPI with DAPL support.
xi. To run Intel MPI applications:

```
mpdboot -n -r ssh --ncpus=
mpiexec -ppn -n 2 /opt/intel/impi/3.1/tests/IMB-3.1/IMB-MPI1
```

The performance is best with NIC MTU set to 9000 bytes.

- **Platform MPI Setup**

i. After you have installed Platform MPI RPM and obtained the license from Platform computing, unset the previously selected MPI as

```
mpi-selector --unset <previous MPI>
```

ii. Edit `.bashrc` and add these lines:

```
export MPI_ROOT=/opt/platform_mpi/
export PATH=$MPI_ROOT/bin:/opt/bin:$PATH
export MANPATH=$MANPATH:$MPI_ROOT/share/man
```

iii. Log out and log back in.
iv. To run Platform MPI over RDMA interface, DAPL 2.0 should be set up as follows:

Enable the Chelsio device by adding an entry at the beginning of the /`etc/dat.conf` file for the Chelsio interface. For instance, if your Chelsio interface name is eth2, then the following line adds a DAT 2.0 device named "chelsio2" for that interface:

```
chelsio2 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth2 0" ""
```

v. Start the License Manager and run the test as:

```
mpirun -prot -e DAPL_MAX_INLINE=64 -UDAPL -hostlist <name of hosts
separated by comma> -np 4 /opt/platform_mpi/tests/IMB-3.2/IMB-MPI1
```

- **Open MPI (Installation and Setup)**

The latest release of Open-MPI-1.4.3 comes with OFED package only. Select Open MPI package while installing OFED package.

Open MPI iWARP support is only available in Open MPI version 1.3 or greater.

Open MPI will work without any specific configuration via the openib btl. Users wishing to performance tune the configurable options may wish to inspect the receive queue values. Those can be found in the "Chelsio T4" section of `mca-btl-openib-device-params.ini`.

To run Open MPI applications:

```
mpirun --host node1,node2 -mca btl openib,sm,self /usr/mpi/gcc/openmpi-
1.4.3/tests/IMB-3.2/IMB-MPI1
```

- **MVAPICH2-1.7 setup**

The latest release of MVAPICH2-1.7 comes with OFED package only. Select MVAPICH2 package while installing OFED package.

i. Set MVAPICH2 with mpi-selector (do this on all nodes).

```
[root@host]# mpi-selector --set mvapich2-1.7
```

ii. Edit `.bashrc` and add these lines:

```
export MVAPICH2_HOME=/usr/mpi/gcc/mvapich2-1.7/
export MV2_USE_IWARP_MODE=1
export MV2_USE_RDMA_CM=1
```

iii. Logout & log back in.

iv. Populate `mpd.hosts` with node names.

v. On each node, create /etc/mv2.conf with a single line containing the IP address of the local T4 interface. This is how MVAPICH2 picks which interface to use for RDMA traffic.

vi. Run the MPI application as :

```
mpirun_rsh -ssh -np 8 -hostfile mpd.hosts $MVAPICH2_HOME/tests/IMB-3.2/IMB-MPI1
```

4.3. Setting up NFS-RDMA

For NFS-RDMA to work we need to compile and build 2 modules - *RNFS-UTILS* and *NFSRDMA* from OFED during installation. The following section explains how to setup NFS-RDMA depending upon the version of OFED used:

4.3.1. Building and Installing NFS-RDMA

- **Using OFED-1.5.3.x**

i. Download OFED-1.5.3.x package from the following location

http://www.openfabrics.org/downloads/OFED/ofed-1.5.3/

ii. Untar the package using the following command:

```
[root@host]# tar -xf <path to OFED package>
```

iii. Navigate to OFED-1.5.3.x directory

```
[root@host]# cd OFED-1.5.3.x
```

iv. Generate a config file:

```
[root@host]#./install.pl -p
```

A config file `ofed-all.conf` will be generated in the same directory.

v. Open the file and append the following entries at the end

```
nfsrdma=y
rnfs-utils=y
```

vi. Save and Exit.
vii. Now, start OFED installation with NFS-RDMA support:

```
[root@host]#./install -c ofed-all.conf
```

Note *If you are using OFED-1.5.2 and installing using GUI mode, the interactive menus will direct you through the install process.*

- **Using OFED-1.5.4**

i. Download OFED-1.5.4 package from the following location

http://www.openfabrics.org/downloads/OFED/ofed-1.5.4/OFED-1.5.4.tgz

ii. Untar the package using the following command:

```
[root@host]# tar -xf <path to OFED package>
```

iii. Navigate to OFED-1.5.4 directory

```
[root@host]# cd OFED-1.5.4
```

iv. Start OFED installation

```
[root@host]#./install.py -all
```

4.3.2. Starting NFS-RDMA

- **Server-side settings**

Follow the steps mentioned below to set up an NFS-RDMA server.

i. Make entry in `/etc/exports` file for the directories you need to export using NFS-RDMA on server as:

```
/share/rdma      *(fsid=0,async,insecure,no_root_squash)
/share/rdma1     *(fsid=1,async,insecure,no_root_squash)
```

Note that for each directory you export, you should have DIFFERENT fsid's.

ii. Load the iwarp modules and make sure peer2peer is set to 1.
iii. Load `xprtrdma` and `svcrdma` modules as:

```
[root@host]#modprobe xprtrdma
[root@host]#modprobe svcrdma
```

iv. Start the nfs service as:

```
[root@host]#service nfs start
```

All services in NFS should start without errors.

v. Now we need to edit the file portlist in the path `/proc/fs/nfsd/`
vi. Include the rdma port 2050 into this file as:

```
[root@host] #echo rdma 2050 > /proc/fs/nfsd/portlist
```

vii. Run `exportfs` to make local directories available for Network File System (NFS) clients to mount.

```
[root@host]#exportfs
```

Now the NFS-RDMA server is ready.

- **Client-side settings**

Follow the steps mentioned below at the client side.

i. Load the iwarp modules and make sure peer2peer is set to 1. Make sure you are able to ping and ssh to the server Chelsio interface through which directories will be exported.
ii. Load the `xprtrdma` module.

```
[root@host]#modprobe xprtrdma
```

iii. Run the `showmount` command to show all directories from server as:

```
[root@host]#showmount –e <server-chelsio-ip>
```

iv. Once the exported directories are listed, mount them as:

```
[root@host]#mount.rnfs <serverip>:<directory> <mountpoint-on-client> -i -o
rdma,port=2050
```

V. WD-UDP

1. Introduction

Chelsio WD-UDP (Wire Direct-User Datagram Protocol) with Multicast is a user-space UDP stack with Multicast address reception and socket acceleration that enables users to run their existing UDP socket applications unmodified.

It features software modules that enable direct wire access from user space to the Chelsio T4 network adapter with complete bypass of the kernel, which results in an ultra-low, three microsecond deterministic latency 10Gb Ethernet solution for high frequency trading and other delay-sensitive applications.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio WD-UDP driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T422-CR
- T420-CX
- T404-BT

1.2. Software Requirements

To use WD-UDP with Chelsio adapters, user needs to install the iWARP drivers as well as the libcxgb4, libcxgb4_sock, libcxgb4_udp, libibverbs, and librdmam libraries. Chelsio provides the iWARP drivers, libcxgb4, libcxgb4_sock and libcxgb4_udp libraries as part of its driver package. The other drivers are provided as part of the Open Fabrics Enterprise Distribution (OFED) package.

1.2.1. Linux Requirements

The Chelsio WD-UDP driver runs on Linux-based platforms and therefore it is a base requirement for running the software.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)

- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1),2.6.32.12-0.7 (64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)*
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35
- Kernel.org linux-2.6.36
- Kernel.org linux-2.6.37
- Kernel.org linux-2.6.39
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

*Limited QA performed

2. Software/Driver Compiling and Loading

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

i. Change your current working directory to driver package directory & run the following command :

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

ii. InfiniBand modules from the OFED package should be loaded before proceeding. To load the WD-UDP driver we need to load the NIC driver & core RDMA drivers first. Run the following commands:

```
[root@host]#modprobe cxgb4
[root@host]#modprobe iw_cxgb4
[root@host]#modprobe rdma_ucm
```

3. Software/Driver Unloading

To unload the WD-UDP driver, run the following command:

```
[root@host]# rmmod iw_cxgb4
```

4. Software/Driver Configuration and Fine-tuning

4.1. Accelerating UDP Socket communications

The libcxgb4_sock library is a LD_PRELOAD-able library that accelerates UDP Socket communications transparently and without recompilation of the user application. This section describes how to use libcxgb4_sock.

By preloading libcxgb4_sock, all sockets created by the application are intercepted and possibly accelerated based on the user's configuration. Once accelerated, data for the UDP endpoint are transmitted or received via HW queues allocated specifically for the accelerated endpoint, bypassing the kernel, the host networking stack and sockets framework, and enabling ultra-low latency and high bandwidth utilization.

Due to HW resource limitations, only a small number of queues can be allocated for UDP acceleration. Therefore only performance critical UDP applications should use libcxgb4_sock.

Only 64 IPv4 UDP sockets / 28 IPv6 UDP Sockets can be accelerated per Chelsio T4 device used.

4.1.1. Application Requirements

Certain application behavior is not supported by libcxb4_sock in this release. If your application does any of the following, it will not work with libcxgb4_sock:

- Calling fork() after creating UDP sockets and using the UDP socket in the child process.
- Using multiple threads on a single UDP socket without serialization. For instance, having one thread sending concurrently with another thread receiving. If your application does this, you need to serialize these paths with a spin or mutex lock.
- Only 1 UDP endpoint is allowed to bind to a given port per host. So if you have multiple processes on the same host binding to the same UDP port number, you cannot use libcxgb4_sock.
- Applications must have root privileges to use libcxgb4_sock.
- Applications requiring bonded T4 interfaces are not currently supported.

The performance benefit observed with libcxgb4_sock will vary based on your application's behavior. While all UDP IO is handled properly, only certain datagrams are accelerated. Non accelerated IO is handled by libcxgb4_sock via the host networking stack seamlessly. Both Unicast and Multicast datagrams can be accelerated, but the datagrams must meet the following criteria:

- Non fragmented. In other words, they fit in a single IP datagram that is <= the T4 device MTU.
- Routed through the T4 acceleration device. If the ingress datagram arrives via a device other than the T4 acceleration device, then it will not utilize the acceleration path. On

egress, if the destination IP address will not route out via the T4 device, then it too will not be accelerated.

4.1.2. Using libcxgb4_sock

The libcxgb4_sock library utilizes the Linux RDMA Verbs subsystem, and thus requires the RDMA modules be loaded. Ensure that your systems load the *iw_cxgb4* and *rdma_ucm* modules:

```
[root@host]# modprobe iw_cxgb4
[root@host]# modprobe rdma_ucm
```

To preload libcxgb4_sock, use one of the commands mentioned below when starting your application:

```
[root@host]# LD_PRELOAD=libcxgb4_sock.so <pathto>/your_application
```

OR

```
[root@host]# wdload <pathto>/your_application
```

If you want to use *libcxgb4_sock*'s debug capabilities, make the following entry in the */etc/syslog.conf* file:

```
*.debug                                /var/log/cxgb4.log
```

then, restart the service:

```
[root@host]# /etc/init.d/syslog restart
```

Now, preload *libcxgb4_sock_debug* using the command mentioned below when starting your application:

```
[root@host]# LD_PRELOAD=libcxgb4_sock_debug.so CXGB4_SOCK_DEBUG=-1
<pathto>/your_application
```

In addition to preloading `libcxgb4_sock.so`, you must create a configuration file that defines which UDP endpoints should be accelerated, their vlan and priority if any, as well as which T4 interface/port should be used. The file `/etc/libcxgb4_sock.conf` contains these endpoint entries. Create this file on all systems using libcxgb4_sock. Here is the syntax:

```
#
# Syntax:
#
# endpoint {attributes} ...
# where attributes include:
#                interface = interface-name
#                port = udp-port-number
#                vlan = vlan-id
#                priority = vlan-priority
#
# e.g.
# endpoint {
#                interface=eth2.5
#                port = 8000 vlan = 5 priority=1
#   }
# endpoint { interface=eth2 port=9999}
#
# endpoints that bind to port 0 (requesting the host allocate a port)

# can be accelerated with port=0:
#
# endpoint {interface=eth1 port=0}
#
```

Assume your T4 interface is eth2. To accelerate all applications that preload libcxgb4_sock using eth2, you only need one entry in `/etc/libcxgb4_sock.conf`:

```
endpoint {interface=eth2 port=0}
```

If you have eth2 and eth3 configured for example, you can define certain endpoints to eth2 and others to eth3:

```
endpoint {interface=eth2 port=9999}
endpoint {interface=eth3 port=8888}
```

For VLAN support, create your VLANs using the normal OS service (like vconfig, for example), then add entries to define the VLAN and priority for each endpoint to be accelerated:

```
endpoint {interface = eth2.5 port=10000}
endpoint {interface = eth2.7 priority=3 port=9000}
```

For Jumbo Frames of 9000B, follow the steps mentioned below:

```
[root@host]# echo 1024 > /proc/sys/vm/nr_hugepages
[root@host]# CXGB4_SOCK_HUGE_PAGES=1 wdload <pathto>/your_application
```

Note

- *Jumbo frames of 9000B are supported only on kernel 2.6.32 and above.*
- *In order to offload IPv6 UDP sockets, please select "low latency networking" as T4 configuration tuning option during installation.*

4.1.3. Using per process config file

The libcxgb4_sock library utilizes the Linux RDMA Verbs subsystem, and thus requires the RDMA modules be loaded. Ensure that your systems load the iw_cxgb4 and rdma_ucm modules:

```
[root@host]# modprobe iw_cxgb4
[root@host]# modprobe rdma_ucm
```

For individual interfaces, create a config file for "accelerate all" configuration on both nodes:

```
[root@host]# cat /root/config1
endpoint {interface=eth1 port=0}
```

To preload libcxb4_sock and load the config file, use the LD_PRELOAD environment variable along with the config file when starting your application:

```
[root@host]# CXGB4_SOCK_CFG=<path to config file>
LD_PRELOAD=libcxgb4_sock.so <pathto>/your_application
```

4.1.4. Example with hpcbench/udp

The udp benchmark from the hpcbench suite can be used to show the benefits of libcxgb4_sock. The hpcbench suite can be found at:

Source: http://hpcbench.sourceforge.net/index.html

Sample: http://hpcbench.sourceforge.net/udp.html

The nodes in this example, r9 and r10, have T4 eth1 configured and the ports are connected point-to-point.

```
[root@r9 ~]# ifconfig eth1|grep inet
          inet addr:192.168.2.111  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:104:465a/64 Scope:Link

[root@r9 ~]#
[root@r10 ~]# ifconfig eth1|grep inet
          inet addr:192.168.2.112  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:104:456a/64 Scope:Link
[root@r10 ~]#
```

For this benchmark, we need a simple “accelerate all” configuration on both nodes:

```
[root@r9 ~]# cat /etc/libcxgb4_sock.conf
endpoint {interface=eth1 port=0}
[root@r9 ~]#

[root@r10 ~]# cat /etc/libcxgb4_sock.conf
endpoint {interface=eth1 port=0}
[root@r10 ~]#
```

On R10, we run udpserver on port 9000 without libcxgb4_sock preloaded, and on port 90001 with preload:

```
[root@r10 ~]# /usr/local/src/hpcbench/udp/udpserver -p 9000 &
[1] 11453
[root@r10 ~]# TCP socket listening on port [9000]

[root@r10 ~]# LD_PRELOAD=libcxgb4_sock.so
/usr/local/src/hpcbench/udp/udpserver -p 9001 &
[2] 11454
[root@r10 ~]# TCP socket listening on port [9001]
[root@r10 ~]#
```

Then on r9, we run udptest to port 9000 to see the host stack UDP latency:

```
[root@r9 ~]# /usr/local/src/hpcbench/udp/udptest -r 5 -a -h 192.168.1.112 -p
9000
```

Running the same test with libcxgb4_sock:

```
[root@r9 ~]# LD_PRELOAD=libcxgb4_sock.so /usr/local/src/hpcbench/udp/udptest
-r 5 -a -h 192.168.1.112 -p 9001
```

4.1.5. Performance tuning on 2.6.18 kernel

To get better performance with WD-UDP using the 2.6.18 kernel, load the iw_cxgb4 modules with the `ocqp_support=0` parameter. For example,

```
modprobe iw_cxgb4 ocqp_support=0
```

4.1.6. Determining if the application is being offloaded

To see if the application is being offloaded, open a window on one of the machines, and run `tcpdump` against the Chelsio interface. If you see minimal UDP output on the interface, then the UDP traffic is being properly offloaded.

VI. iSCSI PDU Offload Target

1. Introduction

This section describes how to install and configure iSCSI PDU Offload Target software for use as a key element in your iSCSI SAN. The software runs on Linux-based systems that use Chelsio or non-Chelsio based Ethernet adapters. However to guarantee highest performance, Chelsio recommends using Chelsio adapters. Chelsio's adapters include offerings that range from stateless offload adapters (regular NIC) to the full line of TCP/IP Offload Engine (TOE) adapters.

The software implements RFC 3720, the iSCSI standard of the IETF. The software has been fully tested for compliance to that RFC and others and it has been exhaustively tested for interoperability with the major iSCSI vendors.

The software implements most of the iSCSI protocol in software running in kernel mode on the host with the remaining portion, which consists of the entire fast data path, in hardware when used with Chelsio's TOE adapters. When standard NIC Adapters are used the entire iSCSI protocol is executed in software.

The performance of this iSCSI stack is outstanding and when used with Chelsio's hardware it is enhanced further. Because of the tight integration with Chelsio's TOE adapters, this software has a distinct performance advantage over the regular NIC. The entire solution, which includes this software, Chelsio TOE hardware, an appropriate base computer system – including a high end disk subsystem, has industry leading performance. This can be seen when the entire solution is compared to others based on other technologies currently available on the market in terms of throughput and IOPS.

1.1. Features

Chelsio's iSCSI driver stack supports the iSCSI protocol in the Target mode. From henceforth "iSCSI Software Entity" term refers to the iSCSI target.

The Chelsio iSCSI PDU Offload Target software provides the following high level features:

- Expanded NIC Support
  - Chelsio TCP Offload Engine (TOE) Support
    - T4 Based HBAs (T4xx Series cards)
  - Non-Chelsio
    - Runs on regular NICs
    - From wireless through 10 GigE links
- Chelsio Terminator ASIC Support
  - Offloads iSCSI Fast Data Path with Direct Data Placement (DDP)
  - Offloads iSCSI Header and Data Digest Calculations
  - Offload Speeds at 1 Gb and 10 Gb

    - Offloads TCP/IP for NAS simultaneously with iSCSI

- Target Specific features
    - Full compliance with RFC 3720
    - Error Recovery Level 0 (ERL 0)
    - CHAP Support, including Mutual Authentication
    - Internet Storage Name Service (iSNS) Client
    - Target Access Control List (ACL)
    - Multiple Connections per Session
    - Multiple Targets
    - Multiple LUNs per Target
    - Multi Path I/O (MPIO)
    - Greater than 2 TB Disk Support
    - Reserve / Release for Microsoft Cluster© Support
    - Persistent Reservation
    - Dynamic LUN Resizing
    - iSCSI Target Redirection
    - Multiple Target device types
        - Block
        - Virtual Block (LVM, Software RAID, EVMS, etc.)
        - Built in RAM Disk
        - Built in zero copy RAM Disk
    - Supports iSCSI Boot Initiators
    - An Intuitive and Feature Rich Management CLI

This chapter will cover these features in detail.

1.2. Hardware Requirements

1.2.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with iSCSI PDU Offload Target software:

- T420-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-BT
- T404-BT

1.2.2. Adapter Requirements

The Chelsio iSCSI PDU Offload Target software can be used with or without hardware protocol offload technology. When used with protocol offload, a Chelsio TOE adapter must be used. There are four modes of operation using the iSCSI PDU Offload Target software on Ethernet-based adapters:

- Regular NIC (non-Chelsio NIC Adapter) – The software can be used in non-offloaded (regular NIC) mode. Please note however that this is the least optimal mode of operating the software in terms of performance.
- TOE HW Acceleration (Uses Chelsio's T4xx Series adapters) – In TOE mode the software takes advantage of the TCP/IP Offload capability of Chelsio's TOE adapter (without the additional benefit of iSCSI HW acceleration).
- iSCSI HW Acceleration (Uses Chelsio's T4xx Series adapters) – In addition to offloading the TCP/IP protocols in hardware (TOE), this mode also takes advantage of Chelsio's ASIC capability of hardware assisted iSCSI data and header digest calculations as well as using the direct data placement (DDP) feature.
- Mix of iSCSI HW Acceleration and TOE HW Acceleration (Uses T4xx Series adapters) – Using a special software mode the stack can be configured to change between iSCSI acceleration or just TOE acceleration, depending if digests are used or not.

1.2.3. Storage Requirements

When using the Chelsio iSCSI target, a minimum of one hardware storage device is required. This device can be any of the device types that are supported (block, virtual block, RAM disk). Multiple storage devices are allowed by configuring the devices to one target or the devices to multiple targets. The software allows multiple targets to share the same device but use caution when doing this.

Chelsio's implementation of the target iSCSI stack has flexibility to accommodate a large range of configurations. For quick testing, using a RAM Disk as the block storage device works nicely. For deployment in a production environment a more sophisticated system would be needed. That typically consists of a system with one or more storage controllers with multiple disk drives attached running software or hardware based RAID.

1.3. Software Requirements

`chiscsi_base.ko` is iSCSI non-offload target mode driver and `chiscsi_t4.ko` is iSCSI PDU offload target mode driver.

cxgb4, toecore, t4_tom and chiscsi_base modules are required by `chiscsi_t4.ko` module to work in offloaded mode. Whereas in iscsi non-offloaded target(NIC) mode, only `cxgb4` is needed by `chiscsi_base.ko` module.

1.3.1. Linux Requirements

The iSCSI PDU Offload Target software runs on Linux-based platforms and therefore it is a base requirement for running the software.

Currently the software is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)*
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)*
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)*
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Kernel.org linux-2.6.34*
- Kernel.org linux-2.6.35*
- Kernel.org linux-2.6.36*
- Kernel.org linux-2.6.37*
- Kernel.org linux-2.6.39*
- Kernel .org linux-3.1

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

*Limited QA performed

Note *chiscsi_t4 compiles on all Kernel.org Linux Kernels 2.6.18+ Kernels, but only tested and guaranteed to work with above mentioned OSes.*

1.3.2. Requirements for Installing the iSCSI Software

When installing the iSCSI software, it is required that the system have Linux kernel source or its headers installed in order to compile the iSCSI software as a kernel module. The source tree may be only header files, as for RHEL5 as an example, or a complete tree. The source tree needs to be configured and the header files need to be compiled. Additionally, the Linux kernel must be configured to use modules.

License Key Requirement

A license key file is required for each copy of the iSCSI software installed. To obtain the key file a binary program called "`chinfotool`" must be run on the host system where the iSCSI software is to be installed. That program generates an information file that contains data about the system. The information file must be sent back to Chelsio where a license key file will be generated. The key file will be sent back and must be installed on the system in order to unlock the software. Details on this mechanism can be found in the next section on software installation.

Evaluation Version

An evaluation version of this software is available. Please contact Chelsio at support@chelsio.com for details on how to obtain the license for this version.

2. Software/Driver Loading

There are two main steps to installing the Chelsio iSCSI PDU Offload Target software. They are:

1. **Installing the iSCSI software** – The majority of this section deals with how to install the iSCSI software.
2. **Configuring the iSCSI software** – Information on configuring the software can be found in a section further into this user's guide.

2.1. Getting the Latest iSCSI Software Stack Driver Software

The iSCSI software stack comes bundled in the Chelsio Unified Wire package which can be downloaded from the Chelsio support website (http://service.chelsio.com). The license key must be obtained through interaction with the Chelsio sales channel and/or the support organization. Please contact sales@chelsio.com or support@chelsio.com for more information.

The iSCSI software is available for use with most installations of the Linux kernel version 2.6. The software is dependent on the underlying NIC adapter driver and thus the limitation on what version of the 2.6 Linux kernel it can run on is mostly dependent on the NIC driver's limitations.

The iSCSI module will be installed in the `/lib/modules/<linux_kernel_version>/kernel/drivers/iscsi` directory. The modules database will be updated by the installer. This allows the iSCSI module to be located when using the `modprobe` utility. The actual module `chiscsi_t4.ko` can be found inside the package under /chiscsi-5.xxx/kernel.

The `iscsictl` tool and the `chisns` tool will be installed in `/sbin`. The `chisns` tool starts the iSNS client. The `iscsictl` tool is provided for configuring and managing the iSCSI targets and iSNS client. It also provides control for iSCSI global settings.

1. Loading the Kernel module

- For RHEL distributions, run modprobe as follows:

```
[root@host]# modprobe chiscsi_t4
```

- For SLES11 sp1, run modprobe as follows:

```
[root@host]# modprobe chiscsi_t4 --allow-unsupported
```

Note

i. *While using rpm-tar-ball for installation*
   a. *Uninstallation will result into chiscsi.conf file renamed into chiscsi.conf.rpmsave, but if again uninstallation is done then it will lead to overwriting of the old chiscsi.rpmsave file.*
   b. *Its advised to take a backup of chiscsi.conf file before you do an uninstallation and installation of new/same unified wire package. As re-installing/upgrading unified-wire package may lead to loss of chiscsi.conf file.*

ii. *Installation/uninstallation using source-tar-ball will neither remove the conf file nor rename it. It will always be intact. However it's recommended to always take a backup of your configuration file for both methods of installation.*

A sample iSCSI configuration file will be installed in `/etc/chelsio-iscsi/chiscsi.conf`. This file should be edited using a standard text editor and customized to fit your environment.

2. Set iSCSI service to automatically start at bootup

The `chelsio-target` service scripts are installed to `/etc/init.d` and the parameters for the script are installed at `/etc/sysconfig/chiscsi`. The script is installed as a system service.

To auto-start the iSCSI target service at a certain runlevel, e.g. runlevel 3, `chkconfig` can be used on Red Hat and Novell / SuSE based systems as follows:

```
[root@host]# chkconfig --level 3 chelsio-target on
```

The `chelsio-target` service scripts do basic checks before starting the iSCSI target service, loads the kernel module, and starts all the targets configured by default. It can also be used to stop the targets, and restart/reload configuration.

Note

For the script to execute properly, make sure the following flag is set on all kernel.org kernels.

 # CONFIG_MODULE_FORCE_LOAD=y

2.2. Obtaining the iSCSI Software License

A license file is required for each copy of the Chelsio iSCSI PDU Offload Target software installed.

The license is tied to the selected NIC present in the system. The license file will be generated depending on your requirement for a Chelsio iSCSI Target.

2.2.1. Linux Requirements

To obtain an iSCSI license key file, which could be either a production or an evaluation version, please follow the steps below.

1. Prepare License information file

A license information file is required for each license. Run the following command to obtain the license information file:

```
[root@host]# chinfotool
```

The chinfotool scans and lists all the NICs in the system, and prompts the user to select one NIC, to which the keyfile will be tied. At the end, it prints out a summary of the license file.

Note *Only one NIC per system needs to be selected. As long as the selected NIC is in the system, the Chelsio iSCSI software will work on any of the NICs in the system.*

For the selected NIC, the following information is gathered:

- PCI vendor ID
- MAC address
- Link speed

Privacy notice: The above information is used strictly by Chelsio Communications for the iSCSI license management. Chelsio Communications, Inc. does not share, sell, rent, or lease the information to any third party.

```
[root@host]# chinfotool
Scanning System for network devices....
License key will be tied to any of the following interfaces.
Please select the interface
1. Interface eth1 with INTEL Adapter
Linkspeed is 1000 Mbps/s MAC is: 00:30:48:00:00:10.
2. Interface eth2 with CHELSIO Adapter
Linkspeed is 10000 Mbps/s MAC is: 00:07:43:00:00:10.
ENTER NUMBER:2

******************************************************
SUMMARY:
Interface with MAC Address 00:07:43:00:00:10 (10000 Mbps) is selected
Please send info file "/etc/xyz_chelsio_infofile" to Chelsio Communications
at support@chelsio.com
Chelsio Support will send back the keyfile.
The keyfile unlocks the software as long as the above interface is present
******************************************************
```

The generated file `hostname_chelsio_infofile` needs to be sent back to Chelsio at support@chelsio.com as instructed by the `chinfotool`.

Note *Please be sure that the selected NIC Adapter is present in the system at all times when using Chelsio's iSCSI software. If it is removed the license will be invalid and the process of obtaining a new license file will need to be restarted. That includes using `chinfotool` to rescan the system and obtaining a new keyfile from Chelsio support.*

2. Install the license file received from Chelsio

Once the information file (generated by `chinfotool` in the previous step) is processed by Chelsio, the user will receive a license file tailored to the system. Copy the file `to /etc/chelsio-iscsi/chiscsi.key`.

```
[root@host]# cp -f chiscsi.key /etc/chelsio-iscsi/chiscsi.key
```

The iSCSI software is now unlocked and is ready to be configured and used.

3. Software/Driver Unloading

1. Unloading the Kernel module

Use the following command to unload the module:

```
[root@host]# rmmod chiscsi_t4
```

4. Software/Driver Configuration and Fine-tuning

The Chelsio iSCSI software needs configuration before it can become useful. The following sections describe how this is done.

There are two main components used in configuring the Chelsio iSCSI software: the configuration file and the iSCSI control tool. This section describes in some detail what they are and their relationship they have with one another.

4.1. Command Line Tools

There are two command line tools, one for control of the iSNS client and one for control of the iSCSI target nodes.

4.1.1. iscsictl

The Chelsio iSCSI control tool, `iscsictl`, is a Command Line Interface (CLI) user space program that allows administrators to:

- Start/Stop the iSCSI Target
- Start the iSNS client
- Get/Set the iSCSI driver global settings
- Get/Set/Remove the iSCSI Target configuration settings
- Retrieve active sessions" information of an iSCSI Target
- Manually flush data to the iSCSI Target disks
- Reload the iSCSI configuration file
- Update the iSCSI configuration file
- Save the current iSCSI configuration to a file

4.1.2. chisns

The Chelsio iSNS client, `chisns`, can be started independently of `iscsictl`.

4.2. iSCSI Configuration File

The iSCSI configuration file is the place where information about the Chelsio iSCSI software is stored. The information includes global data that pertains to all targets as well as information on each specific iSCSI target node. Most of the information that can be placed in the configuration file has default values that only get overwritten by the values set in the configuration file.

There are only a few global configuration items that can be changed. An example is “iscsi_offload_mode” which specifies the hardware protocol offload method that should be used (e.g. “TOE”, “ULP”, or “AUTO” modes) for all iSCSI targets.

There are many specific parameters that can be configured, some of which are iSCSI specific and the rest being Chelsio specific. An example of an iSCSI specific item is “HeaderDigest” which is defaulted to “None” but can be overridden to “CRC32C”. An example of a Chelsio specific configurable item is “ACL” (for Access Control List). “ACL” is one of the few items that have no default.

Before starting any iSCSI target, an iSCSI configuration file must be created. An easy way to create this file is to use the provided sample configuration file and modify it. This file can be named anything and placed in any directory but it must be explicitly specified when using iscsictl by using the –f option. To avoid this, put configuration file in the default directory (`/etc/chelsio-iscsi`) and name it the default file name (`chiscsi.conf`).

4.2.1. “On the fly” Configuration Changes

Parameters for the most part can be changed while an iSCSI node is running. However, there are exceptions and restrictions to this rule that are explained in a later section that describes the details of the iSCSI control tool `iscsictl`.

4.3. A Quick Start Guide for Target

This section describes how to get started quickly with a Chelsio iSCSI target. It includes:

- Basic editing of the iSCSI configuration file.
- Basic commands of the iSCSI control tool including how to start and stop a target.

4.3.1. A Sample iSCSI Configuration File

The default Chelsio iSCSI configuration file is located at `/etc/chelsio-iscsi/chiscsi.conf`. If this file doesn’t already exist then one needs to be created.

To configure an iSCSI target, there are three required parameters (in the form of key=value pairs) needed as follows:

- TargetName – A worldwide unique iSCSI target name.
- PortalGroup – The portal group tag associating with a list of target IP address (es) and port number(s) that service the login request. The format of this field is a Chelsio specific iSCSI driver parameter which is described in detail in the configuration file section.
- TargetDevice – A device served up by the associated target. A device can be:
    - A block device (for example, /dev/sda)
    - A virtual block device (for example, /dev/md0)
    - A RAM disk
    - A regular file

A target can serve multiple devices, each device will be assigned a Logical Unit Number (LUN) according to the order it is specified (i.e., the first device specified is assigned LUN 0, the second one LUN 1, …, and so on and so forth). Multiple TargetDevice key=value pairs are needed to indicate multiple devices.

Here is a sample of a minimum iSCSI target configuration located at `/etc/chelsio-iscsi/chiscsi.conf`:

```
target:
        TargetName=iqn.2006-02.com.chelsio.diskarray.san1
        TargetDevice=/dev/sda
        PortalGroup=1@192.0.2.178:3260
```

The `TargetDevice` value must match with the storage device in the system. The `PortalGroup` value must have a matching IP address of the Ethernet adapter card in the system.

For more information about TargetDevice configuration please refer to the later chapter titled "Target Storage Device Configuration".

4.3.2. Basic iSCSI Control

Control of the Chelsio iSCSI software is done through `iscsictl`, the command line interface control tool. The following are the basic commands needed for effective control of the target.

Start Target: To start all of the iSCSI targets specified in the iSCSI configuration file, execute `iscsictl` with the "`-S`" option followed by "`target=ALL`".

```
[root@host]# iscsictl –f /etc/chelsio-iscsi/chiscsi.conf –S target=ALL
```

To start a specific target execute `iscsictl` with "`-S`" followed by the target.

```
[root@host]# iscsictl –f /etc/chelsio-iscsi/chiscsi.conf –S
target=iqn.2006-02.com.chelsio.diskarray.san1
```

Stop Target: To stop the all the iSCSI target(s), execute `iscsictl` with "`-s`" option followed by "`target=ALL`".

```
[root@host]# iscsictl –s target=ALL
```

To stop a specific target execute `iscsictl` with "`-s`" followed by the target name.

```
[root@host]# iscsictl -s target=iqn.2006-02.com.chelsio.diskarray.san1
```

View Configuration: To see the configuration of all the active iSCSI targets, execute `iscsictl` with "`-c`" option.

```
[root@host]# iscsictl -c
```

To see the more detailed configuration settings of a specific target, execute `iscsictl` with "`-c`" option followed by the target name.

```
[root@host]# iscsictl -c target=iqn.2006-02.com.chelsio.diskarray.san1
```

View Global Settings: To see Chelsio global settings, execute `iscsictl` with "`-g`" option.

```
[root@host]# iscsictl -g
```

Change Global Settings: To change Chelsio global settings, execute `iscsictl` with "`-G`" option.

```
[root@host]# iscsictl -G iscsi_offload_mode=AUTO
```

View Help: To print help to stdout, execute `iscsictl` with "`-h`" option.

```
[root@host]# iscsictl -h
```

4.4. The iSCSI Configuration File

The iSCSI configuration file consists of a series of blocks consisting of the following types of iSCSI entity blocks:

1. global
2. target

There can be only one global entity block whereas multiple target entity blocks are allowed. The global entity block is optional but there must be at least one target entity block.

An entity block begins with a block type (global or target). The content of each entity block is a list of parameters specified in a "key=value" format. An entity block ends at the beginning of the next entity block or at the end-of-file.

The parameter list in an entity block contains both:

- iSCSI parameters that override the default values
- Parameters that facilitate passing of control information to the iSCSI module

All lines in the configuration file that begin with "#" character are treated as comments and will be ignored. White space is not significant except in key=value pairs.

For the "key=value" parameters the <value> portion can be a single value or a list of multiple values. When <value> is a list of multiple values, they must be listed on one line with a comma "," to separate their values. Another way to list the values instead of commas is to list their values as key=value pairs repeatedly, each on a new line, until they are all listed.

There are three categories of key=value parameter, the first category belongs to the global entity block whereas the second and third categories belong to target and initiator entity blocks:

1. The Chelsio Global Entity Settings of key=value pairs
2. The iSCSI Entity Settings of key=value pairs
3. The Chelsio Entity Settings of key=value pairs

The following sub-sections describe these three categories and list in tables the details of their key=value parameters.

4.4.1. Chelsio System Wide Global Entity Settings

Description

Chelsio System Wide Global Entity Parameters pass system control information to the iSCSI software which affects all targets in the same way. More detail of the these parameters below can be found in a later section entitled "System Wide Parameters".

Table of Chelsio Global Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
iscsi_offload_mode	"AUTO" "TOE" "ULP"	"AUTO"	No	Defines the offload mode **AUTO**: iSCSI software will make the decision. If the connection goes through Chelsio'sHBA which has the iSCSI acceleration enabled, then ULP. **TOE**: Use Chelsio HBA TCP Offloading Engine (TOE) capabilities. **ULP**: Use Chelsio iSCSI hardware acceleration in terminator ASIC including Direct Data Placement. *NOTES: CLI will allow the user to change the offload mode to TOE even though there is no TOE functionality existing. This is because, with a TOE network adaptor card, its TOE capability could be disabled anytime, in which case the traffic will be treated as normal NIC traffic.*
iscsi_auth_order	"ACL" "CHAP"	"CHAP"	No	Authorization order for login verification on the target. Valid only when a target's ACL_Enable=Yes **ACL**: ACL first then CHAP **CHAP**: CHAP first then ACL *Applies to Target(s) Only*
iscsi_target_vendor_id	a string of maximum of 8 characters	"CHISCSI"	No	The target vendor ID part of the device identification sent by an iSCSI target in response of SCSI Inquiry command.

4.4.2. iSCSI Entity Settings

Description

iSCSI Entity Parameters pass iSCSI protocol control information to the Chelsio iSCSI module. This information is unique for each entity block. The parameters follow the IETF iSCSI standard RFC 3720 in both definition and syntax. The descriptions below are mostly from this RFC.

Table of iSCSI Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
MaxConnections	1 to 65535	1	No	Initiator and target negotiate the maximum number of connections requested/acceptable.
InitialR2T	"Yes" "No"	"Yes"	No	To turn on or off the default use of R2T for unidirectional and the output part of bidirectional commands.
ImmediateData	"Yes" "No"	"Yes"	No	To turn on or off the immediate data.
FirstBurstLength	512 to 16777215 (2^{24} - 1)	65536	No	The maximum negotiated SCSI data in bytes of unsolicited data that an iSCSI initiator may send to a target during the execution of a single SCSI command.
MaxBurstLength	512 to 16777215 (2^{24} - 1)	262144	No	The maximum negotiated SCSI data in bytes, of a Data-In or a solicited Data-Out iSCSI sequence between the initiator and target.
DefaultTime2Wait	0 to 3600	2	No	The minimum time, in seconds, to wait before attempting an explicit / implicit logout or connection reset between initiator and target.
DefaultTime2Retain	0 to 3600	20	No	The maximum time, in seconds, after an initial wait.
MaxOutstandingR2T	1 to 65535	1	No	The maximum number of outstanding R2Ts per task.
DataPDUInOrder	"Yes" "No"	"Yes"	No	To indicate the data PDUs with sequence must be at continuously increasing order or can be in any order. *Chelsio only supports "Yes".*
DataSequenceInOrder	"Yes" "No"	"Yes"	No	To indicate the Data PDU sequences must be transferred in continuously non-decreasing sequence offsets or can be transferred in any order. *Chelsio only supports "Yes".*
ErrorRecoveryLevel	0 to 2	0	No	To negotiate the recovery level supported by the node. *Chelsio only supports 0.*
HeaderDigest	"None" "CRC32C"	"None"	Yes	To enable or disable iSCSI header Cyclic integrity checksums.
DataDigest	"None" "CRC32C"	"None"	Yes	To enable or disable iSCSI data Cyclic integrity checksums.
AuthMethod	"CHAP" and "None"	"None, CHAP"	Yes	To choose an authentication method during login phase.
TargetName	"<target name>"		No	A worldwide unique iSCSI target name. *Target only.*
TargetAlias	"<target alias>"		No	A human-readable name or description of a target. It is not used as an identifier, nor is it for authentication. *Target only.*
MaxRecvDataSegmentLength	512 to 16777215 (2^{24} - 1)	8192	No	To declare the maximum data segment length in bytes it can receive in an iSCSI PDU.
OFMarker	"Yes" "No"	"No"	No	To turn on or off the initiator to target markers on the connection. *Chelsio only supports "No".*

IFMarker	"Yes" "No"	"No"	No	To turn on or off the target to initiator markers on the connection. *Chelsio only supports "No".*
OFMarkInt	1 to 65535	2048	No	To set the interval for the initiator to target markers on a connection.
IFMarkInt	1 to 65535	2048	No	To set the interval for the target to initiator markers on a connection.

4.4.3. Chelsio Entity Settings

Description

Chelsio Entity Parameters pass control information to the Chelsio iSCSI module. The parameters are specific to Chelsio'simplementation of the iSCSI node (target or initiator) and are unique for each entity block. The parameters consist of information that can be put into three categories:

1. Challenge Handshake Authentication Protocol (CHAP).
2. Target specific settings. All of the following parameters can have multiple instances in one target entity block (i.e., they can be declared multiple times for one particular target).

- Portal Group
- Storage Device
- Access Control List (ACL)

Table of Chelsio Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
Chelsio CHAP Parameter (Target)				
Auth_CHAP_Target	"<user id>" :"<secret>"		No	CHAP user id and secret for the target. **<user id>** must be less than 256 characters. Commas "," are not allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The target user id and secret are used by the initiator to authenticate the target while doing mutual chap. *NOTE: The double quotes are required as part of the format.*
Auth_CHAP_Initiator	"<user id>" :"<secret>"		Yes	CHAP user id and secret for the initiator. **<user id>** must be less than 256 characters. Commas "," are not

				allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The initiator user id and secret are used by the target to authenticate the initiator. *NOTE: The double quotes are required as part of the format.*
Auth_CHAP_Challenge Length	16 to 1024	16	No	CHAP challenge length
Auth_CHAP_Policy	"Oneway" or "Mutual"	"Oneway"	No	Oneway or Mutual (two-way) CHAP
Chelsio Target Specific Parameter				
PortalGroup	<portal group tag> @<target IP address> [:<port number>] . . . [,<target IP address> [:<port number>]] [,timeout= <timeout value in milliseconds>] **[,[portalgrouptag1, portalgrouptag2,… portalgrouptagn]**		Yes	The portal group name associates the given target with the given list of IP addresses (and optionally, port numbers) for servicing login requests. It"s required to have at least one per target. **<portal group tag>** is a unique tag identifying the portal group. It must be a positive integer. **<target IP address>** is the IP address associated with the portal group tag. **<port number>** is the port number associated with the portal group tag. It is optional and if not specified the well-known iSCSI port number of 3260 is used. **<timeout>** is optional, it applies to all the portals in the group. The timeout value is in milliseconds and needs to be multiple of 100ms. It is used to detect loss of communications at the iSCSI level. *NOTE: There can be multiple target IP address/port numbers per portal group tag. This enables a target to operate on multiple interfaces for instance.* **<portalgrouptagX>**The portalgroup to which login requests should be redirected to. *NOTE: There can be multiple redirection target portalgroups specified for a particular target portal group and the redirection will happen to these in a round robin manner.*
ShadowMode	"Yes" "No"	"No"	No	To turn ShadowMode on or off for iSCSI Target Redirection

TargetSessionMaxCmd	1 to 2048	64	No	The maximum number of outstanding iSCSI commands per session.
TargetDevice	<path/name> [,FILE|MEM|BLK] [,NULLRW] [,SYNC] [,RO] [,size=xMB] [,ScsiID=xxxxxx] [,WWN=xxxxxxxx]		No	A device served up by the associated target. The device mode can be a: • Block Device (e.g. /dev/sda) • Virtual Block Device (e.g. /dev/md0) • RamDisk • Regular File **<path/name>** is the path to the device - with the exception of when a RAM Disk is specified, where it is a unique name given to the device. If multiple RAM Disks are used for a target then each name must be unique within the target. **NULLRW** specifies that random data is returned for reads, and for writes data is dropped. Useful for testing network performance. **SYNC** specifies that the device will function in the write-through mode (i.e., the data will be flushed to the device before the response is returned to the initiator). *NOTE: SYNC is only applicable with FILE mode.* **RO** specifies the device as a read-only device. **FILE** specifies this device should be accessed via the kernel‟s VFS layer. This mode is the most versatile, and it is the default mode in the cases where there is no mode specified. **BLK** specifies this device should be accessed via the kernel‟s block layer. This mode is suitable for high-speed storage device such as RAID Controllers. **MEM** specifies this device should be created as a RAM Disk. **size**=xMB is used with “MEM”, to specify the RamDisk size. If not specified, the default RamDisk size is 16MB (16 Megabytes). The minimum value of x is 1 (1MB) and the maximum value is limited by system memory. **ScsiID**=xxxxxx is a 16 character unique value set for multipath aware iSCSI initiator host.

				WWN=xxxxxx is a 16 character unique value set for multipath aware iSCSI initiator host. When multipath aware initiator host is accessing the storage Logical Unit Number(LUN) via multiple iSCSI session, ScsiID and WWN values must be set for the TargetDevice. These values will be returned in Inquiry response (VPD 0x83). Multiple TargetDevice key=value pairs are needed to indicate multiple devices. There can be multiple devices for any particular target. Each device will be assigned a Logical Unit Number (LUN) according to the order it is specified (i.e., the first device specified is assigned LUN 0, the second one LUN 1, …, and so on and so forth). *NOTE: **FILE** mode is the most versatile mode, if in doubt use **FILE** mode.*
ACL_Enable	"Yes" "No"	"No"	No	Defines if Chelsio'sAccess Control List (ACL) method will be enforced on the target: **Yes:** ACL is enforced on the target **No:** ACL is not enforced on the target
ACL	[iname=<name1>][;<sip=<sip1>][;dip=<dip1>][;lun=<lun_list:permissions>]		Yes	The ACL specifies which initiators and how they are allowed to access the LUNs on the target. **iname=<Initiator Name>** specifies one or more initiator names, the name must be a fully qualified iSCSI initiator name. **sip=<Source IP address>** specifies one or more IP addresses the initiators are connecting from. **Dip=<Destination IP address>** specifies one or more IP addresses that the iSCSI target is listening on (i.e., the target portal IP addresses). *NOTE: when configuring an ACL at least one of the above three must be provided:* • *iname, and/or* • *sip, and/or* • *dip.* **lun=<lun list>:<permission>** controls how the initiators access the luns. The supported value for **<lun list>** is

				ALL. **<permissions>** can be: **R**: Read Only **RW** or **WR**: Read and Write If permissions are specified then the associated LUN list is required. If no **lun=<lun list>:[R|RW]** is specified then it defaults to **ALL:RW**. *NOTE: For the Chelsio Target Software release with lun-masking included,* ***<lun list>** is in the format of **<0..N | 0~N | ALL>*** *Where:* ***0..N**: only one value from **0** through **N*** ***0~N**: a range of values between **0** through **N*** ***ALL**: all currently supported LUNs.* *Multiple lists of LUN numbers are allowed. When specifying the list separate the LUN ranges by a comma.*
RegisteriSNS	"Yes" "No"	"Yes"	No	To turn on or off exporting of target information via iSNS

4.4.4. Sample iSCSI Configuration File

Following is a sample configuration file. While using iSCSI node (target), irrelevant entity block can be removed or commented.

```
#
# Chelsio iSCSI Global Settings
#
global:
       iscsi_offload_mode=AUTO
       iscsi_auth_order=ACL
#
# an iSCSI Target "iqn.2006-02.com.chelsio.diskarray.san1"
# being served by the portal group "5". Setup as a RAM Disk.
#
target:
       TargetName=iqn.2006-02.com.chelsio.diskarray.san1
       # lun 0: a ramdisk with default size of 16MB
       TargetDevice=ramdisk,MEM
       PortalGroup=5@192.0.2.178:3260
#
```

```
# an iSCSI Target “iqn.2005-8.com.chelsio:diskarrays.san.328”
# being served by the portal group "1" and "2"
#
target:
       #
       # iSCSI configuration
       #
       TargetName=iqn.2005-8.com.chelsio:diskarrays.san.328
       TargetAlias=iTarget1
       MaxOutstandingR2T=1
       MaxRecvDataSegmentLength=8192
       HeaderDigest=None,CRC32C
       DataDigest=None,CRC32C
       ImmediateData=Yes
       InitialR2T=No
       FirstBurstLength=65535
       MaxBurstLength=262144

       #
       # Local block devices being served up
       # lun 0 is pointed to /dev/sda
       # lun 1 is pointed to /dev/sdb

       TargetDevice=/dev/sda,ScsiID=aabbccddeeffgghh,WWN=aaabbbcccdddeeef
       TargetDevice=/dev/sdb

       #
       # Portal groups served this target
       #

       PortalGroup=1@102.50.50.25:3260
       PortalGroup=2@102.60.60.25:3260

       #
       # CHAP configuration
       #
       Auth_CHAP_Policy=Mutual
       Auth_CHAP_target=“iTarget1ID”:“iTarget1Secret”
```



```
        Auth_CHAP_Initiator="iInitiator1":"InitSecret1"
        Auth_CHAP_Initiator="iInitiator2":"InitSecret2"
        Auth_CHAP_ChallengeLength=16

        #
        # ACL configuration
        #
        # initiator "iqn.2006-02.com.chelsio.san1" is allowed full access
        # to this target
        ACL=iname=iqn.2006-02.com.chelsio.san1

        # any initiator from IP address 102.50.50.101 is allowed full access
        # of this target
        ACL=sip=102.50.50.101

        # any initiator connected via the target portal 102.60.60.25 is
        # allowed full access to this target
        ACL=dip=102.60.60.25

        # initiator "iqn.2005-09.com.chelsio.san2" from 102.50.50.22 and
        # connected via the target portal 102.50.50.25 is allowed read only
        # access of this target
        ACL=iname=iqn.2006-
02.com.chelsio.san2;sip=102.50.50.22;dip=102.50.50.25;lun=ALL:R
```

4.5. Challenge-Handshake Authenticate Protocol (CHAP)

The Chelsio iSCSI software supports Challenge-Handshake Authentication Protocol (CHAP). CHAP is a protocol that is used to authenticate the peer of a connection and uses the notion of a challenge and response, (i.e., the peer is challenged to prove its identity).

The Chelsio iSCSI software supports two CHAP methods: oneway and two way. CHAP is not supported for discovery sessions.

4.5.1. Oneway CHAP authentication

With Oneway CHAP (also called unidirectional CHAP) the target uses CHAP to authenticate the initiator. The initiator does not authenticate the target. This method is the default method.

For Oneway CHAP, the initiator CHAP id and secret are configured and stored on a per-initiator with Chelsio Entity parameter "Auth_CHAP_Initiator".

4.5.2. Mutual CHAP authentication

With mutual CHAP (also called bidirectional CHAP), the target uses CHAP to authenticate the initiator. The initiator uses CHAP to authenticate the target.

For mutual CHAP, in addition to the initiator CHAP id and secret, the target CHAP id and secret are required. They are configured and stored on a per target basis with Chelsio Entity parameter "Auth_CHAP_Target".

4.5.3. Adding CHAP User ID and Secret

A single `Auth_CHAP_Target` key and multiple `Auth_CHAP_Initiator` keys could be configured per target:

```
target:
      TargetName=iqn.2006-02.com.chelsio.diskarray.san1
      TargetDevice=/dev/sda PortalGroup=1@192.0.2.178:8000
      Auth_CHAP_Policy=Oneway
      Auth_CHAP_Initiator="remoteuser1":"remoteuser1_secret"
      Auth_CHAP_Initiator="remoteuser2":"remoteuser2_secret"
      Auth_CHAP_Target="targetid1":"target1_secret"
```

In the above example, target "`iqn.2005-com.chelsio.diskarray.san1`" has been configured to authenticate two initiators, and its own id and secret are configured for use in the case of mutual CHAP.

4.5.4. AuthMethod and Auth_CHAP_Policy Keys

By setting the iSCSI key AuthMethod and the key Auth_CHAP_Policy, a user can choose whether to enforce CHAP and if Mutual CHAP needs to be performed.

The AuthMethod key controls if an initiator needs to be authenticated or not. The default setting of AuthMethod is "None,CHAP".

The Auth_CHAP_Policy key controls which CHAP authentication (Oneway or mutual) needs to be performed if CHAP is used. The default setting of Auth_CHAP_Policy is "Oneway".

On an iSCSI node, with AuthMethod=None,CHAP:

- Auth_CHAP_Policy=Oneway, Chelsio iSCSI target will accept a relevant initiator if it does
    a) no CHAP
    b) CHAP Oneway or
    c) CHAP Mutual
- Auth_CHAP_Policy=Mutual, the Chelsio iSCSI target will accept a relevant initiator if it does
    a) no CHAP or
    b) CHAP Mutual

With AuthMethod=None, regardless the setting of the key Auth_CHAP_Policy, the Chelsio iSCSI target will only accept a relevant initiator if it does no CHAP.

With AuthMethod=CHAP, CHAP is enforced on the target:

i. Auth_CHAP_Policy=Oneway, the iSCSI target will accept a relevant initiator only if it does
    a) CHAP Oneway or
    b) CHAP Mutual
ii. Auth_CHAP_Policy=Mutual, the iSCSI node will accept a relevant initiator only if it does
    a) CHAP Mutual

4.6. Target Access Control List (ACL) Configuration

The Chelsio iSCSI target supports iSCSI initiator authorization via an Access Control List (ACL).

ACL configuration is supported on a per-target basis. The creation of an ACL for a target establishes:

- Which iSCSI initiators are allowed to access it
- The type of the access: read-writeor read-only
- Possible SCSI layer associations of LUNs with the initiator

More than one initiator can be allowed to access a target and each initiator‟s access rights can be independently configured.

The format for ACL rule is as follows:

```
ACL=[iname=<initiator name>][;<sip=<source ip addresses>]
    [;dip=<destination ip addresses>][;lun=<lun_list>:<permissions>]

target:
      TargetName=iqn.2006-02.com.chelsio.diskarray.san1
      TargetDevice=/dev/sda

      PortalGroup=1@102.50.50.25:3260
      PortalGroup=2@102.60.60.25:3260

      # initiator “iqn.2006-02.com.chelsio.san1” is allowed
      # full read-write access to this target
      ACL=iname=iqn.2006-02.com.chelsio.san1

      # any initiator from IP address 102.50.50.101 is allowed full
      # read-write access of this target
```

```
ACL=sip=102.50.50.101

# any initiator connected via the target portal 102.60.60.25
# is allowed full read-write access to this target
ACL=dip=102.60.60.25

# initiator "iqn.2005-09.com.chelsio.san2" from 102.50.50.22
# and connected via the target portal 102.50.50.25 is allowed
# read only access of this target
ACL=iname=iqn.2006-
02.com.chelsio.san2;sip=102.50.50.22;dip=102.50.50.25;lun=ALL:R
```

4.6.1. ACL Enforcement

To toggle ACL enforcement on a per-target base, a Chelsio keyword "`ACL_Enable`" is provided:

- Setting "`ACL_Enable=Yes`" enables the target to perform initiator authorization checking for all the initiators during login phase. And in addition, once the initiator has been authorized to access the target, the access rights will be checked for each individual LU the initiator trying to access.
- Setting "`ACL_Enable=No`" disable the target to perform initiator authorization checking.

When a target device is marked as read-only (RO), it takes precedence over ACL"s write permission (i.e., all of ACL write permission of an initiator is ignored).

4.7. Target Storage Device Configuration

An iSCSI Target can support one or more storage devices. The storage device can either be the built-in RAM disk or actual backend storage.

Configuration of the storage is done through the Chelsio configuration file via the key-value pair `TargetDevice`.

When option `NULLRW` is specified, on writes the data is dropped without being copied to the storage device, and on reads the data is not actually read from the storage device but instead random data is used. This option is usefuly for measuring network performance.

The details of the parameters for the key TargetDevice are found in the table of Chelsio Entity Settings section earlier in this document.

4.7.1. RAM Disk Details

For the built-in RAM disk:

- The minimum size of the RAM disk is 1 Megabyte (MB) and the maximum is limited by system memory.
- To use a RAM disk with a Windows Initiator, it is recommended to set the size >= 16MB.

To configure an ramdisk specify `MEM` as the device mode:

```
TargetDevice=<name>,MEM,size=xMB
```

Where: `<name>` Is a unique name given to the RAM Disk. This name identifies this particular ramdisk. If multiple RAM Disks are configured for the same target, the name must be unique for each RAM Disk.

`x` Is the size of the RAM Disk in MB. It‟s an integer between 1 - max, where max is limited by system memory. If this value is not specified the default value is 16 MB.

```
target:
#<snip>
  # 16 Megabytes RAM Disk named ramdisk1
  TargetDevice=ramdisk1,MEM,size=16MB
#<snip>
```

4.7.2. FILE Mode Storage Device Details

The FILE mode storage device is the most common and versatile mode to access the actual storage attached to the target system:

- The FILE mode can accommodate both block devices and virtual block devices.
- The device is accessed in the exclusive mode. The device should not be accessed (or active) in any way on the target system.
- Each device should be used for one and only one iSCSI target.
- “SYNC” can be used with FILE mode to make sure the data is flushed to the storage device before the Target responds back to the Initiator.

To configure a FILE storage device specify `FILE` as the device mode:

```
TargetDevice=<path to the storage device>[,FILE][,SYNC]
```

Where: `<path>` Is the path to the actual storage device, such as /dev/sdb for a block device or /dev/md0 for a software RAID. The path must exist in the system.

`SYNC` When specified, the Target will flush all the data in the system cache to the storage driver before sending response back to the Initiator.

4.7.3. Example Configuration of FILE Mode Storage

Below is an example:

```
target:
#<snip>
 # software raid /dev/md0 is accessed in FILE mode
TargetDevice=/dev/md0,FILE
#<snip>
```

4.7.4. BLK Mode Storage Device Details

The BLK mode storage device is suitable for high-speed storage attached to the target system:

- The BLK mode can accommodate only block devices.
- The device is accessed in the exclusive mode. The device should not be accessed (or active) in any way on the target system.
- Each device should be used for one and only one iSCSI target.

To configure a block storage device specify BLK as the device mode:

```
TargetDevice=<path to the storage device>,BLK
```

Where: `<path>` Is the path to the actual storage device, such as /dev/sdb. The path must exist in the system.

```
target:
#<snip>
 # /dev/sdb is accessed in BLK mode
 TargetDevice=/dev/sdb,BLK
#<snip>
```

4.8. Target Redirection Support

An iSCSI Target can redirect an initiator to use a different IP address and port (often called a portal) instead of the current one to connect to the target. The redirected target portal can either be on the same machine, or a different one.

4.8.1. ShadowMode for Local vs. Remote Redirection

The ShadowMode setting specifies whether the Redirected portal groups should be present on the same machine or not. If ShadowMode is enabled, the redirected portal groups are on a different system. If it is disabled then the redirected portal groups must be present on the same system otherwise the target would fail to start.

Below is an example with ShadowMode enabled:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261.

     PortalGroup=1@10.193.184.81:3260,[2]
     PortalGroup=2@10.193.184.85:3261

     # the PortalGroup "2" is NOT presented on the same system.
     ShadowMode=Yes
 #<snip>
```

Below is an example with ShadowMode disabled:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261

    PortalGroup=1@10.193.184.81:3260,[2]
    PortalGroup=2@10.193.184.85:3261
    # the PortalGroup "2" IS present on the same system
    ShadowMode=No
#<snip>
```

4.8.2. Redirecting to Multiple Portal Groups

The Chelsio iSCSI Target Redirection allows redirecting all login requests received on a particular portal group to multiple portal groups in a round robin manner.

Below is an example Redirection to Multiple Portal Groups:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261 and 10.193.184.85:3262 in a
     # Round Robin Manner.

     PortalGroup=1@10.193.184.81:3260,[2,3]
     PortalGroup=2@10.193.184.85:3261
     PortalGroup=3@10.193.184.85:3262
     ShadowMode=No
#<snip>
```

4.9. The command line interface tools “iscsictl” & “chisns”

4.9.1. iscsictl

`iscsictl` is the tool Chelsio provides for controlling the iSCSI target. It is a Command Line Interface (CLI) that is invoked from the console. Its usage is as follows:

```
iscsictl <options> <mandatory parameters> [optional parameters]
```

The mandatory and optional parameters are the **key=value** pair(s) defined in RFC3720, or the **var=const** pair(s) defined for Chelsio iSCSI driver implementation. In this document, the key=value is referred to as “pair”, and var=const is referred to as “parameter” to clarify between iSCSI protocol‟s pair value(s), and Chelsio iSCSI driver‟s parameter value(s). Note that all **value** and **const** are case sensitive.

4.9.2. chisns

`chisns` is the command line tool for controlling the iSNS client. This is a simple tool that starts the iSNS client with a client and server parameter.

4.9.3. iscsictl options

Options	Mandatory Parameters	Optional Parameters	Description
-h			Display the help messages.
-v			Display the version.
-f	<[path/] filename>		Specifies a pre-written iSCSI configuration text file, used to start, update, save, or reload the iSCSI node(s). This option must be specified with one of the following other options: "**-S**", "**-U**", or "**-W**". For the "**-S**" option "**-f**" must be specified first. All other options will ignore this "**-f**" option. If the "**-f**" option is not specified with the commands above the default configuration file will be used. It"s name and location is: /etc/chelsio-iscsi/chiscsi.conf The configuration file path and filename must conform to Linux standards. For the format of the iSCSI configuration file, please see "Format of The iSCSI Configuration File" section earlier in this document.
-k	<key>[=<val>]		Specifies an iSCSI Entity or Chelsio Entity parameter. This option can be specified after "**-c**" option to retrieve a parameter setting..
-c	target=<name> [,name2 . . . ,<nameN>]		Display the Chelsio iSCSI target configuration. **target=<name>** parameter: Where **name** is the name of the node whose information will be returned. **name** can be one or more string of names, separated by a comma, **<name1[,name2,…,nameN] | ALL>** A **name** of **ALL** returns information on all targets. **ALL** is a reserved string that must be uppercase. Example: **iscsictl -c target=iqn.com.cc.it1** **Iscsictl -c target=iqn.com.cc.target1 -k TargetAlias** The **<name>** parameter can also be specified as one or more parameter on the same command line, separated by a comma, **target=<name1>, <name2>, … ,<nameN>** The **target=<name>** parameter(s) are optional and if not specified all active Chelsio iSCSI targets(s) configuration(s) will be displayed. If **target=ALL** is specified or no parameters are specified the output will be abbreviated. Specify specific targets to get detailed configuration data.

			If the **target=<name>** option is specified, the -k <key> option can optionally be specified along with this option to display only the selected entity parameter setting. Example: **iscsictl -c target=iqn.com.cc.target1 -k HeaderDigest**
-F		target=<name> -k lun=<value>	Flush the cached data to the target disk(s). **target=<name>** parameter: Where **name** is the name of the target to be flushed. **name** can be one or more string of names, separated by a comma, **<name1[,name2,…,nameN] | ALL>** A **name** of **ALL** will cause all the target data to be flushed. **ALL** is a reserved string that must be uppercase. The **target=name** parameter is optional. If no **target=name** parameter is specified, it is the same as specifying **target=ALL**. The -k lun=<value> option is optional. It can be used to further specify a particular lun to be flushed. Example: To flush all the targets in the system: **iscsictl -F** To flush a particular target: **iscsictl -F target=iqn.com.cc.it1** To flush only the lun 0 of a particular target: **iscsictl -F target=iqn.com.cc.it1 -k lun=0**
-g			Display the Chelsio iSCSI global settings. The global parameters printed are: **iscsi_offload_mode** **iscsi_auth_order** **iscsi_target_vendor_id *** *The iscsi_target_vendor_id will be printed only if it has been changed from the default value.
-G	<var=const>		Set the Chelsio iSCSI global settings. **var=const** parameter: Where **var=const** can be: **iscsi_offload_mode=<AUTO | TOE | ULP>** **iscsi_auth_order=<ACL | CHAP>** **iscsi_target_vendor_id=<string of max. 8 characters>** Example: **iscsictl -G iscsi_auth_order=ACL** The **var=const** parameter(s) are mandatory. If the **var=const** parameter is not specified, the command will be denied.

			If any of the specified **var=const** parameter is invalid, the command will reject only the invalid parameters, but will continue on and complete all other valid parameters if any others are specified.
`-s`	`target=<name>`		Stop the specified active iSCSI targets. **target=<name>** parameter: *See the description of option -c for the target=<name> parameter definition.* The **target=<name>** parameter is mandatory. If no **target=<name>** parameter is specified, the command will be denied. If the **target=<name>** parameter is specified, only the specified targets from the **target=<name>** parameters will be stopped. If **target=ALL** is specified, all active targets will stop.
`-S`	`target=<name>`		Start or reload the iSCSI targets. **target=<name>** parameter: Where **name** is the name of the target(s) that will be started or reloaded. The **target=<name>** parameter can be specified as one or more parameter on the same command line, separated by a space, **target=<name1> target=<name2> … target=<nameN>** The **target=<name>** parameter can also be, **target=ALL** A **name** of **ALL** starts or reloads all targets specified in the configuration file. **ALL** is a reserved string that must be uppercase. The **target=<name>** parameter is optional. If this command line option is specified without the -f option, the default configuration file `/etc/chelsio-iscsi/chiscsi.conf` will be used. Rules, 1. If the **target=<name>** parameter is specified, only the targets from the list will be started or reloaded. 2. If **target=ALL** is specified, all targets specified from the iSCSI configuration file will be started or reloaded. 3. If the **target=<name>** parameter is not specified, all active targets configurations will be reloaded from the configuration file while those targets are running. All non-active targets specified will not be loaded / started. For Rules 1-3, if the specified targets are currently active (running), they will get reloaded. For Rules 1 & 2, if the specified targets are not

			currently active, they will be started. For Rules 2 & 3, please note the differences – they are not the same! The global settings are also reloaded from the configuration file with this option.
-r	target=<name>	-k initiator=<name>	Retrieve active iSCSI sessions under a target. **target=<name>** parameter: Where **name** must be a single target name. If **target=<name>** parameter is specified as target=<name>, the sessions can be further filtered based on the remote node name with optional –k initiator=<name> option. Examples: **iscsictl -r target=iqn.com.cc.it1** **iscsictl -r target=iqn.com.cc.it1 -k initiator=iqn.com.cc.ii1** The first **target=<name>** parameter is mandatory. If it is not specified, the command will be denied.
-U			Update the specified iSCSI configuration file to include the current iSCSI global settings and the active iSCSI targets" configuration. *Will not delete the non-active targets' configuration from the specified file. The order of the targets' configuration might get re-organized afterward.* The -f option MUST be specified along with this option.
-W			Overwrite the specified iSCSI configuration file with ONLY the current iSCSI global settings and the active iSCSI targets" configuration to the specified iSCSI configuration file. *Will delete any non-active targets' configuration from the specified file.* The -f option MUST be specified along with this option.
-h			Display the help messages.
	server=<IP address> [:<port>]	id=<isns entity id> query=<query interval>	Start the Chelsio iSNS client. **server=<IP address>[:<port>]** where **server** is the iSNS server address. The port is optional and if it"s not specified it defaults to 3205. The server with the ip address is mandatory and if it"s not specified the, the command will be denied. **id=<isns entity id>** where **id** is the iSNS entity ID used to register with the server. It defaults to <hostname>. **query=<query interval>** where **query** is the initiator query interval (in seconds). It defaults to 60 seconds. Examples: chisns server=192.0.2.10 chisns server=192.0.2.10:3205 id=isnscln2 query=30

			In the first example the minimum command set is given where the IP address of the iSNS server is specified. In the second example a fully qualified command is specified by also setting three optional parameters. Here, the mandatory IP address and the corresponding optional port number are specified. Also set is the iSNS entity ID to "`isnscln2`" as well as the query interval to 30 seconds.

4.10. Rules of Target Reload (i.e. "on the fly" changes)

After a target has been started its settings can be modified via reloading of the configuration file (i.e., `iscsictl -S`).

The following parameters cannot be changed once the target is up and running otherwise the target reload would fail:

- TargetName
- TargetSessionMaxCmd

The following parameters **CAN** be changed by reloading of the configuration file. The new value will become effective **IMMEDIATELY** for all connections and sessions:

- TargetDevice
- PortalGroup
- ACL_Enable
- ACL

The following parameter **CAN** be changed by reloading of the configuration file. The new value will **NOT** affect any connections and sessions that already completed login phase:

- TargetAlias
- MaxConnections
- InitialR2T
- ImmediateData
- FirstBurstLength
- MaxBurstLength
- MaxOutstandingR2T
- HeaderDigest
- DataDigest
- MaxRecvDataSegmentLength
- AuthMethod
- Auth_CHAP_Initiator
- Auth_CHAP_Target

- Auth_CHAP_ChallengeLength
- Auth_CHAP_Policy

The following parameters **SHOULD NOT** be changed because only one valid value is supported:

- DataPDUInOrder (support only "Yes")
- DataSequenceInOrder (support only "Yes")
- ErrorRecoveryLevel (support only "0")
- OFMarker (support only "No")
- IFMarker (support only "No")

The following parameters can be changed but would not have any effect because they are either not supported or they are irrelevant:

- DefaultTime2Wait (not supported)
- DefaultTime2Retain (not supported)
- OFMarkInt (irrelevant because OFMarker=No)
- IFMarkInt (irrelevant because IFMarker=No)

4.11. System Wide Parameters

The Chelsio iSCSI software provides three system wide parameters that can be controlled through the configuration file, through the use of the command line "`iscsictl -G`". The finer points of these parameters are described in detail here:

4.11.1. iscsi_offload_mode

`iscsictl` is the tool Chelsio provides for controlling the iSCSI target. It is a Command Line Interface (CLI) that is invoked from the console. Its usage is as follows:

Options: "AUTO", "TOE", or "ULP", defaults to "AUTO"

- **TOE**: When selected the iSCSI software attempts to offload the TCP connection. If the connection can be offloaded, it operates that connection with TCP/IP offloaded in the Chelsio TOE hardware.
- **ULP**: This is also known as iSCSI Hardware Acceleration. Like TOE mode, when selected the software attempts to offload the TCP connection. If the connection can be offloaded, it operates that connection with TCP/IP offloaded in the Chelsio TOE hardware. Additionally (and differing from TOE mode), it also enables hardware calculations of the header and data digest and performs Direct Data Placement (DDP) into host memory.
- **AUTO**: When selected the iSCSI software chooses either TOE mode or ULP mode automatically. It will select ULP mode when the connection is going through a Chelsio iSCSI acceleration enabled HBA The purpose of this mode is to optimize the performance of the Chelsio acceleration hardware.

Note *iscsi_offload_mode has no meaning when the iSCSI software is used on a non-TOE based NIC.keyfile from Chelsio support.*

4.11.2. iscsi_auth_order

Options: "ACL" or "CHAP", defaults to "CHAP"

On an iSCSI target when `ACL_Enable` is set to "`Yes`", iscsi_auth_order decides whether to perform CHAP first then ACL or perform ACL then CHAP.

- **ACL:** When setting `iscsi_auth_order=ACL`, initiator authorization will be performed at the start of the login phase for a iSCSI normal session: upon receiving the first iscsi_login_request, the target will check it"s ACL, if this iscsi connection does not match any ACL provisioned, the login attempt will be terminated.

- **CHAP**: When setting `iscsi_auth_order=CHAP`, initiator authorization will be performed at the end of the login phase for an iSCSI normal session: before going to the full feature phase, the target will check it"s ACL, if this iscsi connection does not match any ACL provisioned, the login attempt will be terminated.

Note *iscsi_auth_order has no meaning when* `ACL_Enable` *is set to "*`No`*" on a target.*

4.11.3. iscsi_target_vendor_id

Options: A string of maximum of 8 characters, defaults to "`CHISCSI`".

The iscsi_target_vendor_id is part of the device identification sent by an iSCSI target in response of a SCSI Inquiry request.

VII. iSCSI PDU Offload Initiator

1. Introduction

The Chelsio T4 series Adapters support iSCSI acceleration and iSCSI Direct Data Placement (DDP) where the hardware handles the expensive byte touching operations, such as CRC computation and verification, and direct DMA to the final host memory destination:

- **iSCSI PDU digest generation and verification**
  On transmitting, Chelsio h/w computes and inserts the Header and Data digest into the PDUs. On receiving, Chelsio h/w computes and verifies the Header and Data digest of the PDUs.

- **Direct Data Placement (DDP)**
  Chelsio h/w can directly place the iSCSI Data-In or Data-Out PDU's payload into pre-posted final destination host-memory buffers based on the Initiator Task Tag (ITT) in Data-In or Target Task Tag (TTT) in Data-Out PDUs.

- **PDU Transmit and Recovery**
  On transmitting, Chelsio h/w accepts the complete PDU (header + data) from the host driver, computes and inserts the digests, decomposes the PDU into multiple TCP segments if necessary, and transmit all the TCP segments onto the wire. It handles TCP retransmission if needed.
  On receiving, Chelsio h/w recovers the iSCSI PDU by reassembling TCP segments, separating the header and data, calculating and verifying the digests, then forwarding the header to the host. The payload data, if possible, will be directly placed into the pre-posted host DDP buffer. Otherwise, the payload data will be sent to the host too.

The cxgb4i driver interfaces with open-iSCSI initiator and provides the iSCSI acceleration through Chelsio hardware wherever applicable.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with iSCSI PDU Offload Initiator Software:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-CX

- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio iSCSI PDU Offload Initiator software runs on Linux-based platforms and therefore it is a base requirement for running the software.

Currently the software is available for the following versions:

- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)*
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)*
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Ubuntu 12.04 , 3.2.0-23
- Kernel.org linux-2.6.34*
- Kernel.org linux-2.6.35*
- Kernel.org linux-2.6.36*
- Kernel.org linux-2.6.37*
- Kernel.org linux-2.6.39*
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

*Limited QA performed

2. Software/Driver Loading

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

Run the following command to load the driver:

```
[root@host]#modprobe cxgb4i
```

On SLES, load the driver with `--allow` option:

```
[root@host]#modprobe cxgb4i --allow
```

If installing the Chelsio Unified Wire package doesn't install the cxgb4i driver properly on SLES, then follow these steps to configure your source tree and then re-install the package:

```
[root@host]# cd /usr/src/linux
[root@host]# make clean
[root@host]# make oldconfig
[root@host]# make scripts
```

Note

- *Sometimes open-iSCSI fails to compile because of dependencies on libopenssl-devel (for SLES11SP1) or openssl-devel(for RHEL 6) packages, so install these packages first and then install the Chelsio Unified Wire package (such issue is not seen while installing package using RPM).*

- *If loading of cxgb4i displays "unkown symbols found" error in dmesg, then kill iscsid, unload all open iSCSI modules and related/dependent modules (like multipath, bnx2i etc) and then reload cxgb4i driver.*

If loading the driver fails, then follow the steps mentioned below and try again.

i. Run the following command to view all the loaded iSCSI modules

```
[root@host]# lsmod | grep iscsi
```

ii. Now, unload them using the following command:

```
[root@host]# rmmod <modulename>
```

3. Software/Driver Unloading

To unload the driver, execute the following commands:

```
[root@host]#rmmod cxgb4i
[root@host]#rmmod libcxgbi
```

4. Software/Driver Configuration and Fine-tuning

4.1. Accelerating open-iSCSI Initiator

The following steps need to be taken to accelerate the open-iSCSI initiator:

4.1.1. Configuring iscsid.conf file

Edit the `iscsi/iscsid.conf` file and change the setting for `MaxRecvDataSegmentLength`:

```
node.conn[0].iscsi.MaxRecvDataSegmentLength = 8192
```

The login would fail for a normal session if `MaxRecvDataSegmentLength` is too big. A error message in the format of `ERR! MaxRecvSegmentLength <X> too big. Need to be <= <Y>.` would be logged to dmesg.

Important

- *Always take a backup of `iscsid.conf` file before installing Chelsio Unified Wire Package. Although the file is saved to `iscsid.rpmsave` after uninstalling the package using RPM, you are still advised to take a backup.*
- *In case of RHEL 6, uninstalling the package using RPM will result into uninstallation of open-iSCSI admin utility. In order to get back the utility, please install open-iscsi-utils provided with OS CD/DVD.*

4.1.2. Configuring interface (*iface)* file

Create an interface file located under *iface* directory for the new transport class *cxgb4i* in the following format:

iface.iscsi_ifacename = <iface file name>
iface.hwaddress = <MAC address>
iface.transport_name = cxgb4i
iface.net_ifacename = <ethX>
iface.ipaddress = <iscsi ip address>

E.g.:-

```
iface.iscsi_ifacename = cxgb4i.00:07:43:04:5b:da
iface.hwaddress = 00:07:43:04:5b:da
iface.transport_name = cxgb4i
iface.net_ifacename = eth3
iface.ipaddress = 102.2.2.137
```

Alternatively, you can create the file automatically by executing the following command:

```
[root@host]#iscsiadm -m iface
```

Here,

- `iface.iscsi_ifacename` denotes the name of interface file in `/etc/iscsi/ifaces/`.
- `iface.hwaddress` denotes the MAC address of the Chelsio interface via which iSCSI traffic will be running.
- `iface.transport_name` denotes the transport name, which is cxgb4i.
- `iface.net_ifacename` denotes the Chelsio interface via which iSCSI traffic will be running.
- `iface.ipaddress` denotes the IP address which is assigned to the interface.

Note

i. The interface file needs to be created in `/etc/iscsi/iscsid.conf`.
ii. If `iface.ipaddress` *is specified, it needs to be either the same as the ethX's IP address or an address on the same subnet. Make sure the IP address is unique in the network.*

4.1.3. Discovery and Login

i. Starting iSCSI Daemon

Start Daemon from */sbin* by using the following command:

```
[root@host]#iscsid -f -d 3
```

Note *if* `iscsid` *is already running, then kill the service and start it as shown above after installing the Chelsio Unified Wire package.*

ii. Discovering iSCSI Targets

To discovery an iSCSI target execute a command in the following format:

iscsiadm -m discovery -t st -p <target ip address>:<target port no> -I <cxgb4i iface file name>

E.g.:-

```
[root@host]# iscsiadm -m discovery -t st -p 102.2.2.155:3260 -I
cxgb4i.00:07:43:04:5b:da
```

iii. Logging into an iSCSI Target

Log into an iSCSI target using the following format:

iscsiadm -m node -T <iqn name of target> -p <target ip address>:<target port no> -I <cxgb4i iface file name> -l

E.g.:-

```
[root@host]#iscsiadm -m node -T iqn.2004-05.com.chelsio.target1 -p
102.2.2.155:3260,1 -I cxgb4i.00:07:43:04:5b:da -l
```

iv. Logging out from an iSCSI Target

Log out from an iSCSI Target by executing a command in the following format:

iscsiadm -m node -T <iqn name of target> -p <target ip address>:<target port no> -I <cxgb4i iface file name> -u

E.g.:-

```
[root@host]#iscsiadm -m node -T iqn.2004-05.com.chelsio.target1 -p
102.2.2.155:3260,1 -I cxgb4i.00:07:43:04:5b:da -u
```

Note *Other options can be found by typing* `iscsiadm --help`

4.2. Auto login from cxgb4i initiator at OS bootup

For iSCSI auto login (via cxgb4i) to work on OS startup, please add the following line to `start()` in `/etc/rc.d/init.d/iscsid` file on RHEL:

```
modprobe -q cxgb4i
```

E.g.:-

```
force_start() {
    echo -n $"Starting $prog: "
    modprobe -q iscsi_tcpmodprobe -q ib_iser
    modprobe -q cxgb4i
    modprobe -q cxgb3i
    modprobe -q bnx2i
    modprobe -q be2iscsi
    daemon brcm_iscsiuio
    daemon $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}
```

VIII. FCoE Full Offload Initiator

1. Introduction

Fibre Channel over Ethernet (FCoE) is a mapping of Fibre Channel over selected full duplex IEEE 802.3 networks. The goal is to provide I/O consolidation over Ethernet, reducing network complexity in the Datacenter.

Chelsio FCoE initiator maps Fibre Channel directly over Ethernet while being independent of the Ethernet forwarding scheme. The FCoE protocol specification replaces the FC0 and FC1 layers of the Fibre Channel stack with Ethernet. By retaining the native Fibre Channel constructs, FCoE will integrate with existing Fibre Channel networks and management software.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with FCoE full offload Initiator driver:

- T420-CR
- T440-CR
- T422-CR
- T440-LP-CR
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio FCoE full offload Initiator driver runs on Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)
- Kernel.org linux-2.6.34*
- Kernel .org linux-3.5

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

* Limited QA performed.

2. Software/Driver Loading

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

To load the driver, execute the following:

```
[root@host]# modprobe csiostor
```

Note *To load the driver on SLES11sp1, execute the following command:*
[root@host]# modprobe csiostor --allow

3. Software/Driver Unloading

To unload the driver:

```
[root@host]# modprobe -r  csiostor
```

Note *If multipath services are running, unload of FCoE driver is not possible. Stop the multipath service and then unload the driver.*

4. Software/Driver Configuration and Fine-tuning

For information on **Configuring Cisco Nexus 5010 switch**, please refer to the previous chapter **FCoE Full Offload Target**.

4.1. FCoE fabric discovery verification

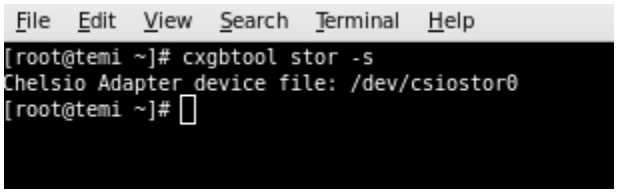
The verification is done using “cxgbtool”.

4.1.1. Verifying the DCBX parameters

To verify the current DCBX information being exchanged, execute the below commands using cxgbtool.

i. First step is to find the adapter number. Find it using the following command

```
[root@host]# cxgbtool stor –s
```



ii. Now execute the following command to check the DCBX information.

```
[root@host]# cxgbtool   stor –a <adapter_no> --dcb-params
```

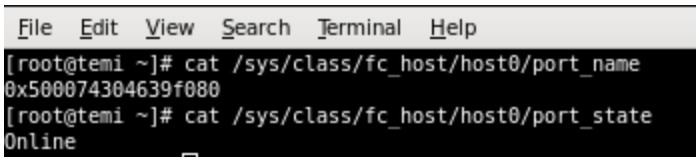
```
File  Edit  View  Search  Terminal  Help
[root@temi ~]# cxgbtool stor -a /dev/csiostor0 --dcb-params
******************** DCBX Paramters[Port:  0] ********************
Priority Group ID of Priority 0         : 0
Priority Group ID of Priority 1         : 0
Priority Group ID of Priority 2         : 0
Priority Group ID of Priority 3         : 1
Priority Group ID of Priority 4         : 0
Priority Group ID of Priority 5         : 0
Priority Group ID of Priority 6         : 0
Priority Group ID of Priority 7         : 0

Bandwidth Percentage :
---------------------
Bandwidth Percentage of Priority Group 0: 50
Bandwidth Percentage of Priority Group 1: 50
Bandwidth Percentage of Priority Group 2: 0
Bandwidth Percentage of Priority Group 3: 0
Bandwidth Percentage of Priority Group 4: 0
Bandwidth Percentage of Priority Group 5: 0
Bandwidth Percentage of Priority Group 6: 0
Bandwidth Percentage of Priority Group 7: 0


Number of Traffic Classes Supported     : 2

Strict Priorate :
----------------
Strict Priorate for Priority 0          : 0
Strict Priorate for Priority 1          : 0
Strict Priorate for Priority 2          : 0
Strict Priorate for Priority 3          : 0
Strict Priorate for Priority 4          : 0
Strict Priorate for Priority 5          : 0
Strict Priorate for Priority 6          : 0
Strict Priorate for Priority 7          : 0

PFC Enabled/Disabled :
----------------------
PFC for Priotity 0                      : Disabled
PFC for Priotity 1                      : Disabled
PFC for Priotity 2                      : Disabled
PFC for Priotity 3                      : Enabled
PFC for Priotity 4                      : Disabled
PFC for Priotity 5                      : Disabled
PFC for Priotity 6                      : Disabled
PFC for Priotity 7                      : Disabled
```

4.1.2. Verifying Local Ports

Once connected to the switch, use the following command to see if the FIP has gone through and a VN_Port MAC address has been assigned.

Verify if all the FCoE ports are online/ready and a successful FIP has taken place using the following command. The **wwpn** and **state** of the initiator local port can be found under sysfs.

```
[root@host]# cat /sys/class/fc_host/hostX/port_name
```



Note

- *The hosts under fc_host depends on the number of ports on the adapter used.*
- *Inorder to identify chelsio fc_host from other vendor fc_host, the WWPN always begins with* ***0x5000743***

Alternatively, the local port information can also be found using:

```
[root@host]# cxgbtool  stor –a <adapter_no> --show-lnode
```

```
File  Edit  View  Search  Terminal  Help
[root@temi ~]# cxgbtool stor -a /dev/csiostor0 --show-lnode
****************[Index:  0]***************
LNode Device ID: 2950
VNPI    : 0xb86
FCFI    : 0xba3
MAC     : 0E-FC-01-67-00-0D

Port Id : 0
Nport id: 67000d
State   : READY

WWPN    : 50:00:74:30:46:39:f0:80
WWNN    : 50:00:74:30:46:39:f0:00
NPIV    : SUPPORTED
Total VPorts :  0

No.of RNodes : 10

Common Service Params:
        Rcv size: 2068
        ED-TOV  : 2000

Class Service Params:
Class 1: NOT SUPPORTED
Class 2: NOT SUPPORTED
Class 3: SUPPORTED
        Initiator ctl           : 0
        Recipient ctl           : 0
        Rcv size                : 2068
        Total concurrent seq    : 0
        EE Credit               : 0
        Open Sequence per Exchange: 0

Class 4: NOT SUPPORTED
****************[Index:  1]***************
LNode Device ID: 68484
VNPI    : 0xb84
FCFI    : 0xba1
MAC     : 0E-FC-01-67-00-02

Port Id : 1
Nport id: 670002
State   : READY

WWPN    : 50:00:74:30:46:3a:71:80
WWNN    : 50:00:74:30:46:3a:71:00
NPIV    : SUPPORTED
Total VPorts :  0

No.of RNodes : 10
```

4.1.3. Verifying the target discovery

To check the targets being discovered use *cxgbtool*. To check the list of targets that are being discovered from a particular FCoE port, use the following commands from *cxgbtool*.

i. Check for the adapter number using the following command.

```
[root@host]# cxgbtool stor -s
```


ii. To check the list of targets discovered from a particular FCoE port, first find out the wwpn of the initiator local port under sysfs. The hosts under fc_host depends on the number of ports on the adapter used.

```
[root@host]# cat /sys/class/fc_host/hostX/port_name
```

iii. After finding the localport, go to the corresponding Remote port under sysfs # **cat /sys/class/fc_remote_ports/rport-X:B:R** where X is the Host ID, B is the bus ID and R is the remote port.

```
File  Edit  View  Search  Terminal  Help
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-0/roles
Fabric Port
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-1/roles
Directory Server
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-2/roles
Management Server
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-3/roles
FCP Initiator
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-4/roles
FCP Initiator
[root@temi ~]# cat /sys/class/fc_remote_ports/rport-0\:0-9/roles
FCP Target
[root@temi ~]#
```

Note *R can correspond to NameServer, Management Server and other initiator ports logged in to the switch and targets.*

Alternatively the locaports can also be found using cxgbtool

```
[root@host]# cxgbtool stor -a  <adapter no>   --show-lnode
```

After finding out the wwpn of the local node, to verify the list of targets being discovered, use the following command.

```
[root@host]# cxgbtool stor -a <adapter_no> --show-rnode --wwn=<wwpn of
lnode>
```

```
File  Edit  View  Search  Terminal  Help
[root@temi ~]# cxgbtool stor -a /dev/csiostor0 --show-rnode --wwn=50:00:74:30:46:3b:73:80

****************[Index:   0]***************
SSNI            : 0x780
VNPI            : 0xb84
FCFI            : 0xba2
WWPN            : 20:06:00:05:73:d5:7a:ff
WWNN            : 20:05:00:05:73:d5:7a:c1
Nport id        : FFFFFE
State           : READY
FCP Flags       : 0
Role            : Fabric

Class Service Params:
Class 1: NOT SUPPORTED
Class 2: NOT SUPPORTED
Class 3: SUPPORTED
Class 4: NOT SUPPORTED

****************[Index:   1]***************
SSNI            : 0x781
VNPI            : 0xb84
FCFI            : 0xba2
WWPN            : 25:0d:00:05:73:d5:7a:c0
WWNN            : 20:05:00:05:73:d5:7a:c1
Nport id        : FFFFFC
State           : READY
FCP Flags       : 0
Role            : Name-Server

Class Service Params:
Class 1: NOT SUPPORTED
Class 2: NOT SUPPORTED
Class 3: SUPPORTED
Class 4: NOT SUPPORTED

****************[Index:   2]***************
SSNI            : 0x783
VNPI            : 0xb84
FCFI            : 0xba2
WWPN            : 25:0b:00:05:73:d5:7a:c0
WWNN            : 20:05:00:05:73:d5:7a:c1
Nport id        : FFFFFA
State           : READY
FCP Flags       : 0
Role            : N-Port

Class Service Params:
Class 1: NOT SUPPORTED
Class 2: NOT SUPPORTED
Class 3: SUPPORTED
```

4.2. Formatting the LUNs and Mounting the Filesystem

Use *lsscsi –g* to list the LUNs discovered by the initiator

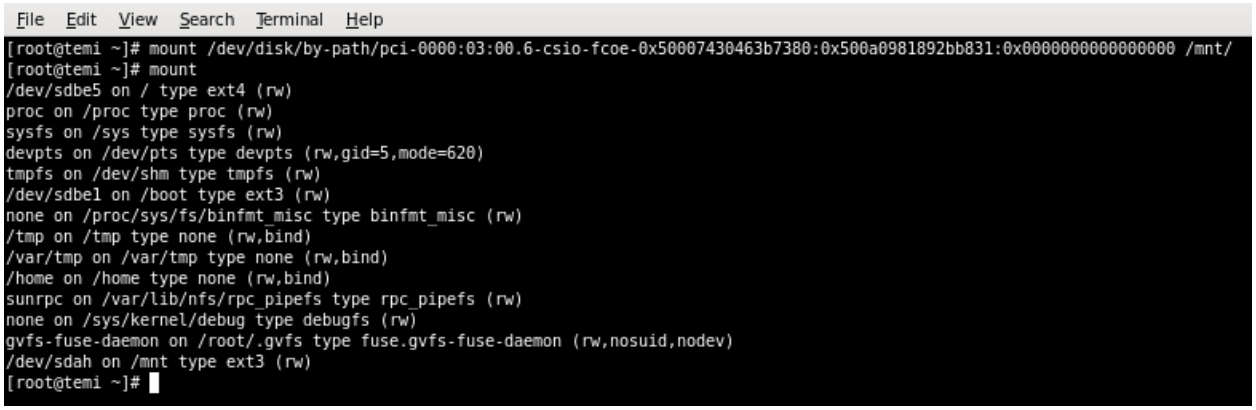
```
[root@host]# lsscsi –g
```

File Edit View Search Terminal Help

```
[root@temi ~]# lsscsi -g
[0:0:0:0]    disk    NETAPP   LUN             8010  /dev/sda   /dev/sg0
[0:0:0:1]    disk    NETAPP   LUN             8010  /dev/sdb   /dev/sg1
[0:0:0:2]    disk    NETAPP   LUN             8010  /dev/sdc   /dev/sg2
[0:0:0:3]    disk    NETAPP   LUN             8010  /dev/sdd   /dev/sg3
[0:0:0:4]    disk    NETAPP   LUN             8010  /dev/sde   /dev/sg4
[0:0:0:5]    disk    NETAPP   LUN             8010  /dev/sdf   /dev/sg5
[0:0:0:6]    disk    NETAPP   LUN             8010  /dev/sdg   /dev/sg6
[0:0:0:7]    disk    NETAPP   LUN             8010  /dev/sdh   /dev/sg7
[0:0:0:8]    disk    NETAPP   LUN             8010  /dev/sdi   /dev/sg8
[0:0:0:9]    disk    NETAPP   LUN             8010  /dev/sdj   /dev/sg9
[0:0:0:10]   disk    NETAPP   LUN             8010  /dev/sdk   /dev/sg10
[0:0:0:11]   disk    NETAPP   LUN             8010  /dev/sdl   /dev/sg11
[0:0:0:12]   disk    NETAPP   LUN             8010  /dev/sdm   /dev/sg12
[0:0:0:13]   disk    NETAPP   LUN             8010  /dev/sdn   /dev/sg13
[0:0:0:14]   disk    NETAPP   LUN             8010  /dev/sdo   /dev/sg14
[0:0:0:15]   disk    NETAPP   LUN             8010  /dev/sdp   /dev/sg15
[0:0:0:16]   disk    NETAPP   LUN             8010  /dev/sdq   /dev/sg16
[0:0:0:17]   disk    NETAPP   LUN             8010  /dev/sdr   /dev/sg17
[0:0:0:18]   disk    NETAPP   LUN             8010  /dev/sds   /dev/sg18
[0:0:0:19]   disk    NETAPP   LUN             8010  /dev/sdt   /dev/sg19
[1:0:0:0]    disk    NETAPP   LUN             8010  /dev/sdu   /dev/sg20
[1:0:0:1]    disk    NETAPP   LUN             8010  /dev/sdv   /dev/sg21
[1:0:0:2]    disk    NETAPP   LUN             8010  /dev/sdw   /dev/sg22
[1:0:0:3]    disk    NETAPP   LUN             8010  /dev/sdx   /dev/sg23
[1:0:0:4]    disk    NETAPP   LUN             8010  /dev/sdy   /dev/sg24
[1:0:0:5]    disk    NETAPP   LUN             8010  /dev/sdz   /dev/sg25
[1:0:0:6]    disk    NETAPP   LUN             8010  /dev/sdaa  /dev/sg26
[1:0:0:7]    disk    NETAPP   LUN             8010  /dev/sdab  /dev/sg27
[1:0:0:9]    disk    NETAPP   LUN             8010  /dev/sdac  /dev/sg28
[1:0:0:10]   disk    NETAPP   LUN             8010  /dev/sdad  /dev/sg29
[1:0:0:11]   disk    NETAPP   LUN             8010  /dev/sdae  /dev/sg30
[1:0:0:12]   disk    NETAPP   LUN             8010  /dev/sdaf  /dev/sg31
[1:0:0:15]   disk    NETAPP   LUN             8010  /dev/sdag  /dev/sg32
[3:0:0:0]    disk    NETAPP   LUN             8010  /dev/sdah  /dev/sg33
[3:0:0:1]    disk    NETAPP   LUN             8010  /dev/sdai  /dev/sg34
[3:0:0:2]    disk    NETAPP   LUN             8010  /dev/sdaj  /dev/sg35
[3:0:0:3]    disk    NETAPP   LUN             8010  /dev/sdak  /dev/sg36
[3:0:0:4]    disk    NETAPP   LUN             8010  /dev/sdal  /dev/sg37
[3:0:0:5]    disk    NETAPP   LUN             8010  /dev/sdam  /dev/sg38
[3:0:0:6]    disk    NETAPP   LUN             8010  /dev/sdan  /dev/sg39
[3:0:0:7]    disk    NETAPP   LUN             8010  /dev/sdao  /dev/sg40
[3:0:0:8]    disk    NETAPP   LUN             8010  /dev/sdap  /dev/sg41
[3:0:0:9]    disk    NETAPP   LUN             8010  /dev/sdaq  /dev/sg42
[3:0:0:10]   disk    NETAPP   LUN             8010  /dev/sdar  /dev/sg43
[3:0:0:11]   disk    NETAPP   LUN             8010  /dev/sdas  /dev/sg44
[3:0:0:12]   disk    NETAPP   LUN             8010  /dev/sdat  /dev/sg45
[3:0:0:13]   disk    NETAPP   LUN             8010  /dev/sdau  /dev/sg46
[3:0:0:14]   disk    NETAPP   LUN             8010  /dev/sdav  /dev/sg47
[3:0:0:15]   disk    NETAPP   LUN             8010  /dev/sdaw  /dev/sg48
[3:0:0:16]   disk    NETAPP   LUN             8010  /dev/sdax  /dev/sg49
```

Alternatively, the LUNs discovered by the Chelsio FCoE initiators can be accessed via easily-identifiable ‘udev’ path device files like:

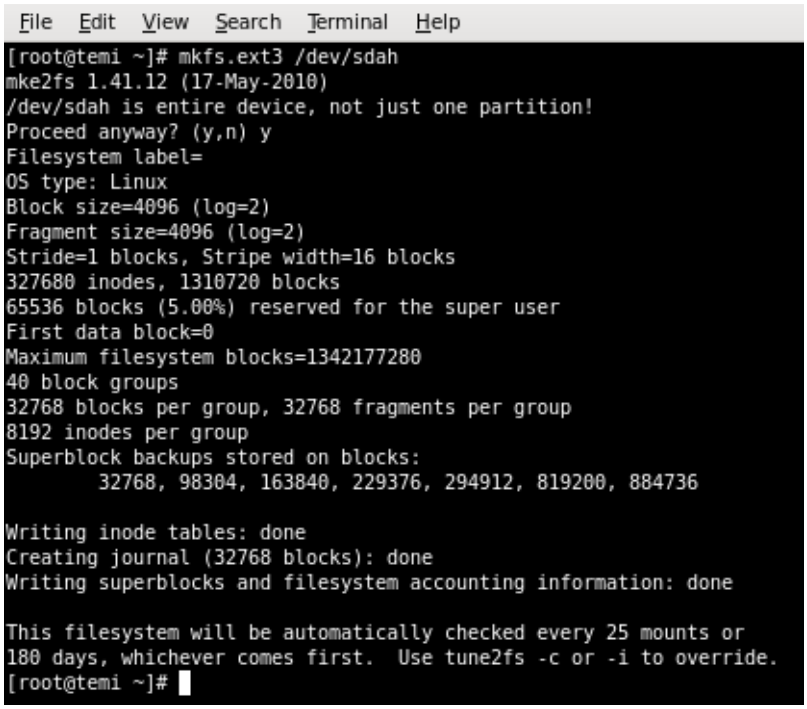
```
[root@host]# ls   /dev/disk/by-path/pci-0000:04:00.0-csio-fcoe
<local_wwpn>:<remote_wwpn>:<lun_wwn>
```



4.3. Creating Filesystem

Create an ext3 filesystem using the following command:

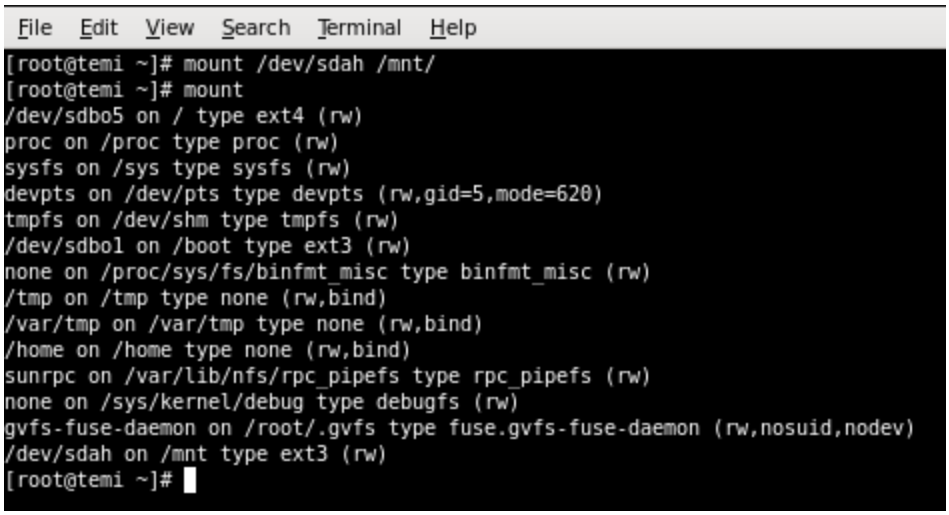
```
[root@host]# mkfs.ext3 /dev/sdx
```



4.4. Mounting the formatted LUN

The formatted LUN can be mounted on the specified mountpoint using the following command:

```
[root@host]# mount /dev/sdx /mnt
```



IX. Offload Bonding driver

1. Introduction

The Chelsio Offload bonding driver provides a method to aggregate multiple network interfaces into a single logical bonded interface effectively combining the bandwidth into a single connection. It also provides redundancy in case one of link fails.

The traffic running over the bonded interface can be fully offloaded to the T4 Adapter, thus freeing the CPU from TCP/IP overhead.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the Chelsio Offload Bonding driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-SO-CR
- T420-CX
- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio Offload Bonding driver runs on Linux-based platforms and therefore it is a base requirement for running the driver. Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)

- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)
- Ubuntu 12.04, 3.2.0-23
- Kernel .org linux-3.1
- Kernel .org linux-3.5

Other kernel versions have not been tested and are not guaranteed to work.
*Limited QA performed

2. Software/Driver Loading

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

To load the driver (with offload support), run the following command:

```
[root@host]  # modprobe bonding
```

3. Software/Driver Unloading

To unload the driver, run the following command:

```
[root@host] # rmmod bonding
```

4. Software/Driver Configuration and Fine-tuning

4.1. Creating a Bond Interface

The Chelsio Offload Bonding driver supports all bond modes for NICs. In case of Chelsio adapters using TOE, only the active-backup (mode=1) and 802.3ad (mode=4) modes are supported. To create a bond, use the following command:

```
[root@host]# modprobe bonding mode=1 miimon=100
```

This would create a bond interface (bondX) in active-backup mode. The interfaces can be added to bond using:

```
[root@host] # ifconfig bond0 up
[root@host] # ifenslave bond0 ethX ethY
```

Load the t4_tom driver to offload the traffic over the bond interface.

```
 [root@host] # modprobe t4_tom
```

For more information on the parameters supported by the bonding driver,

```
[root@host] # modinfo bonding
```

4.2. Network Device Configuration

Please refer to the operating system documentation for administration and configuration of network devices.

Note *Some operating systems may attempt to auto-configure the detected hardware and some may not detect all ports on a multi-port adapter. If this happens, please refer to the operating system documentation for manually configuring the network device.*

X. Offload IPv6 driver

1. Introduction

The growth of the Internet has created a need for more addresses than are possible with IPv4. Internet Protocol version 6 (IPv6) is a version of the Internet Protocol (IP) designed to succeed the Internet Protocol version 4 (IPv4).

Chelsio's Offload IPv6 driver provides support to fully offload IPv6 traffic to the T4 adapter.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio Offload IPv6 driver:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH
- T422-CR
- T420-SO-CR
- T420-CX
- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

The Chelsio Offload IPv6 driver runs on 64-bit Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7

- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27
- Ubuntu 12.04, 3.2.0-23
- Kernel.org linux-2.6.34
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

2. Software/Driver Loading

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

After installing the driver package and rebooting the host, the offload IPv6 driver loads automatically. However, it is possible to manually load the driver by typing the command below:

```
[root@host]# modprobe ipv6
```

3. Software/Driver Unloading

To unload the driver, run the following command:

```
[root@host]# rmmod ipv6
```

4. Software/Driver Configuration and Fine-tuning

4.1. Offloading IPv6 traffic

Once the IPv6 Offload driver is loaded successfully, load the t4_tom driver:

```
[root@host]# modprobe t4_tom
```

All the IPv6 TCP traffic will be offloaded now.

4.2. Network Device Configuration

Please refer to the operating system documentation for administration and configuration of network devices.

Note

Some operating systems may attempt to auto-configure the detected hardware and some may not detect all ports on a multi-port adapter. If this happens, please refer to the operating system documentation for manually configuring the network device.

XI. Bypass Driver

1. Introduction

Chelsio's B420 and B404 Bypass Adapters are Ethernet cards that provide bypass functionality and an integrated L2, L3, and L4 Ethernet switch. The integrated switch allows for selective bypass on a per-packet basis at line rate.

To use the bypass adapters, you must have both the Chelsio NIC driver and the bypass CLI user space application loaded.

1.1. Features

B404 is a four-port 1G and B420 is a two-port 10G, short-range or long range, low profile PCI-Express Host Bus Adapters (HBAs) using T4 ASIC.

- **Bypass**
  - Software programmable behavior on power fails – either Bypass Mode or Drop Mode.
  - Firmware control of Bypass / Normal / Drop Modes when T4 timer expires.
  - Bypass control via software
  - Drop Mode control by putting the associated PHY in reset.

- **Selective Bypass** – Programmable HW traffic classification and redirection without host intervention in normal mode

- **Product function**:

The Bypass Adapters can operate in 4 modes.

- Bypass Mode
- Normal Mode
- Disconnect Mode
- Selective Bypass Mode

- **Bypass Mode**

In this mode the Bypass adapter switches the packet from one port to another port. That is, in B420, port 0 to port 1 and vice versa; and in B404, port 0 to port 1 and port 2 to port 3 and vice versa.

- **Normal Mode**

The Bypass Adapters can be programmed to function as a NIC. In this mode all the packets are redirected to the host.

- **Disconnect Mode**

The Bypass cards can also be programmed to drop all the packets.

- **Selective Bypass**

In Normal mode, the Bypass Adapters can be programmed to perform redirection of packets depending on the certain portion of the packet. The specification of the match criteria is called a **rule**. When a rule is matched an **action** is applied to the ingress packet. The actions that are supported are *drop*, *forward* and *input*.

- The *drop* action causes the packet to be discarded.
- The *forward* action causes the packet to bypass the host from any port to any port.
- Finally, the *input* action directs the packet to the host where it can be processed by the application.

1.2. Hardware Requirements

1.2.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio Bypass driver:

- B420-SR
- B404-BT

1.3. Software Requirements

1.3.1. Linux Requirements

The Chelsio Bypass driver runs on Linux-based platforms and therefore it is a base requirement for running the driver.

Currently the driver is available for the following versions:

- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)*
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit) *
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)*
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7(64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)*
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)*
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Ubuntu 12.04, 3.2.0-23
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35*
- Kernel.org linux-2.6.36
- Kernel.org linux-2.6.37
- Kernel.org linux-2.6.39*
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

* Limited QA performed.

2. Software/Driver Loading

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

Run the following command to load the Bypass driver:

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

3. Software/Driver Unloading

Run the following command to unload the Bypass driver:

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

4. Software/Driver Configuration and Fine-tuning

4.1. Starting ba server

4.1.1. For IPv4 only

Execute the following command to start the ba server only for IPv4:

```
[root@host]# ba_server –i ethX
```

4.1.2. For IPv4 and IPv6

Execute the following command to start the ba server for IPv4 and IPv6:

```
[root@host]# ba_server -6 –i ethX
```

4.2. Bypass API (CLI)

A CLI will be created that implements the Bypass API as specified below. This CLI will then communicate the requests to the SDK server. The API will contain the following elements:

- Bypass Management (watchdog, state)
- Redirect Management

Bypass management provides a means of setting the watchdog timeout as well as enabling and disabling it. It allows setting a default state and a current state.

The redirect management element will interface with the SDK server to take the appropriate actions to manage tables and rules.

4.2.1. Bypass Management

The bypass CLI will have the following syntax:

```
[root@host]# bypass ethX get|set --key [value]
```

E.g.

```
[root@host]# bypass eth1 set --current_state normal
```

Here is a list of Bypass commands:

Bypass command list

Command	Key	Value	Return
bypass ethX get	default_state		bypass|disconnect
bypass ethX set	default_state	bypass|disconnect	
bypass ethX get	current_state		bypass|disconnect|normal
bypass ethX set	current_state	bypass|disconnect|normal	
bypass ethX get	watchdog		enabled|disabled
bypass ethX set	watchdog	enable|disable|lock|ping	
bypass ethX get	watchdog_timeout		timeout in milliseconds
bypass ethX set	watchdog_timeout	timeout in milliseconds (maximum is 60000)	

4.2.2. Operating the Switch (Examples)

The switch has three modes of operation; *bypass*, *drop*, and *normal*. These three modes can be selected using the bypass CLI. The mode can be selected for the current operating state or for the default state that is applied when powering up or after failure detection.

4.2.3. Bypass State

To obtain the current state of the switch:

```
[root@host]# bypass ethX get --current_state
```

To set the state of the switch:

```
[root@host]# bypass ethX set --current_state [bypass|disconnect|normal]
```

Getting the default state:

```
[root@host]# bypass ethX get --default_state
```

Setting the default state:

```
[root@host]# bypass ethX set --default_state [bypass |disconnect]
```

4.2.4. Using the bypass watchdog timer

The watchdog timer is used to ensure that if there is a software failure, the switch will enter the default state.

First set the timeout value for the watchdog in milliseconds:

```
[root@host]# bypass ethX set --watchdog_timeout 3000
```

Start pinging the watchdog:

```
[root@host]# while true; do bypass ethX set --watchdog ping; sleep 1; done
```

Enable the watchdog:

```
[root@host]# bypass ethX set --watchdog enable
```

4.2.5. Redirect Management

The bypass adapter has the ability to redirect packets based upon rules. Rules can be grouped into **tables**. When a rule is added into a table, it is assigned an index. Within a table, rules are evaluated by index number from low to high. When a rule matches the action associated with it, that rule is performed and higher number rules are not evaluated. Rules are not evaluated until a table is activated.

The redirect CLI has the following syntax:

```
[root@host]# redirect ethX command --key [value] …
```


Redirect Command List

Command	Key	Value	Return
redirect ethX list			list of all configured tables and rules
redirect ethX add update match	table	table id (defaults to table 1)	Add a rule to a table. Update the specified rule with new keys. Match specified keys to a rule in a table.
	index	rule index. If not specified the rule is appended. Otherwise it is inserted at the specified index.	
	proto	icmp|icmp6|tcp|udp|any (defaults to any)	
	srcaddr	Source IP address	
	dstaddr	Destination IP address	
	vlan	VLAN id	
	action	input|forward|drop (default input)	
	port	If action is “input”, then this is the port to which the packet is directed.	
	srcmask	Source IP address mask (Default is 255.255.255.255)	
	dstmask	Destination IP address mask (Default is 255.255.255.255)	
	srcport	Source port Number	
	dstport	Destination Port number	
	srcportmask	Source port Number Mask*	
	dstportmask	Destination port Number Mask*	
	ipv6	Use this option if IPv6 address are used while adding Redirect rule	
	srcaddr6	IPv6 Source IP address	
	dstaddr6	IPv6 Destination IP address	
	srcmask6	IPv6 Source IP address mask (Default: **FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF**)	
	dstmask6	IPv6 Destination IP address mask (Default:**FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF**)	
redirect ethX delete	table	table id	Delete the table
	index	rule index	Delete a rule from a table.
redirect ethX purge	table	table id	Remove all rules from the specified table
redirect ethX move	table	table id	Check that the new rule id is valid and move the source rule to new id if it doesn’t contain any existing rule

	old_id	Existing rule index.	
	new_id	New rule index.	
redirect ethX count	table	table id	Display a count of the number of packets and bytes has matched a rule.
	index	rule index	
redirect ethX create_table	table	table id	Create a new table that is used to hold a set of rules.
redirect ethX activate_table	table	table id	Activate a table so that all the rules in it will be active.
redirect ethx deactivate_table	table	table id	Deactivate a table and its associated rules.
redirect ethX delete_table	table	table id	Delete a table and the rules in it.
redirect ethX dump			dump all tables and rules as commands

* The matching algorithm for `srcportmask` and `dstportmask` is

(Ingress_Packet-Field & Filter-Mask) == (Filter-Value & Filter-Mask)

4.2.6. Managing persistent tables and rules

This example creates a new table with table id 1 and then adds a rule to drop all ICMP packets for port 0. The rules in a table are not active until the table is activated.

Create a table:

```
[root@host]# redirect ethX create_table --table 1
```

Add a rule to the newly created table:

```
[root@host]# redirect ethX add --table 1 --proto icmp --action drop --port 0
```

Activate the table:

```
[root@host]# redirect ethX activate_table --table 1
```

The `redirect dump` command can be used to save the currently configured tables and rules into a shell script.

To make the current configured rules & tables persistent, redirect the output to `/etc/ba.cfg` file only:


```
[root@host]# redirect ethX dump > /etc/ba.cfg
```

where the `/etc/ba.cfg` is read by the *bad* service at boot time.

To apply the saved configuration after machine reboots, start the *bad* service. This service is available only in the IPv4 mode. For IPv6, the *ba_server* needs to be started manually.

```
[root@host]# service bad start
```

To stop the service:

```
[root@host]# service bad stop
```

To restart the service:

```
[root@host]# service bad restart
```

XII. WD Sniffing and Tracing

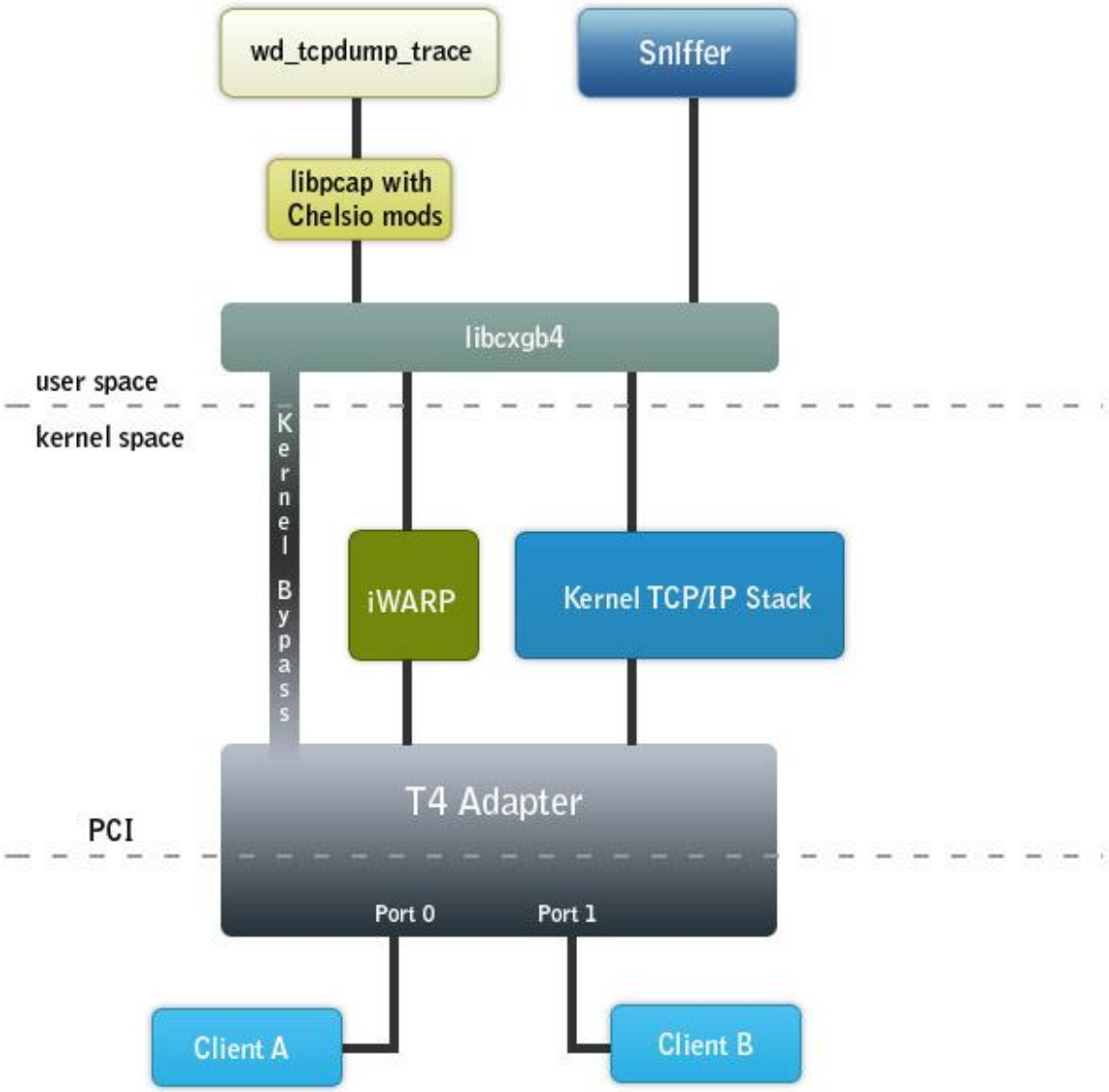
1. Theory of Operation

The objective of these utilities (*wd_sniffer* and *wd_tcpdump_trace*) is to provide sniffing and tracing capabilities by making use of T4's hardware features.

- Sniffer- Involves targeting specific multicast traffic and sending it directly to user space.
    a) Get a Queue (raw QP) idx.
    b) Program a filter to redirect specific traffic to the raw QP queue.

- Tracer - All tapped traffic is forwarded to user space and also pushed back on the wire via the internal loop back mechanism
    a) Get a Queue (raw QP) idx
    b) Set the T4 adapter in loop back
    c) Connect Client A and B to ports 0 and 1 or ports 2 and 3.
    d) Enable tracing.

In either mode the targeted traffic bypasses the kernel TCP/IP stack and is delivered directly to user space by means of an RX queue.



Schematic diagram of T4 sniffer and tracer

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the tools:

- T420-CR
- T420-LL-CR
- T440-CR
- T440-LP-CR
- T420-BCH

- T422-CR
- T420-CX
- T420-BT
- T404-BT

1.2. Software Requirements

1.2.1. Linux Requirements

WD Sniffing and Tracing Utilities run on Linux-based platforms and therefore it is a base requirement for running the tools.

Currently the utility is available for the following version:

- Redhat Enterprise Linux 5 update 3 kernel (RHEL5.3), 2.6.18-128.el5 (64-bit)
- Redhat Enterprise Linux 5 update 4 kernel (RHEL5.4), 2.6.18-164.el5 (64-bit)
- Redhat Enterprise Linux 5 update 5 kernel (RHEL5.5), 2.6.18-194.el5 (64-bit)
- Redhat Enterprise Linux 5 update 6 kernel (RHEL5.6), 2.6.18-238.el5 (64-bit)
- Redhat Enterprise Linux 5 update 7 kernel (RHEL5.7), 2.6.18-274.el5 (64-bit)
- Redhat Enterprise Linux 5 update 8 kernel (RHEL5.8), 2.6.18-308.el5 (64-bit)
- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6 (64-bit)
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6 (64-bit)
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6 (64-bit)
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6 (64-bit)
- Suse Linux Enterprise Server 11 kernel (SLES11) , 2.6.27.19-5 (64-bit)
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1),2.6.32.12-0.7 (64-bit)
- Suse Linux Enterprise Server 11 SP2 kernel (SLES11SP2), 3.0.13-0.27 (64-bit)*
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13 (64-bit)*
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14 (64-bit)*
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35
- Kernel.org linux-2.6.36
- Kernel.org linux-2.6.37
- Kernel.org linux-2.6.39
- Kernel .org linux-3.1
- Kernel .org linux-3.5

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

*Limited QA performed

2. Installation and Usage

2.1. Installing basic support

iw_cxgb4 (Chelsio iWARP driver) and *cxgb4* (Chelsio NIC driver) drivers have to be compiled and loaded before running the utilities. Refer to the **Software/Driver Loading** section for each driver and follow the instructions mentioned before proceeding.

2.2. Using Sniffer (wd_*sniffer)*

1. **Setup**:

Wire filter sniffing requires 2 systems with one machine having a T4 card.

The machines should be setup in the following manner:

Machine A <---------> Machine B

192.168.1.100 192.168.1.200

2. **Procedure**:

On the Device Under Test (DUT), start sniffer.

```
[root@host]# wd_sniffer -T 20 -s 1000 -I <MAC address of interface to sniff>
```

Start traffic on the PEER and watch the sniffer.

The sniffer will receive all packets as fast as possible, update the packet count, and then discard the data. Performance is a full 10Gbps for packet size 1000.

2.3. Using Tracer (*wd_tcpdump_trace)*

1. **Setup**:

Wire tapping requires 3 systems with one machine having a T4 card with two or more ports. The machines should be setup in the following manner:

DUT: Machine B

PEER: Machine A <-----> (port 0) (port 1) <-----> PEER: Machine C

192.168.1.100 IP-dont-care IP-dont-care 192.168.1.200

2. Procedure:

Run `wd_tcpdump_trace -i iface` on the command prompt where *iface* is one of the interfaces whose traffic you want to trace. In the above diagram its port 0 or port 1.

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

Try ping or ssh between machines A and B. The traffic should successfully make it from end to end and wd_tcpdump_trace on the DUT should show the tapped traffic.

XIII. UDP Segmentation Offload and Pacing

1. Introduction

Chelsio's T4 series of Unified Wire Adapters provide extensive support for UDP Segmentation offload. This feature can be particularly useful to users who transport video using large frames over UDP. Without this feature, the video stream needs to be broken up into frames by the kernel, thereby using a great deal of CPU cycles in the process. With this feature, the breaking up of the stream into frames is done in the T4 chip, thereby freeing up the CPU to do other useful work on the system. Another notable advantage of this feature is that when the kernel does the disassembly to the stream, it sends the data as IP fragments. When the T4 does the disassembly, it sends the data out as individual UDP datagrams, which works much better in the video environment.

An additional feature, also useful in video applications, is the ability to pace the transmission rate of UDP traffic. This helps in lowering drop rates as well as reduces the burstiness of UDP traffic. To use this feature, users associate a bit rate and average packet size expected for this traffic class, to each of the traffic classes using the Chelsio supplied *cxgbtool* utility. To associate a particular UDP socket to a socket class, *setsockopt()* is used.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the UDP Segmentation Offload and Pacing driver.

- T420-CR
- T420-CX
- T440-CR
- T404-BT
- T422-CR
- T440-LP-CR
- T420-LL-CR

1.2. Software Requirements

1.2.1. Linux Requirements

The driver is developed to run on 64-bit Linux-based platforms and therefore it is a base requirement for running the driver.
Currently the driver is available for the following versions:

- Redhat Enterprise Linux 6 base kernel (RHEL6.0), 2.6.32-71.el6
- Redhat Enterprise Linux 6 update 1 kernel (RHEL6.1), 2.6.32-131.0.15.el6
- Redhat Enterprise Linux 6 update 2 kernel (RHEL6.2), 2.6.32-220.el6
- Redhat Enterprise Linux 6 update 3 kernel (RHEL6.3), 2.6.32-279.el6
- Suse Linux Enterprise Server 11 SP1 kernel (SLES11SP1), 2.6.32.12-0.7
- Fedora release 13 (FC 13), 2.6.33.3-85.fc13
- Fedora release 14 (FC 14), 2.6.35.6-45.fc14
- Kernel.org linux-2.6.34
- Kernel.org linux-2.6.35

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

2. Software/Driver Loading

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

Run the following commands to load the driver:

```
[root@host]# modprobe cxgb4
[root@host]# modprobe t4_tom
```

Though normally associated with the Chelsio TCP Offload engine, the *t4_tom* module is required in order to allow for the proper redirection of UDP socket calls.

3. Software/Driver Unloading

Reboot the system to unload the driver.

4. Software/Driver Configuration and Fine-tuning

4.1. Modifying the application

To use the UDP offload functionality, the application needs to be modified. Follow the steps mentioned below:

i. Determine the UDP socket file descriptor in the application through which data is sent

ii. Declare and initialize two variables in the application:

```
int fs=1442;
int cl=1;
```

- *fs* is the frame size in bytes. This value must equal the total frame size on the wire. That means the headers associated with the frame namely the UDP, IP, and ethernet headers must be taken into account. In the example above, the IO frame is 1400 bytes and the total of the associated headers is 42 bytes, hence *fs*=1442.
- *cl* is the UDP traffic class that the user wishes to assign the data stream to. This value needs to be in the range of 0 to 7.

The application will function according to the parameters set for that traffic class.

iii. Add socket option definitions:

In order to use *setsockopt()* to set the options to the UDP socket, the following two definitions need to be made:

- SO_FRAMESIZE used for setting frame size, which has the value 291.
- SOL_SCHEDCLASS used for setting UDP traffic class, which has the value 290.

```
# define SO_FRAMESIZE 291
# define SOL_SCHEDCLASS 290
```

iv. Use the *setsockopt()* function to set socket options:

```
//Get the UDP socket descriptor variable
setsockopt (sockfd , 17, SO_FRAMESIZE, &fs, sizeof(fs));
setsockopt (sockfd , 17, SOL_SCHEDCLASS, &cl, sizeof(cl));
```

where:

- *sockfd* : The file descriptor of the UDP socket

- *protocol level* : IPPROTO_UDP for UDP
- *SO_FRAMESIZE / SOL_SCHEDCLASS* : the option being set
- *&fs* / *&cl*: pointer to the framesize and class variables
- *sizeof(fs)* / *sizeof(cl)* : the size of the variables

v. Now, compile the application.

4.2. Configuring UDP traffic class

Now that the application has been modified to associate the application's UDP socket to a particular UDP traffic class, the pacing of that socket's traffic can be set using the *cxgbtool* utility. The command and its parameters are explained below:

```
[root@host]# cxgbtool ethX config_sched_class channel <Channel No.> class
<UDP class> max <Bit rate> pktsize <Frame size>
```

Here,

- *ethX* is the Chelsio interface
- *Channel No.* is the port on which data is flowing (0-3)
- *UDP class* is the UDP traffic class (0-7) set for *SOL_SCHEDCLASS* socket option in the application in section 4.1.
- *Bit rate* is the bit rate (Kbps) for this UDP stream.
- *Frame size* is the frame size on wire in bytes; it should be equal to the value set for *SO_FRAMESIZE* socket option in the application in section 4.1.

Example:
The user wants to transfer UDP data on port 0 of the T4 card using the UDP Segmentation Offload and Pacing option. The application has been modified as shown in section 4.1. In order to set a Bit rate of 10Mbps for traffic class 1 with a frame size of 1442 on port 0, the following invocation of *cxgbtool* is used:

```
[root@host]# cxgbtool ethX config_sched_class channel 0 class 1 max 10000
pktsize 1442
```

Important

To get an accurate bit rate per class, the IO size, which is set by the application, should be a multiple of the packet size (excluding the header size). In the example above, the IO size should be set to a multiple of 1400 (excluding the header size of 42 bytes).

Important

Linux Unified Wire currently supports 10240 offload UDP connections. If your application has the chance of using more than 10240 offload UDP connections, you should have code to support this limit. If not keeping track of the number of connections in use, the user can check the return code from a send() or sendto() call. If that fails, then the application will need to close this socket and open a new one that uses the kernel UDP stack. The application should not use setsockopt() to set the Chelsio UDP offload parameters.

XIV. Appendix A

1. Troubleshooting

- ***Cannot bring up T4 interface***

Make sure you have created the corresponding network-script configuration file as stated in Section 5.2. If the file does exist, make sure the structure and contents are correct. A sample is given in Section 5.1. Another reason may be that the IP address mentioned in the configuration file is already in use on the network.

- ***Cannot ping through T4 interface***

First, make sure the interface was successfully brought up using ifup ethX (where ethX is your interface) and that it is linked to an IP address, either static or obtained through DHCP.

You then may want to check whether the destination host (i.e. the machine you are trying to ping) is up and running and accepts ICMP requests such as ping. If you get a return value of 0 when doing a `cat /proc/sys/net/ipv4/icmp_echo_ignore_all` on the remote host that means it is configured to reply to incoming pings. Change `ipv4` to `ipv6` in the path if you are using IPv6. Note that this is a Linux-only tip.

If you have more than one interface wired to the network, make sure you are using the right one for your outgoing ping requests. This can be done by using the `-I` option of the ping command, as shown in the following example:

```
[root@host]# ping -I eth1 10.192.167.1
```

Where 10.192.167.1 is the machine you want to ping.

- ***Configuring firewall for your application***

In many cases the firewall software on the systems may prevent the applications from working properly. Please refer to the appropriate documentation for the Linux distribution on how to configure or disable the firewall.

- ***FCoE link not up***

Always enable LLDP on the interfaces as FCoE link won't come up until and unless a successful LLDP negotiation happens.

- ***priority-flow-control mode on the switch***

On the switch, make sure priority-flow-control mode is always set to auto and flow control is disabled.

- ***Configuring Ethernet interfaces on Cisco switch***

Always configure Ethernet interfaces on Cisco switch in trunk mode.

- ***Binding VFC to MAC***

If you are binding the VFC to MAC address in case of Cisco Nexus switch, then make sure you make the Ethernet interface part of both Ethernet VLAN and FCoE VLAN.

- **Cisco nexus switch reporting “pauseRateLimitErrDisable”**

If in any case the switch-port on the Cisco nexus switch is reporting “pauseRateLimitErrDisable”, then perform an Ethernet port shut/no shut.

2. 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).

May 16, 2011

CHELSIO END USER LICENSE AGREEMENT

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, nontransferable license to:

(i) install and use the Software on a single computer system or on multiple workstations, systems and servers that incorporate a Chelsio network adapter and may be accessed by multiple users from multiple locations; and (ii) 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, as 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 for profit, 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 recorded on the original media 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. 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 not meeting the above limited warranty 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.

Chelsio reserves the right to modify this license agreement at any time without notice, and any modified version of this agreement shall supercede any earlier versions.

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

Chelsio Communications, Inc.
370 San Aleso Ave.
Sunnyvale, CA 94085

