DBL v3.1.6 User Guide

DBL v3.1.6 User Guide

Version 3.1.6

October 4, 2019

Preface

ARIA Cybersecurity Solutions, which includes ARIA SDS, Myricom network adapters, and nVoy security appliances, are designed and manufactured by the High Performance Products Division of CSP Inc.

ARIA Cybersecurity Solutions Product Development
C/O CSPi
175 Cabot Street, Suite 210
Lowell, MA 01854
Tel: (800) 325-3110
ARIA_support@ariacybersecurity.com
http://www.ariacybersecurity.com/support

This document describes the run-time DBL® software for the ARIA Cybersecurity Solutions Myricom 10-Gigabit Ethernet products.

The document is intended for system and networking architects looking for a tailored solution with a focus on low latency combined with increased throughput. In this context, the DBL product provides a deployable solution by using a combination of advanced software stacks and 10GbE adapters. The document assumes that readers are familiar with the C language, using the GNU development tools, and general computer maintenance.

Software developers interested in directly utilizing the advanced features of Myricom products through the DBL interfaces, should refer to the DBL API Reference Manual. Software documentation and downloads are available from the ARIA Customer Portal.

Additional product information can be found on the ARIA website at:
http://www.ariacybersecurity.com/network-adapters

Contact Technical Support via the ARIA Customer Portal:
http://www.ariacybersecurity.com/support/
or via email:
ARIA_support@ariacybersecurity.com

Table of Contents

1 Introduction

DBL® is an optional, user-level, software interface for accelerating applications whose performance depends on network latency. Examples of applications that will benefit from DBL include algorithmic financial trading, VoIP, online gaming engines, and others where reduced network latency is beneficial.

DBL software with Myri-10G 10-Gigabit Ethernet Network Adapters leverages user-level, kernel-bypass, messaging techniques originally developed for high performance computing (HPC) applications, and applies them to IP communication over Ethernet in a multicast and/or unicast environment. DBL is interoperable and wire-compatible with all standard TCP and UDP implementations.

Unlike IP communication through the host operating system, which allows the full set of IP networking services to potentially hundreds of user application threads, but at the cost of latency in the operating-system protocol stack, DBL takes advantage of kernel-bypass firmware in Myri-10G Network Adapters to allow high-priority user threads to send and receive IP frames directly. The typical user-level read latency (delay from packet arrival at the adapter network port until it is available to the application) of DBL on modern hosts is about 1.5us and the half-round-trip pingpong latency is about 3.5us.

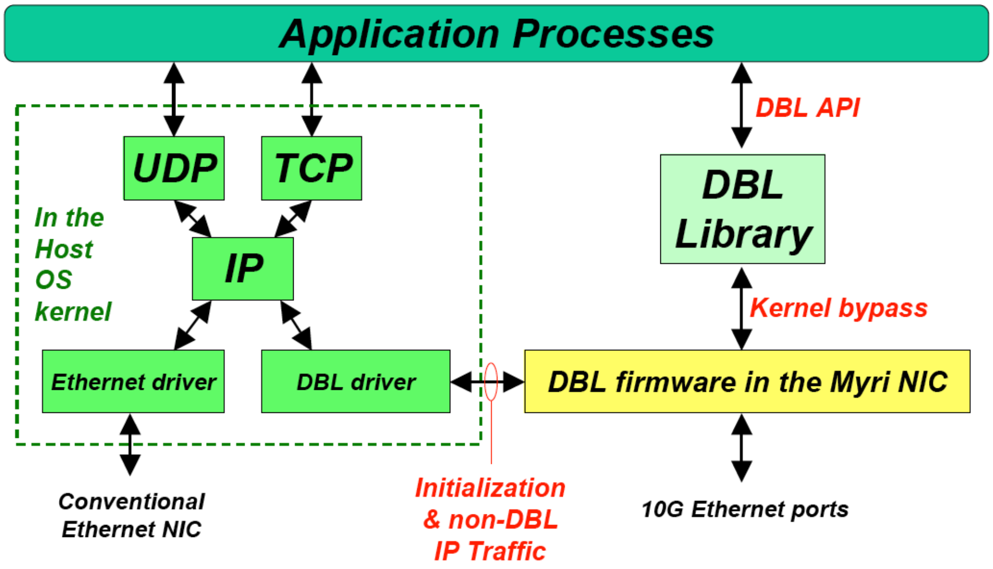


Figure 1. DBL Interface vs. Conventional Ethernet

DBL is not intended to completely replace IP communication through the host operating system, nor to improve performance in situations where overall system performance depends on multiplexing many user application threads over fewer available computing resources (or CPU cores). Rather, DBL will improve the throughput, response time, and transaction rates of user application threads that have dedicated computing resources. Thus, DBL best suits cases where target latencies, response times, and jitter would otherwise be hampered by the operating-system protocol stack and scheduling policies.

DBL software distributions are currently available for Linux and Windows. **Chapter 2: Installation** provides detailed hardware and software installation instructions, as well as instructions to run the DBL API tests and example programs.

DBL v3.1.6 can be used in two different modes:

- **TA Mode: Transparent Acceleration for TCP/UDP.** This option is preferred for customers who wish to run their existing sockets-based applications without modifications. Instructions to run an application in DBL TA Mode are provided in **Chapter 3: Transparent Sockets Acceleration Software**.
- **API Mode: DBL API for TCP/UDP.** Applications can be written to target an API that supports low-latency TCP or UDP receive and send operations (multicast and unicast). The **DBL API documentation** is available in HTML format in the **/share/doc/dbl-api/** directory of the software distribution. The full DBL API is accessible from C and Java on all supported platforms (Linux and Windows).

Chapter 4: Tuning provides tuning guidelines for the DBL software. **Chapter 5: Troubleshooting** details common DBL installation, usage, and performance issues. This chapter also provides a detailed explanation of timestamping with the DBL software.

Appendix: DBL Counters provides a detailed explanation of all DBL software and hardware counters.

Appendix: DBL Software Limitations/Restrictions explains the current software implementation limitations and restrictions.

1.1 New Features in the DBL v3.1.6 Software

The release of the DBL v3 software introduces the following features/enhancements to the DBL software.

- **API Extension for TCP**

  In DBL v2 the DBL API only supported UDP channels. With the introduction of DBL v3.1.3 and later, an extension of the DBL API became available to add support for TCP channels. The original concept remains the same: channels are created when calling dbl_bind with a given channel type (UDP or TCP) and a port on which to bind. (See the **DBL API documentation** for details).

- **DBL-specific socket extensions (MSG_WARM, SIO_GETNICTIME, SO_TIMESTAMP)**

  Three DBL-specific sockets extensions are available for Linux and Windows in the DBL v3.1.6 software. For details, please read **Section 3.3 DBL-Specific Socket Extensions** and the **DBL API Documentation**.

- **TCP congestion management modules**

  Congestion control algorithms supported by the DBL TCP API belong to the loss-based family of algorithms. We support the following algorithms: newreno, cubic, and htcp. By default, we use the cubic algorithm since it is well suited for latency sensitive applications. Refer to **Chapter 3** of this guide or the comments in the **dbl.conf** file for instructions on usage.

- **Windows non-LSP TA mode**

  On Windows, starting with DBL v3.1.3 and later, two solutions for the DBL TA mode for TCP and UDP sockets (Winsock) are offered. Note that only one DBL TA method (LSPs or DLL preload) may be active/registered at a time. The non- LSP (DLL preload) TA mode achieves binary transparency and has the benefit of better performance, lowering latency by 0.5-5.0us depending on the complexity of the socket operation. Please refer to **Section 3.4 Windows DBL TA Implementation** for details.

- **Arista Networks DANZ timestamping support**

  Timestamping support with Arista 7150 series 10GbE switches is available in the DBL v3.1.3 and later software. For details, please refer to **Chapter 2** of this document.

- **Windows Server 2016 or 2012 R2 support**

- **myri_ptpv2d support on Windows**

  DBL v3.1.3 and later provides a PTP daemon which was previously only available to Linux.

- **MSI-X support on Windows**

  MSI-X support on Windows is available in DBL 3.0.3.52632 and later.

2 Installation

Use of the Myricom High-Speed, Low-Latency networking solutions requires the installation of one or more Network Adapters and DBL software. Once installed, the software must be activated via appropriate license keys.

2.1 Quickstart - Installation Overview

The installation process consists of three major areas to be addressed:

A. Physical installation of network adapter(s) into host computer(s) and connection of appropriate cable.

B. Installation of DBL Network Adapter Driver Software products and configuration of run-time operation.

C. Configuration of Software License Keys on each host.

Myricom products are designed to be compatible with prevailing industry standards and typically install quickly without significant user effort.

Generalized instructions about each of the three installation areas are given in subsequent sections as a guide which is sufficient to achieve success for most current computing platforms.

NOTE: DBL contains a **myri_dbl** driver that must be run instead of the default **myri10ge** driver typically associated to Myri-10G Ethernet network adapters.

Detailed differences in hardware and networking requirements for individual sites is beyond the scope of this manual and customers should contact ARIA Technical Support if they experience difficulty configuring Myricom solutions in their unique environments.

2.2 Hardware Installation Instructions

Installation of a network adapter is a straight-forward process, as follows:

1. Shutdown host system applications and Operating System.
2. Turn off computer power and unplug power cord.
3. Open the computer case and locate PCI-Express expansion card slots on the motherboard. Take care not to be confused with legacy PCI card slots which have different physical dimensions and electrical specification (see Figure 2). For best performance, the network adapter should be placed in a slot which supports an 8-lane PCI-Express link (x8). To verify the hardware installation for maximum performance, please read **Section 5.1.1 Hardware Installation/ Performance**. Some systems will have longer x16 PCIe slots which may also support short x8 cards (verify motherboard specifications for compatibility as mechanical fit alone does not guarantee electrical operation). Note that not all x8 connectors are actually wired as x8 slots; check motherboard specifications if you are unsure. Check that there is a free slot, then remove the mounting screw of the protective bracket plate (if present) covering the selected slot and set aside the plate. If there are no free slots then it will be necessary to remove an unneeded card to make room for the network adapter.

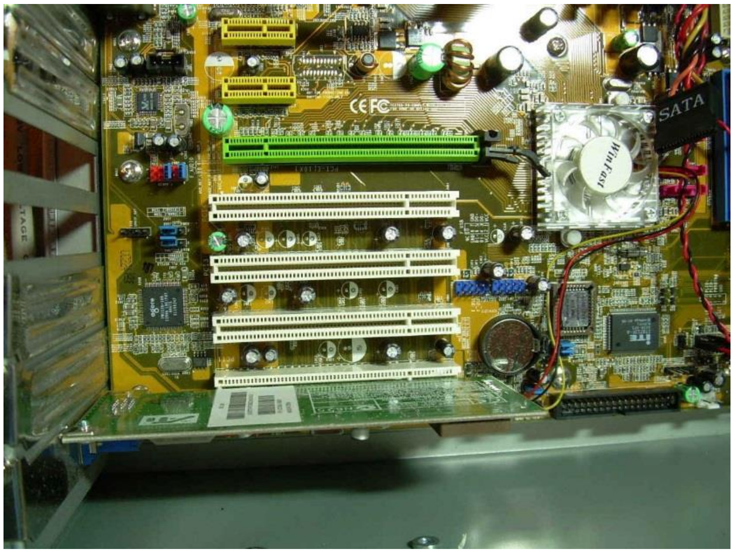


Figure 2. PCI-Express (green) and Legacy PCI (white) Card Slots

4. CAUTION: When handling network adapters it is important to follow anti-static procedures to avoid accidently damaging integrated circuits on the card due to static electric discharge. Avoid working in damp areas or walking on carpet while transporting bare cards. It is recommended to use an anti-static wrist strap with the ground wire attached to a metal frame in the computer case or its power supply. Alternatively, hold the card carefully by its metal mounting bracket and gently touch the metal case of the

computer power supply with your free hand to safely ground any static charge before proceeding.

5. Locate the shipping package for the Myri-10G Network Adapter and remove the sealed protective sleeve containing the card. Open the seal on the protective sleeve and remove the adapter (see Figure 3). Be careful to avoid touching the gold connectors on the bottom edge of the card.

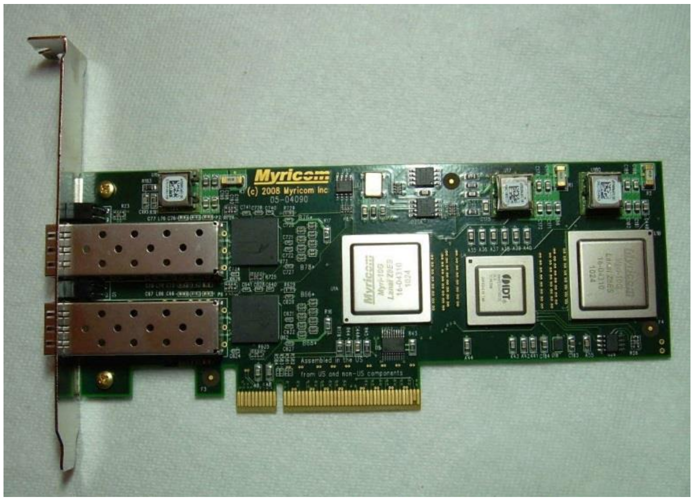


Figure 3. Typical Myri-10G Card

6. Move the card to the empty slot in the computer case. Line up the gold PCI connectors and indexing tab on the edge of the card with the PCI-Express slot, ensuring that the ports and mounting bracket are facing the back of the computer (see Figure 4). Carefully press the card into the slot, making sure to press evenly on both edges until the card is firmly seated. When the card is correctly installed it will “click” into place.

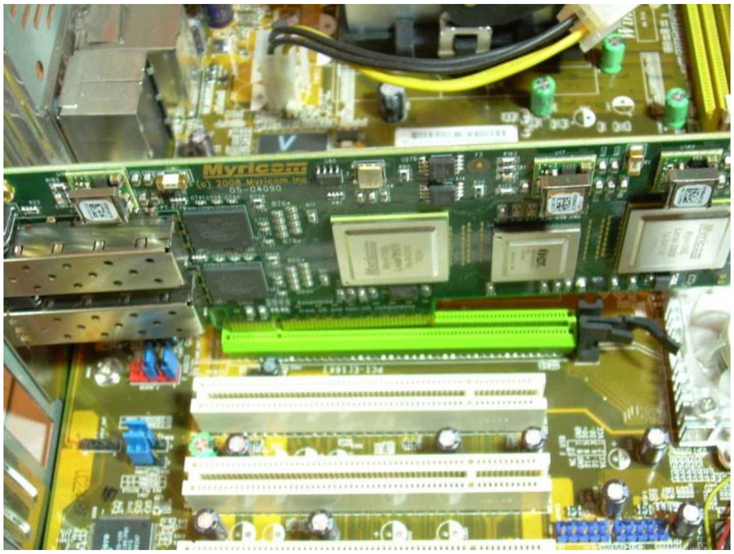


Figure 4. Card Insertion

7. Secure the card to the computer case with the screw removed from the bracket plate previously (see Figure 5). To verify the hardware installation for maximum performance, please read **Section 5.1.1 Hardware Installation/ Performance**.

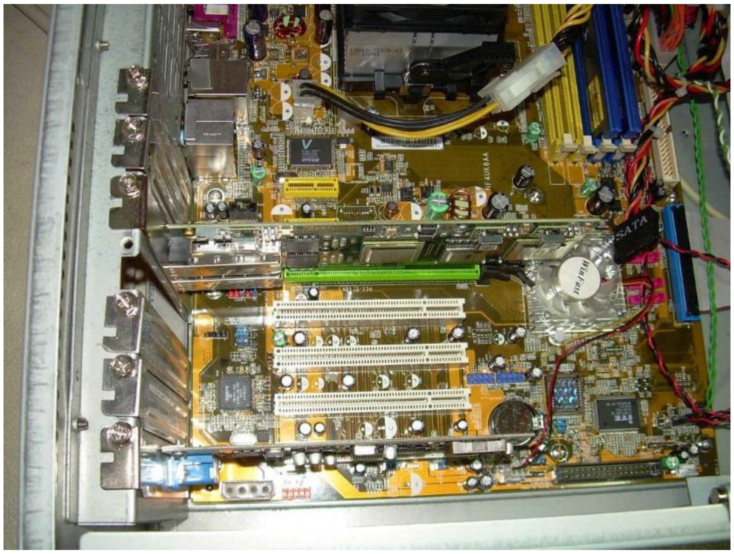


Figure 5. Card Fully Seated

8. Close the computer case and attach the power plug.
9. Attach appropriate cable to the exposed port(s) on back of the network adapter, taking care not to kink the cable.

10. For all Myri-10G network adapters except the 10G-PCIE2-8C2-2S-SYNC, the physical network adapter installation is now complete.
11. For the 10G-PCIE2-8C2-2S-SYNC adapter, you will additionally need a 50-ohm coaxial cable (such as RG58) with an SMB plug on one end to connect to the 10G-PCIE2-8C2-2S-SYNC adapter. The other end should be whatever is needed to connect to the timecode generator. Most timecode sources use BNC connectors, so for a source with this connector, the cable would need a BNC male connector.

    10G-PCIE2-8C2-2S-SYNC network adapters connect to CDMA or GPS IRIG-B00x timecode sources. These IRIG-B00x timecode outputs may also be called "unmodulated" or "DCLS" or "TTL into 50 ohm" outputs. Units which do not have enough outputs for the number of adapters to be connected will need a fanout buffer to drive multiple outputs from a single input.

    By default, the software assumes that the 10G-PCIE2-8C2-2S-SYNC adapter is connected to a timecode generator and timesource (hardware) timestamping is desired. If the SYNC adapter is not connected to a timecode generator, and host timestamping will be utilized, the **myri_dbl** driver must be loaded with the **myri_timesource=0** load-time option. Otherwise, DBL will return zero for the timestamps. For details, please read **Section 2.3 DBL Software Driver Installation Instructions.**

2.3 DBL Software Driver Installation Instructions

The DBL Software is distributed in a single package file containing:

- A special Network Adapter driver to replace the normal Ethernet driver
- A dynamic library of software modules
- Test programs to demonstrate DBL functionality

Begin the installation by first obtaining a copy of the appropriate DBL software package for your system by following these steps.

1. Using your normal Internet browser, login through the **ARIA Customer Portal** at http://www.ariacybersecurity.com/support/.

2. From the top-level menu bar select the **Groups** tab, navigate to the **DBL v3 – Lanai Group**, and the appropriate software distribution file.

The downloaded file format may be either .rpm or .tgz for Linux or .msi for Windows. A different installation process is required for each format and is explained in the three subsequent sections.

2.3.1 Linux Installation Process for Binary RPM (.rpm)

Continued install process for Linux hosts for RPM-based systems running a standard derivative of a RedHat-based Linux distribution (RHEL, Centos, Fedora).

The RPM is a binary RPM with the exception of the kernel module which is distributed as source. When the RPM is installed, the kernel module is compiled against the currently running kernel. For a successful installation, the **kernel- devel** package that corresponds to the currently running kernel (as reported by **uname -a**) must be installed. As for development packages, the installation will also require the **gcc** package for kernel module compilation and the **jdk** package for java runtime support. Users with alternative installations may prefer the binary tarball method described next.

1. When the download of the file is complete, open a terminal window and type the following command using the actual <version_info> of the downloaded file name:

```
# rpm -i myri_dbl-<version_info>*rhel-5342.x86_64.rpm
```

As a result of a successful installation, the package will be installed in **/opt/dbl** and the kernel module is automatically started. Upon next reboot, the kernel module will also be automatically started from /etc/init.d.

(Optional) The --prefix option to rpm can be used as the package is relocatable.

2. (Optional): Manual stopping and starting of the kernel module can by controlled by using the **start** or **stop** commands with the **myri_start_stop** script:

```
# /opt/dbl/sbin/myri_start_stop start
```

The basic installation of the DBL software is complete.

3. A Local Area Network address must be configured for each port of the Myri-

10G Network Adapter(s) before it can be used on the network. Manually assign the appropriate IP address on all ports on all host systems to establish the DBL enabled network.

The installation of the DBL-enabled Myri-10G Network Adapter is now complete.

Note that with some Linux distributions, the GUI Package Manager can be used to perform the package install rather than the command line used above.

Module Parameters (Optional)

To determine the list of optional module load-time parameters, type:

```
# modinfo /opt/dbl/sbin/myri_dbl.ko
```

If you are using SFP+-terminated copper “direct attach” cables with the Myri-10G network adapters and you experience link up/down connectivity issues or bad crc errors, please read about the load-time option **myri_serdes_mode** in **Section 5.1.2 Hardware Cabling Connectivity Issues**.

Timestamping Support (Optional)

Timestamping support in DBL API mode is enabled by default. However, for performance reasons, in DBL 2.1 and later, timestamping support with DBL TA mode is disabled by default. Note that this setting is a change in the default behavior. For implementation details of how this change affects DBL API and TA customers please read **Section 5.2.12 Network Adapter Timestamps**.

In the DBL release with SYNC adapter support, three extra parameters were added to the DBL module:

- **myri_timesource**

  By default, the DBL software assumes that the 10G-PCIE2-8C2-2S-SYNC adapter is connected to a timecode generator (**myri_timesource=1**) and that timesource (hardware) timestamping mode is desired. If the SYNC adapter is not connected to a timecode generator, and host timestamping will be utilized, the **myri_dbl** driver must be loaded with the **myri_timesource=0** load-time option. Otherwise, DBL will return zero for the timestamps.

  On Linux, the instructions to set **myri_timesource=0** and thus switch to using host timestamps on the SYNC adapter are:

  ```
  # /opt/dbl/sbin/myri_start_stop start myri_timesource=0
  ```

  On non-SYNC adapters, the default is **myri_timesource=0** and host timestamping.

- **myri_timesource_loss_mode**

  Set **myri_timesource_loss_mode=1** for free-running (default), and set value to 0 for zeroing timestamps.

- **myri_timesource_pps**

  Set **myri_timesource_pps=0** to disable the exporting of a pps interface that can be used by ntp.

For additional details on timestamping, please read **Section 5.2.12 Network Adapter Timestamps**.

In DBL 2.1 and later, timestamping with DBL TA mode is disabled by default. To utilize timestamping with DBL TA mode, please read **Section 3.2: Customizing Transparent Acceleration with dbl.conf.** (Note that this setting is a change in the default behavior. In DBL 2.0, timestamping was enabled by default in DBL TA mode.)

Arista Timestamping Support (Optional)

Timestamping support with Arista 7150 series 10GbE switches is available in DBL v3 and later. DBL supports the alternate timestamping provided by the Arista 7150 Switch EOS. For more details, please read:

https://eos.arista.com/timestamping-on-the-7150-series/

The technique supported by DBL is the append (before-fcs in the Arista terminology), which will add 4 bytes at the end of each packet, as well as extra packets to describe the time references for computations to be made (keyframes in the Arista terminology).

Arista timestamping support is enabled by default (**MYRI_ARISTA_ENABLE_TIMSTAMPING=1**).

- **MYRI_ARISTA_ENABLE_TIMESTAMPING = 0,1**

  MYRI_ARISTA_ENABLE_TIMESTAMPING enables the support for Arista timestamping, and is activated as soon as the process receives its first keyframe. The Arista switch does not guarantee delivery of keyframes when it is under heavy load. It is possible for the arrival of the first keyframe which enables time stamping to begin to be delayed. Also, under these conditions, the arrival interval of keyframes may be greater than the timestamp wrap time of approximately 6.1 seconds. In this case DBL will continue to approximate timestamps until a new keyframe arrives. Therefore, a lapse of time may be needed to receive the first keyframe. By default, **MYRI_ARISTA_ENABLE_TIMESTAMPING=1.**

  On Linux, if you are using 10G-PCIE2-8C2-2S-SYNC adapters, hardware timestamping must be disabled. Execute the following command:

  ```
  # /opt/dbl/sbin/myri_start_stop start myri_timesource=0
  ```

  To verify that Arista timestamping has been enabled, the following message will appear in the kernel log.

  ```
  myri_dbl: INFO: ** Arista timestamping [on] supported modes: append (before-fcs)
      * features: keyframe check min.rate [on] kf check ptp [on] kf kernel dup [off]
  ```

- **MYRI_ARISTA_PARAM_KF_DST_IP = <IP ADDRESS>**

  MYRI_ARISTA_PARAM_KF_DST_IP must be set to the IP address of the keyframe generator (the Arista switch). By default, the address is set to 1.1.1.0. For more details, please consult

  http://www.aristanetworks.com/docs/Manuals/ConfigGuide.pdf

The following three environment variables are optional for Arista timestamping support.

1. **MYRI_ARISTA_PARAM_KF_CHECK_PTP = 0,1**

MYRI_ARISTA_PARAM_KF_CHECK_PTP is an optional environment variable to check if the PTP synchronization is working on the switch. By default, **MYRI_ARISTA_PARAM_KF_CHECK_PTP=1**.

2. **MYRI_ARISTA_PARAM_KF_CHECK_RATE = 0,1**

   MYRI_ARISTA_PARAM_KF_CHECK_RATE is an optional environment variable to check if the keyframes sent are sufficient to provide accuracy. By default, **MYRI_ARISTA_PARAM_KF_CHECK_RATE=1**.

3. **MYRI_ARISTA_PARAM_KF_KERNDUP = 0, 1**

   By default, **MYRI_ARISTA_PARAM_KF_KERNDUP=0**. If enabled, this feature will duplicate keyframe packets to the host stack.

Uninstalling the Binary RPM

To uninstall the Linux DBL RPM, execute the following commands:

```
$ sudo /opt/dbl/sbin/myri_start_stop stop
$ sudo rpm -e myri_dbl
```

2.3.2 Linux Installation Process for Binary tarball (.tgz)

If you downloaded the binary tarball (.tgz) for Linux, please follow these installation instructions.

1. When the download of the file is complete, first remove any existing /opt/dbl directory, then type the following commands using the actual <version_info> of the downloaded file name:

```
# cd /opt
# gunzip -c <path>/myri_dbl-<version_info>*x86_64.tgz | tar xvf -
# mv myri_dbl-<version_info>*x86_64  dbl
# cd /opt/dbl
# sbin/rebuild.sh
```

2. Now load the drivers by typing:

```
# /opt/dbl/sbin/myri_start_stop start
```

The basic installation of the DBL software is complete.

3. A Local Area Network address must be configured for each port of the Myri-

10G Network Adapter(s) before it can be used on the network. Manually assign the appropriate IP address on all ports on all host systems to establish the DBL enabled network.

The installation of the DBL-enabled Myri-10G Network Adapter is now complete.

Module Parameters (Optional)

To determine the list of optional module load-time parameters, type:

```
# modinfo /opt/dbl/sbin/myri_dbl.ko
```

If you are using SFP+-terminated copper "direct attach" cables with the Myri-10G network adapters and you experience link up/down connectivity issues or bad crc errors, please read about the load-time option **myri_serdes_mode** in **Section 5.1.2 Hardware Cabling Connectivity Issues**.

Timestamping Support (Optional)

Timestamping support in DBL API mode is enabled by default. However, for performance reasons, in DBL 2.1 and later, timestamping support with DBL TA mode is disabled by default. Note that this setting is a change in the default behavior. For implementation details of how this change affects DBL API and TA customers please read **Section 5.2.12 Network Adapter Timestamps**.

In the DBL release with SYNC adapter support, three extra parameters were added to the DBL module:

- **myri_timesource**

  By default, the DBL software assumes that the 10G-PCIE2-8C2-2S-SYNC adapter is connected to a timecode generator (**myri_timesource=1**) and that timesource (hardware) timestamping mode is desired. If the SYNC adapter is not connected to a timecode generator, and host timestamping will be utilized, the **myri_dbl** driver must be loaded with the **myri_timesource=0** load-time option. Otherwise, DBL will return zero for the timestamps.

  On Linux, the instructions to set **myri_timesource=0** and thus switch to using host timestamps on the SYNC adapter are:

  ```
  # /opt/dbl/sbin/myri_start_stop start myri_timesource=0
  ```

  On non-SYNC adapters, the default is **myri_timesource=0** and host timestamping.

- **myri_timesource_loss_mode**

  Set **myri_timesource_loss_mode=1** for free-running (default), and set value to 0 for zeroing timestamps.

- **myri_timesource_pps**

  Set **myri_timesource_pps=0** to disable the exporting of a pps interface that can be used by ntp.

For additional details on timestamping, please read **Section 5.2.12 Network Adapter Timestamps**.

In DBL 2.1 and later, timestamping with DBL TA mode is disabled by default. To utilize timestamping with DBL TA mode, please read **Section 3.2: Customizing Transparent Acceleration with dbl.conf.** (Note that this setting is a change in the default behavior. In DBL 2.0, timestamping was enabled by default in DBL TA mode.)

Arista Timestamping Support (Optional)

Timestamping support with Arista 7150 series 10GbE switches is available in DBL v3 and later. DBL supports the alternate timestamping provided by the Arista 7150 Switch EOS. For more details, please read:

https://eos.arista.com/timestamping-on-the-7150-series/

The technique supported by DBL is the append (before-fcs in the Arista terminology), which will add 4 bytes at the end of each packet, as well as extra packets to describe the time references for computations to be made (keyframes in the Arista terminology).

Arista timestamping support is enabled by default (**MYRI_ARISTA_ENABLE_TIMSTAMPING=1**).

- **MYRI_ARISTA_ENABLE_TIMESTAMPING = 0,1**

  MYRI_ARISTA_ENABLE_TIMESTAMPING enables the support for Arista timestamping, and is activated as soon as the process receives its first keyframe. The Arista switch does not guarantee delivery of keyframes when it is under heavy load. It is possible for the arrival of the first keyframe which enables time stamping to begin to be delayed. Also, under these conditions, the arrival interval of keyframes may be greater than the timestamp wrap time of approximately 6.1 seconds. In this case DBL will continue to approximate timestamps until a new keyframe arrives. Therefore, a lapse of time may be needed to receive the first keyframe. By default, **MYRI_ARISTA_ENABLE_TIMESTAMPING=1.**

  On Linux, if you are using 10G-PCIE2-8C2-2S-SYNC adapters, hardware timestamping must be disabled. Execute the following command:

  ```
  # /opt/dbl/sbin/myri_start_stop start myri_timesource=0
  ```

  To verify that Arista timestamping has been enabled, the following message will appear in the kernel log.

  ```
  myri_dbl: INFO: ** Arista timestamping [on] supported modes: append (before-fcs)
      * features: keyframe check min.rate [on] kf check ptp [on] kf kernel dup [off]
  ```

- **MYRI_ARISTA_PARAM_KF_DST_IP = <IP ADDRESS>**

  MYRI_ARISTA_PARAM_KF_DST_IP must be set to the IP address of the keyframe generator (the Arista switch). By default, the address is set to 1.1.1.0. For more details, please consult the appropriate documentation from Arista:
  https://www.arista.com/en/support/product-documentation

The following three environment variables are optional for Arista timestamping support.

1. **MYRI_ARISTA_PARAM_KF_CHECK_PTP = 0,1**

   MYRI_ARISTA_PARAM_KF_CHECK_PTP is an optional environment variable to check if the PTP synchronization is working on the switch. By default, **MYRI_ARISTA_PARAM_KF_CHECK_PTP=1**.

2. **MYRI_ARISTA_PARAM_KF_CHECK_RATE = 0,1**

   MYRI_ARISTA_PARAM_KF_CHECK_RATE is an optional environment variable to check if the keyframes sent are sufficient to provide accuracy. By default, **MYRI_ARISTA_PARAM_KF_CHECK_RATE=1**.

3. **MYRI_ARISTA_PARAM_KF_KERNDUP = 0, 1**

   By default, **MYRI_ARISTA_PARAM_KF_KERNDUP=0**. If enabled, this feature will duplicate keyframe packets to the host stack.

Uninstalling the Binary tarball (.tgz)

To uninstall the Linux DBL tarball (.tgz), execute the following commands:

```
$ sudo /etc/init.d/myri_start_stop stop
$ sudo rm -rf /opt/dbl/
$ sudo rm -f /etc/init.d/myri_start_stop
```

2.3.3 Windows Installation

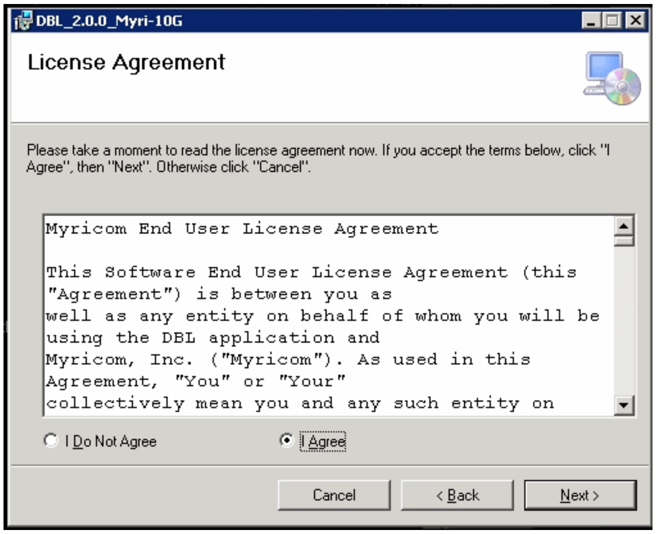
Installation using Windows Installer (.msi)

NOTE:

- Installation must be done from an account with administrator privileges and permissions.
- VLAN support is included in the DBL MSI package (see Section 2.3.3.5).
- In DBL 2.1 and later, timestamping with DBL TA mode is disabled by default. If you need Windows DBL TA timestamping support, contact ARIA Technical Support.
- The .msi file must be on a local or regular network drive; a Remote Desktop-mounted device will not work.
- .NET 2.0 or later is required. If this is not installed, the installer will open a warning and provide a way to download it.

1. When the download of the file is complete, invoke the MSI installer with a GUI and step through these dialogue pages.

a. License Agreement



License Agreement

Please take a moment to read the license agreement now. If you accept the terms below, click "I Agree", then "Next". Otherwise click "Cancel".

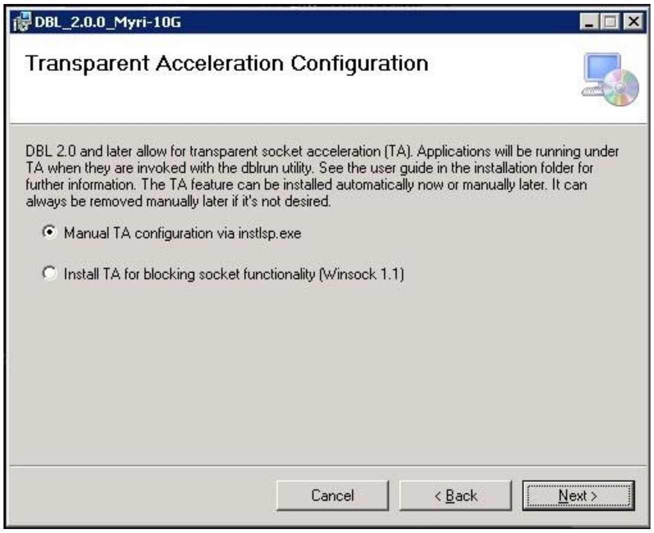
```
Myricom End User License Agreement

This Software End User License Agreement (this
"Agreement") is between you as
well as any entity on behalf of whom you will be
using the DBL application and
Myricom, Inc. ("Myricom"). As used in this
Agreement, "You" or "Your"
collectively mean you and any such entity on
```

○ I Do Not Agree ● I Agree

You must agree to the license to continue.

b. Transparent Acceleration (TA) Configuration



Transparent Acceleration Configuration

DBL 2.0 and later allow for transparent socket acceleration (TA). Applications will be running under TA when they are invoked with the dblrun utility. See the user guide in the installation folder for further information. The TA feature can be installed automatically now or manually later. It can always be removed manually later if it's not desired.

● Manual TA configuration via instlsp.exe

○ Install TA for blocking socket functionality (Winsock 1.1)

For Transparent Acceleration Configuration, there are 2 options:

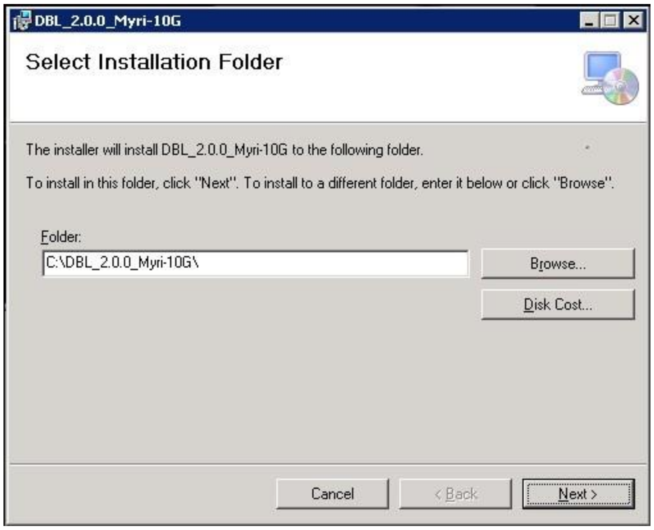
2. Manual TA configuration.

   Select this option if you want to install the utility manually.

   If this option is specified, Transparent Acceleration may be installed manually using the steps in the Transparent Acceleration section below.

3. Install TA for blocking socket functionality (Winsock 1.1 or 2.2)
   Select this option to install the automated TA configuration.

c. Select Installation Folder



DBL_2.0.0_Myri-10G

Select Installation Folder

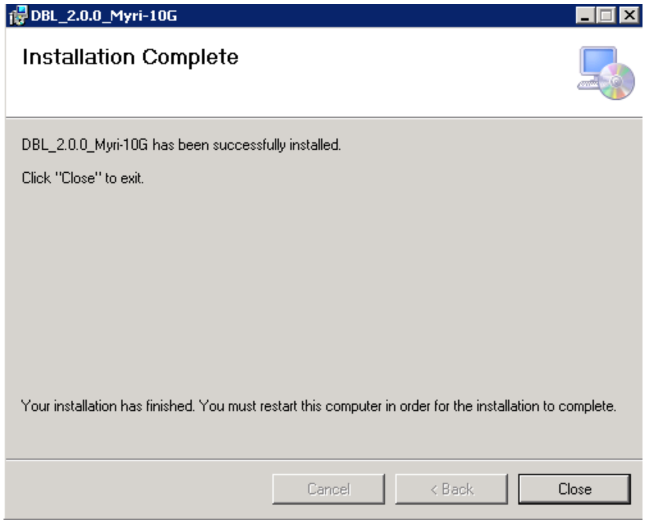
The installer will install DBL_2.0.0_Myri-10G to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\DBL_2.0.0_Myri-10G\

Specify the location to install the software.

d. Ready to Install



DBL_2.0.0_Myri-10G

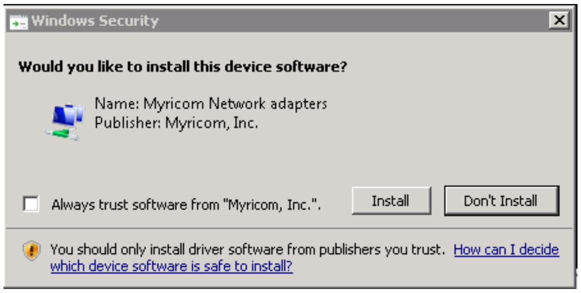
Installation Complete

DBL_2.0.0_Myri-10G has been successfully installed.

Click "Close" to exit.

Your installation has finished. You must restart this computer in order for the installation to complete.

If a security alert like the one below appears during the installation, click on Install/Continue.



Windows Security

Would you like to install this device software?

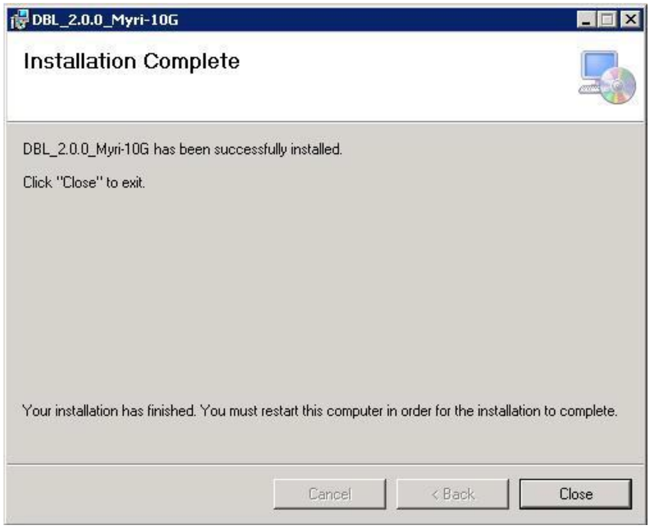
Name: Myricom Network adapters
Publisher: Myricom, Inc.

☐ Always trust software from "Myricom, Inc.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

4．Finish Up

After the installation wizard is complete, the computer may need to be restarted to complete the installation. If that is needed, there will be a note on the final page of the installer.



DBL_2.0.0_Myri-10G

Installation Complete

DBL_2.0.0_Myri-10G has been successfully installed.

Click "Close" to exit.

Your installation has finished. You must restart this computer in order for the installation to complete.

5. When finished, set the IP address and other network settings as explained on the Microsoft website.

Confirming the MSI Installation

Open the Device Manager (Start->Run->devmgmt.msc) to confirm that the software has been successfully installed. The software installation is successful when the text **Myri-10G PCIe NIC with DBL** appears under **Device Manager -> Network adapters.**

If an error has been encountered during the software installation, the output of **Device Manager -> Network adapters** will either show no DBL devices or a yellow exclamation mark.

If the MSI installation fails when running the MSI installer, please send ARIA Technical Support the log obtained by adding /log filename to the MSI install. For example,

```
c:\> dbl-<version_info>*x64_wnet_wlh.msi /log dbl_install_log.txt
```

Refer to **Chapter 5: Troubleshooting** for further details.

If you are using SFP+-terminated copper “direct attach” cables with the Myri-10G network adapters and you experience link up/down connectivity issues or bad crc errors, please read about the load-time option **myri_serdes_mode** in **Section 5.1.2 Hardware Cabling Connectivity Issues**.

Installing using Windows msiexec

If the MSI cannot be installed, the files can be extracted from the MSI without running any Custom Actions by using the following command line:

```
$ msiexec /a dbl-<version_info>*x64_wnet_wlh.msi /qb
TARGETDIR="C:\DBL_Myri-10G" INSTALLLEVEL=5
```

Or, for default settings, use the following command line:

```
$ msiexec /i dbl-<version_info>*x64_wnet_wlh.msi INSTALLLEVEL=5
```

Afterwards, please use the 'Update Driver' Network Adapter option.

Deploying for Windows imaging

To deploy DBL for Windows imaging, the files can be extracted from the MSI by using the following command line:

```
$ msiexec /a dbl-<version_info>*x64_wnet_wlh.msi /qb
TARGETDIR="C:\DBL_Myri-10G"
```

Teaming/VLAN Configuration

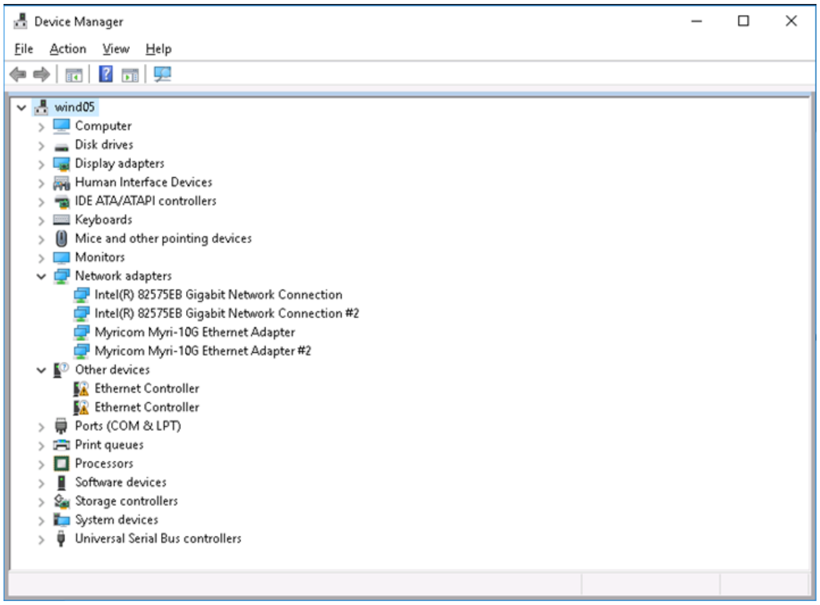
The DBLv3 VLAN/Team driver enables two DBLv3 network adaptor ports to team up.

NOTE	For illustrative purposes, these instructions describe how to configure teaming on Windows Server 2016.

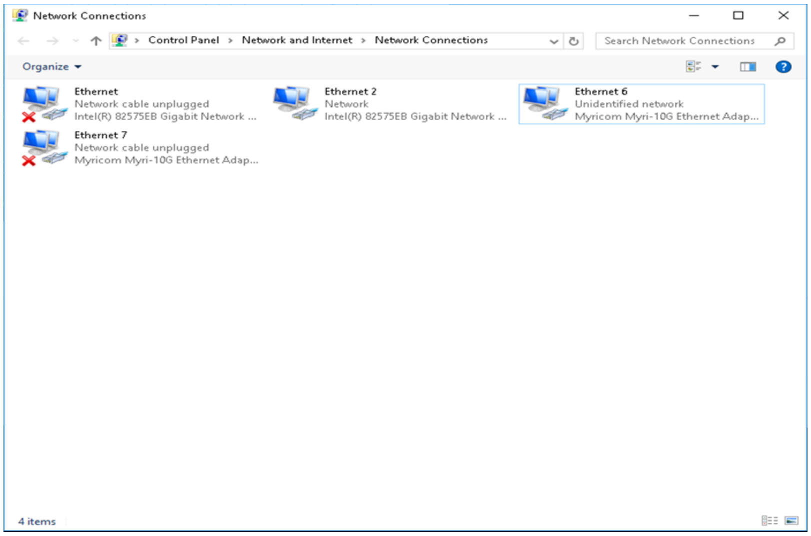
The VLAN/Team built within the Myricom Ethernet driver and DBL releases will not appear when installed into a Windows Server 2016 system. Windows Server 2016, however, will allow you to configure teaming/VLAN using the operating system network configuration options as described below.

Team the two DBLv3.1.6 network adapters as follows:

1. Verify Windows Server detects the appropriate network adapters. The adapters should be shown in the Device Manager and Network Connections.



- wind05
  - Computer
  - Disk drives
  - Display adapters
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
    - Intel(R) 82575EB Gigabit Network Connection
    - Intel(R) 82575EB Gigabit Network Connection #2
    - Myricom Myri-10G Ethernet Adapter
    - Myricom Myri-10G Ethernet Adapter #2
  - Other devices
    - Ethernet Controller
    - Ethernet Controller
  - Ports (COM & LPT)
  - Print queues
  - Processors
  - Software devices
  - Storage controllers
  - System devices
  - Universal Serial Bus controllers


Network Connections
Control Panel > Network and Internet > Network Connections
Search Network Connections
Organize
4 items

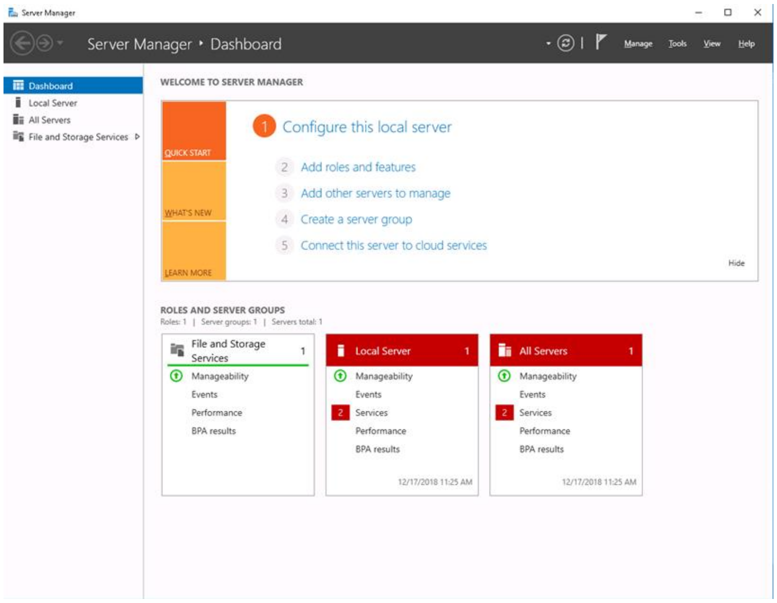
Ethernet
Network cable unplugged
Intel(R) 82575EB Gigabit Network ...

Ethernet 2
Network
Intel(R) 82575EB Gigabit Network ...

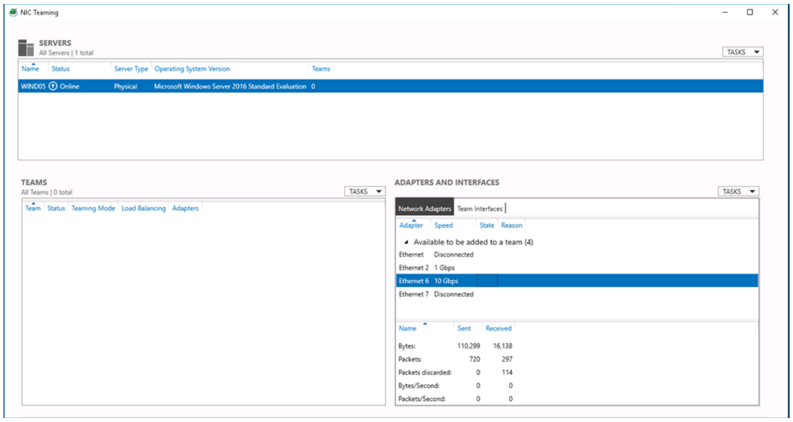
Ethernet 6
Unidentified network
Myricom Myri-10G Ethernet Adap...

Ethernet 7
Network cable unplugged
Myricom Myri-10G Ethernet Adap...

2. Open the Server Manager Dashboard.



3. Select Local Server and enable teaming on the desired adapter.



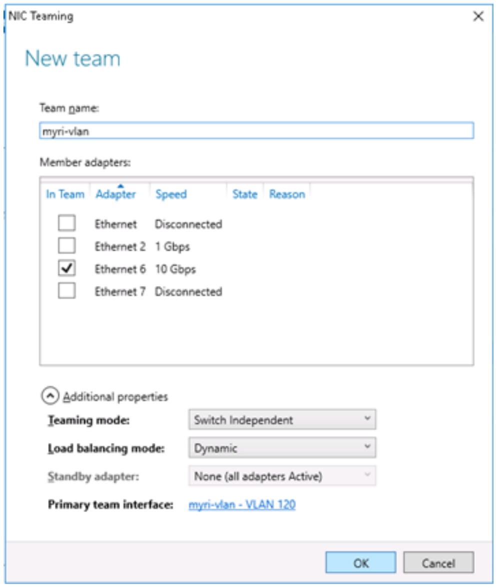
Name	Status	Server Type	Operating System Version	Teams
WIND05	Online	Physical	Microsoft Windows Server 2016 Standard Evaluation	0

Team	Status	Teaming Mode	Load Balancing	Adapters

Adapter	Speed	State	Reason
Available to be added to a team (4)			
Ethernet	Disconnected		
Ethernet 2	1 Gbps		
Ethernet 6	10 Gbps		
Ethernet 7	Disconnected		

Name	Sent	Received
Bytes:	110,299	16,138
Packets:	720	297
Packets discarded:	0	114
Bytes/Second:	0	0
Packets/Second:	0	0

4. Create a new team, provide a name for the team, select the appropriate member adapters, and configure the additional properties (Teaming mode, Load balancing mode, and Primary team interface.



NIC Teaming

New team

Team name: myri-vlan

Member adapters:

In Team	Adapter	Speed	State	Reason
☐	Ethernet	Disconnected		
☐	Ethernet 2	1 Gbps		
☑	Ethernet 6	10 Gbps		
☐	Ethernet 7	Disconnected		

Additional properties

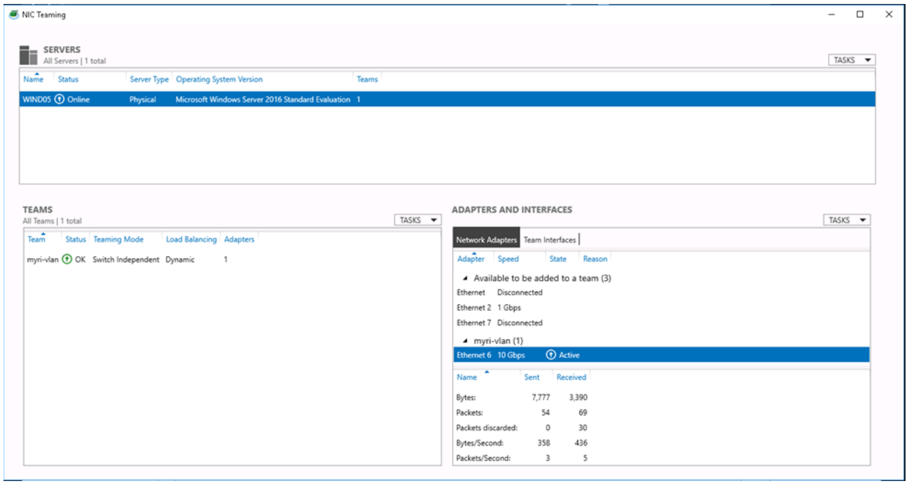
Teaming mode: Switch Independent

Load balancing mode: Dynamic

Standby adapter: None (all adapters Active)

Primary team interface: myri-vlan - VLAN 120

5. Click **OK** to create the team.
6. Verify the new VLAN/team is displayed in the Teams list.



NIC Teaming

SERVERS

All Servers | 1 total

Name	Status	Server Type	Operating System Version	Teams
WIND05	Online	Physical	Microsoft Windows Server 2016 Standard Evaluation	1

TEAMS

All Teams | 1 total

Team	Status	Teaming Mode	Load Balancing	Adapters
myri-vlan	OK	Switch Independent	Dynamic	1

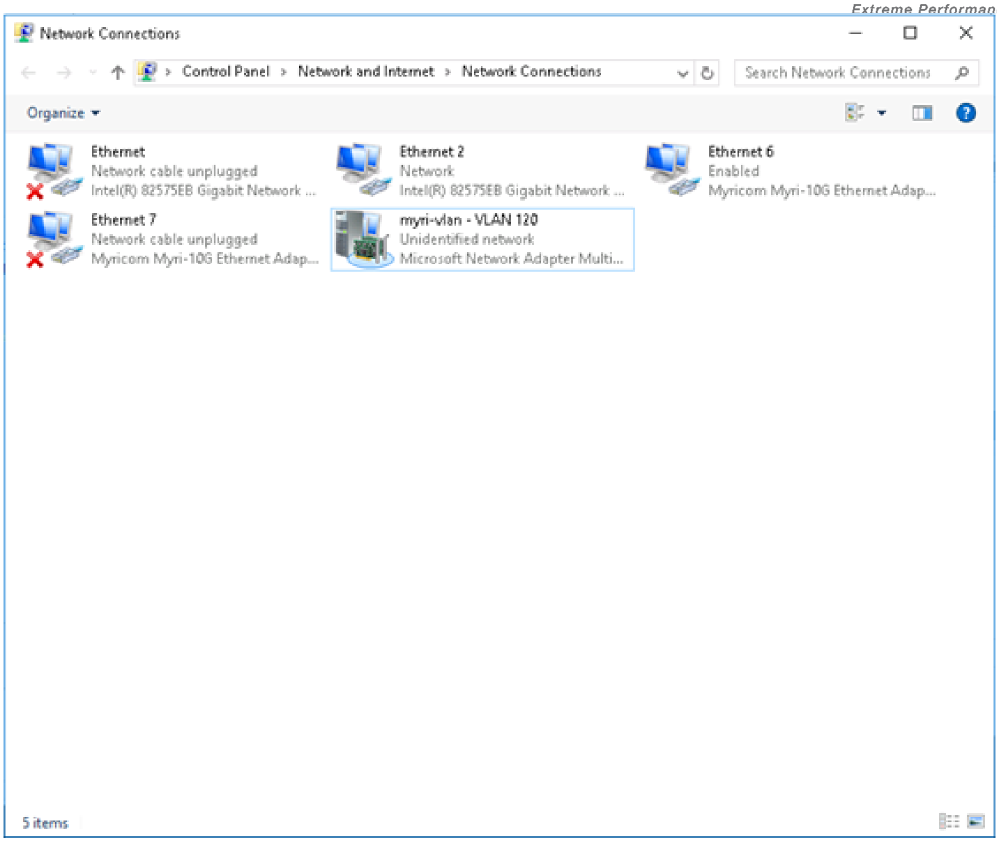
ADAPTERS AND INTERFACES

Network Adapters | Team Interfaces

Adapter	Speed	State	Reason
Available to be added to a team (3)			
Ethernet	Disconnected		
Ethernet 2	1 Gbps		
Ethernet 7	Disconnected		
myri-vlan (1)			
Ethernet 6	10 Gbps	Active	

Name	Sent	Received
Bytes:	7,777	3,390
Packets:	54	69
Packets discarded:	0	30
Bytes/Second:	358	436
Packets/Second:	3	5

7. Verify the VLAN/Team adapter is available in Network Connections.



Network Connections
Control Panel › Network and Internet › Network Connections
Search Network Connections
Organize ▾
5 items

Ethernet
Network cable unplugged
Intel(R) 82575EB Gigabit Network ...

Ethernet 2
Network
Intel(R) 82575EB Gigabit Network ...

Ethernet 6
Enabled
Myricom Myri-10G Ethernet Adap...

Ethernet 7
Network cable unplugged
Myricom Myri-10G Ethernet Adap...

myri-vlan - VLAN 120
Unidentified network
Microsoft Network Adapter Multi...

Transparent Acceleration (TA)

Manual Installation

If manual installation of Transparent Acceleration (TA) was specified during installation, an LSP will need to be enabled. Otherwise, this step is automatically performed.

To enable the TA via the LSP, open an elevated cmd prompt, and navigate to the directory where the software was installed. Enter the following command:

```
$ instlsp.exe -i -d %WINDIR%\System32\ipdbl_lsp11.dll
```

or

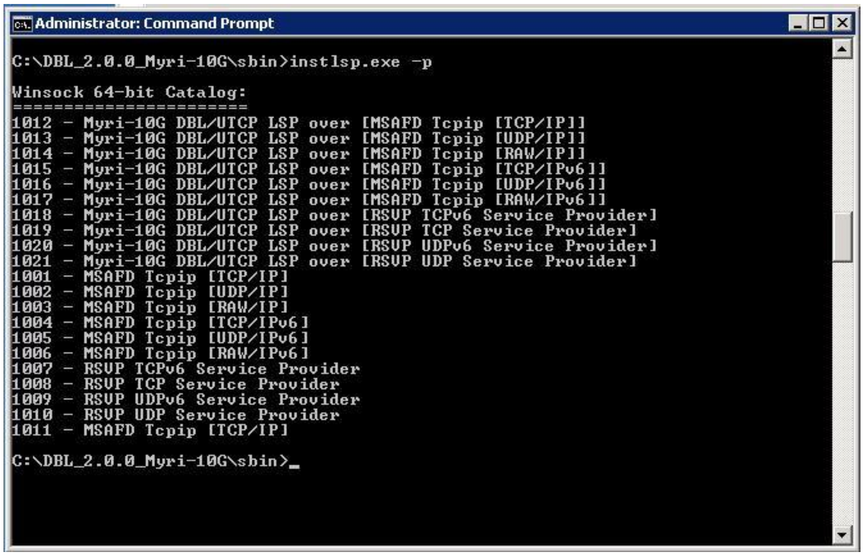
```
$ instlsp.exe -i -d %WINDIR%\System32\ipdbl_lsp22.dll
```

for Winsock 1.1 support or Winsock 2.2 support, respectively.

The following command can be used to check that the LSP is installed:

```
$ instlsp.exe -p
```

The system should display something like the output below. Look for the Myri-10G entries for DBL.



```
C:\DBL_2.0.0_Myri-10G\sbin>instlsp.exe -p

Winsock 64-bit Catalog:
=======================
1012 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [TCP/IP]]
1013 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [UDP/IP]]
1014 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [RAW/IP]]
1015 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [TCP/IPv6]]
1016 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [UDP/IPv6]]
1017 - Myri-10G DBL/UTCP LSP over [MSAFD Tcpip [RAW/IPv6]]
1018 - Myri-10G DBL/UTCP LSP over [RSVP TCPv6 Service Provider]
1019 - Myri-10G DBL/UTCP LSP over [RSVP TCP Service Provider]
1020 - Myri-10G DBL/UTCP LSP over [RSVP UDPv6 Service Provider]
1021 - Myri-10G DBL/UTCP LSP over [RSVP UDP Service Provider]
1001 - MSAFD Tcpip [TCP/IP]
1002 - MSAFD Tcpip [UDP/IP]
1003 - MSAFD Tcpip [RAW/IP]
1004 - MSAFD Tcpip [TCP/IPv6]
1005 - MSAFD Tcpip [UDP/IPv6]
1006 - MSAFD Tcpip [RAW/IPv6]
1007 - RSVP TCPv6 Service Provider
1008 - RSVP TCP Service Provider
1009 - RSVP UDPv6 Service Provider
1010 - RSVP UDP Service Provider
1011 - MSAFD Tcpip [TCP/IP]

C:\DBL_2.0.0_Myri-10G\sbin>_
```

Transparent Acceleration Usage

To enable TA for a specific process, simply use the `dblrun.exe` command which contains several parameters to steer the acceleration. Most notably, it will pass the `DBL_IP_ACCELERATION=1` environment variable to the specified application. A manual setting is not recommended, but if the user wishes to do so, the following environment variables must be set.

Windows Environment Variables

To use TA manually, without dblrun.exe, the `DBL_IP_ACCELERATION` environment variable must be set to 1.

Optional environment variables are:

```
DBL_IP_ADDRESS=w.x.y.z // in UDP mode if an INADDR_ANY address is
used, the TA will use this IP address to bind to an IP:PORT (TCP
does support INADDR_ANY)

DBL_IP_VERBOSE=1 // gives some additional information to stdout.
e.g. which sockets are accelerated

DBL_CPU_AFFINITYMASK=mask // mask for cpu binding

DBL_ENABLE_TCP=0 //don't accelerate TCP

DBL_ENABLE_UDP=0 // don't accelerate UDP

DBL_IP_LOGFILE=string // path to file for tracing information

DBL_USE_BSD_SHIM=1 // to use the BSD stack for UDP to
overcome the 32-endpoint limitation (DBL API)
```

Timestamping Support (Optional)

Timestamping support in DBL API mode is enabled by default. However, for performance reasons, in DBL 2.1 and later, timestamping support with DBL TA mode is disabled by default. Note that this setting is a change in the default behavior. For implementation details of how this change affects DBL API and TA customers please read **Section 5.2.12 Network Adapter Timestamps**.

In the DBL release with SYNC adapter support, three extra parameters were added to the DBL module:

- **myri_timesource**

  By default, the DBL software assumes that the 10G-PCIE2-8C2-2S-SYNC adapter is connected to a timecode generator (**myri_timesource=1**) and that timesource (hardware) timestamping mode is desired. If the SYNC adapter is not connected to a timecode generator, and host timestamping will be utilized, the **myri_dbl** driver must be loaded with the **myri_timesource=0** load-time option. Otherwise, DBL will return zero for the timestamps.

  On Windows, the instructions to set **myri_timesource=0** and thus switch to using host timestamps on the SYNC adapter are as follows.

  Create the following registry key (followed by a reboot):

```
   REG ADD HKLM\SYSTEM\CurrentControlSet\services\dbl /v myri_timesource
/t
   REG_DWORD /d 0
```

  The Windows EventViewer will then show:

```
myri_dbl INFO: Timesource function disabled by registry entry
```

  On non-SYNC adapters, the default is **myri_timesource=0** and host timestamping.

- **myri_timesource_loss_mode**

  Set **myri_timesource_loss_mode=1** for free-running (default), and set value to 0 for zeroing timestamps.

- **myri_timesource_pps**

  Set **myri_timesource_pps=0** to disable the exporting of a pps interface that can be used by ntp.

For additional details on timestamping, please read **Section 5.2.12 Network Adapter Timestamps**.

In DBL 2.1 and later, timestamping with DBL TA mode is disabled by default. To utilize timestamping with DBL TA mode, please read **Section 3.2: Customizing Transparent Acceleration with dbl.conf.** (Note that this setting is a change in the default behavior. In DBL 2.0, timestamping was enabled by default in DBL TA mode.)

Arista Timestamping Support (Optional)

Timestamping support with Arista 7150 series 10GbE switches is available in DBL v3 and later. DBL supports the alternate timestamping provided by the Arista 7150 Switch EOS. For more details, please read:

https://eos.arista.com/timestamping-on-the-7150-series/

The technique supported by DBL is the append (before-fcs in the Arista terminology), which will add 4 bytes at the end of each packet, as well as extra packets to describe the time references for computations to be made (keyframes in the Arista terminology).

Arista timestamping support is enabled by default (**MYRI_ARISTA_ENABLE_TIMSTAMPING=1**).

- **MYRI_ARISTA_ENABLE_TIMESTAMPING = 0,1**

  MYRI_ARISTA_ENABLE_TIMESTAMPING enables the support for Arista timestamping, and is activated as soon as the process receives its first keyframe. The Arista switch does not guarantee delivery of keyframes when it is under heavy load. It is possible for the arrival of the first keyframe which enables time stamping to begin to be delayed. Also, under these conditions, the arrival interval of keyframes may be greater than the timestamp wrap time of approximately 6.1 seconds. In this case DBL will continue to approximate timestamps until a new keyframe arrives. Therefore, a lapse of time may be needed to receive the first keyframe. By default, **MYRI_ARISTA_ENABLE_TIMESTAMPING=1.**

  On Windows, if you are using 10G-PCIE2-8C2-2S-SYNC adapters, hardware timestamping must be disabled. Create the following registry key (followed by a reboot):

```
REG ADD HKLM\SYSTEM\CurrentControlSet\services\dbl /v myri_timesource
/t
REG_DWORD /d 0
```

  To verify that Arista timestamping has been enabled, the Windows EventViewer will then show:

```
myri_dbl: INFO: ** Arista timestamping [on] supported modes: append (before-fcs)
    * features: keyframe check min.rate [on] kf check ptp [on] kf kernel dup [off]
```

- **MYRI_ARISTA_PARAM_KF_DST_IP = <IP ADDRESS>**

  MYRI_ARISTA_PARAM_KF_DST_IP must be set to the IP address of the keyframe generator (the Arista switch). By default, the address is set to 1.1.1.0. For more details, please consult

  http://www.aristanetworks.com/docs/Manuals/ConfigGuide.pdf

The following three environment variables are optional for Arista timestamping support.

1. **MYRI_ARISTA_PARAM_KF_CHECK_PTP = 0,1**

   MYRI_ARISTA_PARAM_KF_CHECK_PTP is an optional environment variable to check if the PTP synchronization is working on the switch. By default, **MYRI_ARISTA_PARAM_KF_CHECK_PTP=1**.

2. **MYRI_ARISTA_PARAM_KF_CHECK_RATE = 0,1**

   MYRI_ARISTA_PARAM_KF_CHECK_RATE is an optional environment variable to check if the keyframes sent are sufficient to provide accuracy. By default, **MYRI_ARISTA_PARAM_KF_CHECK_RATE=1**.

3. **MYRI_ARISTA_PARAM_KF_KERNDUP = 0, 1**

   By default, **MYRI_ARISTA_PARAM_KF_KERNDUP=0**. If enabled, this feature will duplicate keyframe packets to the host stack.

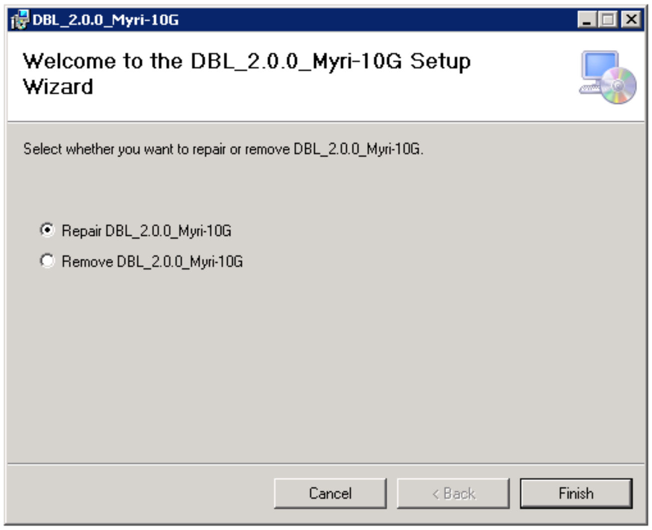
Uninstalling on Windows

The recommended method to uninstall the DBL software on Windows is using the MSI installer.

Using the Installer

If the software was installed using the installer (.msi), running it again will include an option to uninstall. Select that option and there will be two choices: **Repair** and **Remove**. Select **Remove** and click **Finish**. When the software has been removed, an “Installation Complete” message will appear.

Alternatively, if the DBL installation has been corrupted in some manner (for example, a file was accidentally deleted), you can select the **Repair** option to reinstall the software.


DBL_2.0.0_Myri-10G

Welcome to the DBL_2.0.0_Myri-10G Setup Wizard

Select whether you want to repair or remove DBL_2.0.0_Myri-10G.

● Repair DBL_2.0.0_Myri-10G

○ Remove DBL_2.0.0_Myri-10G

Manual Removal

The steps must be done in this order. This listing is a general procedure; you may not have installed all the drivers listed here.

1. **Delete all virtual adapters first and then the teaming/VLAN driver.**
   a. Open Network Connections.
   b. Right-click on the connection for "Myricom Myri-10G Ethernet Adapter" and select Properties.
   c. Select "Teaming/VLAN Driver ..."
   d. If present, click on Uninstall.
   e. Close the connection's properties dialog.
2. **Remove DBL**
   a. Open Device Manager (Start->Run->devmgmt.msc), and uninstall all Myricom Adapters.
   b. **Remove INF and PNF files.** This procedure removes all references to the driver and prevents a stale file from being used in the future.
   c. Open a command window (Start->Run->cmd).

```
cd %windir%\inf
findstr Myricom oem*.inf
```

   d. **Delete all Myricom oem inf and pnf files**. E.g., if oem6 was found with the findstr,

```
del oem6*
```

   e. **Delete sys and dat files**

```
cd %windir%\system32\drivers\
del myri_dbl.sys
```

f. Restart the machine

```
shutdown /r /t 0
```

Once the machine reboots, Windows will try to find the driver for the Myri-10G adapters. It should fail because you deleted the inf and oem files.

2.4 License Key Configuration

A separate license key will be provided by ARIA Customer Support for each purchased network adapter and related licensed software products. A one-time activation step must be performed to record the license key information in the flash memory of the network adapter. This step ensures proper functioning of all licensed features on that card and its use with related software products.

If you purchased DBL v3.1.6 licenses at the same time as your Myri-10G network adapters, the licenses will be pre-installed on the adapters and you should skip the **myri_license -f** procedure in this section of the document. If you are installing DBL on an adapter that was not purchased with the DBL license pre-installed, then you must install the license key in the system using the **myri_license -f** procedure below.

2.4.1 License Key File

A license key file consists of one record for each network adapter. The file is a simple text format that can be viewed with any common text editor. Each record consists of an encrypted key which is locked to a specific network adapter serial number, followed by a string describing the licensed features enabled for that card.

The following is a sample license key file.

```
569f-f54d-1fb3-f194:1:123456:DBL:V3 # s/n 123456
6471-6557-e163-954a:1:123456:DBL:V3.1 # s/n 123456
# limited time trial evaluation of new product
c93b-9fc0-3570-d21b:1:123456:DBL:V3:T1299100709 # s/n 123456 (expires mm/
dd/yyyy)
```

Any records with a leading “#” are ignored and can be used to record any local site comments.

2.4.2 License Key Activation Process

Normally an email will be sent from ARIA Customer Support to the customer containing the list of license keys for the products and network adapter serial numbers requested. The license records will be delimited in the email as follows.

```
...
```

```
==== start license file ====
    .
    .
    .
=== end license file ===
...
```

Simply copy the text license records between the file delimiters comments in the email and paste into a new file using a text editor. Save the file with any file name, (represented below as <license_file>) in a convenient location.

Open a terminal window as user “root” on Linux or open a command-prompt window as user “admin” on Windows systems. Now run the following command, supplying the assigned file name.

```
# /opt/dbl/sbin/myri_license -f <license_file>
```

The application **myri_license** should be present in the sbin directory of the product(s) you have previously installed (e.g., **/opt/dbl/sbin**)

This procedure will program the license keys into the network adapters installed in the host on which you run the **myri_license** command. **myri_license** must be run on each machine that contains Myri-10G Network Adapters to be licensed, but a single file may be used to hold all licenses for convenience.

To verify that the software license(s) have been successfully installed, run the following command:

```
$ /opt/dbl/bin/myri_nic_info
```

If any of the software licenses are listed as invalid, additional diagnostic information may be obtained by examining the kernel log output (dmesg) or running the command:

```
$ /opt/dbl/bin/myri_nic_info --license
```

Important note:

> The **myri_license -f <license_file>** command merges the license keys from the **<license_file>** with any existing license keys already programmed on the adapter.
>
> To determine the license key string(s) currently programmed on an adapter, you can either run the **myri_license** command without any arguments, or the **myri_info** command without any arguments, and it will return the license key.
>
> If you accidentally overwrite a license key on an adapter, it is possible to obtain the missing license key string from **ARIA Technical Support.** Please include the six-digit serial number (or the MAC address) for the network adapter in your inquiry.

2.4.3 License Key Upgrade Process

When licenses have been purchased for new features to be used with existing network adapters, it will be necessary to rerun **myri_license** with an updated license file on the host system containing the network adapters.

Just add the new license key records to your local license file using a text editor and rerun the **myri_license** command to activate the new features.

2.4.4 Multiple License Keys

If you have purchased both DBL and Sniffer10G licenses for the same Myri-10G network adapter(s), both licenses can exist concurrently on the adapter and/or in the myri_license license file. However, only one driver can be loaded at a given time on a specific network port on the adapter. The driver load-time option **myri_bus** or **myri_mac** can be used to selectively load DBL or Sniffer10G (or Myri10GE) on a specific network port on the adapter. Contact ARIA Technical Support for details.

Before attempting to run DBL applications, it should be verified that the DBL driver is loaded on the specific adapter's network port. To verify that the DBL driver is loaded, run the **/opt/dbl/sbin/myri_info** command and examine the **Running MCP** section of the output for text that references **dbl-<version>**. Otherwise, if the adapter has two ports or there are multiple adapters installed on the same hosts, there may be confusion with the wrong driver loaded on the wrong adapter port at a given time.

2.4.5 License Key Portability

Once a network adapter has been activated using the appropriate license key, it is permanently enabled with those features. The card may be moved to other host systems if desired and will continue to operate.

2.4.6 License key maintenance for defective card (RMA Process)

In the rare event of failure of a network adapter, contact ARIA Technical Support to obtain a replacement card using the Return Merchandise Authorization (RMA) process. A new license key record will be provided with the replacement network adapter.

Just add the new license key record(s) to your local license_file using a text editor and rerun the myri_license command to activate the replacement network adapter.

Note that it is not necessary to delete the original license key record in the file as it will be silently ignored.

2.5 Running basic tests to verify DBL installation

DBL software installation is normally completed without difficulty on the first attempt. However, it is recommended that some simple tests be performed to verify DBL operation on your systems and provide basic familiarization with normal operation.

2.5.1 Test / Example Programs

On Linux DBL v3.1.6, test / example programs are available from **/opt/dbl/bin/tests** of the install directory in binary form and in **share/examples** in source form.

On Windows DBL v3.1.6, these test / example programs are available in binary and source form from the **[INSTALL_DIR]\bin\tests** directory.

> **Note**: The undocumented example programs in this directory are provided as additional coding examples, and no instructions for their use are provided.

The following test / example programs demonstrate different aspects of the DBL API and how to use its features.

Many of the DBL test/example programs require a **-p <port>** as a command-line argument. The port number refers to the physical PHY connector port on the network adapter. The ports are enumerated starting at 0. If the network adapter has two physical ports, the port closest to the PCI connector is assigned port 0 and the other port is assigned port 1. Port 0 also corresponds to the lower MAC address of a two-port adapter. If there are multiple adapters installed in the server, the port numbers for the successive adapters are assigned according to the order in which the adapters are detected by the BIOS. Refer to the output of **myri_nic_info -B** to determine the port numbering.

Invoke the test program with the command-line option **-?** to obtain a usage summary of the command-line options.

- **dbl_simple_send.c:**

  Example program using the DBL API function dbl_send(). Client counterpart to **dbl_simple_recv**.

  Example (client):

```
# ./dbl_simple_send -h 192.168.1.1 -p 3333 -l 192.168.1.2 -P 3333 -i 10000 -S 100
```

  Usage: `./dbl_simple_send [options]`

  Command-line [options]:

```
[ -?, --help ]
```

```
    Prints help information and usage.
[ -h, --remote <remote_address> ]
    Specify remote address.
    Default: DBL_REMOTE.
[ -p, --port <port> ]
    Use remote <port> for sending/receiving(1024-65535).
    Default: 3333.
[ -l, --local <local_address> ]
    Specify address of local product interface.
    Default: DBL_LOCAL.
[ -P, --localport <port> ]
    Use local <port> (1024-65535).
    Default: 3333.
[ -i, --iterations <iterations> ]
    Set number of send/receive iterations. Must be
    set equally on client and server. Default:
    DBL_ITERATIONS, or 10000 if undefined.
[ -S, --size <size> ]
    Set size of packet in bytes to send/receive (min).
    Default: 1.
```

- **dbl_simple_recv.c:**

  Example program using the DBL API function dbl_recv(). Server counterpart to **dbl_simple_send**.

  Example (on server):

  ```
  # ./dbl_simple_recv -l 192.168.1.1 -p 3333 -i 10000
  ```

  Usage: `./dbl_simple_recv [options]`

  Command-line [options]:

  ```
  [ -?, --help]
      Prints help information and usage.
  [ -p, --port <port> ]
      Use remote <port> for sending/receiving (1024-65535).
      Default: 3333.
  [ -j, --mcast <mcast_address> ]
      Specify the multicast address to join.
      Default: DBL_MCAST.
  [ -l, --local <local_address> ]
      Specify address of local product interface.
      Default: DBL_LOCAL.
  ```

```
[ -i, --iterations <iterations> ]
   Set number of send/receive iterations. Must be
   set equally on client and server. Default:
   DBL_ITERATIONS, or 10000 if undefined.
[ -C, -recvq <bytes> ]
   Modify size of receive request.
[ -M, --recvmode <recv_mode> ]
   Select receive mode.
           0 - Non-blocking (busy-poll) [default]
           1 - DBL-internal blocking
           2 - Poll using DBL file descriptor
           3 - Non-blocking via EAGAIN
           4 - (Windows only) Receive non-blocking with event signal
   If no receive modes are set, DBL will internally busy-poll.
```

- **dbl_pingpong.c:**

Measures UDP pingpong latency using the DBL API. When the client and server machines are connected back-to-back (switchless), the expected half-round trip latency is 3 to 4 microseconds.

Example:

On the “server”:

```
# ./dbl_pingpong -s -l 192.168.1.1 -p 3333 -i 10000
```

On the “client”:

```
# ./dbl_pingpong -h 192.168.1.1 -l 192.168.1.2 -p 3333 -i 10000
```

Usage: `./dbl_pingpong [options]`

Command-line [options]:

```
[ -?, --help]
   Prints help information and usage.
[ -s, --server ]
   Run as server.
[ -h, --remote <remote_address> ]
   Specify remote address.
   Default: DBL_REMOTE.
[ -p, --port <port> ]
   Use [remote] <port> for sending/receiving (1024-65535).
   Default: 3333.
[ -l, --local <local_address> ]
   Specify address of local product interface.
   Default: DBL_LOCAL.
```

```
[ -i, --iterations <iterations> ]
   Set number of send/receive iterations. Must be
   set equally on client and server. Default:
   DBL_ITERATIONS, or 10000 if undefined.
[ -t, --timeout <seconds> ]
   Set time to run before terminating application.
[ -S, --size <start_size> ]
   Set size of packet to send/receive (min).
   Default: 1.
[ -E, --endsize <end_size> ]
   Set size of packet to send/receive (max).
   Default: 2048.
[ -b, --blocking ]
   Use blocking receive.
[ -d, --dup ]
   Duplicate packets to kernel.
```

- **dbl_perf_test.c:**

Measures the point-to-point RTT latency of DBL. By default, this program also runs a test over multicast UDP. If specified, a short parallel I/O test will be run as well.

Example:

On the “server”:

```
#./dbl_perf_test -s -l 192.168.1.1 -P 3333 -i 10000 -V
```

On the “client”:

```
# ./dbl_perf_test -h 192.168.1.1 -p 3333 -l 192.168.1.2 -i 10000 -V
```

Usage: `./dbl_perf_test [options]`

Command-line [options]:

```
[ -?, --help ]
   Prints help information and usage.
[ -s, --server ]
   Run as server.
[ -h, --remote <remote_address> ]
   Specify remote address.
   Default: DBL_REMOTE.
[ -p, --port <port> ]
   Use remote <port> (1024-65535).
   Default: 3333.
[ -l, --local <local_address> ]
   Specify address of local product interface.
   Default: DBL_LOCAL.
[ -P, --localhost <port> ]
   Use local <port> (1024-65535).
   Defaults to remote port or 3333 if not specified.
[ -i, --iterations <iterations> ]
   Set number of send/receive iterations. Must be
   set equally on client and server. Default:
   DBL_ITERATIONS, or 10000 if undefined.
[ -V, --verbose ]
   Print verbose output for debugging.
[ -Y, --piotest ]
   Run a short PIO performance test.
```

Note: If the -i (or -V) command-line argument is used, it must be specified on both the server and the client.

- **dbltcp_pingpong.c**

Measures TCP pingpong latency using the DBL API. When the client and server machines are connected back-to-back (switchless), the expected half-round trip latency is 4 to 5 microseconds.

Example:

On the “server”:

```
$ dbltcp_pingpong -s -l 10.0.0.9
Test will be using Type DBL_TCP, Proto DBL_MYRI
Starting DBL test with socket type [DBL_TCP], and proto [DBL_MYRI]
Setting up to listen()
Setting up to accept()
Got connection from 10.0.0.10:3333
Client went away
```

On the “client”:

```
$ dbltcp_pingpong -h 10.0.0.9 -l 10.0.0.10 -E 2
Test will be using Type DBL_TCP, Proto DBL_MYRI
Starting DBL test with socket type [DBL_TCP], and proto [DBL_MYRI]
elapsed = 91712 us, 4.59 per HRT, size 1
```

Usage: `./dbltcp_pingpong [options]`

Command-line [options]:

```
[ -? ]
   Prints help information and usage.
[ -s ]
   Run as server.
[ -h <remote_address> ]
   Specify remote address. Default: DBL_REMOTE.
```

```
[ -p <port> ]

   Use <port> for sending/receiving (1024-65535). Default: 3333.

[ -l <local_address> ]

   Specify address of local interface. -a may also be used. Default:
   DBL_LOCAL.

[ -i <iterations> ]

   Set number of iterations. Must be set equally on client and server.
   Default: DBL_ITERATIONS, or 10000 if undefined.

[ -t <seconds> ]

   Terminate application after <seconds>.

[ -S <start_size> ]

   Set size of smallest packet to send/receive. Default: 1.

[ -E <end_size> ]

   Set size of largest packet to send/receive. Default: 4000.
```

In addition, the env var DBL_ADD_MIX_CHANNELS (e.g export DBL_ADD_MIX_CHANNELS=5) will demonstrate the mix of DBL and TCP channels (required on both ends) on a device.

- **tcp_pingpong.c**:

Simple point-to-point pingpong latency test over TCP. If specified, the test may be run over UDP instead. Latency results may be improved if the test is run in IP acceleration mode with the dblrun script.

Example:

On the “server”:

```
# ./tcp_pingpong -s -l 192.168.1.1 -p 3333 -i 10000
```

On the “client”:

```
# ./tcp_pingpong -h 192.168.1.1 -l 192.168.1.2 -p 3333 -i 10000
```

Usage: `./tcp_pingpong [options]`

Command-line [options]:

```
[ -?, --help ]

   Prints help information and usage.

[ -s, --server ]

   Run as server.

[ -h, --remote <remote_address> ]

   Specify remote address.

   Default: DBL_REMOTE.

[ -p, --port <port> ]

   Use [remote] port for sending/receiving (1024-65535).

   Default: 3333.

[ -l, --local <local_address> ]
```

```
   Specify address of local product interface.
   Default: DBL_LOCAL.
[ -i, --iterations <iterations> ]
   Set number of send/receive iterations. Must be
   set equally on client and server. Default:
   DBL_ITERATIONS, or 10000 if undefined.
[ -w, --warmup <warmup_count> ]
   Set number of warmup sends to do from client before starting main
   loop.
   Default: 10000
[ -g, --delay <usecs> ]
   Set delay.
[ -t, --timeout <seconds> ]
   Set time to run before terminating application.
[ -S, --size <start_size> ]
   Set size of packet to send/receive (min).
   Default: 1.
[ -E, -endsize <end_size> ]
   Set size of packet to send/receive (max).
   Default: 2048.
[ -u, --udp ]
   Set socket type to UDP.
   Must be set on both client and server.
[ -6, --ipv6 ]
   Set socket family to IPv6.
[ -f, --fork ]
   Fork the process after accept.
[ -z, --affinity <value> ]
   Set process affinity using bitmask.
[ -G, --overlap <overlap_mode> ]
   Select overlap mode
   0 - Blocking [default]
   1 - Select
   2 - Poll
   3 - Non-blocking (FIONBIO)
   4- (Windows only) Overlap/WSAGetOverlappedResult
   5- (Windows only) IOCP WSARecvs
   6 - (Windows only) WSAEventSelect
[ -X ]
   Reports send() cost statistics.
```

2.5.2 Tool Programs

The DBL distribution also provides diagnostic tool programs, in **sbin/** and **bin/**, as documented below. Since the **/sbin/** tools must be run as root, you must either add **/opt/dbl/sbin/** to your **PATH**, or execute the command using the full path (e.g., **/opt/dbl/sbin/myri_license**). The majority of these tools are only needed for obtaining diagnostic information for error reporting. There are undocumented tool programs in these directories which should only be run as instructed by ARIA Technical Support.

- **sbin/myri_bug_report:** (Linux only) A Linux diagnostic/ bug report script which can be sent to ARIA Technical Support. For example:

  ```
  # /opt/dbl/sbin/myri_bug_report > output.txt
  ```

  Please read **Chapter 5: Troubleshooting** for details.

- **sbin\myri_dmesg.ps1**: (Windows only) A diagnostic powershell script for ARIA Technical Support. Please read **Chapter 5: Troubleshooting** for details.

- **sbin/myri_info:** Provides hardware information (e.g., serial number, mac address) for the adapter, as well as the version of firmware running on the adapter.

  Usage:

  ```
  # /opt/dbl/sbin/myri_info [options]
  ```

  Command-line [options]:

  ```
  [ -b <board_num> ]
     Only print info about card instance <board_num>.
  [ -v ]
     Verbose.
  [ -vv ]
     Very verbose.
  ```

- **sbin/myri_license:** Programs software license keys into the network adapter(s) installed in the host. Please read **Section 2.4: License Key Configuration** for details. Note that myri_license specified with no arguments reports licenses currently loaded.

  Usage:

  ```
  # /opt/dbl/sbin/myri_license [options]
  ```

  Command-line [options]:

  ```
  [ -b <unit> ]
     Is used to specify an adapter for reporting. Default is all
     adapters if -b is not specified.
  [ -c ]
     Clears/removes license(s) on the adapter.
  ```

```
[ -f <license_file> ]
   Load licenses from <license_file> into all matching adapters.
```

- **sbin\team_config:** (Windows only) Command-line tool to install the VLAN driver and configure the VLAN adapters. Please read **Section 2.3.3.5** for more details. For a full listing of all command-line arguments, specify **team_config.exe -h**.
- **bin/dbltool**: Provides netstat-style results for DBL TA mode. It will show counters for packets dropped by the stack. Since DBL is kernel-bypass, packets dropped by DBL will not appear there, but will instead be visible in myri_counters output.

  Usage:

  ```
  $ /opt/dbl/bin/dbltool -p <pid> <cmd> [cmd options ...]
  ```

  Command-line [options]:

  ```
  [ -p <pid> ]
     pid of the process you want to examine.
  <cmd>
     The command you would like to send. The supported commands are
     mtcp_config, mtcp_stats, and mtcp_routes.
  ```

- **bin/myri_bandwidth**: Shows the instantaneous bandwidth of data going through a given network adapter, which is mostly useful when displayed at an interval (-i option).

  Usage:

  ```
  $ /opt/dbl/bin/myri_bandwidth [options]
  ```

  Command-line [options]:

  ```
  [ -b <board_num> ]
     Board number or MAC address. Default: 0.
  [ -t <secs> ]
     Time interval (seconds). Default: 1.
  [ -k ]
     Keep running.
  [ -p ]
     Display counters for each port on multi-port adapters.
  [ -a ]

     Also display aggregated counters for each iteration.
  [ -A ]
     Only display aggregated counters for each iteration.
  [ -h ]
     Help.
  ```

- **bin/myri_counters**: Generates output for low-level network adapter counters. For details, please read **Appendix: DBL Counters**.

  Usage:

  ```
  $ /opt/dbl/bin/myri_counters [options]
  ```

  Command-line [options]:

  ```
  [ -p <port_num> ]
     Port number or MAC address. Default: 0.
  [ -c ]
     Clear/reset the counters.
  [ -q ]
     Quiet: display only non-zero counters.
  [ -i ]
     Display host interrupt counters.
  [ -x ]
     Expert: display all counters.
  [ -h ]
     Help.
  ```

- **bin/myri_dmabench**: .Verifies the PCI DMA read and DMA write bandwidth for the PCIe slot for which the adapter is installed. Please read **Chapter 5: Troubleshooting** for details.

  Usage:

  ```
  $ /opt/dbl/bin/myri_dmabench [options]
  ```

  Command-line [options]:

  ```
  [ -b <board_num> ]
     Board number or MAC address. Default: 0.
  [ -s <dma_size> ]
     DMA size. Default: 4096. Must be a power of two between 1 and 4096.
  [ -r ]
     Test only read (send) DMA.
  [ -w ]
     Test only write (recv) DMA.
  [ -i <iters> ]
     Number of iterations. Default: 64.
  [ -a ]
     Test all (power of 2) DMA sizes.
  ```

- **bin/myri_endpoint_info**: Shows which processes consume some adapter-level resources known as endpoints. It can be useful to know which process IDs are using DBL.

Usage:

```
$ /opt/dbl/bin/myri_endpoint_info [options]
```

Command-line [options]:

```
[ -b <board_num> ]
   Board number or MAC address. Default: 0.
[ -h ]
   Help.
```

- **bin/myri_nic_info:** Provides diagnostic information on the number of adapters installed, which driver is loaded, and the status of the software license(s) for each adapter.

  Usage:

  ```
  $ /opt/dbl/bin/myri_nic_info [options]
  ```

  Command-line [options]:

  ```
  [ -B ]
     Display board numbers.
  [ -m ]
     Comma separated output (for machine parsing).
  [ -a ]
     Print for all known adapters.
  [ --license]
     Print software licensing status for all known adapters.
  ```

- **bin/myri_port_failover**: A utility that can be used to control the failover policy on the 10G-PCIE-8B-2* adapters and the 10G-PCIE-8B-2I and 10G-PCIE-8B2-4I BladeCenter adapters. Please read **Chapter 5: Troubleshooting** for details.

  Usage:

  ```
  $ /opt/dbl/bin/myri_port_failover [ -b<unit> ] [ -0 | -1 | -s | -r<0|1> ]
  ```

  Command-line [options]:

  ```
  [ -b <unit> ]
     Uses NIC/Controller rank #unit. Default: 0.
  [ -0 ]
     Set P0 as primary link for failover purposes (P1 is backup only).
  [ -1 ]
     Set P1 as primary link for failover purposes (P0 is backup only).
  [ -s ]
     Symmetric failover (only switch link if current port is down).
  [ -r <0|1> ]
  ```

```
Disable/enable sending a rarp pkt on failover (to update switch
tables)
```

2.5.3 Running DBL API tests on Linux

Test / example programs are available from **/opt/dbl/bin/tests** of the install directory in binary form and in **share/examples** in source form.

The file **dbl_pingpong** is a simple point-to-point test to measure the UDP pingpong latency using the DBL API.

The file **dbltcp_pingpong** is a simple point-to-point test to measure the TCP pingpong latency using the DBL API.

The file **tcp_pingpong** is a simple point-to-point test to measure the UDP (or TCP) pingpong latency using standard IP.

The following test will verify the correct installation of the DBL software and the network adapter hardware connectivity, as well as demonstrate the expected low-latency UDP and TCP performance of the DBL API.

This example assumes that IP addresses have been assigned on two hosts, with **10.0.0.1** for the “client” and **10.0.0.2** for the “server”.

Note:

- A network configuration with IP addresses on the same subnet is **not** supported.
- The DBL API semantics do not support loopback communication. Thus, for DBL API mode, the sender port/adapter and receiver port/adapter must be located on different Myri-10G network adapters.

On both systems insure that **/opt/dbl/bin/tests/** has been added to the current path variable.

Simple (Standard IP) pingpong latency test for UDP

On the “server” host system, open a command window and type the following command.

```
$ tcp_pingpong -u -s -l 10.0.0.2
```

On the “client” host system, open a command window and type the following command.

```
$ tcp_pingpong -u -l 10.0.0.1 -h 10.0.0.2
```

Successful execution of the test will report a sequence of communications between the two machines. Note the actual delays reported on your systems to compare with the next step. Kill the running tasks when done.

DBL API pingpong latency test for UDP

Now repeat the UDP latency test using the DBL API.

On the “server” host system, open a command window and type the following command.

```
$ dbl_pingpong -s -l 10.0.0.2
```

On the “client” host system, open a command window and type the following command.

```
$ dbl_pingpong -l 10.0.0.1 -h 10.0.0.2
```

The resulting output should show accelerated UDP traffic with much smaller latency results than conventional Ethernet. Some variability in individual “ping” numbers is normal. Kill the running tasks when done.

When the two machines are connected back-to-back (switchless), the expected UDP pingpong latency is 3-4 microseconds. IThe speed of the CPU/ host is also a very important factor in the latency performance of the host. f the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

Simple (Standard IP) pingpong latency test for TCP

On the “server” host system, open a command window and type the following command.

```
$ tcp_pingpong -s -l 10.0.0.2
```

On the “client” host system, open a command window and type the following command.

```
$ tcp_pingpong -l 10.0.0.1 -h 10.0.0.2
```

Successful execution of the test will report a sequence of communications between the two machines. Note the actual delays reported on your systems to compare with the next step. Kill the running tasks when done.

DBL API pingpong latency test for TCP

Now repeat the TCP latency test using the DBL API.

On the “server” host system, open a command window and type the following command.

```
$ dbltcp_pingpong -s -l 10.0.0.9
Test will be using Type DBL_TCP, Proto DBL_MYRI
Starting DBL test with socket type [DBL_TCP], and proto [DBL_MYRI]
Setting up to listen()
Setting up to accept()
Got connection from 10.0.0.10:3333
Client went away
```

On the “client” host system, open a command window and type the following command.

```
$ dbltcp_pingpong -h 10.0.0.9 -l 10.0.0.10 -E 2
Test will be using Type DBL_TCP, Proto DBL_MYRI
Starting DBL test with socket type [DBL_TCP], and proto [DBL_MYRI]
elapsed = 91712 us, 4.59 per HRT, size 1
```

The resulting output should show accelerated TCP traffic with much smaller latency results than conventional Ethernet. Some variability in individual “ping” numbers is normal. Kill the running tasks when done.

When the two machines are connected back-to-back (switchless), the expected TCP pingpong latency is 4-5 microseconds. The speed of the CPU/ host is a very important factor in the latency performance of the host. For a fast host, it may be possible to achieve 3-4 microseconds TCP pingpong latency. If the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

3 Transparent Sockets Acceleration Software

After the DBL software has been installed, TCP and UDP acceleration can be invoked by following the instructions in **Section 3.1: TCP and UDP Acceleration using DBL**. **Sections 3.2** and **3.3** provide DBL TA implementation details for the Linux and Windows operating systems, respectively. Step-by-step testing instructions are provided in **Section 3.4: Running DBL TA tests on Linux**. If the dblrun script cannot be used, manual configuration instructions are detailed in **Section 3.5: Manual Configuration of Transparent Acceleration (TA)**.

3.1 TCP and UDP Acceleration using DBL

After the DBL software has been installed, TCP and UDP acceleration can be invoked on an existing program binary (no coding changes or recompilation required) by running the program prefixed with "**dblrun**" on the command line.

```
dblrun [options] executable arguments

OPTIONS:
     -i ip_address   DBL IP Address of Accelerating Device for UDP TA
                     to support autobind [myri0]
     -l libdir       Full path to preload directory [/opt/dbl/lib]
     -f logfile      Full path to log file [off, e.g. /tmp/log.txt]
     -d              Bypass acceleration under ld_preload [off]
     -v              Verbose startup [off]
     -u              UDP acceleration only [off]
     -t              TCP acceleration only [off]
     -c cpu_number   Process Affinity mask [off]
     -C <path_to>/dbl.conf Use non-default dbl.conf for application
     -b              Multiplexes UDP sockets using the BSD stack
     -w              (Windows only) Forward cntrl-x events to target
                     application
     -?              This usage information
```

To determine if the DBL TA mode is being activated when running the application, examine the output of **dblrun -v <appname>**.

For example, when using Windows DBL, you will see the following output (for TCP):

```
WSPSocket: Enabled fd 276 for TCP TA
```

where TA stands for Transparent Acceleration and means that the fd is accelerated.

If you are using Windows DBL TA mode and you do not see the “TCP TA” text in the output, verify that the LSP is installed by running the command

```
$ instlsp.exe -p
```

If the LSP is not installed, follow the instructions in **Section 3.4** and then it will be possible to accelerate applications with **dblrun.exe <appname>.**

The DBL stack for UDP acceleration only supports up to 32 DBL endpoints. If you have more sockets under DBL TA (Transparent Acceleration), then use the

```
$ dblrun -b myapp.exe
```

option to multiplex those (UDP) sockets using the general TA BSD stack. This suggestion also applies if several applications are accelerated concurrently and few endpoints are available (see the output of **myri_endpoint_info**).

The **dblrun -i** option only applies for UDP traffic in case TA mode uses the DBL API (default). It will not affect any TCP traffic nor the UDP traffic when run with **-b**. Its only purpose is to offer support for bind to INADDR_ANY. In this case a DBL interface can be specified on which traffic is expected.

If the sockets application fails to run successfully in Transparent Acceleration (TA) mode, please follow the troubleshooting procedures in **Section 5.2.5 TA Failure**.

Important Notes for Testing:

- DBL TA mode is restricted to UDP and TCP. Other protocols such as SOCK_RAW are **not** accelerated.
- Loopback communication on the loopback address (e.g IP 127.0.0.1) is only supported in DBL TA mode (dblrun), starting with DBL version 2.1.0.51507.
- When using loopback with DBL TA mode (dblrun), a socket is demoted/ unaccelerated and the traffic is passed over internal OS dependent paths which may be faster than sending packets via DBL.
- A network configuration with IP addresses on the same subnet is **not** supported.

3.2 Customizing Transparent Acceleration using dbl.conf

The **dbl.conf** file, located in the DBL_ROOT directory in the DBL distribution, customizes the DBL TA (Transparent Acceleration) and DBL TCP API configuration. By default, when using DBL in TA (Transparent Acceleration) mode, all TCP and UDP ports are accelerated, delayed acks are disabled, and timestamping on adapters is disabled. This behavior may be changed by modifying the **dbl.conf** configuration file as described below:

- **restrict which TCP and UDP ports are accelerated**

```
dbl.tcp.ports_include=1-65535
dbl.udp.ports_exclude=1-1023
```

Users can restrict which sockets are accelerated by using the include/exclude notation in the **dbl.conf** file. On TCP, excluded ports are all non-autobound ports: destination port in a client connect() and source port in a server listen(). On UDP, excluded ports are only honored for server binds().

- **increase the maximum number of UDP sockets which can be accelerated**

  ```
  dbl.udp.many_sock=0
  ```

  By default, **dbl.udp.many_sock=0**. When this value is set to 0, a maximum of 32 accelerated UDP sockets are available per adapter system-wide. Setting dbl.udp.many_sock to 1 removes this limitation at the expense of a little latency.

- **enable delayed acks**

  ```
  net.inet.tcp.delayed_ack=1
  net.inet.tcp.delacktime=10
  ```

  By default, enable delayed ACK at 10ms.

- **assign CPU affinity masks to service threads (a specific core)**

  ```
  mtcp.thread_softclock_cpu_affinity_mask=0x1
  mtcp.thread_monitor_cpu_affinity_mask=0x1
  ```

  The affinity mask is used to prevent the service thread from eventually polluting an execution, which could happen, since the thread is sometimes scheduled by kernels. This option allows the user to pin the service thread to cores according to the best strategy for the application. The mask will not cause starvation since the work is very light (mainly related to tcp timeout checks).

  The user can selectively assign the service threads to specific cores or let them be scheduled by the OS scheduler.

  The **mtcp.thread_softclock_cpu_affinity_mask** and **mtcp.thread_monitor_cpu_affinity_mask** refer to the two service threads:

  – one very inactive 'monitor thread' used for managing the connection with our OS internal netlink,

  – one slightly more active 'clock thread', whose work it is to compute timeouts and arm the timer for connections' timeouts.

- **enable timestamping**.

  ```
  mtcp.hw_timestamping=0
  ```

  In DBL 2.1 and later, timestamping in DBL TA mode is disabled by default to minimize latency. Linux users can enable timestamping via setting **mtcp.hw_timestamping=1** in the **dbl.conf** file. This timestamping option in **dbl.conf** affects whether or not you receive a timestamp from the **SO_TIMESTAMP[NS]** feature. Windows does not support SO_TIMESTAMP[NS] at the NDIS level but we added this feature for transparent acceleration.

Therefore you can #define SO_TIMESTAMP 0x0400 to be used in setsockopt and you will be able to retrieve timestamps (similar to what Linux offers). Please note that if you are using a 10G-PCIE2-8C2-2S-SYNC adapter not connected to a timecode generator, you must also have loaded the **myri_dbl** driver with **myri_timesource=0**. Otherwise, DBL will return zero for the timestamps. For additional details, please read **Section 5.2.12 Network Adapter Timestamps**. If you need Windows DBL TA timestamping support, contact ARIA Technical Support.

- **tcp congestion management control algorithms**

  ```
  # net.inet.tcp.cc.algorithm=newreno
  ```

  Congestion control algorithms supported by DBL belong to the loss-based family of algorithms. We support the following algorithms: newreno, cubic, and htcp. By default, we use the cubic algorithm like Linux since it is well suited for latency sensitive applications.

Refer to the comments in the **dbl.conf** file for instructions on usage.

Note:

- If you would like to specify a **dbl.conf** for only a specific application, you can use the **-C <full_path_to>/dbl.conf** command-line argument to **dblrun**. In this way, the **dbl.conf** settings will only apply to one application, and not to all DBL TA applications.
- Using the **dblrun -f logfile** option, you can determine which actions are taken when DBL examines the **dbl.conf** file.
- If the **dbl.conf** file cannot be found then
  - when using **-C** command-line option, the target application will not be started.
  - when not using the **-C** command-line option, and the (default) **dbl.conf** file in the **DBL_ROOT** directory cannot be found, a warning is logged but DBL TA mode is initiated.

3.3 DBL-Specific Socket Extensions

Three DBL-specific sockets extensions are available for Linux and Windows in the DBL v3.1.6 software.

- **MSG_WARM**

  Socket applications may suffer degraded TCP send performance from cache misses if the code path has not been called recently. With DBL v3.1.6 a socket flag MSG_WARM may be used in send() function calls to reduce these TCP

cold cache symptoms. This flag allows an application to occasionally call send(s, buf, len, MSG_WARM) to cache the TCP output code lines (by not injecting any data) and avoid cache misses.

The MSG_WARM flag is defined in dbl.h.

```
dbl.h:#define         MSG_WARM                0x20000
```

See the **DBL API Documentation** for additional details.

- **SIO_GETNICTIME**

  The SIO_GETNICTIME sockets extension is available in DBL 3.1.3.52782 and later.

  On a UDP/TCP socket, an application can use the ioctl (Linux) or WSAIoctl (Windows) function call to retrieve the current NIC time. This function call can be used to associate a timestamped packet and the ongoing time it takes while the packet is traversing through the application's stages.

  A variable (called by reference) of struct dbl_ticks (dbl_ticks_t) (see dbl.h) needs to be passed in to query the current NIC time. SIO_GETNICTIME is defined in dbl.h.

  ```
  dbl.h: #define SIO_GETNICTIME 0x5465
  ```

  See also dbl_getticks() in the **DBL API Documentation**.

- **SO_TIMESTAMP**

  While Windows does not support the SO_TIMESTAMP[NS] option, the DBL TA mode introduces support for it. Since Windows headers do not know about the define, you will need to define it as follows:

  ```
  #define SO_TIMESTAMP 0x0400
  ```

  This option can be used in setsockopt() and the application will be able to retrieve timestamps (similar to what Linux offers).

  For additional details on timestamping, please read **Section 5.2.12 Network Adapter Timestamps**.

3.4 Linux DBL TA Implementation

On Linux, the socket acceleration is done via a library preload. It is the **libdbl_preload.so** library which intercepts socket function calls and maps UDP and TCP applications so that they can benefit from DBL.

The concept relies on dynamically linked libraries and will remap socket function calls. The functionality can also depend on the behavior of the installed **libc** version. Therefore, it is strongly recommended that you run the examples given in the next section to verify functionality with your systems before proceeding to more complex environments.

For DBL TA mode, the timestamp is obtained via the standard socket **SO_TIMESTAMP[NS]** option. For additional details on timestamping, please read **Section 5.2.12 Network Adapter Timestamps**.

If the sockets application fails to run successfully in Transparent Acceleration (TA) mode, please follow the troubleshooting procedures in **Section 5.2.5 TA Failure**.

3.5 Windows DBL TA Implementation

On Windows, starting with DBL 3.x, two solutions for the DBL TA mode for TCP and UDP sockets (Winsock) are offered. Note that only one DBL TA method (LSPs or DLL preload) may be active/registered at a time.

3.5.1 Acceleration as an LSP implementation

This method is based on the concept of Layered Service Providers (LSPs). Starting with DBL v2, both IFS and non-IFS LSPs are provided so that applications can run unmodified with accelerated modes. The differentiation is that an IFS LSP can only provide Winsock 1.1 functionality. Applications that would like to accelerate Winsock 2.2 function calls (overlapped calls, IOCP support), must use the non-IFS LSP (ipdbl_lsp22.dll). It should be noted that Winsock 2.2 encapsulates Winsock 1.1 functionality, and that a DBL TA installation per host can only enable one mode, either Winsock 1.1 or Winsock 2.2.

This DBL TA mode can be configured as part of the MSI installation (see **Section 2.3.3.3**), or manually configured via instlsp.exe after installing the DBL drivers. You should verify that the LSP is installed by running the command

```
$ instlsp.exe -p
```

The Windows DBL TA configuration can be changed by activating either one of the LSPs. To change the TA (Transparent Acceleration for TCP/UDP) mode from Winsock1.1 to Winsock2.2 support or vice versa, execute the following procedure.

1. To **query status**, use the instlsp.exe tool in [INSTALLDIR]\sbin and run

   ```
   $ instlsp.exe -p
   ```

2. To **remove**, use the instlsp.exe tool in [INSTALLDIR]\sbin and run

   ```
   $ instlsp.exe -f
   ```

3. **Reboot**, if the LSP had to be removed.

4. **Run**

   ```
   $ instlsp.exe -i -d %WINDIR%\System32\ipdbl_lsp11.dll
   ```

   or

   ```
   $ instlsp.exe -i -d %WINDIR%\System32\ipdbl_lsp22.dll
   ```

   for Winsock 1.1 support or Winsock 2.2 support, respectively.

3.5.2 Acceleration via DLL preload [BETA]

This implementation transparently connects the application and the Winsock interface. Residing above any LSPs, this method allows for lower latencies. Similar to the first method, a Winsock 1.1 and a Winsock 2.2 compliant implementation is provided. The installation is as follows:

1. To **query status**, use the instdllinj.exe tool in [INSTALLDIR]\sbin to determine if a DLL is registered

   ```
   $ instdllinj.exe -p
   ```

2. To **remove**, use the instdllinj.exe tool in [INSTALLDIR]\sbin and run

   ```
   $ instdllinj.exe -f
   ```

3. **Reboot,** if the DLL had to be removed.

4. **Run**

   ```
   $ instdllinj.exe -i -d %WINDIR%\System32\ipdbl_inj11.dll
   ```

   or

   ```
   $ instdllinj.exe -i -d %WINDIR%\System32\ipdbl_inj22.dll
   ```

   for Winsock 1.1 support or Winsock 2.2 support, respectively.

Note that **dblrun.exe** must be used to accelerate your application. dblrun.exe will accelerate using the method which was registered. Also note, that only one method can be active/registered at a time.

If the sockets application fails to run successfully in Transparent Acceleration (TA) mode, please follow the troubleshooting procedures in **Section 5.2.5 TA Failure**.

3.6 Running DBL TA tests on Linux

On Linux, the DBL test programs are located in **/opt/dbl/bin/tests**.

On Windows, the DBL test programs are located in **INSTALL_DIR\bin\tests**.

As a basic test, let us measure the UDP pingpong and TCP pingpong performance of DBL Transparent Acceleration (TA) using the test program **tcp_pingpong** between a server machine and a client machine. Please compare performance results when the two machines are connected back-to-back (switchless) and when the two machines are connected through a 10GbE switch.

tcp_pingpong is a pure sockets program which bounces TCP (or UDP) packets from one host to another host and prints latency results in microseconds.

It is recommended to run the provided examples both with standard IP (without **dblrun**) and in IP acceleration mode (with **dblrun**) to check the system configuration and performance improvement.

Important Notes for Testing:

- Loopback communication on the loopback address (e.g IP 127.0.0.1) is only supported in DBL TA mode (dblrun), starting with DBL version 2.1.0.51507.
- When using loopback with DBL TA mode (dblrun), a socket is demoted/ unaccelerated and the traffic is passed over internal OS dependent paths which may be faster than sending packets via DBL.
- A network configuration with IP addresses on the same subnet is **not** supported.

3.6.1 Simple (Standard IP) pingpong latency test for UDP

On both machines:

```
$ cd /opt/dbl/bin/tests
$ PATH=$PATH:/opt/dbl/bin:/opt/dbl/bin/tests
```

Server host:

```
$ tcp_pingpong -u -s -l <local_ip>
```

Client host:

```
$ tcp_pingpong -u -l <local_ip> -h <server_ip> -i 50000
```

Successful execution of the test will report a sequence of communications between the two machines. Note the actual delays reported on your systems to compare with the next step (accelerated). Kill the running tasks when done.

If the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

3.6.2 Accelerated pingpong latency test for UDP

Now repeat the same UDP test to demonstrate Transparent Acceleration (TA) for UDP.

On both machines:

```
$ cd /opt/dbl/bin/tests
$ PATH=$PATH:/opt/dbl/bin:/opt/dbl/bin/tests
```

Server host:

```
$ dblrun tcp_pingpong -u -s -l <local_ip>
```

Client host:

```
$ dblrun tcp_pingpong -u -l <local_ip> -h <server_ip> -i 50000
```

The resulting output should show accelerated UDP traffic with much smaller latency results than with conventional Ethernet. Some variability in individual “ping” numbers is normal. Kill the running tasks when done.

When the two machines are connected back-to-back (switchless), the expected UDP pingpong latency is 3-4 microseconds. The speed of the CPU/host is a very important factor in the latency performance of the host. If the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

If the sockets application fails to run successfully in Transparent Acceleration (TA) mode, please follow the troubleshooting procedures in **Section 5.2.5 TA Failure**.

3.6.3 Simple (Standard IP) pingpong latency test for TCP

On both machines:

```
$ cd /opt/dbl/bin/tests
$ PATH=$PATH:/opt/dbl/bin:/opt/dbl/bin/tests
```

Server host:

```
$ tcp_pingpong -s -l <local_ip>
```

Client host:

```
$ tcp_pingpong -l <local_ip> -h <server_ip> -i 50000
```

Successful execution of the test will report a sequence of communications between the two machines. Note the actual delays reported on your systems to compare with the next step (accelerated). Kill the running tasks when done.

If the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

3.6.4 Accelerated pingpong latency test for TCP

Now repeat the same TCP test to demonstrate Transparent Acceleration (TA) for TCP.

Server host:

```
$ dblrun tcp_pingpong -s -l <local_ip>
```

Client host:

```
$ dblrun tcp_pingpong -l <local_ip> -h <server_ip> -i 50000
```

The resulting output should show accelerated TCP traffic with much smaller latency results than with conventional Ethernet. Some variability in individual “ping” numbers is normal. Kill the running tasks when done.

When the two machines are connected back-to-back (switchless), the expected TCP pingpong latency is 4-5 microseconds. The speed of the CPU/host is a very important factor in the latency performance of the host. For a fast host, it may be possible to achieve 3-4 microseconds TCP pingpong latency. If the two machines are connected through a 10GbE switch, it is important to note that the latency of the 10GbE switch varies depending upon the make/model of the switch.

If the sockets application fails to run successfully in Transparent Acceleration (TA) mode, please follow the troubleshooting procedures in **Section 5.2.5 TA Failure**.

3.7 Manual Configuration of Transparent Acceleration (TA)

3.7.1 Linux

Using **dblrun** is the preferred method to enable DBL Transparent Acceleration (TA) for the application. A manual setting is not recommended, but if you are unable to use **dblrun**, the DBL TA mode must be manually configured by setting the **LD_PRELOAD=<path_to>/libdbl_preload.so** environment variable at the command line of the application. E.g.,

```
$ LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ./testprg [options] [params]
```

This **LD_PRELOAD** environment variable forces the Linux loader to use the socket-related functions in the DBL library rather than the default libc functions. We recommend that **LD_PRELOAD** only be specified on the command line of the application rather than being globally exported.

Optional environment variables which may be set are:

- **DBL_IP_ADDRESS=a.b.c.d**

  DBL_IP_ADDRESS specifies the IP address of the DBL-enabled network adapter whose sockets are to be accelerated. It will be queried if the application’s **bind()** function call is using either **INADDR_ANY**, a multicast address or an auto-bind is required.

  For example:

```
$ DBL_IP_ADDRESS=10.0.0.1 LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ↵
./testprg [options] [parameters]
```

- **DBL_IP_PORTRANGE=x:y**

  For example, to only map sockets with ports in the range [4000 .. 5000] ↵
  to be accelerated:

```
$ DBL_IP_PORTRANGE=4000:5000 LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ↵
./testprg [options] [parameters]
```

- **DBL_ENABLE_TCP=0**

  Do not accelerate TCP.

```
$ DBL_ENABLE_TCP=0 LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ↵
./testprg [options] [parameters]
```

- **DBL_ENABLE_UDP=0**

  Do not accelerate UDP.

```
$ DBL_ENABLE_UDP=0 LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ↵
./testprg [options] [parameters]
```

- **DBL_USE_BSD_SHIM=1**

  Use the BSD stack for UDP to overcome the 32-endpoint limitation (DBL API).

```
$ DBL_USE_BSD_SHIM=1 LD_PRELOAD=/opt/dbl/lib/libdbl_preload.so ↵
./testprg [options] [parameters]
```

Note: The **DBL_IP_ADDRESS** is currently only required for **INADDR_ANY** on UDP sockets and **DBL_IP_PORTRANGE** is currently only supported on UDP sockets.

3.7.2 Windows

Using **dblrun.exe** is the preferred method to enable DBL Transparent Acceleration (TA) for the application. A manual setting is not recommended, but if you are unable to use **dblrun.exe**, the DBL TA mode must be manually configured by setting the **DBL_IP_ACCELERATION** environment variable to 1 at the command prompt prior to invoking the application. I.e.,

```
$ set DBL_IP_ACCELERATION=1
$ ./testprg.exe [options] [parameters]
```

To verify that DBL TA mode is being utilized for the application, you would additionally set the **DBL_IP_VERBOSE** environment variable to 1, and the LSP or DLL preload method will provide startup information detailing which sockets are accelerated.

Optional environment variables which may be set are:

- **DBL_IP_ADDRESS=a.b.c.d**

  In UDP mode if an INADDR_ANY address is used, the TA will use this IP address to bind to an IP:PORT (TCP does support INADDR_ANY)

- **DBL_IP_VERBOSE=1**

  Provides some additional information to stdout. e.g. which sockets are accelerated

- **DBL_CPU_AFFINITYMASK=mask**

  Mask for cpu binding

- **DBL_ENABLE_TCP=0**

  Do not accelerate TCP.

- **DBL_ENABLE_UDP=0**

  Do not accelerate UDP.

- **DBL_IP_LOGFILE=string**

  The path to the file for tracing information.

- **DBL_USE_BSD_SHIM=1**

  Use the BSD stack for UDP to overcome the 32-endpoint limitation (DBL API).

4 Tuning

The performance of the Myricom High-Speed, Low-Latency networking solutions in real customer environments will vary depending upon the particular details of the network configuration, end-user applications, and transaction workloads.

The key to maximum performance improvements is latency which may be accelerated as much as 4x for UDP and 2.5x for TCP transactions in optimal situations. At present there is no direct way to measure these effects to determine real acceleration in a particular environment. Also it must be appreciated that bulk data transfers and other application-specific delays will reduce the effective user application transaction rate in real world situations.

However, critical applications often have internal measurements of effective transaction rates. It is recommended that a study be made of performance before installing a Myricom solution and compared with performance after. Ideally a repeatable workload should be used. If that is not possible, sample enough daily traffic to establish typical performance. If a noticeable improvement was not experienced after installation of the Myricom solution then verify that proper installation of network adapters and software features was performed.

The expected performance of the DBL API mode is demonstrated in **Chapter 2: Installation**, and the expected performance of the DBL TA mode is detailed in **Chapter 3: Transparent Sockets Acceleration Software**.

CPU binding (e.g., using taskset or numactl on Linux) can be very useful for optimizing performance for DBL. On hosts with multiple CPU sockets, some CPUs are physically closer to the adapter and/or memory and perform better than others. Some systems have multiple PCI-Express root ports, for example AMD machines, and some PCI Express slots can connect to one root port and are more easily accessible from one CPU socket, while another PCI-Express slot will be connected to another root port and are more easily accessible from another CPU socket. Some PCIe slots on a machine may have deeper PCI bridging than others. Extra bridge chips between the CPU and the adapter will result in higher latency. If there seem to be lots of bridge chips, trying a different PCI-Express slot may improve latency for DBL. (The tool **myri_info** can be used to see the bridges between the CPU and adapter).

On Windows, DBL v3.1.6 uses an Adaptive Interrupt Moderation (AIM) Scheme (See
Network Configuration, Configure, Advanced Tab). Enabling this feature yields

lower latency for any socket using this interface even without using TA. It can be used to enhance socket applications which rely on overlapping functionality and IOCP support (Winsock 2) which is available in DBL TA mode. However, please note that to achieve the lowest latency an application should use poll/select rather than IOCP, as the latter requires a context switch. It is recommended to compare the results for the test mentioned in Section 3.2.2 in default mode versus AIM mode without using TA / dblrun.exe. Currently, AIM is disabled by default and needs to be enabled.

Contact ARIA Technical Support for further assistance if results are below expectations.

5 Troubleshooting

The defective operation of a network adapter is usually easily determined. When this situation occurs, contact ARIA Technical Support to initiate the RMA (Return Merchandise Authorization) process to obtain a replacement for the defective network adapter.

If the network appears to be functioning correctly but application transaction rates have not significantly improved after installation of the Myri-10G Network Adapters and DBL software, it is possible that the proprietary software features are not properly enabled.

Remember that the default operation of the Myri-10G Network Adapters is to provide standard 10Gbit Ethernet performance. Only those hosts with valid DBL licenses loaded will demonstrate accelerated performance.

It may be necessary to repeat the activation process on all suspect host machines using the latest customer site license file to insure proper status of all network adapters.

5.1 Hardware Issues

5.1.1 Hardware Installation/Performance

Your host/motherboard may have either PCIe 3.0 (“Gen3”), PCIe 2.0 (“Gen2”), or PCIe 1.1 (“Gen1”) PCI-Express slots.

Both x8 "Gen2" and x8 "Gen1" 10-Gigabit Network Adapters are available for purchase.

> **Important Note**: Myri-10G "Gen1" PCI Express Network Adapters are compatible with “Gen3” and "Gen2" slots in hosts; the adapter auto-negotiates operation in the widest available mode (x8, x4, x2, or x1) supported by the slot into which it is installed, and at the 2.5 GT/s data rate. Similarly, Myri-10G "Gen2" PCI Express Network Adapters are compatible with “Gen3” slots and auto-negotiate operation in the widest available mode (x8, x4, x2, or x1) supported by the slot into which it is installed, and at the highest data rate (5 or 2.5 GT/s), but these "Gen2" PCI Express Adapters cannot achieve full performance in "Gen1" PCI Express slots in a host.

Which type of Myri-10G network adapter is installed? Gen1 or Gen2?

The Product code for the adapter will indicate if it is a "Gen1" (2.5 GT/s) PCI-Express Network Adapter or if it is a "Gen2" (5.0 GT/s) PCI-Express Network

Adapter. If the product code includes PCIE2 in its name, then the adapter is a "Gen2" (5.0 GT/s) PCI-Express Network Adapter. For example, 10G-PCIE2-8B2-2S. Similarly, if the product code has PCIE in its name, then the adapter is a "Gen1" (2.5 GT/s) PCI-Express Network Adapter. For example, 10G-PCIE-8B-S.

For optimal performance, if you have installed an x8 “Gen2” Myri-10G network adapter, verify that the adapter reports x8 "Gen2" (5.0 GT/s) PCIe link speed. There are two ways to determine if the "Gen2" Myri-10G Network Adapter is installed into a "Gen2" PCI Express slot: the output of **/opt/dbl/sbin/myri_info**, or (Linux only) the output of **lspci -vvv**.

Example header information from the output of **myri_info** for a “Gen2” Myri-10G Network Adapter:

```
pci-dev at 05:00.0 vendor:product(rev)=14c1:0008(01)
        behind bridge downstream-port: 04:02.0 111d:806a (x8.1/x8.2)
        behind bridge upstream-port: 03:00.0 111d:806a (x8.2/x8.2)
        behind bridge root-port: 00:03.0 8086:340a (x8.2/x16.2)
Myri-10G-PCIE-8B -- Link x8
```

The ".2" in the **pci-dev** output indicates that this is a PCIe 2.0 "Gen2" slot.

For PCIE2 adapters, the interesting component in the **pci-dev** output is the second line in the bridge series.

More specifically:

- **behind bridge upstream-port: 03:00.0 111d:806a (x8.2/x8.2)** indicates that our adapter is currently running at x8 Gen2 speed (and that is also its maximum capability).
- **behind bridge root-port: 00:03.0 8086:340a (x8.2/x16.2)** indicates the same link as seen from the motherboard side. The link is observed running at the same width-speed x8.2 from this side than from the adapter side. In addition, the motherboard slot is advertised as x16-able.
- The **x8.1** lines are about the internal links between the PLX/IDT chip and each of our controllers inside our adapter, which are x8.1 (x8 at Gen1 speed), but there is one link for each of the two Z8ES chips.
- The **Myri-10G-PCIE-8B -- Link x8** also indicates that the adapter is optimally running at x8 speed.

Alternatively, if your operating system has the **lspci** command, look at the **lspci -vvv** output to examine the Link speed (**Lnk Sta**) for the PLX or IDT chips.

If you have a two-port **8C** adapter, e.g., **10G-PCIE2-8C2-2S**, the Myri-10G network adapter has a PLX bridge chip. There will be 4 PLX chip entries in the **lspci** output, as well as 2 entries for 10G-PCIE-8B. The extra PLX entry is a downstream port which can be ignored. If the network adapter is installed into an x8 PCIe "Gen2" slot, then the **Lnk Sta** (a.k.a., link status) of one of the PLX chip entries should be listed as **5 GT/s**, and the **Lnk Sta** of the other three PLX chip entries will be listed as 2.5 GT/s. Otherwise, if you only see 2.5 GT/s listed for all 4 PLX chips, the network adapter is installed into a "Gen1" PCIe slot.

If you have a two-port **8B** adapter, e.g., **10G-PCIE2-8B2-2S**, the Myri-10G network adapter has an IDT bridge chip. There will be 3 IDT chip entries in the **lspci** output, as well as 2 entries for 10G-PCIE-8B. If the network adapter is installed into an x8 PCIe "Gen2" slot, then the **Lnk Sta** of one of the IDT chip entries should be listed as **5 GT/s**, and the **Lnk Sta** of the other two IDT chip entries will be listed as 2.5 GT/s. Otherwise, if you only see 2.5 GT/s listed for all 3 IDT chips, the network adapter is installed into a "Gen1" PCIe slot.

For similar performance reasons, if you have installed an x8 “Gen1” Myri-10G network adapter, verify that the adapter reports x8 “Gen1” (2.5 GT/s) PCIe link speed.

5.1.2 Hardware Cabling Connectivity Issues

If you are using Myri-10G “8C” adapters (10G-PCIE2-8C2-2S or 10G-PCIE2-8C2-2S-SYNC) with SFP+-terminated copper “direct attach” cables and you experience link up/down connectivity issues or bad crc errors, try loading the myri_dbl driver with the load-time option **myri_serdes_mode=2**.

On Linux, issue the following command:

```
    # /opt/dbl/sbin/myri_start_stop start myri_serdes_mode=2
```

On Windows, the equivalent instructions are:

Create the following registry key (followed by a reboot):

```
    REG ADD HKLM\SYSTEM\CurrentControlSet\services\dbl /v myri_serdes_mode
/t
    REG_DWORD /d 2
```

We have documented cases where certain direct attach cables work better with 10G-PCIE2-8C2-2S-SYNC adapters when SFP+ settings for the serdes chip on the adapter are used instead of direct attach settings.

All of the possible options to **myri_serdes_mode=X** are as follows:

0: Autodetect (default)

1: Force use of direct attach cable settings.

2: Force use of SFP+ 10GBase-SR/10GBase-LR settings.

3: Force use of SFP+ 10GBase-LRM settings.

All other values are reserved for future use.

If you are using Myri-10G “8B” adapters (e.g., 10G-PCIE2-8B2-2S), the load-time option **myri_serdes_mode=X** will be ignored. If you experience link up/down connectivity issues or bad crcs errors with the Myri-10G “8B” adapters and direct attach cables, please try a shorter length direct attach cable or replace the direct attach cable with an SFP+ transceiver module and fiber cable.

5.1.3 Timing Discontinuities with 10G-PCIE2-8C2-2S-SYNC Adapters

When using a 10G-PCIE2-8C2-2S-SYNC adapter connected to a timesource generator, it is **very** important to have a stable timesource. If the timesource generator encounters an intermittent loss of connectivity with the GPS signal, timing discontinuities will cause one or both of the following two symptoms.

1. loss of synchronization with timesource requiring a re-synchronization
2. link up/down issues

The first symptom is the expected behavior with timing discontinuities, and the kernel log will contain messages indicating the successful resynchronization.

If a resynchronization attempt fails after a timing discontinuity, the kernel log will contain repeated messages indicating "Regained communication with timesource" and the message "Timecode features not working". For example:

```
[Fri Nov 15 18:14:15 2013] myri_dbl INFO: Enabling timesource
timestamping.
[Fri Nov 15 18:14:15 2013] myri_dbl INFO: Timestamps freerun on timesource
loss.
[Fri Nov 15 18:14:15 2013] myri_dbl INFO: 0: Regained communication with
timesource.
[Fri Nov 15 18:14:15 2013] myri_dbl INFO: 1: Regained communication with
timesource.
[Fri Nov 15 18:14:16 2013] myri_dbl INFO: 0: Synched with timesource at
1384560856 seconds.
[Fri Nov 15 18:14:17 2013] myri_dbl INFO: 1: Synched with timesource at
1384560856 seconds.
[Thu Nov 21 00:22:15 2013] myri_dbl INFO: 0: Regained communication with
timesource.
[Thu Nov 21 00:22:15 2013] myri_dbl INFO: 1: Regained communication with
timesource.
[Thu Nov 21 00:22:16 2013] myri_dbl INFO: 0: Regained communication with
timesource.
[Thu Nov 21 00:22:16 2013] myri_dbl INFO: 1: Regained communication with
timesource.
[Thu Nov 21 00:23:27 2013] myri_dbl INFO: 1: Timecode features not working
[Thu Nov 21 00:23:28 2013] myri_dbl INFO: 1: Something bad happened during
timesource read. This might be fatal.
```

If you see messages like these in your logs, you likely have an older timestamp microcontroller firmware on the 10G-PCIE2-8C2-2S-SYNC adapter and should upgrade to the newest firmware to avoid having the adapter permanently fail in its synchronization with the timesource. With the upgraded firmware there will still be a loss of synchronization and a resynchronization, but it will not result in a permanent failure. This problem only manifests itself in the presence of timing discontinuities in the input timesource.

To resolve this resynchronization issue, you will need to upgrade the timestamp microcontroller's firmware on the adapter using the **cf_prog** tool in the **SYNC Adapter Tool Kit.** Contact ARIA Technical Support to obtain this tool kit.

If the second symptom (link up/down issues) occurs, you will additionally see messages in the kernel log similar to:

```
[Fri Jan 3 22:39:25 2014] myri_dbl INFO: eth5: Link0 is DOWN
[Fri Jan 3 22:39:27 2014] myri_dbl INFO: eth5: Link0 is UP
```

There is no hardware workaround for this link up/down issue. Please investigate why you are experiencing time discontinuities with your GPS timesource. We have only seen discontinuities such as these when we disconnected the antenna for the timesource, waited a few minutes, and then reconnected the antenna. Is your GPS antenna not positioned with a full view of the sky? Bad antenna cabling? Interference from some other source?

Both issues can be avoided by having a stable timesource.

5.2 Software Installation/System Configuration

If an issue is encountered with installation, usage, or performance, please send the output of **myri_bug_report** or **myri_dmesg.ps1** to ARIA Technical Support. The output of this script may contain vital information to speed the resolution of the issue.

/opt/dbl/sbin/myri_bug_report is a diagnostic script included in the Linux DBL v3.1.6 and v2 software distribution. it is used to collect diagnostic information about a customer’s system configuration, such as uname output, processor files such as cpuinfo and interrupts, lspci, kernel messages, ethtool, myri_counters, etc. This script must be run as root.

[INSTALLDIR]\sbin\myri_dmesg.ps1 is a powershell script in the Windows DBL v3.1.6 and v2 software distribution which extracts logging information from the Windows Logs and prints them to stdout. It will show whether the DBL license is valid, which driver is loaded, and any error statements. This script must be run as administrator. (If you cannot use powershell, inspect the **Event Viewer -> Windows Logs** for error messages related to the DBL software.)

5.2.1 Linux RPM/TGZ Installation failure

If an error occurs during the Linux RPM installation, please send us the full output from the rpm command as well as the output in the kernel log and **/tmp/ myri_dbl.log**.

If an error occurs during the Linux TGZ installation, please send us the full output from the **sbin/rebuild.sh** command. Please note that to build a DBL kernel module, the source kernel tree must be configured to match the running kernel. For example, with RedHat, you must install the kernel-devel package and the kernel-headers package from the RedHat distribution.

5.2.2 Windows MSI Installation failure

If the Windows MSI installation fails when running the MSI installer, please send ARIA Technical Support the log obtained by adding /log filename to the MSI install. For example,

```
c:\> dbl-<version_info>*x64_wnet_wlh.msi /log dbl_install_log.txt
```

If the MSI installation fails with an ambiguous message saying that "There is a problem with this Windows Installer Package. A program run as part of setup did not finish as expected". please do the following.

Please verify that an earlier removal has unlocked all processes from DBL related files. A way to determine if DBL is still in use is to run from a command prompt:

```
c:\> tasklist /m dbl.dll
```

The following output will show process(es) still using DBL related software:

```
Image Name                   PID Modules

============ ========= ======================

svchost.exe                 8948 dbl.dll
```

A reboot is required.

Alternatively, the Windows DBL software can be installed using Windows msiexec as described in **Section 2.3.3.3: Installing with Windows msiexec** and **Section 2.3.3.4: Deploying for Windows imaging**.

5.2.3 Bonding for Failover in TA mode

DBL provides limited bonding support (DBL TA mode only and used for failover only) in DBL 2.1.0.51345 and later.

Using this failover mode in DBL TA (dblrun) requires no additional configuration of the bonding driver. The bonding driver is configured to bond the ethernet DBL interfaces as usual for failover, and DBL TA mode will use that configuration. Under Linux, this feature requires configuring and using the bonding.ko module for a failover bond with the ethernet devices exposed by DBL. Under Windows, this feature requires configuring the supplied teaming driver for a failover team.

When TA mode is used, if it detects that the device you are using is part of a failover bond, TA mode executes a **dbl_open** on each underlying device and then the native bonding driver detects that a link is down and selects a new active slave. TA mode then detects that the active slave has changed and switches to using that slave for its operation.

5.2.4 Failover and myri_port_failover

myri_port_failover is a utility that can be used to control the failover policy on the 10G-PCIE-8B-2* adapters and the 10G-PCIE-8B2-4I BladeCenter adapters.

The usage information for **myri_port_failover** is:

```
myri_port_failover [ -b<unit> ] [ -0 | -1 | -s | -r<0|1> ]
    Options:
     -b <unit>: uses NIC/Controller rank #unit (default 0)
     -0: set P0 as primary link for failover purposes (P1 is backup-only)
     -1: set P1 as primary link for failover purposes (P0 is backup-only)
     -s: symmetric failover (only switch link if current port is down)
     -r<0|1>: disable/enable sending a rarp pkt on failover
         (to update switches tables)
```

With no options (other than -b<unit>), it will only display settings/status.

Without options, it will print the current settings, for example:

```
Currently active-port: P0
Rarp-on-failover: enabled
Port-priority: symmetric
P0-link is UP
P1-link is DOWN
```

As can be seen in the usage information, options -0, -1, and -s choose the failover policy.

This utility is particularly useful with the BladeCenter adapters to determine the I/O Module that is being used by the Myri-10G High Speed Expansion Card (HSEC) in an IBM BladeCenter blade. Instructions for this procedure are described below.

One Myri-10G High Speed Expansion Card (HSEC) type for the IBM BladeCenter H is available for purchase.

The 10G-PCIE-8B2-4I HSEC has 4 network ports for performance and failover (20Gb/s throughput). For the 10G-PCIE-8B2-4I expansion card (which has two interfaces), one interface will be connected to I/O Module slot 7 and 8, and the other interface will be connected to I/O Module slots 9 and 10.

The mapping of logical ethernet interfaces (ie. ethX) to physical interfaces is set by the PCI bus structure, which is motherboard-dependent. To determine which logical ethernet interface is connected to which I/O Module slot you must view the interfaces' MAC addresses.

The interface with the lower MAC address will be connected to I/O Module slots 7 and 8, while the interface with the higher MAC address will be connected to I/O Module slots 9 and 10.

Here is an example:

If the 4I adapter has MAC addresses 00:60:dd:46:ef:a8 and 00:60:dd:46:ef:aa, then

```
00:60:dd:46:ef:a8 port 0 connected to I/O Module slot 7
00:60:dd:46:ef:a8 port 1 connected to I/O Module slot 8

00:60:dd:46:ef:aa port 0 connected to I/O Module slot 9
00:60:dd:46:ef:aa port 1 connected to I/O Module slot 10
```

The functions **sbin/myri_info** and **bin/myri_nic_info** will provide the MAC addresses. From this output, you can determine which MAC address gets mapped to which interface.

If you are trying to use the failover port, then you need to either use the **myri_ethport** tool from the myri-tools package, or use **bin/myri_port_failover**, included in DBL
2.0.4 or later.

5.2.5 TA Failure

If the sockets application fails to run in DBL Transparent Acceleration (TA) mode, please follow these troubleshooting procedures to isolate the source of the failure.

When reporting the error to ARIA Technical Support please send the log generated when using the -f option in dblrun, e.g.

```
$ dblrun -f logfile myapp.exe
```

Note: Since the logfile may be very large for some applications, please try to reproduce the error while using the smallest number of connections (feeds) and the minimum number of variables that still show the error, and then send us the logfile for this smaller configuration.

Troubleshooting Steps

- **Does the problem occur when using UDP or TCP or both?**

  Determine if the problem occurs when using UDP or TCP by running the application with combinations of

```
$ dblrun -u -t
```

where specifying both -u and -t will use the base provider.

- **How many UDP sockets are you trying to accelerate?**

  The DBL stack for UDP acceleration only supports up to 32 DBL endpoints. If you have more sockets under DBL TA (Transparent Acceleration), then use the

  ```
  $ dblrun -b myapp.exe
  ```

  option to multiplex those (UDP) sockets using the general TA BSD stack. This suggestion also applies if several applications are accelerated concurrently and few endpoints are available (see the output of **myri_endpoint_info**).

- **How are the Myri-10G network adapters connected?**

  Loopback communication on the loopback address (e.g IP 127.0.0.1) is only supported in DBL TA mode (dblrun), starting with DBL version 2.1.0.51507.

  When using loopback with DBL TA mode (dblrun), a socket is demoted/ unaccelerated and the traffic is passed over internal OS dependent paths which may be faster than sending packets via DBL.

- **Do you have a network configuration with IP addresses on the same subnet?**

  Starting with DBL 2.1.0.51753, the following error message is printed when the situation occurs:

  "Two IPs on the same subnet detected. TA mode is prevented for this configuration (see FAQ). Exiting."

  In earlier releases of DBL, you would see the following error message in the dblrun -f output.

  ```
  WARN dbl_route_change: Route request failed 17
  ```

  The problem reported by "WARN dbl_route_change: Route request failed 17" is most likely due to a duplicate route in the system's routing table. There are a couple of ways that this situation can arise:

  - if you have manually added a duplicate route to the routing table. Or,
  - if you have two different interfaces on the same subnet. DBL does not support a network configuration with IP addresses on the same subnet. Having two interfaces on the same subnet causes confusion about which interface to use even when using the normal Linux tcp stack.

  Please examine your system routing table to see if there are any duplicate entries where the destination/netmask are the same, and also check that you are not putting two or more interfaces on the same subnet.

- **LD_PRELOAD error message?**

If you are using Linux DBL TA mode, do you see the following LD_PRELOAD error message?

```
   ERROR: ld.so: object '/opt/dbl/lib/libdbl_preload.so' from LD_PRELOAD
cannot be preloaded: ignored.
```

Linux DBL only supports 64-bit applications. If you see the above error message, this is an indication that the target application is 32-bit.

- **Winsock error message?**

  If you are using Windows DBL TA mode, do you see the following Winsock error message?

  ```
  20.09.2012 13:44:35.960 | 1948 | ERROR | 10021001 | dispatcher::issue
  false read result: [error 10045: "The attempted operation is not
  supported for the type of object referenced."]
  ```

  This Winsock error messages indicates that you are using the LSP11 for Winsock1.1 applications but the application is making overlapped calls for which the LSP22 needs to be registered.

  For Winsock 2.2, you need to register the library ipdbl_lsp22.dll. For LSP installation instructions, please read **Section 3.4 Windows DBL TA Implementation**.

- **Are you running a Windows .NET application built with AnyCPU?**

  If you are using Windows DBL TA mode with a .NET application, do you see the following error message?

  ```
  (dblrun info) Unable to accelerate the [64-bit] application.
     Please install the DBL TA LSP for the corresponding chains
  ```

  Since .NET applications can be built for x86,x64 and AnyCPU, please note that when compiling for AnyCPU the bitness cannot be determined and dblrun.exe will give a warning. In this case the LSP dll which matches the native OS bitness needs to be registered.

  .NET apps specifically targeting x64 or x86 do not see that warning. If you specify **dblrun -v your.NetApp** and see a (transp info) in the command prompt it confirms the appropriate DBL TA chains have been intercepted.

5.2.6 Bad License Key

If there is no valid license loaded on a network adapter, **dblrun** will print an error message of the following form:

```
   License check failed on board 0, sn=<serial>: Invalid key signature
```

Additional diagnostic information may be obtained by examining the kernel log output (dmesg) or running the command:

```
$ /opt/dbl/bin/myri_nic_info --license
```

Please contact ARIA Technical Support to obtain an appropriate key.

5.2.7 Expired License Key

If a DBL application complains of an expired license key, please examine the output of **/opt/dbl/bin/myri_nic_info** or **/opt/dbl/sbin/myri_info** to determine if a valid DBL license key is present on the adapter and that the DBL driver is loaded properly on the network port. The Running MCP line of text in the **myri_info** output will identify which driver and version of software that is running.

5.2.8 Multiple License Keys

If you have purchased both DBL and Sniffer10G licenses for the same Myri-10G network adapter(s), both licenses can exist concurrently on the adapter and/or in the myri_license license file. However, only one driver can be loaded at a given time on a specific network port on the adapter. The driver load-time option **myri_bus** or **myri_mac** can be used to selectively load DBL or Sniffer10G (or Myri10GE) on a specific network port on the adapter. Contact ARIA Technical Support for details.

Before attempting to run DBL applications, it should be verified that the DBL driver is loaded on the specific adapter's network port. To verify that the DBL driver is loaded, run either the **/opt/dbl/bin/myri_nic_info** command or **/opt/dbl/sbin/myri_info** command and examine the **Running MCP** section of the output for text that references **dbl-<version>**. Otherwise, if the adapter has two ports or there are multiple adapters installed on the same hosts, there may be confusion with the wrong driver loaded on the wrong adapter port at a given time.

If a license key is accidentally removed from an adapter, contact **ARIA Technical Support** with the 6-digit serial number (or MAC address) of the adapter and we will resend the license key to you.

5.2.9 Software Counters

The tool **myri_counters** provides low-level DBL hardware and software counters for traffic that goes through the device when it is in DBL mode.

On Linux:

```
$ /opt/dbl/bin/myri_counters
```

On Windows:

```
$ [INSTALL_DIR]\bin\myri_counters.exe
```

By default, the **myri_counters** output is only displayed for port/board 0. Two-port adapters appear to **myri_counters** as different ports/boards. If you have a two-port network adapter installed in the host, you will need to specify the command-line argument **-p <port_num>** to obtain the counters output for each port. The variable **port_num** is an integer value from 0 to n-1, where n is the number of

Myri-10G network adapters installed in the host and running the DBL driver. For example:

```
% /opt/dbl/bin/myri_counters -p 0
% /opt/dbl/bin/myri_counters -p 1
```

Note that the space between the "p" and the number is optional. Also, if a host contains two two-port adapters, you would use -p0 and -p1 for the ports of the first adapter and -p2 and -p3 for the ports of the second adapter.

To clear / reset the counters requires root privileges. To clear the counters on a specific port of a network adapter use:

```
# /opt/dbl/bin/myri_counters -p <port_num> -c
```

For a detailed listing of the command-line arguments to **myri_counters**, please read **Section 2.5.2: Tool Programs**. The DBL software counters reported by **myri_counters** are defined in **Appendix: DBL Counters**.

5.2.10 Functional failures

Some sockets-based applications may experience a functional failure due to the behavior of DBL being different from that of the regular Operating System stack. On Linux, an important portion of the transparent Sockets service is devoted to emulation of BSD-style Linux sockets. If such a failure is encountered, the following command can be involved to save tracing output:

```
$ dblrun -f output <application_name>
```

This output file or part of the file can then be sent to ARIA Technical Support.

5.2.11 Performance

When the network performance is below expectation, please follow these steps to isolate the problem:

1. Verify that the transparent socket acceleration is effectively loaded.
   a. Examine the banner on the screen when starting DBL for information in the following format:

   ```
   <process_name>[<pid>] DBL 2.0.0.<build_id> Copyright 2011 Myricom Inc.
   ```

   b. Use **myri_license** to verify the status of the license (see Section 2.4.2: License Key Activation Process). Repeat License activation if necessary.

2. Monitor **myri_counters** to verify that traffic is in fact being accelerated and the network is not experiencing “receive data buffering” overflow conditions. For each network adapter, check the runtime counter values.

   a. First examine values of the “DBL_* ” counters to verify that traffic is going directly to a userspace stack.

   b. Next examine the value of “Receive Queue full”, and “Net Overflow Drop” for traffic that the network adapter and stack cannot handle fast enough. Both will be zero in normal operation. If a non-zero value is reported, contact ARIA Technical Support for assistance to isolate the problem further.

   By default, the **myri_counters** output is only displayed for port/board 0. Two-port adapters appear to **myri_counters** as different ports/boards. If you have a two-port network adapter installed in the host, you will need to specify the command-line argument **-p <port_num>** to obtain the counters output for each port. The variable **port_num** is an integer value from 0 to n-1, where n is the number of Myri-10G network adapters installed in the host and running the DBL driver.

Please also note that CPU binding (e.g., using taskset or numactl on Linux) can be very useful for optimizing performance for DBL. Please refer to **Chapter 4: Tuning** for details.

5.2.12 Network Adapter Timestamps

With DBL 2.0.2.50517 and later, DBL supports socket timestamps. Timestamping support is available in two modes: host timestamping mode, and timesource timestamping mode. Host timestamping mode is available for all supported Myri-10G network adapters. Timesource (hardware) timestamping mode is only available for 10G-PCIE2-8C2-2S-SYNC network adapters connected to a timecode generator. For software installation details about timestamping and module parameters, please read **Section 2.3 DBL Software Driver Installation Instructions**.

The timestamp that is made available through the socket interface (using the SO_TIMESTAMP[NS] socket option) is the adapter-based nanoseconds transposed to system host time (as typically returned by **gettimeofday**). When each packet arrives at the network adapter, a raw timestamp (or tick) is taken from a free-running counter and is attached to the packet as it is transferred to the host. In order to make sense of the timestamp with respect to system time, the network adapter tick is transposed to system host time by using a simple arithmetic operation. The parameters for the transposition are determined by periodically synchronizing the host’s clock source with the network adapter's clock source to account for clock skew.

Windows does not support SO_TIMESTAMP[NS] at the NDIS level but we added this feature for transparent acceleration. Therefore you can #define SO_TIMESTAMP 0x0400 to be used in setsockopt and you will be able to retrieve timestamps (similar to what Linux offers).

NOTE for non-SYNC network adapters: NTP Daemons should be disabled to ensure monotonically increasing timestamps. Services that implement the network time protocol can cause variations in system host time that are larger than the inter-packet arrival times. While the network adapter's raw timestamps are always monotonically increasing, forward or backward jumps in time may be observed once these timestamps are transposed if relatively large correction factors are applied to the host's clock source.

Synchronization

There are different levels of timestamp synchronization.

1. Network Adapter-to-host synchronization (local sync)
2. Host-to-host synchronization (global sync)
3. Network Adapter-to-global synchronization (global sync, needs extra hardware)

Most systems use something like ntpd to implement (2.) -- there are more precise options by using PTP but that only moves the accuracy from milliseconds to tens of microseconds. Since both protocols are not typically run on dedicated networks and because there is typically a lot of host overhead in processing the protocol, the time can only be so accurate.

DBL provides all three levels of timestamp synchronization. The network adapter has its own free-running clock over which timestamps are taken. These raw timestamps are transferred with each packet to the host. Two types of parameters are then necessary:

1. At startup, the network adapter timestamp was read once to compare it against the system's current time.
2. Over time, an asynchronous process synchronizes the network adapter and host clocks every second to account for drift between host and network adapter clocks (since the clock sources are two different crystals).

The timestamps returned to the users are translated to host-level timestamps based on the two above parameters. The computation is fairly inexpensive, it consists of a multiply and shift. The network adapter timestamps are always monotonically increasing. This property can only be maintained if ntpd/ptpd are disabled since there is no guarantee that these systems will not cause a significant time jump in adjustment when compared to the inter-packet arrival times that can be seen at 10Gbit/s speeds.

When using the 10G-PCIE2-8C2-2S-SYNC adapter, it is possible to implement (3.) by connecting the network adapter with a coax connector that carries an IRIG-B signal. The network adapter internally adjusts its timestamping mechanism to synchronize with the PPS information from the IRIG-B and the supplemental time values that are also encoded in this signal allow the host library to determine the second associated to the pulse. This approach is immune to whatever issues can arise by using a synchronization mechanism (like PTP) that is less accurate than 10Gbit/s frequencies.

Implementation of Time Synchronization

Here are the components involved in time synchronization with Myri-10G “8B” and “8C” network adapters.

Host Timestamping

1. The network adapter timestamps packets using its own free-running clock with an accuracy of ~1.5us on 8B and 8C network adapters. These timestamps are "raw" in that they are subject only to one clock and have not undergone any transformation.
2. The myri_dbl driver runs a background process every second to synchronize the host and network adapter clock. The host clock is what applications would use when calling gettimeofday() whereas the network adapter clock refers to the adapter's free-running clock. This process involves sampling both clocks over PCI express and is accurate to ~200ns. The sampling runs asynchronously with respect to all other running applications and produces functional parameters that can be supplied to userspace applications to transform "raw" network adapter timestamps into host timestamps.

   Note: This entry only applies to host timestamp mode and has nothing to do with the SYNC adapter when connected to a timesource. It describes how the adapter converts adapter time to host time for the purpose of presenting a timestamp on the packet in the host time domain. The host's time is never adjusted by any of our software at any time, even when connected to a timesource.
3. Users of the DBL API (or DBL TA) always return absolute timestamps. If the underlying adapter is an 8B network adapter, the raw timestamps will be converted to host timestamps and as such will be accurate to gettimeofday() within ~200ns (because of the limitations explained in #2).

Timesource (Hardware) Timestamping

1. The network adapter timestamps packets using its own free-running clock with an accuracy of ~1.5us on 8B and 8C network adapters. These timestamps are "raw" in that they are subject only to one clock and have not undergone any transformation. If you have a 10G-PCIE2-8C2-2S-SYNC network adapter that

is connected to a IRIG-B signal, the free-running clock has the additional property that it is synchronized with an external timesource accurate to ~12ns. In this case, the "raw" timestamps are also absolute timestamps because of the additional hardware on the board.

2. Users of the DBL API (or DBL TA) always return absolute timestamps. With a IRIG-B connected SYNC network adapter, the raw timestamps are absolute and referenced to the IRIG-B signal and will not undergo transformations with respect to the host system time.
3. Note (for Linux only):

   Since the host time is never adjusted by the DBL software, if you would like to sync the host's clock to the timesource, the software exports a PPS output to the system. This PPS output can be used as an input into ntp. For more details, please contact ARIA Technical Support.

6 Appendix: DBL Counters

To obtain the values of the DBL hardware and software counters since the driver was loaded (or since the counters were reset via **myri_counters -c)**, use the tool **myri_counters**.

On Linux:

```
% /opt/dbl/bin/myri_counters
```

On Windows:

```
% [INSTALL_DIR]\bin\myri_counters.exe
```

By default, the myri_counters output is only displayed for port 0. Two-port adapters appear to myri_counters as different ports. If you have a two-port network adapter installed in the host, you will need to specify the command-line argument -p <port_num> to obtain the counters output for each port. For example:

```
% /opt/dbl/bin/myri_counters -p 0
% /opt/dbl/bin/myri_counters -p 1
```

Note that the space between the "p" and the number is optional. Also, for example, if a host contains two two-port adapters, you would use -p0 and -p1 for the ports of the first adapter and -p2 and -p3 for the ports of the second adapter.

Usage Summary:

```
$ ./myri_counters -h
Usage: myri_counters [args]
-p - Port number [0] or Ethernet MAC
-c - clear the counters
-q - quiet: show only nonzero counters
-i - show host interrupt counters
-x - expert: show all counters
-h - help
```

To clear / reset the counters requires root privileges. To clear the counters on a specific port of a network adapter use:

```
# /opt/dbl/bin/myri_counters -p <port_num> -c
```

All of the counters are 64-bit counters. The majority of the counters are for developer use only and will not be defined below.

If there are additional questions, please send the output of **/opt/dbl/bin/myri_bug_report** or **[INSTALL_DIR]\bin]myri_counters.exe** to ARIA Technical Support.

Glossary for the DBL Counters.

- **Lanai uptime (seconds**)

  The time (in seconds) since the DBL driver was loaded.

- **Counters uptime (seconds)**

  The time (in seconds) since the DBL counters were reset (myri_counters -c).

- **Net send KBytes (Port 0)**

  The amount of data sent (in kilobytes).

- **Net recv KBytes (Port 0)**

  The amount of data received (in kilobytes).

- **Ethernet send**

  The number of Ethernet packets sent through the regular OS driver.

- **Ethernet send (PIO)**

  The number of Ethernet packets sent via PIO.

- **Ethernet Small recv**

  The number of small (typically 256 bytes) Ethernet packets passed through the OS driver.

- **Ethernet Big recv**

  The number of big (bigger than small, but less than the Ethernet MTU size) Ethernet packets passed through the OS driver.

- **Ethernet recv down**

  The number of Ethernet packets that could not be delivered because the ethernet interface was down. if this count is high, please check if you have ifconfig up'ed the ethernet device.

- **Ethernet recv overrun**

  The OS Ethernet driver does not consume packets as fast as the network adapter is giving them to it. In normal operation, this should not happen. It may happen if the host is very loaded.

- **Ethernet re-recv**

  The OS Ethernet driver is not consuming ethernet packets fast enough and the firmware keeps trying to deliver them.

- **Ethernet recv oversized**

  The network adapter received a packet that is larger than the configured MTU.

- **DBL send PIO**

  The number of DBL packets sent through the PIO path. (small messages, by default less than 1K)

- **DBL send DMA**

  The number of DBL packets sent through the DMA path. (large messages, by default greater than 1K)

- **DBL recv**

  The number of packets received by DBL.

- **DBL recv DMA**

  The number of DBL packets received through the DMA path. (large messages, by default greater than 1K)

- **DBL Receive Queue Full**

  A nonzero value for this counter indicates an application flow control issue. Packets cannot be received because internal buffering is exhausted due to the application not consuming the incoming messages fast enough.

- **DBL Drop endpoint closed**

  The number of packets dropped because the user level endpoint that is supposed to receive it is closed at this time.

- **DBL Drop parity recovery**

The number of packets dropped while the user level endpoint is still recovering from a parity error.

- **Flow-control Pause recv**

  The

- **Net send Raw**

  The amount of raw data sent (in kilobytes).

- **Interrupts**

  The number of MSI hardware interrupts and legacy hardware interrupts.

- **Wake interrupt**

  Developer use only.

- **Wake race**

  Developer use only.

- **Wake endpoint closed**

  Developer use only.

- **Net send queued**

  Developer use only.

- **Event Queue full**

A nonzero value for this counter indicates an application flow control issue. Packets cannot be received because internal buffering is exhausted due to the application not consuming the incoming messages fast enough.

- **RX DataQ race**

  Developer use only.

- **Fragmented request**

  Developer use only.

- **Net bad PHY/CRC32 drop (Port 0)**

  Packet dropped due to bad PHY check or bad Ethernet CRC. If the number is high, the link is bad.

  As a general rule, a non-zero value for "Net bad PHY/CRC32" indicates that either a cable/module is not seated properly or that the component is damaged and needs to be replaced. The higher the value of this counter the more likely it is to eventually cause a link connectivity failure. The damaged component is usually a cable. However, the source of the damage can be any of the hardware components along the path from the adapter to the switch. This includes the adapter port/connector, SFP+ transceiver module, cable, or the switch port.

  Isolate the damaged component by replacing each possibility with a spare, clear the myri_counters (# myri_counters -c), generate traffic, and recheck the value of this counter in the myri_counters.

- **Net overflow drop (Port 0)**

  The number of instances where the network adapter hardware had to drop packets due to lack of internal buffering. This count will increase if the incoming packet rate is higher than the network adapter processing rate, and if the link flow control is disabled (always when DBL is running).

If this counter has a non-zero value, you should also verify that CPU Frequency Scaling is disabled for the processor.

- **Net Recv PAUSEs**

  The number of ethernet pause packets received.

- **Net recv alternate channel**

  Developer use only.

- **Net Alt drop**

  Developer use only.

- **Ethernet Multicast filter drop**

  The number of packets dropped by the network adapter multicast filter. This can occur when a multicast packet is received and the corresponding multicast group has not been joined by the OS.

- **Ethernet Unicast filter drop**

  The number of packets dropped by the network adapter unicast filter. This can occur when the ethernet destination mac address does not match the network adapter mac address and the network adapter is not in promiscuous mode.

- **Out of send handles**

  Developer use only.

- **User request type unknown**

  Developer use only.

- **Spurious user request**

  Developer use only.

7 Appendix: DBL Software Limitations/ Restrictions

DBL 3.0 does not support the following operations:

- Performance may suffer on workloads that rely on thread-based approaches for demultiplexing incoming packets, especially those where there are more active threads than host CPU cores. Latency-sensitive applications should be careful when using threads as this may involve expensive operating system scheduling which produces less predictable behavior and generally makes OS bypass less effective.
- (Linux-only) **fork()** support is limited and should not be relied upon for transparent sockets acceleration. If connections are open, the parent/child may not be able to continue with DBL. Servers that use the **fork()** + **accept()** model will not result in a child socket being accelerated.
- No UDP loopback on send (applies to multicast and unicast). A process that is subscribed to multicast group A through accelerated UDP sockets on port P will not receive packets that are sent through port P.
- Passing of accelerated sockets from one process to another as regular file descriptors is not supported. On Windows this is typically done via the WSADuplicateSocket call.
- The DBL API semantics do not support loopback communication. Thus, for DBL API mode, the sender port/adapter and receiver port/adapter must be located on different Myri-10G network adapters.
- A network configuration with IP addresses on the same subnet is not supported.

DBL 3.0 has the following restrictions:

- DBL TA mode is restricted to UDP and TCP. Other protocols such as SOCK_RAW are **not** accelerated.
- Loopback communication on the loopback address (e.g IP 127.0.0.1) is only supported in DBL TA mode (dblrun), starting with DBL version 2.1.0.51507. When using loopback with DBL TA mode (dblrun), a socket is demoted/ unaccelerated and the traffic is passed over internal OS dependent paths which may be faster than sending packets via DBL.
- Only busy-polling is supported as means of obtaining low latency and there is no reliance on interrupts to send and receive through accelerated data paths.

poll/select/epoll methods that mix accelerated and non-accelerated sockets may still incur some OS-induced interrupts.

- (Linux-only) Cannot use an accelerated file descriptor as a **FILE *** (i.e. cannot **fdopen** an existing accelerated socket).
- DBL provides limited bonding support (DBL TA mode only and used for failover only) in DBL 2.1.0.51345 and later.