aria
CYBERSECURITY
SOLUTIONS

Machine Vision Accelerator

USER GUIDE

February 19, 2021

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.

PRELIMINARY NOTICE

This document is in the preliminary stage and is subject to change without notice.

Publishing Information

Revision	Date	Details
1.3.0	February 19, 2021	Initial release.

Address

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

https://www.ariacybersecurity.com

Contents

Chapter 1

Introduction

The ARIA™ Machine Vision Accelerator (MVA) software greatly improves the performance of machine vision applications processing data from GigE Vision devices. MVA dramatically reduces the host processor overhead while providing maximum throughput when receiving GigE Vision Stream Protocol (GVSP) content. MVA was designed for easy integration into existing GigE Vision Software Development Kits (SDKs) and native application.

Users of this software should be familiar with the AIA GigE Vision Specification version 1.x, http://www.machinevisiononline.org/, and the GenICam Standard, http://www.genicam.org/.

MVA leverages Myri-10G programmable 10-Gigabit Ethernet network interface cards (NICs) with custom firmware to divert GVSP data directly to user-space memory, bypassing the operating system and legacy network software stacks. MVA offloads the reassembly of GVSP data blocks from individual packets on the wire, avoiding intermediate memory copies and context switch overhead.

MVA is composed of a user library, driver, and firmware running on the embedded processor of the Myri-10G network adapter. The MVA functionality is leveraged through the MVA API, available as a set of C programming language functions. The API documentation is available in PDF and HTML format in /opt/mva/share/doc/ (Linux) or C:\MVA_Myri-10G\share\doc (Windows). Example programs are available in /opt/mva/share/examples/ (Linux) or C:\MVA_Myri-10G\share\examples (Windows).

The MVA software distribution is available for Linux and Windows.

1.1 Benefits

Myricom MVA is a high-performance, hardware-based framer application. MVA runs on a Myricom PCIe NIC that is installed in a PCIe slot of an x86 server. With this hardware, the MVA driver:

- Supports a native network path for communicating over the control plane with an external camera or other network services.
- Supports incoming GigE Vision packets that are handled separately from the general network path. These packets are decapsulated and framed into a specific block format and transferred to the host memory subsystem using Direct Memory Access (DMA). Because these blocks bypass the kernel space, they are immediately accessible in the user space, increasing performance.
- Provides simple software API methods for the host application. With these methods, the driver can manage memory for the blocks, receive the blocks, and gather network information for statistics.
- Supports a poll-based method for monitoring the stream for packets.
- Identifies blocks that are missing one or more packets and handles them differently based on the MVA configuration.

To track packets, the MVA uses IDs and states, which are part of the block and data packets. See the following section for details.

1.2 Blocks

When the MVA adapter receives packets, it frames them into a block consisting of a leader packet, the payload packets, and a trailer packet. Each block is then assigned an ID.

The leader packet consists of metadata pertaining to the block. This metadata includes the block ID and the current state of the block. Possible states are:

- **not-available** – A block is currently not available for the host application to process.
- **available** – The block is complete and is ready for the host application to consume.
- **incomplete** – The block resides in host memory, but one or more payload packets are missing. The return API on the host application then issues a packet resend request to the adapter in an attempt to retrieve the missing and remaining packets. If the packet resend is successful and the block is complete, the state changes to available.
- **incomplete-final** – The block resides in memory and is missing one or more payload packets. The MVA adapter will no longer attempt to retrieve the missing packets, and the block remains incomplete.

The payload portion of the block contains the data packets, each of which is assigned an ID. These IDs are used to frame the block in sequential order and to determine if the block is complete. If no packets are missing, the block is marked as available and consumed by the host application. If, however, a packet is missing, the block is marked as incomplete. Depending on the configuration, the adapter may attempt to retrieve the missing packets and complete the block. This attempt is made several times until the block is completed (i.e., marked as available) or the number of retries has expired. If unsuccessful, the block is marked as incomplete-final, and no more resends are attempted.

NOTE:	The MVA adapter uses the packet resend feature to complete blocks; it does not re-order packets.

The trailer packet marks the end of the block. Once received, and no packets are missing, the block is marked as available. The MVA application then polls the stream for more packets, marking the next block as not-available until the block is full again.

Because all blocks are assigned a block ID, the driver can detect when it receives an unexpected block, or one that is out-of-order. How the driver handles these unexpected blocks is dependent on a flag passed as part of the packet resend request. Possible options are:

- Drop the unexpected block.
- Return the unexpected block to the host application.
- Return only unexpected blocks with an ID greater than the expected ID. Any blocks with IDs less than the unexpected block are dropped.
- Return only those unexpected blocks with an ID less than the expected ID. Any blocks with IDs greater than the unexpected block are dropped.

Chapter 2

Installing

The use of Myricom high-speed, low-latency networking solutions requires installing one or more Myricom ARC Series C-Class network adapters and the MVA software. Once installed, you must activate the software with the appropriate license key.

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

1. Physical installation of network adapter(s) into host computer(s) and connection of appropriate cable.
2. Installation of MVA Network Adapter Driver Software products and configuration of run-time operation.
3. Configuration of Software License Keys on each host.

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:	MVA contains a myri_mva 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. Please contact ARIA Support at ARIA_support@ariacybersecurity.com if you experience difficulty configuring Myricom solutions in unique environments.

2.1 Installing the Adapter

Installing the physical NIC requires installing the adapter into a host system and then installing the drivers so the adapter can communicate over the PCIe bus.

NOTE:	Before installing the adapter, record the MAC address. You may need this to retrieve the IP address of the adapter from your DHCP server.

To install the adapter:

⚠	The adapter contains materials and parts that are susceptible to damage caused by Electrostatic Discharge (ESD). Use static-safe procedures when handling the card. When handing network adapters it is important to follow anti-static procedures to avoid accidentally 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.

1. Close all active applications and shut down the operating system.
2. Turn off the computer and disconnect the power cord.

3. Open the computer chassis to find an available PCIe slot. Make sure not to use a legacy PCI card slot which has different physical dimensions and electrical specifications. 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 *Hardware Installation and Performance on page 100*. 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. Find 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, remove a card to make room for the network adapter.

***Figure 2-1:** x8 and x16 PCIe Slots*

NOTE:	Place the adapter in an expansion slot that is connected to the CPU that will run your application.

4. Locate the shipping package for the MVA adapter and remove the sealed protective sleeve containing the card. Open the seal on the protective sleeve and remove the adapter. Be careful to avoid touching the gold connectors on the bottom edge of the card.
5. 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. 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.

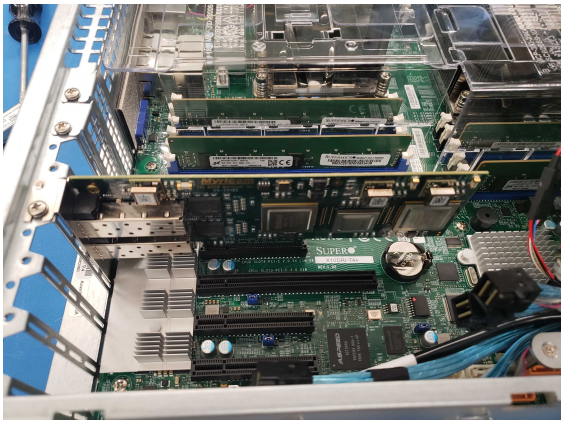


Figure 2-2: Card Inserted

6. Secure the card to the computer case with the screw removed from the bracket plate previously.
7. Close the computer chassis and connect the power cord.
8. Attach appropriate cable to the exposed port(s) on back of the network adapter, taking care not to kink the cable.
9. If using MVA on 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 IRIGB00x 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_MVA driver must be loaded with the myri_timesource=0 load-time option. Otherwise, MVA will return zero for the timestamps. For details, please see *Adapter Timestamps on page 35*.

2.2 Installation Options

There are several options for installing the MVA software and driver. You can either use an InstallAnywhere GUI or the console. The MVA application runs on Windows or Linux.

Linux platforms support local or remote installations. You must, however, be logged into the host machine with root privileges.

Windows platforms support local installations only. You must be logged in using an account with administrative privileges.

2.2.1 Installing MVA Remotely Using GUI

NOTE:	Windows does not support remote installations.

To install MVA remotely InstallAnywhere:

1. Launch the executable file (ARIA_MVA_setup.bin). Once the installer has completed preparing the files, an Introduction window is displayed.

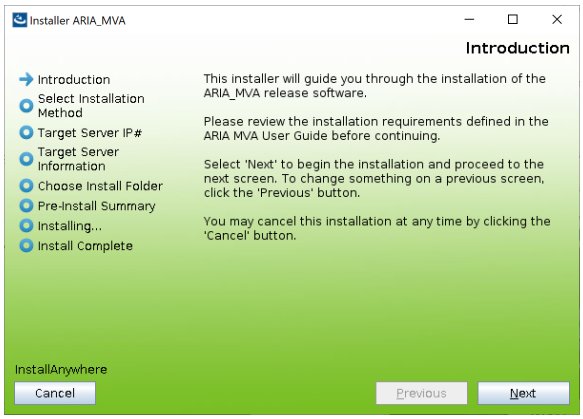


Figure 2-3: InstallAnywhere Introduction

2. Click **Next**. You are prompted select the platform.

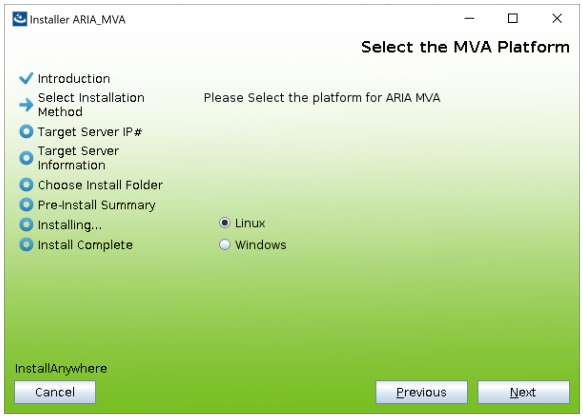


Figure 2-4: Choose Platform

3. Select the Linux and click **Next**. You are prompted to select the installation method.

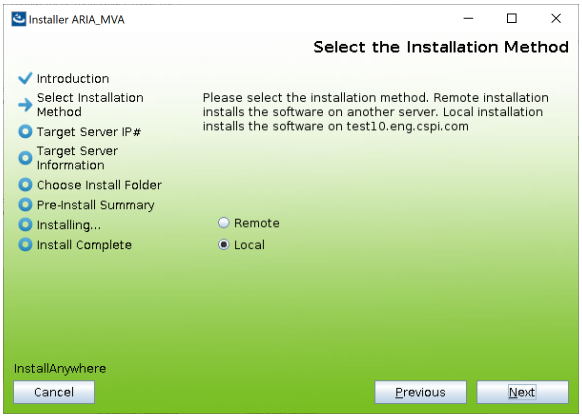


Figure 2-5: Select Installation Method

4. Select Remote and click **Next**. You are prompted to enter the IP address or host name of the target server.

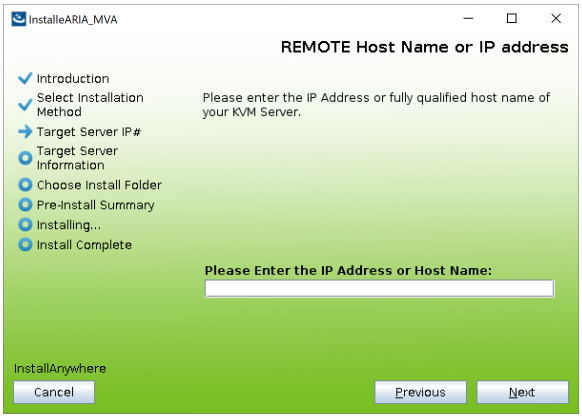


Figure 2-6: Enter IP Address or Host Name

5. Enter the IP address or host name of the host system or VM and click **Next**.

NOTE:	When you click **Next**, the installer attempts to ping the host. If the server does not respond, the installer displays an error message.

The next two screens prompt you for account credentials.

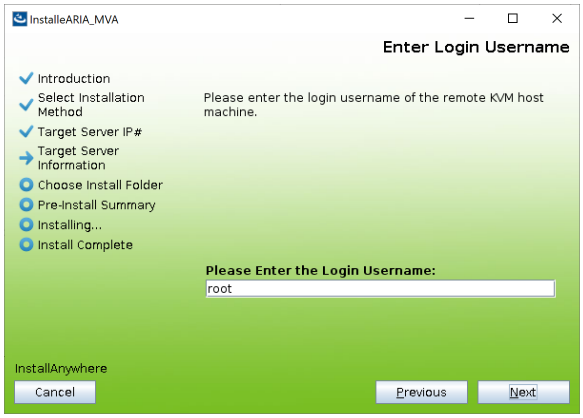


Figure 2-7: Enter Username

6. Enter the username for an account that has root privileges on the host machine and click **Next**.

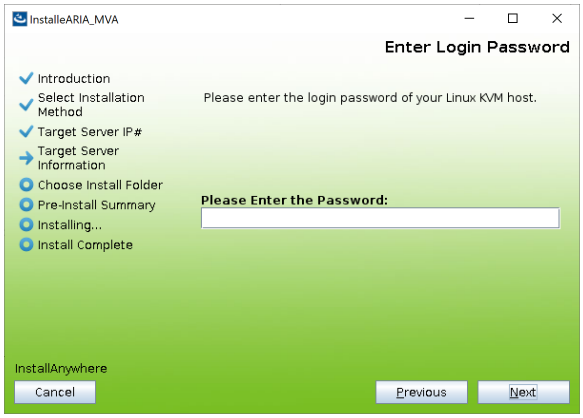


Figure 2-8: Enter Password

7. Enter the password for the account and click **Next**. You are prompted to enter a folder on the local machine for installing documents and supporting files.

NOTE:	When you click **Next**, the installer attempts to log into the host with the credentials you entered. If they are invalid, the installer displays an error message.

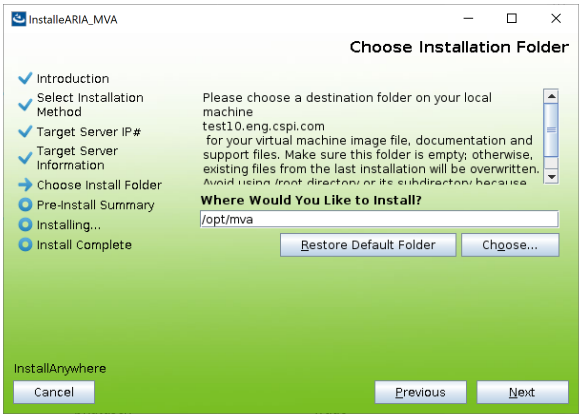


Figure 2-9: Choose Local Folder

8. *Either* accept the default destination folder,

   Or

   Click **Choose** to select a new folder. You can also click **Restore Default Folder** to reset the destination path to the default (opt/mva).

NOTE:	If the account username and password credentials do not have write permissions for the destination folder, the installer displays an error message and will not allow you to continue the installation.

9. Click **Next**. You are prompted to select the installation folder on the remote machine.

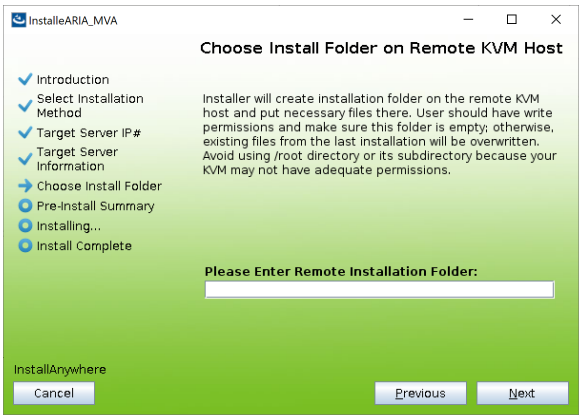


Figure 2-10: Choose Remote Folder

10. Enter the path of the destination folder and click **Next**. A pre-installation summary displays.

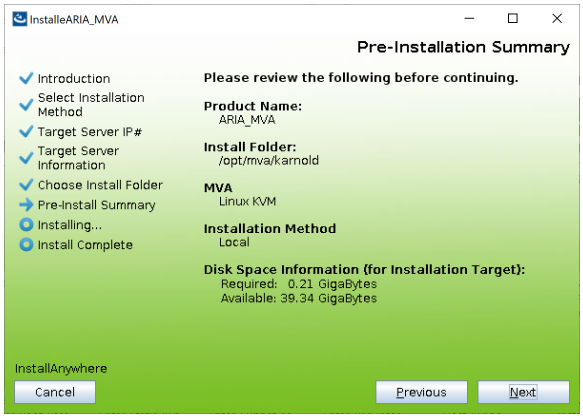


Figure 2-11: Pre-Installation Summary

11. Click **Next** to continue. You are prompted to install the package.

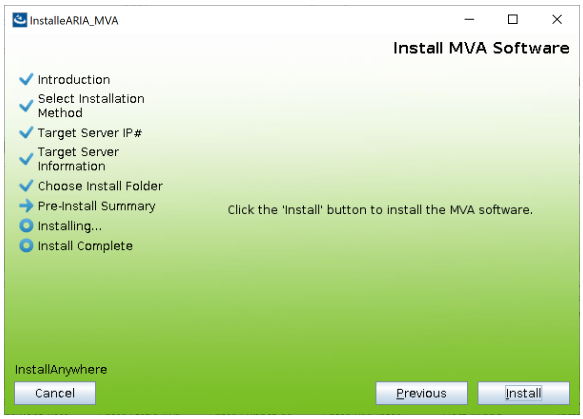


Figure 2-12: Click Install

12. *Either* click **Install** to install the package,

    Or

    Click **Previous** to edit any previous entries.

    The installer displays the progress.

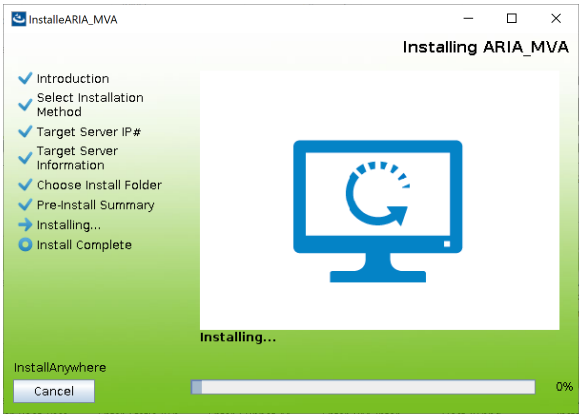


Figure 2-13: Installation Progress

13. Click **OK** when the installation completes to exit the installer.

Figure 2-14: Installation Successful

2.2.2 Installing MVA Locally Using GUI

To install MVA locally using InstallAnywhere:

1. Launch the executable file.

   For Linux, this is ARIA_MVA_setup.bin.

   For Windows, this is ARIA_MVA_setup.exe.

   Once the installer has completed preparing the files, an Introduction window is displayed.

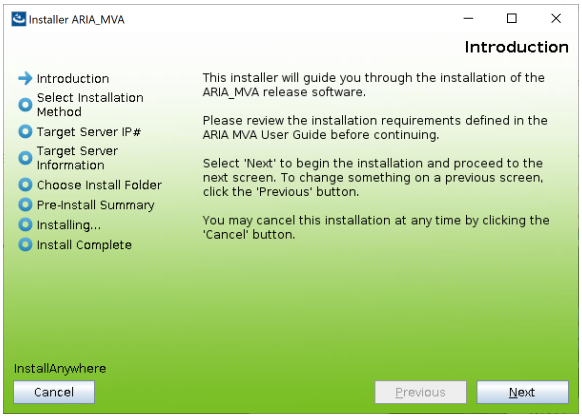


Figure 2-15: InstallAnywhere Introduction

2. Click **Next**. You are prompted select the platform.

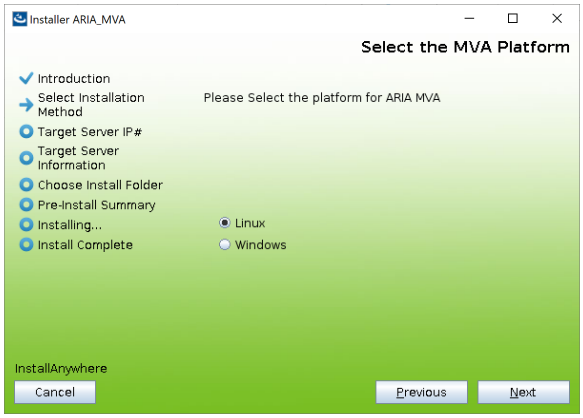


Figure 2-16: Choose Platform

3. Select the platform for the server where the NIC is installed and click **Next**. You are prompted to select the installation method.

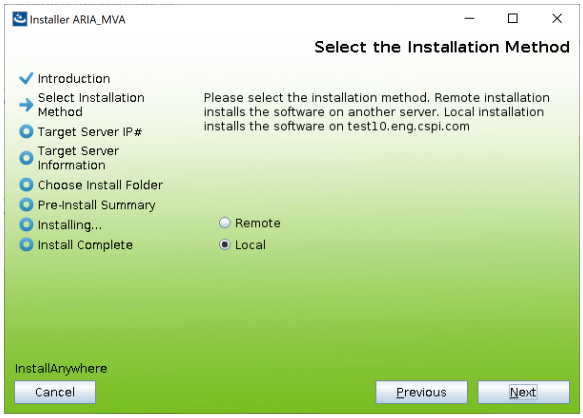


Figure 2-17: Select Installation Method

4. Select Local and click **Next**. You are prompted to enter the installation folder.

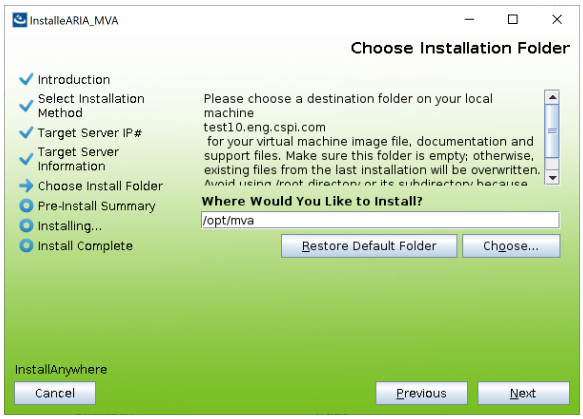


Figure 2-18: Choose Local Folder

5. *Either* accept the default destination folder,

   Or

   Click **Choose** to select a new folder. You can also click **Restore Default Folder** to reset the destination path to the default (opt/mva for Linux and C:\MVA_Myri-10G for Windows).

6. Click **Next**. A pre-installation summary displays.

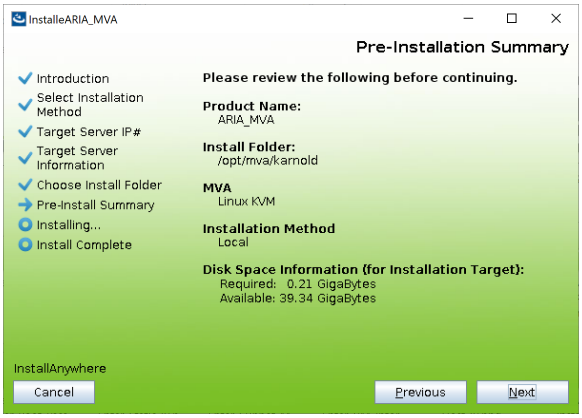


***Figure 2-19:** Pre-Installation Summary*

7. Click **Next** to continue. You are prompted to install the package.

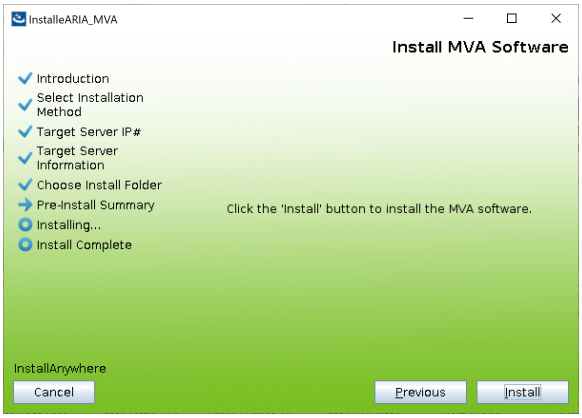


***Figure 2-20:** Click Install*

8. *Either* click **Install** to install the package,

   Or

   Click **Previous** to edit any previous entries.

   The installer displays the progress.

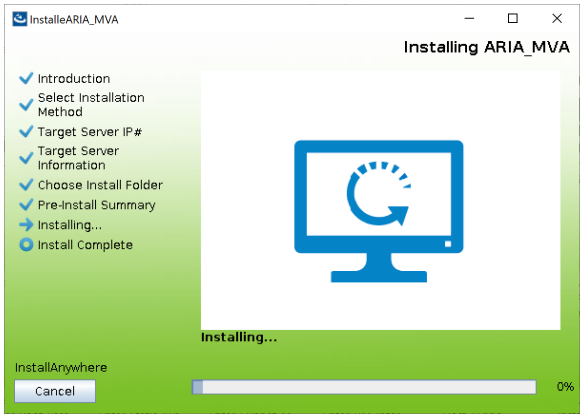


Figure 2-21: Installation Progress

9. Click **OK** when the installation completes to exit the installer.

Figure 2-22: Installation Successful

2.2.3 Installing MVA Using the Console

The installer also supports a command-line interface (CLI). When using the CLI, enter `back` to navigate to a previous prompt. To cancel the installation, type `quit`.

NOTE:	The installer only supports the CLI option on Linux-based machines.

To install MVA using the CLI:

1. Log into the local or remote machine using the root account.
2. Change the directory to the location where ARIA_MVA_setup.bin is stored.
3. Enter the complete file name, followed by `-i console` and press **[ENTER]**. For example, to launch the Linux installer, enter `./ARIA_MVA_setup.bin -i console`. The installer introduction is displayed.

```
===============================================================================
==
Introduction
------------
This installer will guide you through the installation of the ARIA MVA
1.1.0.0 software. Please review the installation requirements defined in
the ARIA MVA User Guide before continuing.
```

```
Respond to each prompt to proceed to the next step in the installation.  If
you want to change something on a previous step, type 'back'.
You may cancel this installation at any time by typing 'quit'.

PRESS <ENTER> TO CONTINUE:
```

4. Press **[ENTER]** to continue. You are prompted to select the platform.

```
================================================================================
Select the MVA Platform
-----------------------
MVA will be installed on a server running Linux KVM.
->1- Linux KVM

ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT:
```

5. Press **[ENTER]** to accept the default (Linux KVM). You are prompted to select the installation method.

```
================================================================================
Installation Method
-------------------
Please select the installation method. Remote installation installs the
software on another server. Local installation installs the software on
<host>. For remote installations, the target server must be running the cor-
rect VM application.

->1- Remote Installation
  2- Local Installation

ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT:
```

6. Select the installation method and follow the prompts. These prompts are identical to those in the GUI.

> For information about deploying MVA with a KVM, see *Installing MVA Remotely Using GUI on page 6*.

Once you have entered the required information and started the installation, the terminal window displays the progress, followed by a notification when the installation completes.

```
================================================================================
Installation Complete
---------------------

Congratulations! ARIA MVA has been successfully installed at <IP>.

Please look at /opt/mva for more information.
```

```
PRESS <ENTER> TO EXIT THE INSTALLER:
```

7. Press **[ENTER]** when the installation completes to exit the installer.

2.3 Passthrough Mode

When using a VM, passthrough mode must be enabled to allow the VM to directly access PCIe devices connected to a host. Information for different VMs is provided in the following sections.

2.3.1 Configuring Passthrough Mode on ESXi VM

VMware ESXi VMs allow a guest operating system on a VM to directly access PCIe devices connected to a host. Each VM can connect to as many as six PCI devices at a time.

Before proceeding, verify:

- The server has Intel Virtualization Technology for Directed I/O (VT-d) or AMD I/O Virtualization Technology (IOV) enabled in the BIOS. This is required to enable DirectPath I/O.
- The virtual machine is running CentOS software version 7.6 or later.
- The MVA adapter is connected to the host and marked as available for passthrough, as described in this section.

To configure passthrough mode:

1. Log into the VMware server.
2. Select **Manage** in the Navigator pane.
3. Click the Hardware tab and PCI Devices to display a list of available devices.

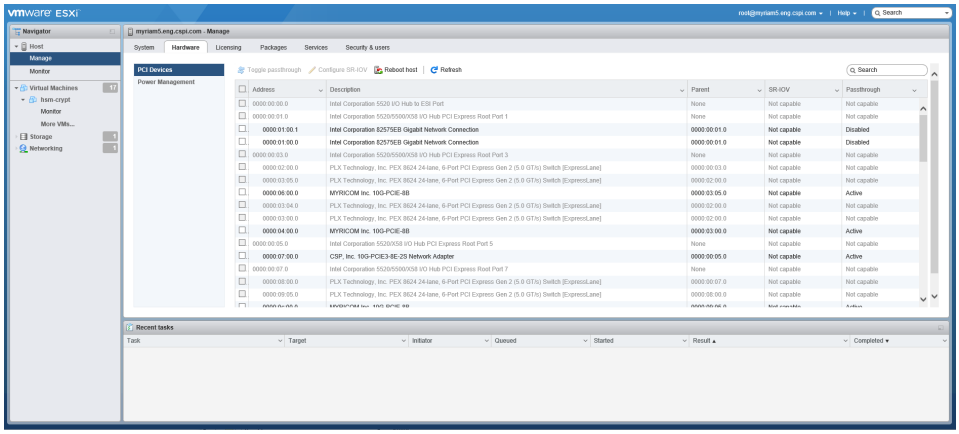


Figure 2-23: PCI Devices

4. Select the adapter in the list. The adapter should display as *Freescale device:Ethernet controller:Freescale Semiconductor Inc Device 8d80*.
5. Click **Toggle passthrough** (). The *Passthrough* value should change to Active.
6. Click **Virtual Machines** in the Navigator pane to display the list of VMs.

7. Right-click on the VM that will use the adapter and select **Edit settings**. The Edit settings dialog box opens.

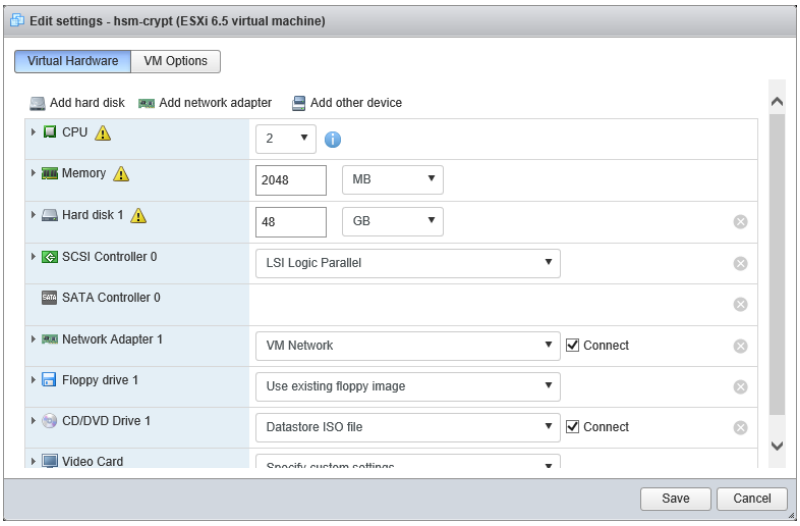


Figure 2-24: Edit VM Settings

8. Click **Virtual Hardware** to display the list of hardware devices the VM can use.
9. Select Add other device > **PCI device**. A new PCI device is added to the hardware list.

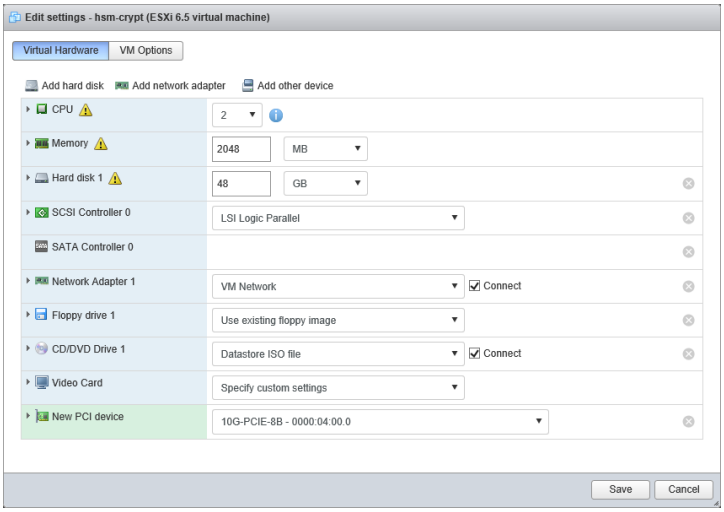


Figure 2-25: Select SIA as New PCI Device

10. Select the SIA model number from the *New PCI device* drop-down box.
11. Click **Save** to save the changes.
12. Cycle power on the VM to have the changes take effect.

Once the MVA adapter is installed and recognized in the VM, you can use IA to install the MVA software and driver.

> For details about installing the software package, see *Installing Drivers and Software on page 1*.

2.3.2 Configuring Passthrough Mode on KVM

KVMs allow a guest operating system on a VM to directly access PCIe devices connected to a host. Each VM can connect to as many as six PCI devices at a time.

Before proceeding, verify:

- The server has Intel Virtualization Technology for Directed I/O (VT-d) or AMD I/O Virtualization Technology (IOV) enabled in the BIOS. This is required to enable DirectPath I/O.
- The virtual machine is running CentOS software version 7.6 or later.
- The MVA adapter is connected to the host and marked as available for passthrough, as described in this section.

To configure passthrough mode:

1. Edit the grub.cfg file to include <processor>_iommu=on and iommu=pt, where <processor> is either intel or amd.

   ```
   vim /etc/default/grub
   "GRUB_CMDLINE_LINUX= intel_iommu=on iommu=pt"
   ```

2. Update the grub file and reboot the VM.

Once the MVA adapter is installed and recognized in the VM, you can use IA to install the MVA software and driver.

> For details about installing the software package, see *Installing Drivers and Software on page 1*.

2.3.3 Configuring Passthrough Mode on Windows VM

If running a Windows VM on a Linux machine, you can enable passthrough mode to allow a guest operating system on a VM to directly access PCIe devices connected to a host. Each VM can connect to as many as six PCI devices at a time.

To configure passthrough mode:

1. Run lspci to determine the bus, slot, and function of the adapter.

   ```
   lspci -nnk | grep MYRI
   ```

   The output shows the bus:slot:function for each card.

   ```
   04:00.0 Ethernet controller [0200]: MYRICOM Inc. Myri-10G Dual-Protocol NIC
   [14c1:0008] (rev 01)
   Subsystem: MYRICOM Inc. 10G-PCIE-8B [14c1:000a]
   Kernel driver in use: myri_mvaKernel modules: myri10ge
   ```

2. Edit the file for the Windows VM.

   ```
   virsh edit <vm-name>
   ```

3. Add the following information, using the bus, slot, and function information for the card to the <devices> section of the file. You must add this information for each port on the card.

   ```
   <hostdev mode='subsystem' type='pci' managed='yes'>
     <driver name='vfio'/>
     <source>
       <address domain='0x0000' bus='0x68' slot='0x00' function='0x0'/>
     </source>
     <alias name='hostdev1'/>
     <address type='pci' domain='0x0000' bus='0x04' slot='0x006' function='0x0'/>
   </hostdev>
   ```

4. Save the file.

Once the MVA adapter is installed and recognized in the VM, you can use IA to install the MVA software and driver.

> For details about installing the software package, see *Installing Drivers and Software on page 1*.

2.4 Configuring the License

A separate license key will be provided by ARIA Cybersecurity Solutions 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 MVA v1.3 licenses at the same time as your network adapters, the licenses will be pre-installed on the adapters and you do not need to follow the myri_license directions in this section of the document. If you are installing MVA on a network adapter that was not purchased with the MVA licenses pre-installed, then you will need to install the license keys in the system.

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:MVA:V1 # s/n 123456
6471-6557-e163-954a:1:123456:MVA:V2.3 # s/n 123456
# limited time trial evaluation of new product
c93b-9fc0-3570-d21b:1:123456:MVA:V1: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.

2.4.1 Activating the License

You should receive an email will be sent from ARIA Support 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 ===
```

To activate the license:

1. Copy the text license records between the file delimiters comments in the email and paste into a new file using a text editor.
2. Save the file with any file name (represented below as <license_file>) in a convenient location.
3. Open a terminal window as user root on Linux or with administrator privileges on Windows and run the following command, supplying the assigned file name.

   Linux

   ```
   /opt/mva/sbin/myri_license -f <license_file>
   ```

   Windows

   ```
   C:\MVA_Myri-10G\sbin\myri_license -f <license_file>
   ```

   The application myri_license is located in the sbin directory of the product(s) you have previously installed. 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 MVA adapters to be licensed, but a single file may be used to hold all licenses for convenience.
4. Verify the software license(s) have been successfully installed.

   Linux

   ```
   /opt/mva/bin/myri_nic_info
   ```

   Windows

   ```
   C:\MVA_Myri-10G\bin\myri_nic_info
   ```
5. If any of the software licenses are listed as invalid, obtain additional diagnosticinformation from the kernel log output (dmesg).

   Linux

   ```
   /opt/mva/bin/myri_nic_info --license
   ```

   Windows

   ```
   C:\MVA_Myri-10G\bin\myri_nic_info --license
   ```

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 Support at ARIA_support@ariacybersecurity.com. Please include the six-digit serial number (or the MAC address) for the network adapter in your inquiry.

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.

2.4.2 Upgrading the License

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.

2.4.3 Maintaining the License for Defective Adapter

In the rare event of failure of a network adapter, contact ARIA Support at ARIA_support@ariacybersecurity.com 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:	It is not necessary to delete the original license key record in the file as it will be silently ignored.

2.4.4 Uninstalling the Driver

To uninstall the driver:

1. Navigate to the folder selected as the destination folder during the installation process. By default, this is /opt/mva (Linux) or C:\MVA_Myri-10G (Windows).
2. Launch Uninstall ARIA_MVA.bin (Linux) or Uninstall ARIA_MVA.exe (Windows). Once the installer has completed preparing the files, you are prompted to uninstall the product.

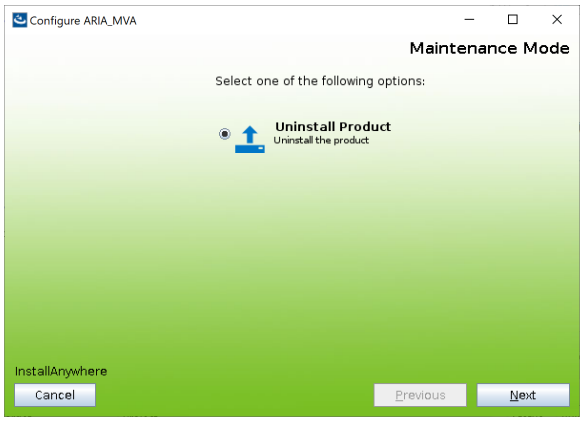


Figure 2-26: Uninstallation Screen

3. Click **Next** to continue. A summary is displayed.

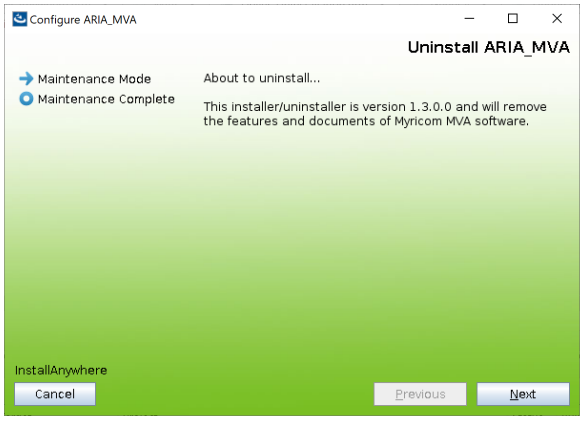


Figure 2-27: Uninstallation Summary

4. Click Next to continue. You are prompted to uninstall the product.

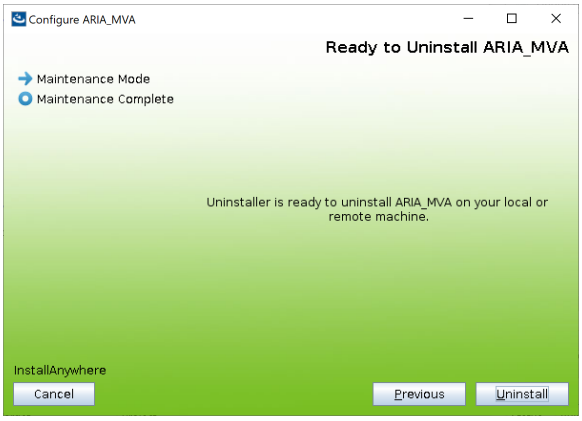


Figure 2-28: Click Uninstall

5. Click **Uninstall**. When the uninstallation completes, the screen indicates if the action was successful.

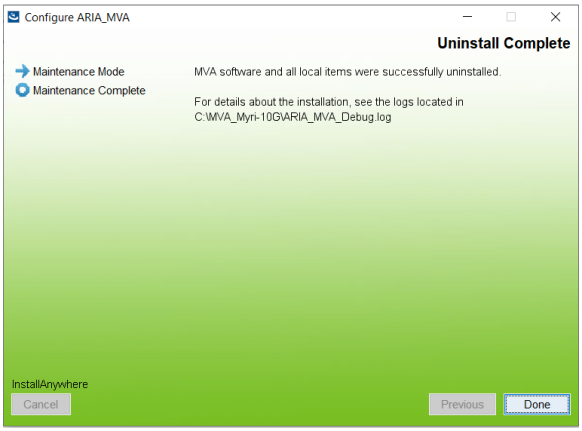


Figure 2-29: Uninstall Complete

6. Click **Done** to exit the application.

You can uninstall the software manually; however, you should only consider this method if using InstallAnywhere was not successful.

To uninstall the MVA software manually on Windows:

1. Open Device Manager (Control Panel > **Device Manager**), and uninstall all Myricom adapters under Network Adapters.
2. Remove INF and PNF files. This procedure removes all references to the driver and prevents a stale file from being used in the future. To do this:
   a. Open a command window (Start > Run > **cmd**), and type the following commands.

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

   b. Delete all Myricom oem files. For example, the following deletes oem6 if found.

```
del oem6*
```

c. Repeat steps a and b to delete the PNF files.

3. Delete sys and dat files.

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

4. Restart the machine.

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

Once the machine reboots, Windows will try to find the driver for the MVA adapters. It should fail because you deleted the .inf and .pnf files.

2.5 Verifying Card Functionality

The myri_nic_info program retrieves adapter information, such as the hardware serial number, MAC address, part number, firmware version, and license information. This helps verify the card is properly installed and recognized by the host.

To retrieve this information in Linux, enter the command in a terminal window.

```
sudo /opt/mva/bin/myri_nic_info
```

To retrieve this information in Windows, enter the command in a terminal window.

```
C:\MVA_Myri-10G\bin\myri_nic_info.exe
```

The console displays the card information, as shown below.

```
# Serial  MAC               ProductCode     Driver Version License
0 491942  00:60:dd:43:48:b3 10G-PCIE2-8B-2S myri_mva-1.3   Valid
1 491943  00:60:dd:43:48:b4 10G-PCIE2-8B-2S myri_mva-1.3   Valid
```

> If no information is returned, see *Troubleshooting on page 99*.

Chapter 3

Testing the Adapter

After the successful installation of the MVA software, it is recommended that some simple tests be performed to verify MVA operation on your systems and provide basic familiarization with normal operation. A variety of test/example programs and tool programs are available and detailed in *Test and Diagnostic Tools on page 75*.

3.1 Identifying Adapter LEDs

Each adapter has two LEDs per port: one green and one amber. These LEDs light to indicate the status of the adapter and network traffic. See the following table for a description of the different colors/states.

Color	State	Indicates
Amber	Blinking Twice Per Second	The adapter is powered on and the LANAI firmware is functioning.
	Off	The adapter has not established a PCIe link or the operating system has not booted yet.
	Solidly Lit	The EEPROM is corrupted. Contact ARIA Support for assistance.
	Blinking Rapidly	There is an issue with the EEPROM. Contact ARIA Support for assistance.
	Blinking Rapidly Five Times, Followed By a Pause	The adapter failed to establish a PCIe link. Contact ARIA Support for assistance.
Green	Solidly Lit	The adapter is powered on and has established a link.
	Off	The adapter has not established a link or Linux has not booted yet.
	Blinking	There is active traffic on the port.

***Figure 3-1:** Port LEDs*

3.2 MVA Testing on Linux

Tests/example programs are available in/opt/mva/bin/tests of the install directory in binary form and in share/examples in source form. The file mva_simple_recv is a simple GVSP receiver showing basic functionality of receiving and counting GVSP data blocks for a single stream. This test is sufficient to verify correct software installation and network adapter hardware connectivity. Please see the usage summary of the mva_simple_recv program on *page 76* for an explanation of the various options.

If a source of GVSP data is available, that may be used to direct traffic into the receiver. If not, a capture file is provided which may be rewritten and replayed with the tcpreplay command to generate traffic.

For best performance, disabling power saving modes in the host BIOS is recommended. Relatively long delays in PCIe response time have been observed on Windows with power saving modes enabled, which can cause network drop overflows. This is especially true for the "Intel C-State" power savings mode. On both systems, ensure /opt/mva/bin/tests/ has been added to the current PATH variable.

These examples assume that IP addresses have been assigned on two 10GbE hosts, with 10.0.0.1 for the Myri-10G MVA-enabled receiver and 10.0.0.2 for the sender on interface eth2. The receiver MAC address is 00:11:22:33:44:55. Use myri_info (see *page 89*), ifconfig (Linux), or ipconfig /all (Windows) to determine the actual MAC address. The connection between the 2 10GbE network adapters is point-to-point and connectivity should be verified with ping.

To rewrite and replay a capture file:

NOTE:	As of the date of this document, a beta version of tcpreplay is available for Windows. See http://tcpreplay.appneta.com/wiki/installation.html for more information.

1. Install tcpreplay and companion commands.

   ```
   sudo yum install tcpreplay
   ```

2. Rewrite the provided capture file stored in /opt/mva/share/examples/.

   ```
   cp /opt/mva/share/examples/gvsp_3frame_9000.cap mvatmp1.cap
   tcpprep --cidr=10.0.0.2 --pcap=mvatmp1.cap --cachefile=mva.cache
   tcprewrite --endpoints=10.0.0.2:10.0.0.1 -c mva.cache --infile=mvatmp1.cap
    --outfile=mvatmp2.cap
   tcprewrite --enet-dmac=00:11:22:33:44:55 --infile=mvatmp2.cap
    --outfile=mva.cap
   rm mvatmp?.cap
   ```

   To run mva_simple_recv:

   1. On the receiving system, run the simple GVSP receiver to accept then drop a single GVSP data block.

      ```
      mva_simple_recv -v 10.0.0.1:60000
      ```

   2. Replay the mva.cap file from the sender.

      ```
      sudo tcpreplay -i eth2 mva.cap
      ```

   If successful, the results will indicate that one data block was received. To obtain diagnostic information from the MVA library, set the MVA_DEBUG and MVA_VERBOSE environment variables prior to the execution of the test program.

   > For more information about variables, see *Settings on page 31*.

```
pkt: 10000, len: 60, ts_hw: 1460582457723094899 ts_host: 1460582457723432378 ts_
diff:337479
pkt: 20000, len: 60, ts_hw: 1460582457727899449 ts_host: 1460582457728224074 ts_
diff:324625
pkt: 30000, len: 60, ts_hw: 1460582457732705989 ts_host: 1460582457733025483 ts_
diff:319494
```

```
pkt: 40000, len: 60, ts_hw: 1460582457737507879 ts_host: 1460582457737826826 ts_
diff:318947
pkt: 50000, len: 60, ts_hw: 1460582457742313319 ts_host: 1460582457742627269 ts_
diff:313950
pkt: 60000, len: 60, ts_hw: 1460582457747118579 ts_host: 1460582457747428999 ts_
diff:310420
pkt: 70000, len: 60, ts_hw: 1460582457751923409 ts_host: 1460582457752229809 ts_
diff:306400
pkt: 80000, len: 60, ts_hw: 1460582457756729069 ts_host: 1460582457757032338 ts_
diff:303269
pkt: 90000, len: 60, ts_hw: 1460582457761532689 ts_host: 1460582457761832804 ts_
diff:300115
```

3.3 MVA Testing on Windows

Tests\example programs are available in C:\MVA_Myri-10G\bin\tests of the install directory in binary form and in share\examples in source form. The file mva_simple_recv.exe is a simple GVSP receiver showing basic functionality of receiving and counting GVSP data blocks for a single stream. This test is sufficient to verify correct software installation and network adapter hardware connectivity. Please see the usage summary of the mva_simple_recv.exe program on *page 76* for an explanation of the various options.

If a source of GVSP data is available, that may be used to direct traffic into the receiver. If not, a capture file is provided which may be rewritten and replayed with the tcpreplay command to generate traffic.

For best performance, disabling power saving modes in the host BIOS is recommended. Relatively long delays in PCIe response time have been observed on Windows with power saving modes enabled, which can cause network drop overflows. This is especially true for the “Intel C-State“ power savings mode. On both systems, ensure C:\MVA_Myri-10G\bin\tests has been added to the current PATH variable.

These examples assume that IP addresses have been assigned on two 10GbE hosts, with 10.0.0.1 for the Myri-10G MVA-enabled receiver and 10.0.0.2 for the sender on interface eth2. The receiver MAC address is 00:11:22:33:44:55. Use myri_info (see *page 89*) or ipconfig /all to determine the actual MAC address. The connection between the two 10GbE network adapters is point-to-point and connectivity should be verified with ping.

To rewrite and replay a capture file:

NOTE:	As of the date of this document, a beta version of tcpreplay is available for Windows. See http://tcpreplay.appneta.com/wiki/installation.html for more information.

1. Install tcpreplay and companion commands.
2. Rewrite the provided capture file stored in C:\MVA_Myri-10G\share\examples.

```
cp \MVA_Myri-10G\share\examples\gvsp_3frame_9000.cap mvatmp1.cap
tcpprep --cidr=10.0.0.2 --pcap=mvatmp1.cap --cachefile=mva.cache
tcprewrite --endpoints=10.0.0.2:10.0.0.1 -c mva.cache --infile=mvatmp1.cap
 --outfile=mvatmp2.cap
```

```
tcprewrite --enet-dmac=00:11:22:33:44:55 --infile=mvatmp2.cap
 --outfile=mva.cap
rm mvatmp?.cap
```

To run mva_simple_recv:

1. On the receiving system, run the simple GVSP receiver to accept then drop a single GVSP data block.

   ```
   mva_simple_recv -v 10.0.0.1:60000
   ```

2. Replay the mva.cap file from the sender.

   ```
   tcpreplay -i eth2 mva.cap
   ```

If successful, the results will indicate that one data block was received. To obtain diagnostic information from the MVA library, set the MVA_DEBUG and MVA_VERBOSE environment variables prior to the execution of the test program.

> For more information about variables, see *Settings on page 31*.

```
pkt: 10000, len: 60, ts_hw: 1460582457723094899 ts_host: 1460582457723432378 ts_
diff:337479
pkt: 20000, len: 60, ts_hw: 1460582457727899449 ts_host: 1460582457728224074 ts_
diff:324625
pkt: 30000, len: 60, ts_hw: 1460582457732705989 ts_host: 1460582457733025483 ts_
diff:319494
pkt: 40000, len: 60, ts_hw: 1460582457737507879 ts_host: 1460582457737826826 ts_
diff:318947
pkt: 50000, len: 60, ts_hw: 1460582457742313319 ts_host: 1460582457742627269 ts_
diff:313950
pkt: 60000, len: 60, ts_hw: 1460582457747118579 ts_host: 1460582457747428999 ts_
diff:310420
pkt: 70000, len: 60, ts_hw: 1460582457751923409 ts_host: 1460582457752229809 ts_
diff:306400
pkt: 80000, len: 60, ts_hw: 1460582457756729069 ts_host: 1460582457757032338 ts_
diff:303269
pkt: 90000, len: 60, ts_hw: 1460582457761532689 ts_host: 1460582457761832804 ts_
diff:300115
```

3.4 MVA Performance Testing on Linux

For best performance, disabling power saving in the host BIOS is recommended. This is especially true for the “Intel C-State“ power savings mode.

The expected performance of the MVA software can be measured using the /opt/mva/bin/tests/mva_simple_recv test program. Example capture files for this test can be found in /opt/mva/share/examples/.

The CPU usage should be measured while receiving data at 10GbE line rate and also check that no packets were dropped. The sending network adapter should send at line rate.

On the receiver:

```
mva_simple_recv -i -1 -t 20
```

On the sender, window 1:

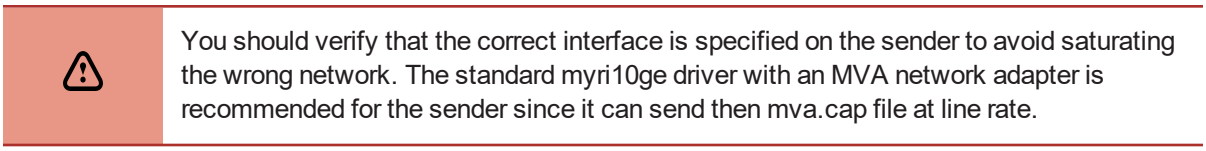
```
mpstat -P ALL 1 (or vmstat)
```

On the sender, window 2:

```
$ tcpreplay -t -K -l 50000 -i eth2 mva.cap
```

Verify CPU usage and that there were no packets dropped.

> For an explanation of the MVA counters reported by /opt/mva/bin/myri_ counters, please see *MVA Counters on page 115*.



3.5 MVA Performance Testing on Windows

For best performance, disabling power saving in the host BIOS is recommended. Relatively long delays in PCIe response time have been observed on Windows with power saving modes enabled, which can cause network drop overflows. This is especially true for the "Intel C-State" power savings mode.

The expected performance of the MVA software can be measured using the C:\MVA_Myri-10G\bin\tests\mva_simple_recv test program. Example capture files for this test can be found in C:\MVA_Myri-10G\share\examples.

The CPU usage should be measured while receiving data at 10GbE line rate and also check that no packets were dropped. The sending network adapter should send at line rate.

On the receiver:

```
mva_simple_recv -i -1 -t 20
```

On the sender (Linux), window 1:

```
perfmon.msc
```

On the sender (Linux), window 2:

```
tcpreplay -t -K -l 50000 -i eth2 mva.cap
```

Verify CPU usage and that there were no packets dropped.

> For an explanation of the MVA counters reported by C:\MVA_Myri-10G\bin\myri_counters, please see *MVA Counters on page 115*.

You should verify that the correct interface is specified on the sender to avoid saturating the wrong network. The standard myri10ge driver with a Myri-10G network adapter is recommended for the sender since it can send then mva.cap file at line rate.

3.6 Programming MVA Applications

You should be familiar with the AIA GigE Vision Specification version 1.0, http://www.machinevisiononline.org/, and the GenICam. Standard, http://www.genicam.org/. The MVA API document is included in the software distribution in /opt/mva/share/doc/ (Linux) or C:\MVA_Myri-10G\share\doc (Windows). All of the necessary software definitions are included in /opt/mva/include/mva.h (Linux) or C:\MVA_Myri-10G\include\mva.h (Windows) and the MVA library is located in /opt/mva/lib/libmva.so (Linux) or C:\MVA_Myri-10G\lib\libmva.dll (Windows). The library in /opt/mva/lib/sockemul/libmva.so (Linux only) is an implementation of the MVA library using only sockets. It can be used to run MVA applications on any type of network adapter for testing and comparison to the accelerated library implementation (/opt/mva/lib/libmva.so or \MVA_Myri-10G\lib\mva.dll), which is Myri-10G specific.

For example usage with the MVA API, refer to the /opt/mva/share/examples/test (Linux) or C:\MVA_Myri-10G\share\examples (Windows) programs mva_simple_recv.c, mva_multi_thread.c (Linux only), mva_multi_dev.c (Linux only), mva_sock_emul.c (Linux only), and tests.c (Linux only).

To compile an MVA application on Linux, execute the following command.

```
gcc -o myprog myprog.c -I /opt/mva/include -L /opt/mva/lib \-lmva -lpthread
```

3.7 Programming Multi-Threaded MVA Applications

Without any serialization, mva_stream_open() and all function calls using the returned stream handle must be executed on the same thread. However, independent threads can handle separate streams without locking. This usage is beneficial when there are multiple devices because threads can be assigned CPU affinity to promote load balancing. See the mva_multi_dev.c example code in /opt/mva/share/examples/ (Linux) or C:\MVA_Myri-10G\share\examples (Windows) for reference.

MVA API function calls for a given stream handle are not thread-safe. The application is responsible for serializing access to MVA API functions that are using the same stream handle. This usage is beneficial when there is one or several high speed devices and frame processing needs to be spread among CPU cores. See the mva_multi_thread.c example code in /opt/mva/share/examples/ (Linux) or C:\MVA_Myri-10G\share\examples (Windows) for reference.

Chapter 4

Settings

The MVA API supports three main functions for opening a stream. These functions are:

- **mva_open_stream()** – Opens a single stream without support for packet resend.
- **mva_open_mcast_stream()** – Opens a multicast stream without support for packet resend.
- **mva_open_stream_pr()** – Opens one or more streams with packet resend support.

Each of these functions support different arguments, each of which is described in the MVA API Reference Manual. There are, however, some flags, environment variables, and a configuration structure that one or more of these functions use. Each of these is described in the following sections.

Component	Description	Page
flags	Sets the MVA_OPEN flag value for the mva_open_stream() and mva_open_mcast_stream() functions. This is part of the configuration structure for the mva_open_stream_pr() function.	31
Environmental Variables	Enables flags while the application runs.	32
Configuration Structure	Defines how the MVA application handles packet resend requests. This applies to the mva_open_stream_pr() function only.	32

Table 1: Function Options

Each of these is described in the following sections.

4.1 flags

Defines how the MVA adapter handles incomplete blocks. If using mva_open_stream_pr(), the flag is passed as part of the packet resend configuration structure. For mva_open_stream() and mva_open_mcast_stream(), the flag is passed as an argument. See the following table for the available flags, their values, and how they impact the MVA adapter functionality.

Flag	Value	Function
MVA_OPEN_ZEROLOSS	0x2	Enables the packet resend feature. This is valid only for the mva_open_stream_pr(). The mva_open_stream() and mva_open_mcast_str() functions ignore this flag.
MVA_OPEN_DROP_INCOMPLETE	0x4	Instructs the adapter to drop any incomplete blocks. If set, the packet resend feature is disabled. This is the default value for all three mva_open functions.
MVA_OPEN_RETURN_INCOMPLETE	0x8	Returns all incomplete blocks to the host application. If set, the packet resend feature is disabled.

Table 2: flags

Although you can specify more than one flag, the adapter ignores the MVA_OPEN_ZEROLOSS flag if set for mva_open_stream() or mva_open_multicast_stream().

4.2 Environment Variables

The MVA API supports two environment variables that assist with troubleshooting. These variables are as follows:

- **MVA_VERBOSE** – Enables verbose mode for logging purposes.
- **MVA_DEBUG** – Enables debug mode, which provides more information than verbose mode, for logging purposes.

Setting these variables to 1 (0x1) enables the flag. For example, the following enables the MVA_DEBUG flag from the command line in Linux.

```
export MVA_DEBUG=0x1
```

NOTE:	You should only set these flags if instructed to do so by ARIA Support.

To set environment variables in Windows, use Windows PowerShell. For example, the following enables the MVA_VERBOSE flag in Windows.

```
Set-Content env:MVA_VERBOSE 1
```

> For more information about environment variables, see *Environment Variables on page 32*.

The flag remains enabled until you unset the flag (i.e., set it back to 0) or you exit the shell.

4.3 Configuration Structure

The mva_open_stream_pr() funciton passes a structure, mva_packet_resend_config, that defines how the adapter handles blocks in terms of dropped packets and unexpected blocks. In addition, you can configure different timers and the maximum number of retries provided the correct flag is set.

See the following table for a summary of the contents of this structure.

Field	Description	Page
flags	Sets the MVA_OPEN flag value.	*31*
block_out_of_order	Sets the MVA_BOOO flag value.	*33*
block_timeout	Defines the number of milliseconds the adapter will wait to complete a block before determining it is incomplete. This is only valid if the MVA_OPEN_ZEROLOSS flag is set.	*33*
resend_timeout	Defines the number of milliseconds the adapter will wait for a response to the packet resend request. This is only valid if the MVA_OPEN_ZEROLOSS flag is set.	*33*
resend_retries	Sets the total number of times the adapter will attempt packet resend requests. This is only valid if the MVA_OPEN_ZEROLOSS flag is set.	*34*

Field	Description	Page
camera_ipv4	Identifies the IP address of the camera.	34
camera_port	Indicates the port to use for camera data.	34
stream_channel_id	Specifies the channel ID of the camera.	34

Table 3: mva_packet_resend_config Fields

See the following sections for the fields that are applicable to mva_open_stream_pr() only.

4.3.1 block_out_of_order

Defines how the MVA adapter handles blocks with an unexpected ID. See the following table for the available flags, their values, and how they impact the MVA adapter functionality.

Flag	Value	Function
MVA_BOOO_ALWAYS_DROP	0x1	Drops any blocks with an unexpected ID.
MVA_BOOO_ALWAYS_RETURN	0x2	Returns all blocks with an unexpected ID to the caller (host application). This is the default flag.
MVA_BOOO_LT_ALWAYS_ RETURN	0x4	Returns blocks with IDs that are less than the next expected block. Any blocks with IDs greater than the next expected block are dropped.
MVA_BOOO_GT_ALWAYS_ RETURN	0x8	Returns blocks with IDs that are greater than the next expected block. Any blocks with IDs less than the next expected block are dropped.

Table 4: block_out_of_order

Because only one flag can be enabled at any time, valid values are 0, 1, 2, 4, and 8.

4.3.2 block_timeout

Specifies the number of milliseconds the polling function on the host application will wait for a block to complete. If this timer expires, the block is marked as incomplete-final, and no packet resend requests are issued. This field is valid only if the MVA_OPEN_ZEROLOSS flag is set. The valid range is 0-4,294,967,295, where 0 indicates to wait indefinitely. This value must be 0 or greater than the value specified in the resend_timeout argument.

NOTE:	If the timeouts (resend_timeout and block_timeout) and the resend_retries arguments are specified, at least one must be non-zero. You cannot assign 0 to all three arguments.

4.3.3 resend_timeout

Specifies the number of milliseconds the polling function on the host application will wait for packets after the packet resend is issued. This field is valid only if the MVA_OPEN_ZEROLOSS flag is set. The valid range is 0-4,294,967,295, where 0 indicates to wait indefinitely. This value must be less than the value specified in the block_timeout argument.

NOTE:	If the timeouts (resend_timeout and block_timeout) and the resend_retries arguments are specified, at least one must be non-zero. You cannot assign 0 to all three arguments.

4.3.4 resend_retries

Defines the maximum number of packet request attempts the adapter will allow. This field is valid only if the MVA_OPEN_ZEROLOSS flag is set. The valid range is 1-4,294,967,295, where 0 indicates unlimited retries. If set to 0, at least one timeout argument (block_timeout or resend_timeout) must be specified.

NOTE:	If the timeouts (resend_timeout and block_timeout) and the resend_retries arguments are specified, at least one must be non-zero. You cannot assign 0 to all three arguments.

4.3.5 camera_ipv4

Specifies the unicast IP v4 address of the camera that will receive the packet resend requests. Because this argument accepts an integer, you must first convert the IP address to hexadecimal. To do this, change each octet to hexadecimal and remove the dots.

For example, 192.168.0.100 converts to 0xC0A80064 as shown below.

192.168.0.11

= 0xC0.0xA8.0x00.0x64

= 0xC0A80064

4.3.6 camera_port

Specifies the port used to send packet resend requests. The valid range is 0-65535.

4.3.7 stream_channel_id

Identifies the channel used for packet resend requests. The valid range is 0-65535.

4.3.8 Adapter Timestamps

MVA supports socket timestamps on the MVA adapters. Timestamping support is available in two modes: host timestamping mode and timesource timestamping mode. Host timestamping mode is available for all supported MVA network adapters. Timesource (hardware) timestamping mode is only available for 10G-PCIE2-8C2-2S-SYNC network adapters connected to a timecode generator.

For SYNC adapter support, three parameters are available in the MVA module:

- **myri_timesource** – By default, the MVA 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 MVA driver must be loaded with the myri_timesource=0 load-time option. Otherwise, MVA will return zero for the timestamps.

  To use host timestamps on the SYNC adapter , include the myri_timesoure option when starting MVA.

  Linux

  ```
  /opt/mva/sbin/myri_start_stop start myri_timesource=0
  ```

  Windows

  ```
  C:\MVA_Myri-10G\sbin\myri_start_stop start myri_timesource=0
  ```

  On non-SYNC adapters, the default is to use host timestamping.

- **myri_timesource_loss_mode** – By default, the timesource loss mode is set to free-running. To zero the timestamp, include the myri_timesource_loss_mode option when starting the MVA driver.

  Linux

  ```
  /opt/mva/sbin/myri_start_stop start myri_timesource_loss_mode=0
  ```

  Windows

  ```
  C:\MVA_Myri-10G\sbin\myri_start_stop start myri_timesource_loss_mode=0
  ```

  By default, myri_timesource_loss_mode is set to 1.

- **myri_timesource_pps** – You can disable exporting a packets per second (pps) interface for use by NTP. To do this, include the myri_timesource_pps option when starting the driver. =0 to disable the exporting of a pps interface that can be used by ntp.

  Linux

  ```
  /opt/mva/sbin/myri_start_stop start myri_timesource_pps=0
  ```

  Windows

  ```
  C:\MVA_Myri-10G\sbin\myri_start_stop start myri_timesource_pps=0
  ```

  By default, exporting the pps interface is enabled (myri_timesource_pps is set to 1).

The timestamp that is made available through the socket interface (using the SO_TIMESTAMP 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 freerunning counter and is attached to the packet as it is transferred to the host. 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 clock source with the network adapter clock source to account for clock skew.

Windows does not support SO_TIMESTAMP at the Network Driver Interface Specification (NDIS) level, but this feature was added 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).

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 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 clock source.

4.3.9 Synchronization

There are different levels of timestamp synchronization.

1. Host-to-host synchronization (global sync)
2. Network adapter-to-host synchronization (local sync)

Most systems use something like ntpd (NTP daemon) to implement host-to-host synchronization. Using Precision Time Protocol (PTP) provides more precise options, but it only moves the accuracy from milliseconds to tens of microseconds. Because both protocols are not typically run on dedicated networks and there is typically a lot of host overhead in processing the protocol, the time can only be so accurate.

MVA 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 local synchronization by connecting the network adapter with a coax connector that carries an IRIGB 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.

> For details of the hardware requirements with the 10G-PCIE2-8C2-2S-SYNC adapter, see *Installing the Adapter on page 3*.

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_mva 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 MVA API 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. In this case, the "raw" timestamps are also absolute timestamps because of the additional hardware on the board.
2. Users of the MVA API 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 MVA 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_support@ariacybersecurity.com.

Chapter 5

Tuning

The performance of the MVA application in real customer environments will vary depending upon the particular details of the network configuration, end-user applications, and transaction workloads. This chapter provides several recommendations designed to achieve the best performance with the MVA adapter.

For best performance, disabling power saving modes in the host BIOS is recommended. Relatively long delays in PCIe response time have been observed on Windows with power saving modes enabled, which can cause network drop overflows. This is especially true for the "Intel C-State" power savings mode.

The expected performance of MVA is detailed in *MVA Performance Testing on Linux on page 28* and *MVA Performance Testing on Windows on page 29*. Contact ARIA Support at ARIA_support@ariacybersecurity.com for further assistance if results are below expectations.

5.1 Before You Begin

Some of the steps defined in this document require you to modify the Linux kernel parameters and reboot your server. Because the boot loader options are different in RHEL/CentOS systems, you must determine the location of the grub.cfg file. In addition, certain Linux packages are required. Instructions for installing these packages as well as determining which boot loader to use are provided in the following sections.

5.1.1 Determining the Location of the Config File

NOTE:	This section applies to systems running CentOS/RedHat.

To determine the location of grub.cfg:

1. List the contents of the boot folder.

   ```
   sudo ls /boot
   ```

2. Determine the system type or boot loader based on the presence of one of the following directories:
   - efi – The system is UEFI-based, and the grub.cfg file is located in /boot/efi/EFI/centos/.
   - grub – The system uses a grub boot loader, and the grub.cfg file is located in /boot/grub/.
   - grub2 – The system uses a grub2 boot loader, and the grub.cfg file is located in /boot/grub2/.

For example, the following output indicates the system is not UEFI-based, and it uses the grub2 boot loader.

```
vmlinuz-3.10.0-957.10.1.el7.x86_64   grub2/
```

5.1.2 Installing Linux Packages

Tuning performance requires two additional Linux packages: tuned and hwloc. In addition, tuna is recommended for CentOS/RedHat. Installing these packages differ for CentOS/RedHat and Ubuntu/Debian systems.

To install the packages on CentOS/RedHat, use yum.

```
sudo yum install hwloc-gui tuned tuna
```

To install the packages on Ubuntu/Debian, use apt.

```
sudo apt update
sudo apt install hwloc tuned-utils
```

5.2 NUMA Nodes and Cores

Some of the recommendations in this guide rely on identifying NUMA nodes and the cores that are assigned to those nodes. Before continuing, make sure you are familiar with the commands used to determine to which NUMA node the adapter is assigned as well as the cores associated with that node.

5.2.1 Finding the NUMA Node Associated with the Adapter

In a NUMA-aware system, you should first determine the NUMA node where the adapter resides. This requires the hwloc package.

> For more information about installing the hwloc package, see *Installing Linux Packages on page 40*.

Once the package is installed, you can view the hardware topology.

```
lstopo
```

Depending on your system, the topology may be displayed in a graphical format as shown in the following figure.

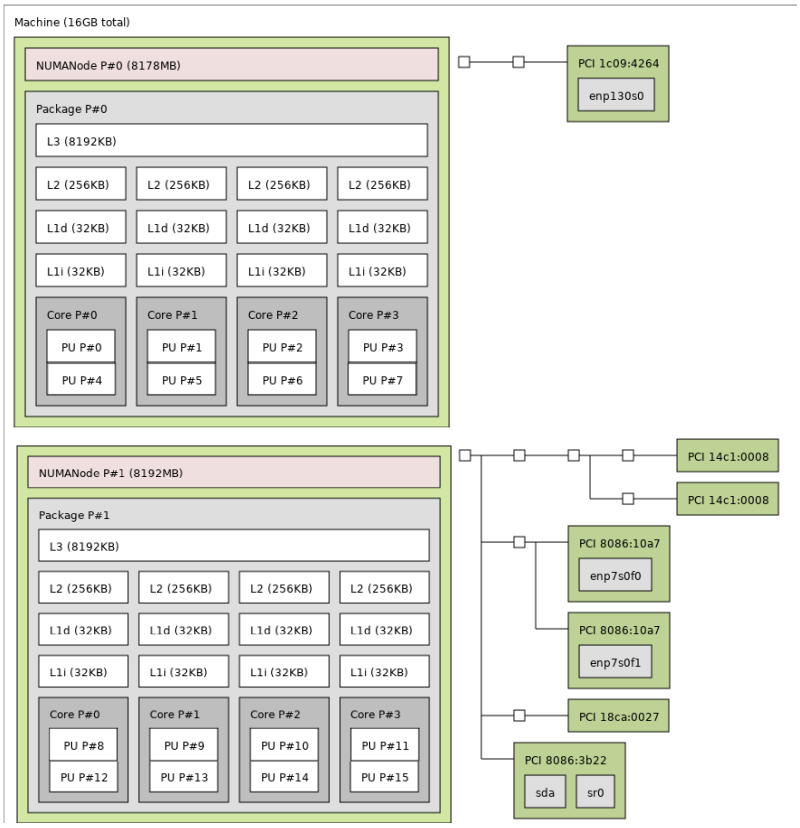


Figure 5-1: lstopo Output

The output shows two NUMA nodes: 0 and 1. A PCI device with the interface name of enp130s0 resides on NUMA node 0. You can then verify the interface is associated with NUMA node 0.

```
sudo cat /sys/class/net/enp130s0/device/numa_node
```

```
0
```

5.2.2 Determining the CPU Configuration

The CPUs and cores assigned to each NUMA node are also displayed in the output of lstopo (see *Figure 5-1*). You can, however, use lscpu or wmic to see a list of the available CPUs and the nodes to which they belong.

Linux

```
lscpu
```

The output displays details about the CPUs installed on the system.

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
```

```
On-line CPU(s) list:   0-7
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 45
Model name:            Intel(R) Xeon(R) CPU E5-2687W 0 @ 3.10GHz
Stepping:              7
CPU MHz:               3399.707
CPU max MHz:           3800.0000
CPU min MHz:           1200.0000
BogoMIPS:              6200.36
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              20480K
NUMA node0 CPU(s):     0-3
NUMA node1 CPU(s):     4-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology non-
stop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2
ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes
xsave avx lahf_lm epb tpr_shadow vnmi flexpriority ept vpid xsaveopt dtherm ida
arat pln pts
```

As shown in the example, the system contains two CPUs, each with 4 cores, and each NUMA node is assigned 4 cores. NUMA node 0 uses cores 0-3 while NUMA node 1 uses cores 4-7. Because core 0 is reserved for kernel operations, you should bind the MVA application to the other cores. In this case, if binding the application to NUMA node 0, you could use cores 1-3.

Windows

You can view CPU details in the Task Manager or with the Windows Management Instrumentation Command (wmic) Line Utility.

If using Task Manager, select the CPU tab to view the processor details.

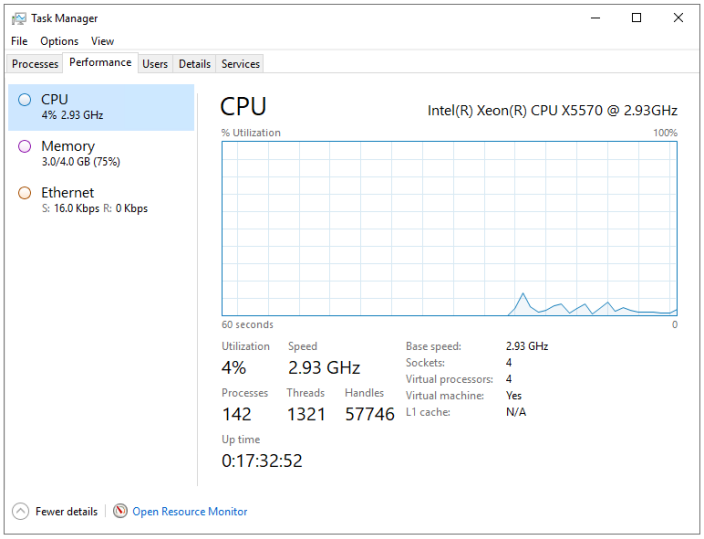


Figure 5-2: Windows Task Manager

If using wmic, retrieve the CPU information using the CPU Get command.

```
wmic
```

This changes the prompt to wmic:root\cli>.

```
CPU Get /Format:List
```

The output displays details about the CPU, including the number of cores.

```
Name=Intel(r) Core(TM) i7-6500U CPU @ 2.50GHz
NumberOfCores=2
NumberOfEnabledCore=0
NumberOfLogicalProcessors=4
```

5.3 Hardware Considerations

One network adapter per NUMA node is recommended. Make sure each adapter is connected to a Gen 3 x8 PCIe slot to root.

5.4 Notes

During the test process, the ARIA development team noted several items. The most interesting was the latency involved when accessing L3 cache versus RAM. Specifically, it took nearly 5 times longer to retrieve data from RAM when compared with accessing the data from L3 cache. CPUs usually require access to memory directly attached to the socket where the CPU is installed. Accessing memory from one socket to memory from another adds latency overhead because it requires traversing the memory interconnect first. Accesses from a single processor to local memory, however, not only have lower latency compared to remote memory accesses, but do not cause contention on the interconnect and the remote memory controllers. Therefore, remote memory accesses should be avoided to increase the overall bandwidth and improve the latency to memory. For this reason, using nonuniform memory access (NUMA) zones is important when tuning for performance

Additional considerations employed include:

- Dual socket motherboard with a balanced memory configuration. Performance is maximized if you balance across the DIMM banks of each CPU. Results are best if an equal amount of RAM is assigned to both CPUs, and performance is maximized when the total memory capacity is balanced across all installed processors.
- Two network adapters; one for each NUMA zone.
- Core 0 isolated and dedicated to the OS.
- Isolated hardware and software interrupt request (IRQ) processing with packet processing bound to a dedicated core.
- Packet processing software bound to remaining isolated cores.

In an attempt to force cores to perform certain duties, core 0 does nothing except generic OS duties, ssh, various kernel threads, and timekeeping.

5.5 Tuning Checklist

MVA is a kernel-bypass driver and does not require much tuning. However, it helps to be aware of issues that may affect latency/packet rate.

In addition to these issues, there are a number of environment variables available in the MVA software for debugging and customizing the software to meet the requirements of the application.

Tuning is recommended to help improve performance. The OS can pre-empt the MVA application when this call takes place. Tuning also helps reduce packet loss due to a drop in the number of available buffers or due to PCI backpressure.

> For more information about variables and counters, see *Settings on page 31* and *MVA Counters on page 115*.

5.5.1 Performance Profile

Description

Both Ubuntu/Debian and CentOS/RedHat operating systems support tuning profiles through the tuned package. When installed, you can use the tuned-adm tool to change the profiles or create a custom profile.

> For more information about tuned and packages available for download, see https://tuned-project.org/.

> For more information about installing tuned, see *Installing Linux Packages on page 40*.

Recommendation

The ARIA Cybersecurity development team recommends setting the system performance profile to *network-latency*. This profile is recommended because it specifies appropriate scheduler, memory, and system energy consumption parameters. In addition, network-latency prevents the kernel from balancing processes across NUMA nodes. This applies to RedHat Enterprise Linux (RHEL) distributions and its derivative, CentOS.

Examples

Linux

Retrieve the current performance profile.

```
sudo tuned-adm active
```

The current profile is returned.

```
Current active profile: balanced
```

Set the profile to network latency.

```
sudo tuned-adm profile network-latency
```

Retrieve the profile to verify the change took effect.

```
sudo tuned-adm active
```

The profile should reflect the change.

```
Current active profile: network-latency
```

Windows

Retrieve the available performance profiles.

```
powercfg /L
```

The available profiles are returned with their respective GUIDs. The active profile is denoted with an asterisk (*).

```
Power Scheme GUID: 381b4222-f694-41f0-9685-ff5bb26-df2e (Balanced)*
Power Scheme GUID: 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c (High performance)
Power Scheme GUID: a1841308-3541-4fab-bc81-f71556f20b4a (Power saver)
```

Set the profile to high performance using the GUID for that power scheme.

```
powercfg /setactive 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c
```

Retrieve the profile to verify the change took effect.

```
powercfg /getactivescheme
```

The profile should reflect the change.

```
Power Scheme GUID: 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c (High performance)
```

5.5.2 PCIe Expansion Slot

Description

Properly seated cards are crucial to ensuring power and data connections are reliable and not compromised. In addition, the slot must be compatible with the correct PCIe link width and physically be a x8 or x16 slot.

Recommendation

You must physically inspect the adapter to ensure it is properly seated. When installed, you should have heard a “click” as it inserted. Once you are confident that it is properly seated, verify the link width (x8) and generation (Gen2 or Gen3) using myri_info.

Example

Run the `myri_info` command to verify the link width is x8.2 (x8, Gen2).

```
/opt/mva/sbin/myri_info
```

```
C:\MVA_Myri-10G\sbin\myri_info
```

The output indicates the adapter is in a Gen3 x16 slot, but the link is Gen2 x8.

```
pci-dev at 03:00.0 vendor:product(rev)=1c09:4264(01)
       behind bridge downstream-port: 02:02.0 111d:806a (x8.1/x8.2)
       behind bridge upstream-port: 01:00.0 111d:806a (x8.2/x8.2)
       behind bridge root-port: 00:01.0 8086:0c01 (x8.2/x16.3)
       10G-PCIE-8B -- Link x8
```

5.5.3 Interrupts and IRQ Affinity

Description

An Interrupt Request (IRQ) is a hardware signal sent to the processor that temporarily stops a running program and allows a special program, an interrupt handler, to run instead. You can send IRQs by a dedicated hardware signal or across a hardware bus as a Message Signaled Interrupt (MSI) information packet. When interrupts are enabled, receipt of an IRQ prompts a switch to interrupt context.

Kernel interrupt dispatch code retrieves the IRQ number and its associated list of registered Interrupt Service Routines (ISRs). The code then calls each ISR in turn. The ISR acknowledges the interrupt and ignores redundant interrupts from the same IRQ and queues a deferred handler to finish processing the interrupt before stopping the ISR from ignoring future interrupts.

Interrupt balancing distributes hardware interrupts across the processors. Although intended to improve performance, in some situations, it may negatively impact performance. Linux systems use a daemon service called irqbalance to distribute these interrupts. This daemon identifies the highest volume interrupt sources and isolates them to a single CPU, spreading the load as much as possible over an entire set of cores and minimizing cache hit rates for IRQ handlers. By default, irqbalance is enabled. You can, however, set the affinity of an interrupt request to a specific set of cores that can service the interrupt.

NOTE:	This section applies to Linux systems only.

> For more information on IRQ affinity, visit https://community.mellanox.com/docs/DOC-2123.

The /proc/interrupts file includes information about the interrupts, including the following:

- IRQ number
- Number of interrupts per core
- Interrupt number handled by each core
- Interrupt type
- Comma-delimited list of drivers registered to receive the interrupt

Disabling irqbalance avoids hardware interrupts in the threads. In real-time deployments, applications are typically dedicated and bound to specific cores; so, the irqbalance daemon is not required.

Recommendation

Because interrupt balancing constantly shifts affinities, it may disrupt the pinned cores. For this reason, you should disable irqbalance and manually bind interrupts to a specific core.

Examples

CentOS/RedHat

NOTE:	The tuna package is recommended for CentOS/RedHat. See *Installing Linux Packages on page 40* for information on installing this package.

Disable irqbalance.

```
sudo systemctl stop irqbalance
sudo systemctl disable irqbalance
```

Find the IRQ associated with each Myricom endpoint.

```
sudo tuna --irqs=myri\* --show_irqs
```

The output shows the IRQ number, the user, and the IRQ affinity. In the following example, the Myricom IRQs are affinitized to core 0.

```
  #  users        affinity
36 myriE0-ep00           0
37 myriE0-ep01           0

...
```

Move all myriC0 interrupts to cores 1-3 on CPU 0.

```
sudo tuna --irqs=myri\* --cpus=1-3 --move --spread
```

Verify your changes.

```
sudo tuna --irqs=myri\* --show_irqs
```

The output should show the IRQ numbers for the endpoints are spread across cores 1-3.

```
  #  users        affinity
35  myriE0-ep00         1
36  myriE0-ep01         2
37  myriE0-ep02         3
38  myriE0-ep03         1
39  myriE0-ep04         2
40  myriE0-ep05         3
41  myriE0-ep06         1
42  myriE0-ep07         2
...
```

Ubuntu/Debian

Disable irqbalance.

```
sudo systemctl stop irqbalance
sudo systemctl disable irqbalance
```

Find the interrupt number of the adapter.

```
sudo cat /proc/interrupts | grep myri
```

The output shows the interrupt number. In this example, the interrupt is number 66.

```
66: 0   0   0   0   1  10150   0   0   PCI-MSI-edge myriE0-ep01
```

Assign the IRQ number to a specific core, in this case, core 1.

```
echo 1 | sudo tee /proc/irq/66/smp_affinity
```

Verify the core is properly assigned.

```
sudo cat /proc/irq/66/smp_affinity
```

The output shows the core mask. In this case, the core that services the interrupts for the SIA is core 1 (bit 1 set).

```
01
```

5.5.4 Core Affinity

Description

Affinity, or core binding, allows you to pin a process to one or more cores so that the process will execute only on the designated core(s). This prevents the OS scheduler from executing the process on any other core or migrating the running process to a different core or socket, which could reduce performance. On hosts with multiple sockets, some sockets are physically closer to the adapter and/or memory and perform better than others. Cores may also be isolated to prevent other OS or less critical processes from running on the designated core, allowing all resources to be dedicated to a specific application.

> For more information, see *Core Isolation on page 57*.

Recommendation

Binding the MVA process to a core is recommended. This ensures the process and its threads have an affinity for a specific core. On NUMA systems, the best performance is generally seen if the processes and their interrupts are bound to the same CPU. For Linux, you can bind an application to a specific core using taskset or numactl to optimize performance for MVA. For Windows, you can set the affinity in the task manager or by using SetThreadAffinityMask. By default, the threads of an application automatically inherit the affinity of the process. In addition, if a core is currently isolated from the kernel scheduler, you can only move processes using tools like taskset or numactl or the function sched_setaffinity().

Some systems have multiple PCIe root ports. For example, in a dual-CPU server some PCIe slots are connected to the root port of CPU 0 and other PCIe slots are connected to the root port of CPU 1. For best performance, the interrupt handler and application should both be bound to the CPU closest to the PCIe slot where the adapter is installed.

Before binding the application to a core, you must first determine the NUMA node where the cores are assigned and which core the interrupt handler is using.

> For more information, see *NUMA Nodes and Cores on page 40* and *Interrupts and IRQ Affinity on page 47*.

Examples

Linux

Bind mva_simple_recv to cores 1-3 using numactl and launch mva_simple_recv.

```
sudo numactl --physcpubind=1-3 /opt/mva/bin/tests/mva_simple_recv
```

```
mva_recv ready to receive
```

Bind mva_simple_recv to cores 1-3 using taskset and launch mva_simple_recv.

```
taskset -c 1-7 /opt/mva/bin/tests/mva_simple_recv
```

```
mva_recv ready to receive
```

Windows

Bind mva_simple_recv to cores 1-3 using the task manager.

1. Open Windows task manager.
2. Select the Details tab.
3. Right-click on mva_simple_recv and select **Set affinity**.
4. Select cores 1-3 and click **OK**.

Bind mva_simple_recv to cores 1-3 using SetThreadAffinityMask(thread, mask).

```
SetThreadAffinityMask(1, 0xe)
```

Bind mva_simple_recv to cores 1-3 when starting the program

```
C:\MVA_Myri-10G\bin\tests\ start /affinity 0xe mva_simple_recv.exe
```

```
mva_recv ready to receive
```

5.5.5 CPU Frequency Scaling

Description

CPU frequency scaling allows the system to adjust the CPU frequency to save power.

Recommendation

Because CPU frequency scaling may lower the frequency, the ARIA Cybersecurity development team recommends disabling this feature so the CPU always runs at the highest speed.

NOTE:	CPU frequency scaling may already be set to “performance” if the performance profile is set to “network-latency” as described in "Performance Profile" on page 45.

Examples

Linux

Determine the current frequency of the CPUs.

```
grep Hz /proc/cpuinfo
```

In the following example, the frequency is only 1.2 GHz, but the CPU speed is 2.4 GHz.

```
model name      : Intel(R) Core(TM)2 CPU          6600  @ 2.40GHz
cpu MHz         : 1197.000
model name      : Intel(R) Core(TM)2 CPU          6600  @ 2.40GHz
cpu MHz         : 1197.000
```

Change CPU governor to “performance” on all CPUs.

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

Verify the changes have been applied.

```
grep Hz /proc/cpuinfo
```

Notice the frequency of both CPUs has changed to 2394 MHz, which is very close to the maximum speed (2.4 GHz). This means they are running in performance mode.

```
model name      : Intel(R) Core(TM)2 CPU          6600  @ 2.40GHz
cpu MHz         : 2394.000
```

```
model name      : Intel(R) Core(TM)2 CPU          6600  @ 2.40GHz
cpu MHz         : 2394.000
```

5.5.6 PCI Bridging

Description

A PCI bridge is a special device that connects two PCI buses together. Some PCIe slots on a machine may have deeper PCI bridging than others. Extra bridge chips between the CPU and the adapter results in higher latency.

Recommendation

If there are a lot of bridge chips, trying a different PCIe slot may improve performance for MVA. To detect the bridges between the CPU and adapter, run myri_info, located in /opt/mva/sbin.

Examples

Linux

Find the list of all MVA PCI devices.

```
lspci | grep -i -e myri
```

The output lists all PCI devices. In this case, the MVA adapter is on bus 3.

```
03:00.0 Ethernet controller: MYRICOM Inc. Myri-10G Dual-Protocol NIC (rev 01)
```

Query the adapter for bridge information.

```
sudo /opt/mva/sbin/myri_info
```

The output shows details about the adapter. In the following example, the adapter is behind one PCI bridge.

```
pci-dev at 03:00.0 vendor:product(rev)=1c09:4264(01)
       behind bridge downstream-port: 02:02.0 111d:806a (x8.1/x8.2)
       behind bridge upstream-port: 01:00.0 111d:806a (x8.2/x8.2)
       behind bridge root-port: 00:01.0 8086:0c01 (x8.2/x16.3)
        10G-PCIE-8B -- Link x8

  EEPROM String-spec:
       MAC=00:60:dd:43:14:12
       SN=500192
       PC=10G-PCIE2-8B2-2S
       PN=09-04669
       TAG=ze_tools-1_4_38
...
```

Windows

1. Open Device Manager and expand the Network adapters.
2. Right-click on the MVA adapter and select **Properties**. The location of the adapter is displayed in the General tab.

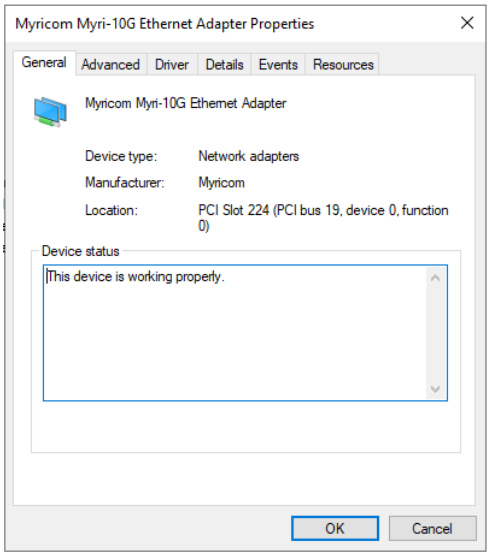


Figure 5-3: Windows Network Adapter Properties

As shown in the above figure, the adapter is on bus 19.

Query the adapter for bridge information.

```
C:\MVA_Myri-10G\sbin\myri_info.exe
```

The output shows details about the adapter. In the following example, the adapter is behind one PCI bridge.

```
pci-dev at 03:00.0 vendor:product(rev)=1c09:4264(01)
       behind bridge downstream-port: 02:02.0 111d:806a (x8.1/x8.2)
       behind bridge upstream-port: 01:00.0 111d:806a (x8.2/x8.2)
       behind bridge root-port: 00:01.0 8086:0c01 (x8.2/x16.3)
        10G-PCIE-8B -- Link x8

  EEPROM String-spec:
       MAC=00:60:dd:43:14:12
       SN=500192
       PC=10G-PCIE23-8B2E-2S
       PN=09-04669
       TAG=ze_tools-1_4_38BOM=B
...
```

5.5.7 Power Saving/Performance Speed

Description

To help mitigate temperature issues, many servers employ power saving techniques, such as Active State Power Management (ASPM), or options for decreasing clock speeds. Both of these options can significantly impact performance.

Recommendation

You should disable all power saving options or configure the server to optimize performance in the BIOS. This includes disabling the following:

- ASPM
- Node Interleaving

In addition, make sure the following are enabled:

- Channel Interleaving
- Intel I/O Acceleration Technology (IOAT)

Data Direct I/O Technology (DDIO) is part of the IOAT features package. When enabled, DDIO forces the adapter to send packets directly to L3 cache, which, as previously indicated, improves performance as the CPU can read data from this location faster than from RAM.

Example

NOTE:	In the following example, an HP ProLiant server was used. The BIOS options may differ for your server.

Set the *Workload Profile* to **General Peak Frequency Compute** in the BIOS.

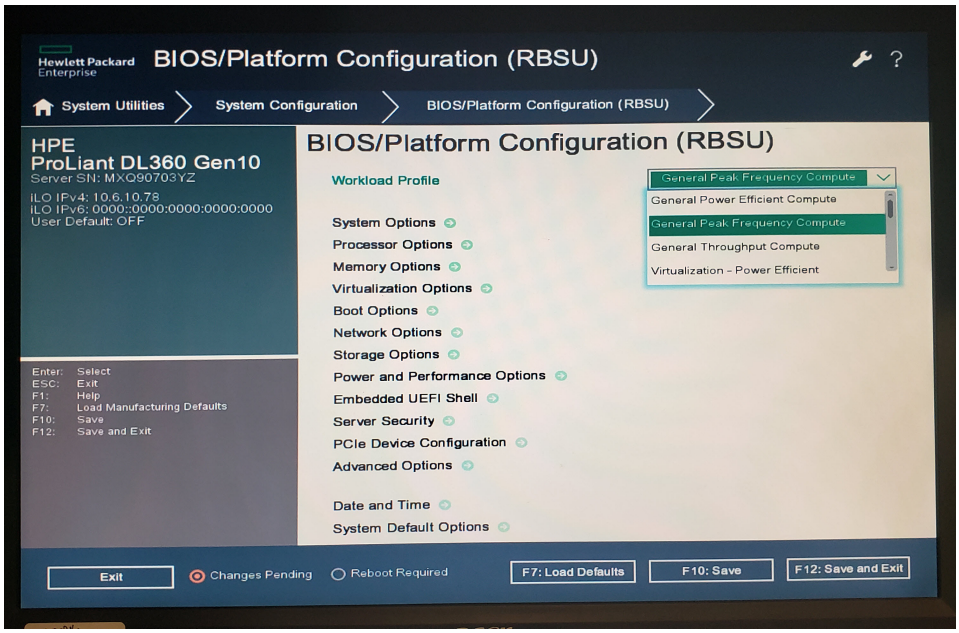


***Figure 5-4:** Set Workload Profile in BIOS*

Click **Power and Performance Options** and set *Energy/Performance Bias* to **Maximum Performance**.

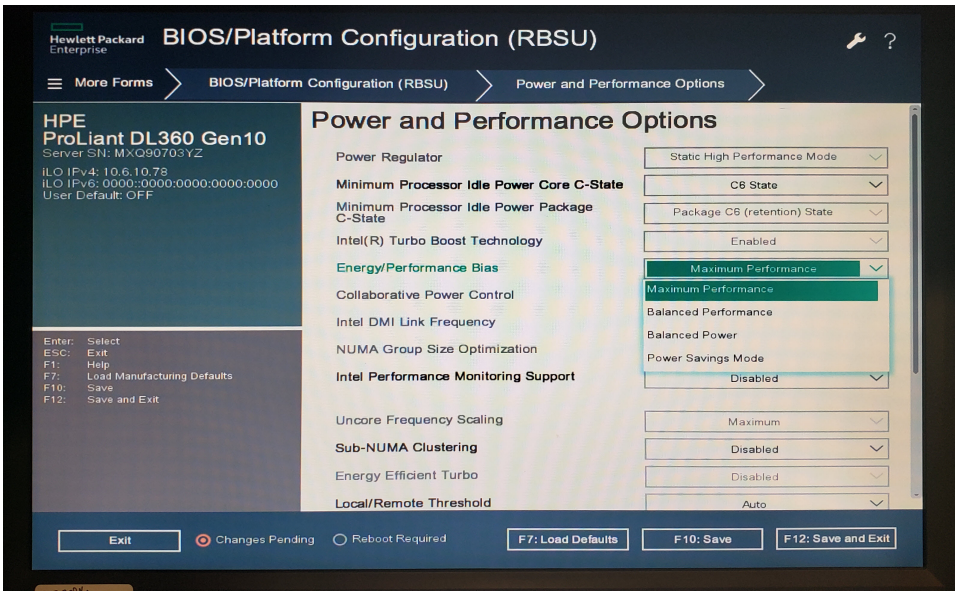


Figure 5-5: Set Power and Performance in BIOS

5.5.8 Hyper-Threading

Description

When hyper-threading is enabled, the operating system detects two logical cores per processor core, which allows the concurrent scheduling of two processes. However, the logical cores share the resources of the CPU. Both logical cores will compete for L1 and L2 cache access, but all cores of the CPU must share the pool of L3 cache. This can lead to poor cache utilization, or cache thrashing, depending on the amount of context needed to maintain packet information during processing.

Recommendation

For best performance, disabling hyper-threading in the BIOS is recommended to avoid resource contention between the cores.

Examples

Linux

Determine if hyper-threading is enabled.

```
lscpu
```

If the number of threads per core is two, hyper-threading is enabled. In the following example, hyper-threading is enabled.

```
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             2
NUMA node(s):          2
```

Windows

Determine if hyper-threading is enabled using Windows Management Instrumentation Command-Line Utility.

```
wmic
```

This changes the prompt to wmic:root\cli>.

```
Get NumberOfCores,NumberOfLogicalProcessors /Format:List
```

If there are twice as many logical processors than cores, hyper-threading is enabled. In the following example, there are four logical processors and two cores, indicating that hyper-threading is enabled.

```
NumberOfCores=2
NumberOfLogicalProcessors=4
```

If hyper-threading is enabled, you must disable it in the BIOS. In the following example, it is disabled on an HP ProLiant server.

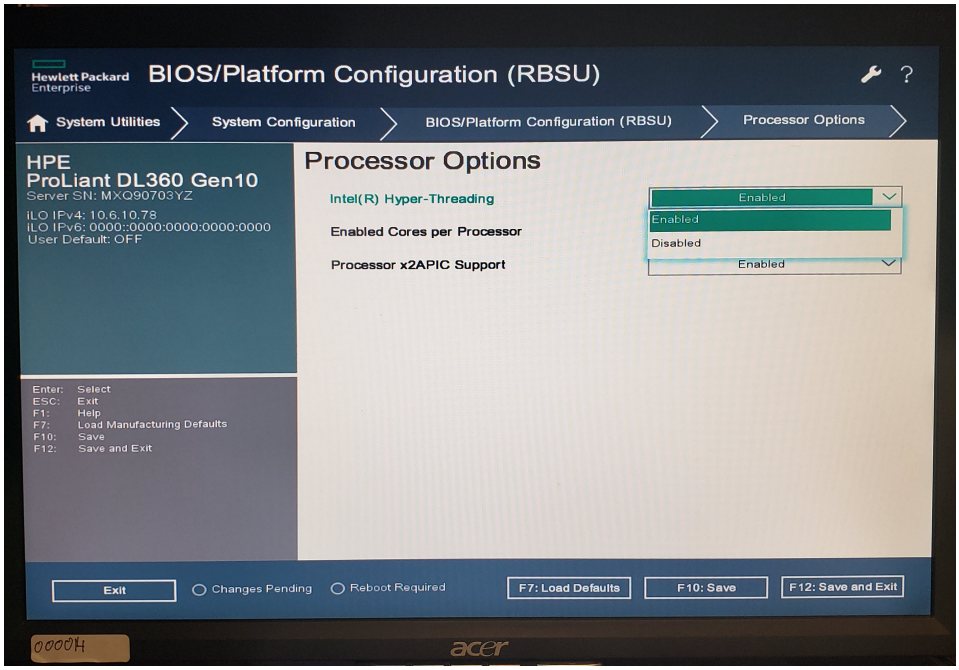


Figure 5-6: Disable Hyper-Threading in BIOS

5.5.9 NUMA Awareness

Description

Non-uniform memory access (NUMA) provides a way to cluster processes and memory into zones to improve performance.

Recommendation

On NUMA For best performance, you should configure MVA to use the CPU closest to the adapter.

To do this, execute the open stream function (mva_open_stream(), mva_open_mcast_stream(), or mva_open_stream_pr()) on the CPU "closest" to the PCIe slot containing the ARC Series adapter.

Latencies and contention across the QPI bridges between the sockets may still arise in some cases.

5.5.10 Core Isolation

NOTE:	This section applies to Linux only.

Description

Isolating a core helps decrease jitter and prevent other processes from pre-empting an application, which affects performance. With core isolation, the process pinned to the core is shielded from other OS services.

Recommendation

The ARIA development team recommends running the MVA application on a core that has been isolated from the kernel scheduler.

Isolating the specific core from the kernel scheduler binds the MVA application to that core. When isolated, the kernel is prevented from scheduling other threads or processes on the core. This results in less jitter as the competing processes or threads are unable to steal cycles.

To isolate the cores, set the following Linux kernel parameters in the GRUB configuration file:

- **mce=ignore_ce** – Disables the poll used to check for correctable errors. This poll, which occurs every five minutes, can cause latency spikes, ECC, and other errors. This essentially disables pre-emptive notifications.
- **isolcpus=<cores>** – Removes the cores listed from the kernel scheduler. Use a comma-separated list (e.g., 2, 4), a range (e.g., 7-15), or a combination of both (2, 4, 7-15) to identify multiple cores. The only way to assign a process to an isolated core is to pin the process to that core.

For GRUB-based Linux systems, use the isolcpu option.

Examples

CentOS/RedHat

NOTE:	Isolating the cores requires modifying kernel parameters, which are stored in grub.cfg. Make sure you understand the location of this file before continuing. See *Determining the Location of the Config File on page 39* for more information.

Determine if any cores are currently isolated.

```
cat /sys/devices/system/cpu/isolated
```

The output lists any cores that are currently isolated. If empty, as in the following example, no cores are currently isolated.

Edit /etc/default/grub and add the settings to the end of the GRUB_CMDLINE_LINUX string. Make sure to keep the text within the quotes.

```
GRUB_CMDLINE_LINUX="crashkernel-auto rd.lvm.lv=centos_venus/root rd.lvm.lv=centos_
venus/swap rhgb quiet mce=ignore_ce isolcpus=1-3"
```

Then build grub.conf and place it in the correct location.

For grub boot loaders:

```
sudo grub2-mkconfig -o /boot/grub/grub.cfg
```

For grub2 boot loaders:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

For UEFI-based systems:

```
sudo grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the cores are isolated.

```
cat /sys/devices/system/cpu/isolated
```

```
1-3
```

Ubuntu/Debian

Determine if any cores are currently isolated.

```
cat /sys/devices/system/cpu/isolated
```

The output lists any cores that are currently isolated. If empty, as shown in the following example, no cores are currently isolated.

```

```

Edit /etc/default/grub and add the settings to the GRUB_CMDLINE_LINUX string.

```
GRUB_CMDLINE_LINUX="mce=ignore_ce isolcpus=1-3"
```

Then update grub.conf.

```
update-grub
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the cores are isolated.

```
cat /sys/devices/system/cpu/isolated
```

```
1-3
```

5.5.11 Tickless Kernel Parameter

Description

Linux systems use a periodic interrupt, referred to as a "tick," to keep track of statistics and time. When this occurs, it is scheduled ahead of other processes. Linux systems, however, offer the nohz_full option. When enabled on a core, all timekeeping activities are moved to non-latency-sensitive cores, thus turning the core "tickless" and reducing interference with user-space tasks.

Recommendation

Enabling nohz_full on all cores running the MVA application (e.g., nohz_full = 1-3) is recommended. When enabled, the timekeeping activities are moved off the cores. The nohz_full Linux kernel parameter is set in the GRUB configuration file.

Examples

CentOS/RedHat

NOTE:	Enabling nohz_full requires modifying kernel parameters, which are stored in grub.cfg. Make sure you understand the location of this file before continuing. See *Determining the Location of the Config File on page 39* for more information.

Determine if nohz_full is already enabled for any cores.

```
sudo cat /proc/cmdline
```

The output displays the options currently set in the boot loader. In the following example, nohz_full is not enabled as it is not included as an option.

```
BOOT_IMAGE=/vmlinuz-4.15.11-1.el7.elrepo.x86_64 root=/dev/mapper/centos-root ro
crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet mce-
e=ignore_ce isolcpus=1-3
```

Edit /etc/default/grub and add `nohz_full=1-3` to the end of the GRUB_CMDLINE_LINUX string. Make sure to keep the text within the quotes.

```
GRUB_CMDLINE_LINUX="crashkernel-auto rd.lvm.lv=centos_venus/root rd.lvm.lv=centos_
venus/swap rhgb quiet mce=ignore_ce isolcpus=1-3 nohz_full=1-3"
```

Build grub.conf and place it in the correct location.

For grub boot loaders:

```
sudo grub2-mkconfig -o /boot/grub/grub.cfg
```

For grub2 boot loaders:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

For UEFI-based systems:

```
sudo grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the boot loader options have changed.

```
sudo cat /proc/cmdline
```

```
BOOT_IMAGE=/vmlinuz-4.15.11-1.el7.elrepo.x86_64 root=/dev/mapper/centos-root ro
crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet mce-
e=ignore_ce isolcpus=1-3 nohz_full=1-3
```

Ubuntu/Debian

Determine if nohz_full is currently enabled for any cores.

```
cat /proc/cmdline
```

The output lists the current boot loader options. In the following example, nohz_full is not enabled on any cores.

```
BOOT_IMAGE=/boot/vmlinuz-4.15.0-47-generic root=UUID=1bb906ea-e2f5-4aab-8b65-
148af80a661d ro net.ifnames=0 mce=ignore_ce isolcpus=1-3
```

Edit /etc/default/grub and add `nohz_full=1-3` to the end of the GRUB_CMDLINE_LINUX string. Make sure to keep the text within the quotes.

```
GRUB_CMDLINE_LINUX="mce=ignore_ce isolcpus=1-3 nohz_full=1-3"
```

Update grub.cfg.

```
update-grub
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the boot loader options have changed.

```
cat /proc/cmdline
```

```
BOOT_IMAGE=/boot/vmlinuz-4.15.0-47-generic root=UUID=1bb906ea-e2f5-4aab-8b65-
148af80a661d ro net.ifnames=0 mce=ignore_ce isolcpus=1-3 nohz_full=1-3
```

5.5.12 Read-Copy-Update Kernel Parameters

Description

The Read-Copy-Update (RCU) system is a lockless mechanism inside the kernel. Sometimes callbacks are queued on cores for future processing when removing memory is safe when performing RCU operations. You can, however, offload RCU callbacks on the cores where the MVA application is running. You can also relieve these cores from polling other cores to waken the RCU offload thread.

Recommendation

Disabling RCU callbacks on the cores where the MVA application is running is recommended. In addition, the cores should not poll other cores that are performing RCU callbacks. These options, rcu_nocbs and rcu_nocb_poll, are kernel parameters and are set in the GRUB configuration file.

Examples

CentOS/RedHat

NOTE:	Offloading RCU callbacks and disabling polling for the affected cores require modifying kernel parameters, which are stored in grub.cfg. Make sure you understand the location of this file before continuing. See "Determining the Location of the Config File" on page 39 for more information.

Edit /etc/default/grub and add `rcu_nocbs=1-3` and `rcu_nocb_poll` to the end of the GRUB_CMDLINE_LINUX string. Make sure to keep the text within the quotes.

```
GRUB_CMDLINE_LINUX="crashkernel-auto rd.lvm.lv=centos_venus/root rd.lvm.lv=centos_venus/swap rhgb quiet mce=ignore_ce isolcpus=1-3 nohz_full=1-3 rcu_nocbs=1-3 rcu_nocb_poll"
```

Build grub.conf and place it in the correct location.

For grub boot loaders:

```
sudo grub2-mkconfig -o /boot/grub/grub.cfg
```

For grub2 boot loaders:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

For UEFI-based systems:

```
sudo grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the boot loader options have changed.

```
sudo cat /proc/cmdline
```

```
BOOT_IMAGE=/vmlinuz-4.15.11-1.el7.elrepo.x86_64 root=/dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet mce-e=ignore_ce isolcpus=1-3 nohz_full=1-3 rcu_nocbs=1-3 rcu_nocb_poll
```

Ubuntu/Debian

Edit /etc/default/grub and add `rcu_nocbs=1-3` and `rcu_nocb_poll` to the end of the GRUB_CMDLINE_LINUX string. Make sure to keep the text within the quotes.

```
GRUB_CMDLINE_LINUX="mce=ignore_ce isolcpus=1-3 nohz_full=1-3 rcu_nocbs=1-3 rcu_nocb_
poll"
```

Update grub.cfg.

```
update-grub
```

The output indicates the status and when the process completes.

```
...
Found linux image: /boot/<image>
Found initrd image: /boot/<image>.img
done
```

Reboot the server.

```
sudo reboot
```

Verify the boot loader options have changed.

```
cat /proc/cmdline
```

```
BOOT_IMAGE=/boot/vmlinuz-4.15.0-47-generic root=UUID=1bb906ea-e2f5-4aab-8b65-
148af80a661d ro net.ifnames=0 mce=ignore_ce isolcpus=1-3 nohz_full=1-3 rcu_
nocbs=1-3 rcu_nocb_poll
```

Chapter 6

Packet Resend

MVA supports the GigE packet resend feature. With this feature, an incomplete block can be detected and recovered by sending a packet resend request to the streaming device, such as a camera. The application can continue to send these requests until the maximum number of retries is met or a complete block is received. Using IDs for tracking, the adapter places the newly received packets in the correct order to ensure the packets are distributed to the host in sequence.

To add flexibility to this feature, MVA supports several flags and configuration settings. These affect how the application responds to missing packets and unexpected blocks. This section describes those settings and some of the use cases they impact.

6.1 Settings

There are several flags that allow you to define how the application handles various situations, such as dropped packets and incomplete blocks. These flags include:

- MVA_OPEN_ZEROLOSS
- MVA_OPEN_DROP_INCOMPLETE
- MVA_OPEN_DROP_RETURN

> For more information about the MVA_OPEN flags, see *flags on page 31*.

In addition, when using the packet resend feature, a configuration structure is passed to the API. This structure consists of different fields designed to control how often the packet resend requests are issued and how long the MVA adapter will wait for a complete block. These include:

- block_timeout
- resend_timeout
- resend_retries
- block_out_of_order

> For more information, see *Configuration Structure on page 32*.

6.2 Scenarios

There are several different scenarios based on the settings and the status of a block. This section provides a description of how the packet resend feature handles the following situations:

- Successful Block Completion
- Successful Packet Resend of One or More Packets
- Successful Multiple Packet Resend Retries

- Failed Packet Resend Requests Exceeded
- Failed Block Timeout Expired
- Failed Resend Timeout Expired
- Incomplete Block Dropped

To explain these situations, a basic flow between the host application (camera application), the MVA-enabled NIC, and the streaming device (camera) is used. See the following figure for an example.

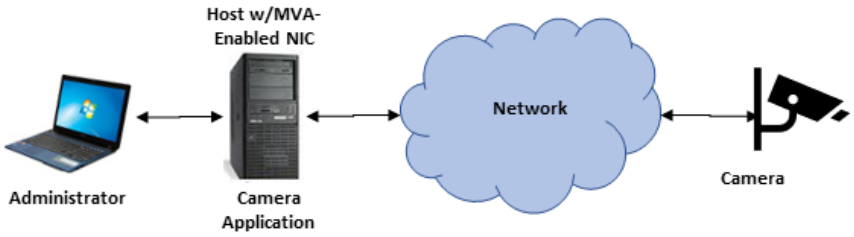


Figure 6-1: Basic Flow Diagram

To illustrate how the counters are incremented, the counter values before and after each scenario are provided. The counters tracked in this section include:

- blocksDropped (Drop)
- blocksReturnedComplete (Comp)
- blocksReturnedIncomplete (Incomp)
- resendRequests (Req)
- resendRequestsSuccessful (OK)
- resendRequestsFailed (Fail)
- resendTimeouts (Timeout)
- maxResendRetries (RetriesX)

In addition, the settings for each case are defined.

6.2.1 Successful Block Completion

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ ZEROLOSS	MVA_BOOO_ALWAYS_ RETURN	1000	0	4

Settings

The packet resend feature is not required when the blocks are completed with the correct packet order. The flow for this scenario is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA adapter for blocks. Because the block is in progress, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. The MVA adapter receives the trailer packet, marking the end of Block 1.
5. The adapter returns a state of available to the host application, which then consumes the block. The blocksReturnedComplete counter increases by 1. The host application queues the buffer in the user space for the next block.
6. The MVA adapter receives the leader packet for the next block (Block 2), and the process repeats.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	1	0	0	0	0	0	0

Ending Counter Values

6.2.2 Successful Packet Resend

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ ZEROLOSS	MVA_BOOO_ALWAYS_ RETURN	1000	0	4

Settings

In this scenario, the MVA adapter informs the host application that the block is incomplete because one or more packets are missing. The API then issues the packet resend request to retrieve the remaining packets, starting with the dropped packet. The steps for this scenario is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in progress, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because Packet 11 is out-of-order, the adapter returns a state of incomplete to the host application and drops any subsequent packets.
6. The receiving API automatically sends a packet resend request, starting with packet ID 10, to the camera. The resendRequests counter increases by 1, and the number of available resend retries decreases by 1.
7. The camera resends the packets, beginning with the ID 10, to the MVA adapter. The resendRequestsSuccessful counter increments by 1.
8. The MVA adapter receives and writes the packets in sequence to the user space.
9. The adapter receives the trailer packet, indicating the end of Block 1.
10. The adapter returns a state of available to the host application. The blocksReturnedComplete counter increases by 1. The host application queues the buffer in the user space for the next block.
11. The MVA application receives the leader packet for the next block (Block 2), and the process repeats.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	1	0	1	1	0	0	0

Ending Counter Values

6.2.3 Successful Multiple Packet Resend Retries

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ ZEROLOSS	MVA_BOOO_ALWAYS_ RETURN	1000	0	4

Settings

In the event multiple packet resend requests are required, the requests are sent until either the block is completed or the maximum number of retries has been exceeded. In this example, the block is eventually completed, but only after two resend requests are issued. The flow for this scenario is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in-flight, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because packet 11 is out-of-order, the adapter returns a state of incomplete to the host application and drops any subsequent packets.
6. The receiving API sends a packet resend request, starting with packet ID 10, to the camera. The resendRequests counter increases by 1, and the number of available resend retries decreases by 1.
7. The camera resends the packets, beginning with the ID 10, to the MVA adapter.
8. The MVA adapter receives and writes the packets in sequence to the user space. The resendRequestsSuccessful counter increments by 1.
9. A packet with ID 15 is dropped, and the next packet the adapter receives is packet ID 16.
10. Because packet 16 is out-of-order, the adapter returns a status of incomplete to the host application and drops any subsequent packets.
11. The polling API automatically sends a packet resend request, starting with packet ID 15, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases by 1.
12. The MVA adapter receives and writes the packets in sequence to the user space. The resendRequestsSuccessful counter increments by 1.
13. The adapter receives the trailer packet, indicating the end of Block 1, and returns a state of available to the host application. The blocksReturnedComplete counter increases by 1. The host application queues the buffer in the user space for the next block.
14. The MVA application receives the leader packet for the next block (Block 2), and the process repeats.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	1	0	2	2	0	0	0

Ending Counter Values

6.2.4 Failed Packet Resends/Retry Count Exceeded

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ ZEROLOSS	MVA_BOOO_ALWAYS_ RETURN	1000	0	3

Settings

As long as an incomplete is returned to the host application, the adapter will issue a packet resend request to the camera. This occurs until the block is complete or the maximum number of retries, which is set to two, is exceeded. In this case, the adapter attempts three resend requests, two of which fail, and then marks the block as incomplete-final. The flow for this scenario is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in progress, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because packet 11 is out-of-order, the adapter returns a state of incomplete to the host application and drops any subsequent packets.
6. The receiving API automatically sends a packet resend request, starting with packet ID 10, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases to 2.
7. The camera resends the packets, beginning with the ID 10, to the MVA adapter.
8. The MVA adapter receives and writes the packets in sequence to the user space. The resendRequestsSuccessful counter increments by 1.
9. A packet with ID 15 is dropped, and the next packet the adapter receives is packet ID 16.
10. Because packet 16 is out-of-order, the adapter returns a status of incomplete to the host application and drops any subsequent packets.
11. The receiving API automatically sends a packet resend request, starting with packet ID 15, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases to 1.
12. The MVA adapter receives packets starting with ID 16, indicating the packet resend was not successful. The resendRequestsFail counter increments by 1.

13. The receiving API automatically sends a packet resend request, starting with packet ID 15, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases to 0.
14. Because packet 16 is out-of-order and no more retries are available, the adapter returns the incomplete block and a status of incomplete-final to the host application. The resendRequestsFailed, blocksReturnedIncomplete, and maxSendRetries counters increment by 1. The host application queues the buffer in the user space for the next block.
15. Remaining packets are dropped until the next block (Block 2) is detected.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	1	3	1	2	0	1

Ending Counter Values

6.2.5 Failed Packet Resends/Block Timeout Expired

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ZEROLOSS	MVA_BOOO_ALWAYS_RETURN	1000	0	3

Settings

In this scenario, the block is not completed before the block_timeout expires. The flow is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in progress, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because packet 11 is out-of-order, the adapter returns a state of incomplete to the host application and drops any subsequent packets.
6. The receiving API automatically sends a packet resend request, starting with packet ID 10, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases to 2.

7. The camera does not see the request in time, and the block_timeout expires. The state changes to incomplete-final, and the incomplete block is returned to the host application. The blocksReturnedIncomplete and resendRequestsFailed counters increase by 1. The host application queues the buffer in the user space for the next block.
8. Remaining packets are dropped until the next block (Block 2) is detected.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	1	1	0	1	0	0

Ending Counter Values

6.2.6 Failed Packet Resends/Resend Timeout Expired

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_ZEROLOSS	MVA_BOOO_ALWAYS_RETURN	0	1000	3

Settings

In this scenario, the block is not completed before the block_timeout expires. The flow is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in progress, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because packet 11 is out-of-order, the adapter returns a state of incomplete to the host application and drops any subsequent packets.
6. The receiving API automatically sends a packet resend request, starting with packet ID 10, to the camera. The resendRequests counter increases by 1, and the number of available retries decreases to 2.
7. The camera does not respond to the request before the resend timeout expires (1 second). The state remains incomplete, and the receiving API sends the packet resend request again. The resendRequests and resendTimeouts counters increment by 1, and the number of available retries decreases to 1.
8. Again, the camera does not respond to the request before the resend timeout expires (1 second). The resendRequests and resendTimeouts counters increment by 1, and the number of available retries decreases to 0.

9. The camera fails to respond before the resend_timeout timer expires. The resendTimeout counter increases by 1, the state changes to incomplete-final, and the incomplete block is returned to the host application. The host application then queues the buffer in the user space for the next block.
10. Remaining packets are dropped until the next block (Block 2) is detected.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	1	3	0	0	3	1

Ending Counter Values

6.2.7 Incomplete Blocks Dropped

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
0	0	0	0	0	0	0	0

Beginning Counter Values

flag	block_out_of_order	block_timeout	resend_timeout	resend_retries
MVA_OPEN_DROP_INCOMPLETE	N/A	N/A	N/A	N/A

Settings

NOTE:	The block_out_of_order, block_timeout, resend_timeout, and resend_retries do not apply to this scenario as they are only valid when the flag is set to MVA_OPEN_ZEROLOSS.

If the MVA_OPEN_DROP_INCOMPLETE flag is enabled, any incomplete blocks are dropped, and no packet resend requests are issued. In this case, as soon as a missing packet is detected, an incomplete-final is sent to the host application, and all packets remaining packets are dropped. An example of the steps for this flow is as follows:

1. The MVA adapter receives the leader packet from the camera. This marks the beginning of the first block (Block 1).
2. The host application begins polling the MVA application for blocks. Because the packets for the block is in-flight, the MVA adapter returns a state of not-available.
3. The MVA adapter continues receiving packets, writing them, in sequence, to the buffer in user space.
4. A packet with ID 10 is dropped, and the next packet the adapter receives is packet ID 11.
5. Because packet 11 is out-of-order and the MVA_OPEN_DROP_INCOMPLETE flag is set, the adapter returns a state of incomplete-final to the host application and drops any subsequent packets. The blocksDropped counter increments by 1. The host application queues the buffer in the user space for the next block.

6. The MVA adapter receives the leader packet from the camera. This marks the beginning of the next block (Block 2). The adapter returns a status of not-available to the host application as it receives packets for the next block.

Drop	Comp	Incomp	Req	OK	Fail	Timeout	RetriesX
1	0	0	0	0	0	0	0

Ending Counter Values

Chapter 7

Test and Diagnostic Tools

The MVA software provides several programs that allow you to test the application as well as run diagnostics. This includes receiving blocks, generating logs, and measuring performance.

Invoke the test or diagnostic program with the option --help to obtain the usage summary of the command-line arguments.

See the following table for a description of these programs.

Program	Description	Page
mva_simple_recv	Receives blocks on a single port.	76
mva_simple_recv_pr	Receives blocks on a single port and enables the packet resend feature.	78
mva_multi_dev	Demonstrates the use of multiple threads.	80
mva_rwdt_recv_pr	Demonstrates the use of multiple threads with the packet resend feature.	82
mva_multi_thread	Serializes access to MVA API functions for multiple threads.	86
mva_set_get_name	Tests the ability to set, retrieve, and reset adapter names.	87
myri_bug_report	Generates a bug report for sending to ARIA Cybersecurity Solutions support.	88
myri_info	Retrieves advanced information about the SIA.	89
myri_license	Displays the current licenses for the adapter.	90
myri_bandwidth	Displays the bandwidth of the adapter(s) at specified intervals.	91
myri_counters	Retrieves or clears adapter counters.	92
myri_dma_bench	Verifies the DMA write bandwidth.	93
myri_endpoint_info	Returns information about any endpoints connected to the adapter.	94
myri_nic_info	Retrieves general information about the SIA.	95
myri_intr_coal	Adds an interrupt coalescing delay.	96

Table 5: Test and Diagnostic Programs

Each of these programs is described in the following sections.

7.1 Test Programs

The test programs are provided in binary and source forms, both of which are located in the install directory. The binary forms are located in /opt/mva/bin/tests (Linux) or C:\MVA_Myri-10G\bin\tests (Windows), and the source forms are located in /opt/mva/share/examples (Linux) or C:\MVA_Myri-10G\share\examples (Windows).

Please read /opt/mva/README.txt (Linux) or C:\MVA_Myri-10G\README.txt (Windows) for a detailed listing of all test programs available, as well as instructions for their use.

These test/example programs demonstrate different aspects of the MVA API and how to use its features. The API documentation can be found in PDF and HTML format in /opt/mva/share/doc/ (Linux) or C:\MVA_Myri-10G\share\docs (Windows). Each of the available programs is described in the following sections.

7.1.1 mva_simple_recv

Receives packets from a single port and displays data about the packets. When stopped by pressing **[CTRL] + C**, a summary is displayed in the console.

Usage

Linux

```
./mva_simple_recv [args] <local_IP>:<port>
```

Windows

```
.\mva_simple_recv [args] <local_IP>:<port>
```

Arguments

See the following table for the available arguments.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv or -v-v.
-T <seconds>	Specifies how long to run the script. If set to -1 (default), the script runs until it is manually stopped by pressing **[CTRL]+C**.
-b <bock_timeout>	Specifies the number of msecs to wait for each block. If -1 (default), there is no timeout period (i.e., the application will wait indefinitely for the next block). If set to 0, the application will continuously poll the host for new blocks.
-i <iters>	Indicates the total number of blocks to receive before the script is terminated. This defaults to 1. If set to -1, the number of iterations is infinite.
-r	Instructs MVA to read the blocks after they are received.
-n <num_bufs>	Indicates the number of buffers to allocate. By default, this is set to the max value of 512.
-p <num_bufs>	Specifies the number of buffers to pre-queue. By default, this is set to the max value of 511.
-D <nanosecs>	Injects a synthetic processing delay based on the specified number of nanoseconds.
-V	Verifies the block IDs are sequential.
-m <mcast_addr>	Instructs MVA to listen on a multi-cast address.
-s <buf_size>	Specifies the size of the buffers. By default, this is 2MB.
<local_IP>	Identifies the IP address of the local MVA-enabled device on which to receive GVSP blocks.
<port>	Indicates the UDP port on the <local_IP> to use for receiving GVSP blocks.
-S	Prints link and time source state every second.

Argument	Meaning
-t	Prints the time between packets.
-c <CPU>	Pins the process to the specified CPU number (starting at 0).
-h (--help)	Displays help.

Table 6: mva_simple_recv Arguments

Return

When mva_simple_recv_pr is stopped, it displays statistics for the application and MVA. See the following table for a list of these statistics.

Statistic	Meaning
Blocks Received	Indicates the total number of blocks the MVA adapter received.
Blocks Non-Zero Status	Specifies the number of blocks that were received with a status that did not equal 0.
Blocks Dropped	Displays the total number of blocks that were dropped.
Poll Retries	Specifies the number of poll requests the host application issued.

Table 7: mva_simple_recv Statistics

Example

Receive blocks for a total of one minute on port 20 for the MVA with an IP address of 192.168.0.100.

```
./mvaf_simple_recv -T 60 192.168.0.100:20
```

```
Stream opened on 192.168.0.100:20
Allocated 64 buffers
Queued 63 buffers (max is 511)

^C
--- APP stats ---
Blocks received:        0
Blocks non-zero status: 0
--- MVA stats ---
Blocks received:        0
Blocks dropped:         0
Poll retries:           0
```

7.1.2 mva_simple_recv_pr

Receives packets from a single port and enables the packet resend feature. When stopped by pressing **[CTRL]** + **C**, a summary is displayed in the console.

Usage

Linux

```
./mva_simple_recv_pr [args] -P <unicast_addr>:<port> <local_IP>:<port>
```

Windows

```
.\mva_simple_recv_pr [args] -P <unicast_addr>:<port> <local_IP>:<port>
```

Arguments

See the following table for the available arguments.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv or -v-v.
-T <seconds>	Specifies how long to run the script. If set to -1 (default), the script runs until it is manually stopped by pressing **[CTRL]+C**.
-i <iters>	Indicates the total number of blocks to receive before the script is terminated. This defaults to 1. If set to -1, the number of iterations is infinite.
-r	Instructs MVA to read the blocks after they are received.
-n <num_bufs>	Indicates the number of buffers to allocate. By default, this is set to the max value of 512.
-p <num_bufs>	Specifies the number of buffers to pre-queue. By default, this is set to the max value of 511.
-D <nanosecs>	Injects a synthetic processing delay based on the specified number of nanoseconds.
-V	Verifies the block IDs are sequential.
-m <mcast_addr>	Instructs MVA to listen on a multi-cast address.
-s <buf_size>	Specifies the size of the buffers. By default, this is 2MB.
<local_IP>	Identifies the IP address of the local MVA-enabled device on which to receive GVSP blocks.
<port>	Indicates the UDP port on the <local_IP> to use for receiving GVSP blocks.
-S	Prints link and time source state every second.
-t	Prints the time between packets.
-c <CPU>	Pins the process to the specified CPU number (starting at 0).
-P <unicast_addr>	Defines the destination IP address for packet resend requests. This argument is required.

Argument	Meaning
-R <num_retries>	Specifies the maximum number of packet resend retries to attempt. **Note:** You must set the resend_timeout, block_timeout, or num_retries to a non-zero value. All three cannot be set to 0.
-E <resend_timeout>	Specifies the number of milliseconds the receiver API will wait for packets after the packet resend is issued. By default, this is 0, which indicates the API will wait indefinitely. **Note:** You must set the resend_timeout, block_timeout, or num_retries to a non-zero value. All three cannot be set to 0.
-b <block_timeout>	Specifies the number of milliseconds the polling function on the host application will wait for a block to complete. By default, this is 0, which indicates the function will wait indefinitely. **Note:** You must set the resend_timeout, block_timeout, or num_retries to a non-zero value. All three cannot be set to 0.
-d <booo>	Sets the block our-of-order flag. By default, this is MVA_BOOO_ALWAYS_RETURN. See *block_out_of_order on page 33* for details about the possible values.
-C <channel>	Defines the channel ID used for packet resend requests.
-h (--help)	Displays help.

Table 8: mva_simple_recv_pr Arguments

Return

When mva_simple_recv_pr is stopped, it displays statistics for the application and MVA. See the following table for a list of these statistics.

Statistic	Meaning
Blocks Received	Indicates the total number of blocks the MVA adapter received.
Blocks Non-Zero Status	Specifies the number of blocks that were received with a status that did not equal 0.
Poll Retries	Specifies the number of poll requests the host application issued.
Last Block ID	Indicates the block ID of the last block received.
Blocks Dropped	Displays the total number of blocks that were dropped.
Blocks Complete	Indicates the total number of complete blocks returned to the host application.
Blocks Incomplete	Specifies the total number of incomplete blocks returned to the host application.
Blocks Not Available	Lists the total number of times a block was not returned because no blocks were available for the host to consume. This counter will increment if a block is requested but the block is not ready to move to user space.
Resend Requests	Indicates the total number of packet resend requests that were issued.
Resend Requests OK	Specifies the total number of resend requests that completed successfully.
Resend Requests Failed	Indicates the total number of resend requests that did not complete.
Block Timeouts	Indicates the number of times the block_timeout timer expired.

Statistic	Meaning
Resend Timeouts	Specifies the number of times the resend_timeout timer expired.
Max Resend Retries	Indicates the total number of times the maximum number of retries was exceeded.
Buffers Allocated	Specifies the number of buffer locations allocated for blocks.
Buffers Freed	Indicates the total number of buffers that were freed.
Buffers Enqueued	Identifies the number of buffers available for blocks.

Table 9: mva_simple_recv_pr Statistics

Example

Receive blocks with packet resend enabled on port 20 for the MVA with an IP address of 192.168.0.100 and a unicast IP address of 192.168.0.101, and set the resend_timeout argument to 600 milliseconds.

```
./mva_simple_recv_pr -E 600 -P 192.168.0.101:20 192.168.0.100:20
```

```
Stream opened on 192.168.0.100:20
Allocated 64 buffers
Queued 63 buffers (max is 511)

^C
--- APP stats ---
Blocks received:        9999
Blocks non-zero status: 0
Poll retries:           3546
Last Block ID:          10000
--- MVA info ---
Blocks received:        9999
Blocks dropped:         0
--- MVA stats ---
Blocks Dropped:         0
Blocks Complete:        9999
Blocks Incomplete:      0
Blocks Not Available:   3546
Resend Requests:        0
Resend Requests OK:     0
Resend Requests Failed: 0
Block Timeouts:         0
Resend Timeouts:        0
Max Resend Retries:     0
Buffers Allocated:      64
Buffers Freed:          0
Buffers Enqueued:       63
```

7.1.3 mva_multi_dev

Demonstrates how independent threads can handle separate streams without locking.

NOTE:	This test program requires Linux.

When stopped by pressing **[CTRL]** + **C**, a summary of the packets received is displayed.

Usage

Linux

```
./mva_multi_dev [args] <local_IP>:<port> [<local_IP>:<port>]
```

Windows

```
.\mva_multi_dev [args] <local_IP>:<port> [<local_IP>:<port>]
```

Arguments

See the following table for the available options.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv or -v-v.
-t	Prints statistics every second.
-T <seconds>	Specifies how long to run the script. If set to -1 (default), the script runs indefinitely until it is manually stopped by pressing **[CTRL]+C**.
-b <bock_timeout>	Specifies the number of msecs to wait for each block. If -1 (default), there is no timeout period (i.e., the application will wait indefinitely for the next block). If set to 0, the application will continuously poll the host for new blocks.
-i <iters>	Indicates the total number of blocks to receive before the script is terminated. This defaults to 1. If set to -1, the number of iterations is infinite.
-p <num_bufs>	Specifies the number of buffers to pre-queue. By default, this is set to 128.
-r	Reads the blocks after they are received.
-D <nanosecs>	Injects a synthetic processing delay based on the specified number of nanoseconds.
-s <buf_size>	Specifies the size of the buffers. By default, this is 2048KB.
<local_IP>	Identifies the IP address of the local MVA-enabled device on which to receive GVSP blocks.
<port>	Indicates the UDP port on the <local_IP> to use for receiving GVSP blocks.
--help	Displays help.

Table 10: mva_multi_dev Arguments

Example

Receive blocks on port 20 for an MVA with an IP address of 192.168.0.100 and port 40 for an MVA with IP address of 192.168.0.200. Run the script until a total of 256 blocks are received.

```
./mva_multi_dev -i 256 192.168.0.100:20 192.168.0.200:40
```

```
Capture using 2 rings and worker threads with default RSS hashing
Worker 0 running on CPU 0
Worker 1 running on CPU 1

3800000 pkts (228000000B) in 16.000 secs (237496 pps), Avg Pkt: 60, Q-fullness=
```

```
0.0% BW (Gbps):  0.114
14900000 pkts (894000000B) in 1.000 secs (14899389 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.152
14900000 pkts (894000000B) in 1.000 secs (14899613 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.152
14800000 pkts (888000000B) in 1.000 secs (14799615 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.104
14900000 pkts (894000000B) in 1.000 secs (14899375 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.152
14900000 pkts (894000000B) in 1.000 secs (14899583 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.152
14900000 pkts (894000000B) in 1.000 secs (14899599 pps), Avg Pkt: 60, Q-full-
ness= 0.0% BW (Gbps):  7.152
7066400 pkts (423984000B) in 1.000 secs (7066152 pps), Avg Pkt: 60, Q-fullness=
0.0% BW (Gbps):  3.392

^C
Ring 0 Packets received,    ring:           0, app:           0, overflow:
    0
Ring 1 Packets received,    ring:   100166400, app:   100166400, overflow:
    0
MVA dropped overflow SW  :            0
NIC dropped overflow HW  :            0
NIC dropped bad PHY/CRC  :            0
Total packets:                 app:   100166400
Average Packet Length:         app:          60 bytes
Total bytes:                   app:  6009984000 (5731 MB)
Total raw bytes + HW align     nic:  4807987200 (4585 MB)
```

7.1.4 mva_rdwt_recv_pr

Receives packets from multiple worker threads and enables the packet resend feature. When stopped by pressing **[CTRL] + C**, a summary is displayed in the console.

Usage

Linux

```
./mva_rdwt_recv_pr [args] -P <unicast_addr>:<port> <local_IP>:<port>
```

Windows

```
.\mva_rdwt_recv_pr [args] -P <unicast_addr>:<port> <local_IP>:<port>
```

Arguments

See the following table for the available arguments.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv or -v-v.
-w <worker_num>	Specifies the number of worker threads to employ.
-T <seconds>	Specifies how long to run the script. If set to -1 (default), the script runs until it is manually stopped by pressing **[CTRL]+C**.
-i <iters>	Indicates the total number of blocks to receive before the script is terminated. This defaults to 1. If set to -1, the number of iterations is infinite.
-r	Instructs MVA to read the blocks after they are received.
-n <num_bufs>	Indicates the number of buffers to allocate. By default, this is set to the max value of 512.
-p <num_bufs>	Specifies the number of buffers to pre-queue. By default, this is set to the max value of 511.
-D <nanosecs>	Injects a synthetic processing delay based on the specified number of nanoseconds.
-V	Verifies the block IDs are sequential.
-m <mcast_addr>	Instructs MVA to listen on a multi-cast address.
-s <buf_size>	Specifies the size of the buffers. By default, this is 2MB.
<local_IP>	Identifies the IP address of the local MVA-enabled device on which to receive GVSP blocks.
<port>	Indicates the UDP port on the <local_IP> to use for receiving GVSP blocks.
-S	Prints link and time source state every second.
-t	Prints the time between packets.
-c <CPU>	Pins the process to the specified CPU number (starting at 0).
-P <unicast_addr>	Defines the destination IP address for packet resend requests. This argument is required.
-R <num_retries>	Specifies the maximum number of packet resend retries to attempt.
-E <resend_timeout>	Specifies the number of milliseconds the receiver API will wait for packets after the packet resend is issued. By default, this is 0, which indicates the API will wait indefinitely. **Note:** You must set the resend_timeout, num_retries, or block_timeout to a non-zero value. All three options cannot be set to 0.
-b <block_timeout>	Specifies the number of milliseconds the polling function on the host application will wait for a block to complete. By default, this is 0, which indicates the function will wait indefinitely. **Note:** You must set either the resend_timeout or block_timeout to a non-zero value. Both cannot be set to 0.
-d <booo>	Sets the block our-of-order flag. By default, this is MVA_BOOO_ALWAYS_ RETURN. See *block_out_of_order on page 33* for details about the possible values.
-C <channel>	Defines the channel ID used for packet resend requests.

Argument	Meaning
--drop-incomplete	Drops incomplete blocks instead of returning them to the host application. This is for internal use only and will generate an error.
--worker-req-id	Starting request id for the worker thread packet resend request. By default, this is 1.
--max-worker-reqs	Defines the maximum number of worker thread packet resend requests. By default, this is 1.
--block-worker <nsecs>	Blocks the worker rthread for <nsecs> after receiving a block.
-h (--help)	Displays help.

***Table 11:** mva_rwdt_recv_pr Arguments*

Return

When mva_rwdt_recv_pr is stopped, it displays statistics for the application and MVA. See the following table for a list of these statistics.

Statistic	Meaning
Blocks Received	Indicates the total number of blocks the MVA adapter received.
Blocks Non-Zero Status	Specifies the number of blocks that were received with a status that did not equal 0.
Poll Retries	Specifies the number of poll requests the host application issued.
Last Block ID	Indicates the block ID of the last block received.
Blocks Dropped	Displays the total number of blocks that were dropped.
Blocks Complete	Indicates the total number of complete blocks returned to the host application.
Blocks Incomplete	Specifies the total number of incomplete blocks returned to the host application.
Blocks Not Available	Lists the total number of times a block was not returned because no blocks were available for the host to consume. This counter will increment if a block is requested but the block is not ready to move to user space.
Resend Requests	Indicates the total number of packet resend requests that were issued.
Resend Requests OK	Specifies the total number of resend requests that completed successfully.
Resend Requests Failed	Indicates the total number of resend requests that did not complete.
Block Timeouts	Indicates the number of times the block_timeout timer expired.
Resend Timeouts	Specifies the number of times the resend_timeout timer expired.
Max Resend Retries	Indicates the total number of times the maximum number of retries was exceeded.
Buffers Allocated	Specifies the number of buffer locations allocated for blocks.
Buffers Freed	Indicates the total number of buffers that were freed.
Buffers Enqueued	Identifies the number of buffers available for blocks.
Stream Stopped	Indicates the number of streams that were stopped.
Stream Started	Specifies the number of streams that were started.

Statistic	Meaning
Stream Blocked	Indicates the number of times a stream was blocked.
Stream Unblocked	Specifies the number of times a stream was unblocked.
Callbacks Invoked	Indicates the number of times the callback function was invoked.
Blocked Blocks Dropped	Specifies the number of blocked blocks that were dropped.

Table 12: mva_rwdt_recv_pr Statistics

Example

Receive blocks with packet resend enabled on port 6000 for the MVA with an IP address of 192.168.0.100 and a unicast IP address of 192.168.0.101. Set the number of workers to 5, with packet resend requests starting at worker 1, add a 50 nanosecond delay, and set the maximum number of packet resend requests per worker to 4.

```
./mva_rwdt_recv_pr -w5 -v 192.168.0.100:60000 -P 192.168.0.101:60001 --max-worker-reqs 4 --worker-req-id 1 --block-worker 50
```

```
Stream opened on 192.168.0.100:20
Allocated 64 buffers
Queued 63 buffers (max is 511)

^C
--- APP stats ---
Blocks received:        10
Blocks non-zero status: 0
Poll retries:           283137842
Last Block ID:          10

--- MVA info ---
Blocks received:        10
Blocks dropped:         0

--- MVA stats ---
Blocks Dropped:         0
Blocks Complete:        9
Blocks Incomplete:      1
Blocks Not Available:   283137842
Resend Requests:        7
Resend Requests OK:     7
Resend Requests Failed: 0
Block Timeouts:         0
Resend Timeouts:        0
Max Resend Retries:     0
Buffers Allocated:      63
Buffers Freed:          0
Buffers Enqueued:       80
Stream Stopped          1
Stream Started          1
Stream Blocked          10
Stream Unblocked        10
```

```
Callbacks Invoked       6
Blocked Blocks Dropped  0
```

7.1.5 mva_multi_thread

Demonstrates how to serialize access to MVA API functions for multiple threads that are using the same stream handle.

Usage

Linux

```
./mva_multi_thread [args] <local_IP>:<port>
```

Windows

```
.\mva_multi_thread [args] <local_IP>:<port>
```

Arguments

See the following table for the available options.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv.
-t	Prints statistics every second.
-w <workers>	Specifies the number of workers.
-T <seconds>	Specifies how long to run the script. If set to -1 (default), the script runs indefinitely until it is manually stopped by pressing **[CTRL]+C**.
-b <block_timeout>	Specifies the number of msecs to wait for each block. If -1 (default), there is no timeout period (i.e., the application will wait indefinitely for the next block). If set to 0, the application will continuously poll the host for new blocks.
-i <iterations>	Indicates the total number of blocks to receive before the script is terminated. This defaults to 1. If set to -1, the number of iterations is infinite.
-p <num_buffers>	Specifies the number of buffers to pre-queue. By default, this is set to 128.
-r	Reads the blocks after they are received.
-D <nanosecs>	Injects a synthetic processing delay based on the specified number of nanoseconds.
-s <buf_size>	Specifies the size of the buffers. By default, this is 2048KB (2MB).
<local_IP>	Identifies the IP address of the local MVA-enabled device on which to receive GVSP blocks.
<port>	Indicates the UDP port on the <local_IP> to use for receiving GVSP blocks.
--help	Displays help.

***Table 13:** mva_multi_thread Arguments*

Example

Receive blocks for three different workers on port 20 for the MVA with an IP address of 192.168.0.100. Display the statistics every second.

```
./mva_multi_thread -t -w 3 192.168.0.100:20
```

```
mva_recv ready to receive

14881288 pkts (892877280B) in 1.000 secs (14880827 pps), Avg Pkt: 60, BW (Gbps):
 7.143
14881328 pkts (892879680B) in 1.000 secs (14880867 pps), Avg Pkt: 60, BW (Gbps):
 7.143
11151056 pkts (669063360B) in 1.000 secs (11150644 pps), Avg Pkt: 60, BW (Gbps):
 5.352

^C
Packets received   in HW:        235421440
Packets reinjected,    app:         0
Packets reflected to netdev:        0
Total bytes received,  app:         14125286400 (13470 MB)
Total bytes received, HW: 14125286400 (13470 MB)
Average Packet Length:    60 bytes
Dropped, NIC overflow:    0
Dropped, ring overflow:   0
Dropped, bad:             0
```

7.1.6 mva_set_get_name

Tests the ability to set and retrieve adapter names as well as return them to the default value.

Usage

Linux

```
./mva_set_get_name [args] "S=<serial_number>" or "M=<MAC_address>"
```

Windows

```
.\mva_set_get_name [args] "S=<serial_number>" or "M=<MAC_address>"
```

Arguments

See the following table for the available options.

Argument	Meaning
-v	Enables verbose mode. To enable very verbose mode, use -vv.
-t	Indicates there are two boards. If specified, you must provide two names in either format.
-f	Forces the program to use "S=123456" and checks for failures.
-m	Tests the name with the maximum number of characters.
-s <size>	Specifies the number of characters in the name.

***Table 14:** mva_set_get_name Arguments*

Example

Verify the adapter accept a new serial number and various MAC addresses.

```
./mva_set_get_name "S=426927" "M=00:60:dd:45:1f:e8" "M=00:60:dd:45:1f:e9"
```

```
mva_set_get_name

ALL Tests PASSED
```

7.2 Diagnostic Programs

The MVA distribution also provides diagnostic tool programs, in sbin/ and bin/, as documented in this section. Since the /sbin/ tools must be run as root, you must either add /opt/mva/sbin/ to your PATH, or execute the command using the full path (e.g., /opt/mva/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 Support. Each of the available programs is described in the following sections.

7.2.1 myri_bug_report

Runs a diagnostic test and generates a bug report as a compressed file. For Windows, you must use myri_bugreport.ps1, which is a Powershell script available with Myri Tools. Contact ARIA Support (ARIA_support@ariacybersecurity.com) for details.

NOTE:	Administrator privileges are required to run the script on Windows.

Usage

Linux

```
./myri_bug_report
```

Windows

```
.\myri_bug_report.ps1 > <file_name>.txt
```

Options

If running the Powershell script on Windows, you must provide the name of the output file (.txt). The Linux command does not support any options.

Example

Generate the bug report.

```
sbin/myri_bug_report
```

```
Will collect system info in file:
bugreport.test9-2016-08-16_160853.txt.gz

Gathering diagnostic information...

bugreport.test9-2016-08-16_160853.txt.gz created
Please send it to ARIA Cybersecurity Solutions Technical Support via the case
you have opened
https://www.cspi.com/ethernet-products/support
or support@ariacybersecurity.com and include a description of your problem.
```

7.2.2 myri_info

Returns network adapter information, such as the hardware serial number, MAC address, firmware version, and clock speed.

Usage

Linux

```
./myri_info [args]
```

Windows

```
.\myri_info [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <board_num>	Specifies the adapter to query. Use myri_nic_info to determine the board number (see page 95).
-v	Returns verbose information, such as the revision number. Use -vv for very verbose information.
-x	Returns information for LX-based cards only.
-z	Returns information for 10G cards only.
-s <spec_file>	Saves the information in the specified file.
-c	Verifies the CRCs on 10G cards.
-n	Does not use memory-mapped input/output.

Table 15: myri_info Arguments

Example

Query board 0.

```
./myri_info -b 0
```

```
pci-dev at 03:00.0 vendor:product(rev)=14c1:0008(01)
       behind bridge downstream-port: 02:02.0 111d:806a (x8.1/x8.2)
       behind bridge upstream-port: 01:00.0 111d:806a (x8.2/x8.2)
       behind bridge root-port: 00:01.0 8086:0c01 (x8.2/x16.3)
10G-PCIE-8B -- Link x8
  EEPROM String-spec:
       MAC=00:60:dd:46:e1:1a
       SN=354212
       PWR=100
       PC=10G-PCIE2-8B2-2S
       PN=09-04080
       TAG=ze_tools-1_4_38

License keys:
       395e-cc2b-016e-8316:2:354212:DBL:V3: # DBL, V3
       8dea-c720-7619-25b0:2:354212:DBL:V2: # DBL, V2
       96f1-7295-523d-b421:2:354212:MVA:V1: # MVA, V1
       0a6d-e106-503f-e174:2:354212:VPUMP:V2: # VPUMP, V2
       f295-8265-f39c-fc69:2:354212:SNF:V2: # SNF, V2
       4621-896e-84eb-5acf:2:354212:SNF:V3: # SNF, V3

EEPROM MCP, PRESENT, length = 106980, crc=0xafd35588
       ETHZ::1.4.59 2014/06/27 21:42:52 self extracting firmware
       Simple-bundle: exec_len = 106976
Running MCP:
       ETH ::1.4.59 -- 2014/06/27 21:42:52 myri10ge netq firmware
```

7.2.3 myri_license

Programs the software license key(s) into the network adapter(s) installed in the host. If no options are provided, the currently loaded licenses are returned.

Usage

Linux

```
./myri_license [args]
```

Windows

```
.\myri_license [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <unit>	Specifies an adapter for reporting. If -b is omitted, all adapters are used.
-l <key>	Loads a license key onto the specified adapter (-b).
-c	Removes the license(s) from the specified adapter.
-f <license_file>	Loads the license(s) from the <license_file> into all matching adapters.
-h (--help)	Displays help.

Table 16: myri_license Arguments

Example

Retrieve information about the licenses installed on all adapters.

```
./myri_license
```

```
96f1-7295-523d-b421:2:354212:MVA:V1: # MVA, V1
```

7.2.4 myri_bandwidth

Shows the instantaneous bandwidth of data going through a given network adapter. This is useful when displayed at intervals (-t option).

Usage

Linux

```
./myri_bandwidth [args]
```

Windows

```
.\myri_bandwidth [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <board_num>	Specifies the board based on the board number or the MAC address. This defaults to board 0.
-t <secs>	Indicates the time interval, in seconds. This defaults to 1.
-k	Runs the script until stopped with **[CTRL] + C**.
-p	Displays counters for each port on multi-port adapters.
-a	Displays aggregated counters for each iteration.
-A	Displays only aggregated counters for each iteration.
-h (--help)	Displays help.

Table 17: myri_bandwidth Arguments

Example

Retrieve bandwidth information for all adapters, displaying the results every five seconds.

```
./myri_bandwidth -t 5
```

```
Total : sent 0 kB (0.00 MB/s) received 0 kB (0.00 MB/s)
```

7.2.5 myri_counters

Generates output for low-level adapter counters.

> For more information about the counters, see *MVA Counters on page 115*.

Usage

Linux

```
./myri_counters [args]
```

Windows

```
.\myri_counters [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-p <board_num>	Returns counter information for a specific board based on the board number (e.g., 0 or 1) or MAC address (e.g., 11:22:33:44:55:66). To obtain the board number and MAC address, issue myri_nic_info (see *page 95* for more information).
-c	Clears the counters.
-q	Returns only non-zero counters.
-i	Includes host interrupt counters.
-x	Returns all counters.
-h (--help)	Displays help.

Table 18: myri_counter Arguments

Example

Retrieve all counters from board 0.

```
./myri_counters -p 0
```

```
Board ca:98:bd:65:68:1b with 1 ports
                  Lanai uptime (seconds):       5351081
              Counters uptime (seconds):       5351081
                        Net send KBytes:             0
                        Net recv KBytes:            81
```

```
                        Ethernet send:              0
                   Ethernet Small recv:             0
                      Ethernet Big recv:            0
                     Ethernet recv down:            0
                  Ethernet recv overrun:            0
                       Ethernet re-recv:            0
                 Ethernet recv oversized:           0
                                MVA recv:           0
                    Drop endpoint closed:           0
                    Drop parity recovery:           0
                       MVA drop no pages:           0
                  MVA hostq prefetch race:          0
                  MVA hostq prefetch race:          0
                 MVA block drop no buffer:          0
                    MVA packet drop order:          0
                 MVA packet drop non-GVSP:          0
                      MVA buffer overflow:          0
                  Flow-control Pause recv:          0
                             Net send Raw:          0
                               Interrupts:          0
                           Wake interrupt:          0
                                Wake race:          0
                    Wake endpoint closed:           0
                         Net send queued:           0
                        Event Queue full:           0
                           RX DataQ race:           0
                      Fragmented request:           0
 Net bad PHY/CRC32 drop (Port 0):                   0
      Net overflow drop (Port 0):                   0
                     Net Recv PAUSEs:               0
      Net recv alternate channel:                   0
                         Net Alt drop:              0
  Ethernet Multicast filter drop:                   0
    Ethernet Unicast filter drop:                   0
                  Out of send handles:              0
       User request type unknown:                   0
           Spurious user request:                   0
                            Drop resend:            0
                       Incomplete final:            0
                             Incomplete:            0
```

7.2.6 myri_dmabench

Verifies the PCI DMA read and DMA write bandwidth for the PCIe slot for which the adapter is installed.

Usage

Linux

```
./myri_dmabench [args]
```

Windows

```
.\myri_dmabench [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <board_num>	Specifies the board based on the board number or the MAC address. This defaults to board 0.
-s <dma_size>	Specifies the DMA size. This must be a power of 2. The valid range is 1-4096, and the default is 4096.
-r	Tests only DMA reads (sends).
-w	Tests only DMA writes (receives).
-i <iterations>	Specifies the number of iterations to run. This defaults to 64.
-a	Tests all DMA sizes (i.e., powers of 2 between 1 and 4096).
-h (--help)	Displays help.

***Table 19:** myri_dmabench Arguments*

Example

Verify the DMA read and write bandwidth for 512 DMA units.

```
./myri_dmabench -s 512
```

```
DMA timings for 00:60:dd:46:e1:1a
       LANai: 364.4 MHz        PCI-E x8        (1MB = 1048576 Bytes)
       DMA read  (send) Bandwidth = 1545.89 MB/sec (512 bytes per DMA)
       DMA write (recv) Bandwidth = 1515.69 MB/sec (512 bytes per DMA)
```

7.2.7 myri_endpoint_info

Identifies which processes, such as endpoints, are consuming adapter-level resources. This also returns the process IDs in use.

Usage

Linux

```
./myri_endpoint_info [args]
```

Windows

```
.\myri_endpoint_info [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <board_num>	Specifies the port MAC address or board number. If omitted, information for port 0 is returned.
-h (--help)	Displays help.

Table 20: myri_endpoint_info Arguments

Example

Retrieve the endpoint information for port 0.

```
bin/myri_endpoint_info
```

```
#  ./bin/myri_endpoint_info
The myri_mva driver is configured to support a maximum of:
               8 endpoints per NIC, 32 NICs per host
================================================================
Board 00:60:dd:46:7c:6c
Endpoint         PID             Command                 Info
<ether>         none            none
0               2234            mva_simple_recv
1               2238            mva_simple_recv
2               2242            mva_simple_recv
3               2246            mva_simple_recv
4               2250            mva_simple_recv
5               2254            mva_simple_recv
6               2258            mva_simple_recv
7               2262            mva_simple_recv
There are currently 8 regular endpoints open
```

7.2.8 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 network adapter.

Usage

Linux

```
./myri_nic_info [args]
```

Windows

```
.\myri_nic_info [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-m (--machine)	Returns a comma-separated list.
-B	Displays the board numbers.
-a (--all)	Returns information for all known adapters.
--license	Returns the license status details.
-h (--help)	Displays help.

***Table 21:** myri_nic_info Arguments*

Example

Retrieve information about the installed adapters in a comma-separated format.

```
./myri_nic_info -m
```

```
0,491942,00:60:dd:43:48:b6,10G-PCIE3-8B-2S,myri_mva,1.3.2.2.54367,Valid
1,491942,00:60:dd:43:48:b7,10G-PCIE3-8B-2S,myri_mva,1.3.2.2.54367,Valid
```

7.2.9 myri_intr_coal

Adds an interrupt coalescing delay.

Usage

Linux

```
./mva_intr_coal [args]
```

Windows

```
.\mva_intr_coal [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b	Specifies the board number (e.g., 0) or the board MAC address.
-s <milliseconds>	Sets the interrupt delay time in milliseconds. By default, this is 0.
-h (--help)	Displays help.

***Table 22:** mva_intr_coal Arguments*

Example

Receive blocks for three different workers on port 20 for the MVA with an IP address of 192.168.0.100. Display the statistics every second.

```
./mva_multi_thread -t -w 3 192.168.0.100:20
```

```
mva_recv ready to receive

14881288 pkts (892877280B) in 1.000 secs (14880827 pps), Avg Pkt: 60, BW (Gbps):
 7.143
14881328 pkts (892879680B) in 1.000 secs (14880867 pps), Avg Pkt: 60, BW (Gbps):
 7.143
11151056 pkts (669063360B) in 1.000 secs (11150644 pps), Avg Pkt: 60, BW (Gbps):
 5.352

^C
Packets received   in HW:        235421440
Packets reinjected,    app:        0
Packets reflected to netdev:       0
Total bytes received,  app:        14125286400 (13470 MB)
Total bytes received, HW: 14125286400 (13470 MB)
Average Packet Length:    60 bytes
Dropped, NIC overflow:    0
Dropped, ring overflow:   0
Dropped, bad:             0
```

Chapter 8

Troubleshooting

This chapter identifies some of the issues you may encounter while using MVA. Each section provides information specific to different issues, such as dropped packets. ARIA Cybersecurity Solutions recommends reviewing this section before contacting ARIA Support. Should you determine the adapter is defective, contact support to initiate a Return Merchandise Authorization (RMA) to obtain a replacement, provided the adapter is still under warranty.

If you experience issues with the MVA adapter, run mva_simple_recv (see *mva_simple_recv on page 76*) and verify the output is correct. You should also retrieve the bug report using myri_bug_report (Linux) or myri_bugreport.ps1 (Windows) as described in *myri_bug_report on page 88*.

8.1 Troubleshooting Chart

The following table lists some issues you may encounter when installing or using the MVA software. Exhausting all possibilities in this table before contacting ARIA Support is recommended.

Issue	Probable Cause	Possible Solution
InstallAnywhere is unable to detect the adapter.	You are attempting to install the software on a VM, but passthrough mode is not enabled.	Make sure passthrough mode is enabled before you attempt to install the software. See *Configuring Passthrough Mode on page 1* for more information.
	The adapter is not properly seated in the PCIe slot.	Use lspci or myri_info to verify the host can communicate with the adapter. If no response is returned, make sure the adapter is receiving adequate power or is properly seated.

The adapter appears to have stopped working.	The driver is not responding.	Restart the driver using the myri_start_stop script.
		`sudo /opt/mva/sbin/myri_start_stop restart`
		`C:\MVA_Myri-10G\sbin\myri_start_stop.exe restart`

Windows returns an error when running myri_bugreport.ps1.	The execution policy is not set.	Set the execution policy to remote signed using the following command. `set-executionpolicy remotesigned`

Table 23: Troubleshooting Chart

8.2 Hardware Issues

Link LED behaviors are not well defined when ARC Series adapters are disabled or when MVA software is uninstalled. In addition, the adapter may not correctly process remote signal loss which may not reflect the true state of the Link.

After installing the software, it is strongly recommended to ping the device before proceeding with testing.

> For more information on LED behavior, see *Testing the Adapter on page 25*

8.2.1 Cabling Issues

If you are using MVA "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_mva driver with the load-time option myri_serdes_mode=2.

On Linux, issue the following command:

```
/opt/mva/sbin/myri_start_stop start myri_serdes_mode=2
```

On Windows, create the following registry key (followed by a reboot):

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

There are 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.

The possible options for myri_serdes_mode=X are:

- 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 MVA "8B" adapters (e.g., 10G-PCIE2-8B2-2S), the load-time option myri_serdes_mode=X is ignored. If you experience link up/down connectivity issues or bad CRC errors with the MVA "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.

8.2.2 Hardware Installation and Performance

Your host/motherboard may have either PCIe 3.0 (Gen3), PCIe 2.0 (Gen2), or PCIe 1.1 (Gen1) PCI-Express slots. ARIA Cybersecurity Solutions sells both Gen2 x8 and Gen1 x8 10-Gigabit network adapters.

For optimal performance, verify that the adapter reports Gen2 x8 (5 GT/s) PCIe link speed, once it is seated in the PCIe expansion slot on the server.

MVA Gen1 PCI Express adapters are compatible with Gen3 and Gen2 slots in hosts; the adapter auto-negotiates operation in the widest available mode (x8) supported by the slot into which it is installed, and at the 2.5 GT/s data rate. Similarly, MVA Gen2 PCI Express adapters are compatible with Gen3 slots in hosts and auto-negotiate operation in the widest available mode (x8 or x16) 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.

The Myricom 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. Gen2 adapters include "PCIE2" in the product code (e.g., 10G-PCIE2-8B2-2S), and Gen1 adapters include "PCIE" (e.g., 10G-PCIE-8B-S. For optimal performance, verify the adapter reports Gen2 x8 (5.0 GT/s) PCIe link speed if you installed an Gen2 x8 MVA network adapter.

There are two ways to determine if the Gen2 MVA adapter is installed into a Gen2 PCIe slot: the output of myri_info; or the output of lspci -vvv.

The following is example information from the output of myri_info for a Gen2 MVA 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 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 the adapter is currently running at Gen2 x8 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 the controllers inside the adapter, which are x8.1 (x8 at Gen1 speed), but there is one link for each of the two LANai chips.
- The Myri-10G-PCIE-8B -- Link x8 also indicates that the adapter is optimally running at x8 speed.

Alternatively, 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 MVA network adapter has a PLX bridge chip. There will be four PLX chip entries in the lspci output, as well as two 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 (AKA 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 four 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 MVA network adapter has an IDT bridge chip. There will be three IDT chip entries in the lspci output, as well as two 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 three IDT chips, the network adapter is installed into a Gen1 PCIe slot. For similar performance reasons, if you have installed an Gen1 x8 MVA network adapter, verify that the adapter reports Gen1 x8 (2.5 GT/s) PCIe link speed.

8.3 Software Installation and Configuration

Should you encounter problems with MVA software installation, usage, or performance, send the bug report script to ARIA Support. The script output contains the vital information required to quickly resolve your software issues.

The diagnostic script myri_bug_report is included in the Linux MVA software distribution. For Windows, the Powershell script myri_bugreport.ps1 is included in Myri Tools, which you must request from ARIA Support (ARIA_support@ariacybersecurity.com). The script is used to collect diagnostic information about a your 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 (Linux) or as an administrator (Windows).

NOTE:	It is very important that you retrieve the bug report script from the /opt/mva/sbin (Linux) or C:\MVA_Myri-10G (Windows) directory; otherwise, important diagnostic information may not be collected.

The [INSTALLDIR]\sbin\myri_dmesg.ps1 Powershell script is included in the Windows MVA software distribution. This script extracts logging information from the Windows logs and prints them to stdout. It will show whether the MVA 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 MVA software.)

8.3.1 Windows Installation Failures

If the Windows installation fails when running Install Anywhere, please send the ARIA_MVA_DEBUG.log file (located in C:\MVA_Myri-10G) to ARIA_support@ariacybersecurity.com.

If the installation fails with an ambiguous message stating that part of the setup did not finish as expected, please verify that an earlier removal has unlocked all processes from MVA-related files. A way to determine if MVA is still in use is to run mva.dll from a command prompt.

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

The output will show process(es) still using MVA-related software.

```
Image Name PID Modules
============ ========== ======================
svchost.exe 8948 mva.dll
```

Reboot the server.

8.3.2 Windows Driver Uninstallation

If the wrong driver was installed on the adapter, you may encounter an issue where you are unable to remove the driver using InstallAnywhere (see *Uninstalling the Driver on page 22*). If this occurs, you can remove the driver using Windows Device Manager or through the Add/Remove Program feature of Windows.

Add/Remove Programs

To remove the driver using the Add/Remove Program option:

1. Open Control Panel.
2. Click Programs > **Programs and Features**.
3. Right-click on the MVA_Myri-10G and select **Uninstall**. When finished, the program should no longer be displayed in the window.
4. Re-install the driver (see *Installing MVA Locally Using GUI on page 11*).

If you continue to have problems, see https://support.microsoft.com/en-us/topic/fix-problems-that-block-programs-from-being-installed-or-removed-cca7d1b6-65a9-3d98-426b-e9f927e1eb4d.

Device Manager

To remove the driver using Device Manager:

1. Open Device Manager.
2. Expand Network adapters.
3. Right-click on the MVA-enabled adapter and select **Uninstall Device**.
4. View the adapter properties to make sure the driver is uninstalled.
5. Re-install the driver (see *Installing MVA Locally Using GUI on page 11*).

If you continue to have problems, see https://support.microsoft.com/en-us/topic/fix-problems-that-block-programs-from-being-installed-or-removed-cca7d1b6-65a9-3d98-426b-e9f927e1eb4d.

8.3.3 License Issues

If there is no valid license loaded on a network adapter, the MVA software 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 myri_nic_info.

Linux

```
/opt/mva/bin/myri_nic_info --license
```

Windows

```
C:\MVA_Myri-10G\bin\myri_nic_info --license
```

Please contact ariasales@ariacybersecurity.com to obtain an appropriate key.

8.3.4 Software Counters

The myri_counters tool provides low-level MVA hardware and software counters for traffic passing through the network adapter.

> For a detailed description of the MVA software counters reported by myri_counters, please see *MVA Counters on page 115*.

8.3.5 Link Status and Time Source

To monitor the status of the link and timesource, use the '-S' command-line option with mva_simple_recv or mva_simple-recv_pr.

Linux

```
./mva_simple_recv -S <local_IP>:<port>
```

Windows

```
.\mva_simple_recv -S <local_IP>:<port>
```

If a 10G-PCIE2-8C2-2S-SYNC adapter is in use and the timesource is connected and synchronized, the following message will appear in the output every second.

```
LINK UP, External PPS: synchronized
LINK UP, External PPS: synchronized
LINK UP, External PPS: synchronized
```

If a non-SYNC adapter is used, the messages will indicate a local time source.

```
LINK UP, Local time source
LINK UP, Local time source
LINK UP, Local time source
```

Note that if the link is not up before mva_simple_recv or mva_simple_recv_pr is executed, the script will fail to start.

8.3.6 Performance

For best performance, disabling power saving modes in the host BIOS is recommended.

> For more information, see *Power Saving/Performance Speed on page 54*.

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

1. Verify the MVA network adapter is installed into an x8 PCIe slot on the motherboard.

   > For more information, see *Installing the Adapter on page 3*.

2. Verify the application was linked successfully with the libmva library.

   Linux

   ldd mva_simple_recv

   ```
   libmva.so => /opt/mva/lib/libmva.so (0x00007f83cb0c2000)
   ```

   Windows

   There is no Windows equivalent of ldd. If you use a dependency walker, look for mva.dll.

3. Use myri_license to verify the status of the license (see *Configuring the License on page 19*). Repeat the license activation if necessary.
4. Set the MVA_DEBUG and MVA_VERBOSE environment variables prior to execution to obtain diagnostic information from the MVA library.
5. Monitor myri_counters to verify that the adapter is receiving traffic (see *Understanding Counters on page 115* for more details). Contact ARIA Support at ARIA_support@ariacybersecurity.com for assistance if there are issues with the counters.

8.4 Advanced Troubleshooting Tools

The /opt/mva/sbin directory (Linux) contains additional scripts that may assist ARIA Support when troubleshooting issues. For Windows, many of these scripts are included with Myri Tools, which are located in C:\Myricom\uba-tools-x64. Because these tools are designed for advanced troubleshooting, you should not run the scripts unless instructed to do so by ARIA Support.

NOTE:	Contact ARIA Support (ARIA_support@ariacybersecurity.com) for information about installing the tools on Windows.

See the following table for a list of these additional tools.

Program	Description	Page
mva_create_dev	Creates Linux files required for communicating with the MVA driver.	106
mva_kpoke	Examines memory associated with the adapter.	106
mva_lmesg	Displays log messages.	107
mva_local_install	Creates the devices and loads the scripts and modules.	108
mva_mdio_rw	Examines PHY/transceiver registers.	109
myri_port_failover	Controls failover behavior.	110
myri_sram_dump	Retrieves information about the adapter SRAM.	111
myri_ze_scan	Displays the LANai registers.	112

***Table 24:** Advanced Troubleshooting Tools*

Each of these is described in the following sections.

8.4.1 myri_create_devs

Creates the Linux device files used for communicating with the MVA driver. The myri_local_install script calls this tool.

Usage

```
./myri_create_devs
```

Arguments

This command does not support any arguments.

Example

Generate the Linux files required for communicating with the MVA driver.

```
./myri_create_devs
```

8.4.2 myri_kpoke

Examines memory associated with the adapter.

Usage

Linux

```
./myri_kpoke [args] <address>.<size> <val>
```

Windows

```
.\myri_kpoke [args] <address>.<size> <val>
```

Arguments

See the following table for the available options.

Argument	Meaning
<address>	Specifies the location to read/write. By default, this specifies a PCI-MEM address. To use a RAM location, include the -m option.
<size>	Indicates the length of a binary dump or the number of bytes based on PCI conventions. Valid values for number of the bytes based on PCI conventions are: b (byte=1 byte), w (word=2 bytes), l (long=4 bytes), and q (quad=8 bytes).
<val>	Specifies the value to write to the <address> location. If omitted, the data in the location is read.
-m	Indicates the <address> is a RAM address rather than a PCI-MEM address.
-i	Specifies the <address> location is in the special I/O port space (not MMIO).
-b <unit>	Indicates the <address> is IO and relative to start of NIC board (<unit>).
-2	Specifies BAR2 units instead of BAR0 units when used with -b.
-e	Accesses EEPROM of the 10G NIC when used with -b.
-c	Accesses the board through debug configuration space registers when used with -b.

Argument	Meaning
-n	Indicates to swap the bytes read or written.
-w	Writes the raw data taken from stdin.
-q	Prints the results only, not the progress.
-r <rabbit-ip>	Accesses memory from the rabbit MQ server.
-g mon-card:slot_2z_rank	Accesses memory from the LANai chip.

Table 25: myri_kpoke Arguments

Examples

Read swapped 32-bit (long) word at IO address 0xde000508.

```
./myri_kpoke 0xde000508.l
```

Read the 32-bit word at physical RAM address 0x20100.

```
./myri_kpoke -m 0x20100.l
```

Dump first MB of SRAM.

```
./myri_kpoke -b 0 0.0x100000
```

Read ISR of board (on little-endian host).

```
./myri_kpoke -nb0 0xfffdf8.l
```

Write swapped word in PCI-MEM.

```
./myri_kpoke -n 0 0xde000508.w 0x55aa
```

8.4.3 myri_lmesg

Displays log messages stored in LANai SRAM.

Usage

Linux

```
./myri_lmesg
```

Windows

```
.\myri_lmesg
```

Arguments

This command does not support any arguments.

Example

Display the LANai log messages.

```
./myri_lmesg
```

```
Assuming print buf at offset 0x1dc8e8, meta at 0x1de8e8
MCP printf2 initialization,magic was 0x0
0.066606:RS=0h,C,PE=8, ETH 1.4.45-ht2100 -- 2009/08/22 19:00:26 myri10ge firm-
ware
0.066701:i2c init: gpo=1 gpi=3, MDI=3, smb=0
0.070769:smb-slave:=0x92(0x49)
0.085991:PCIe l=0xff (0/0x0) FC: P=807f0e00h,NP=807f0200h,CPL=807f0e00h
23.484440:RS=0h,W,PE=c, ETH 1.4.57 -- 2013/10/23 13:58:51 myri10ge firmware
23.484523:i2c init: gpo=1 gpi=3, MDI=3, smb=0
23.493961:xfi_phy_init
```

8.4.4 myri_local_install

Creates the devices, sets up the initialization scripts, and loads the modules. This script is automatically run when the driver is installed.

Usage

Linux

```
./myri_local_install [--module]
```

Windows

```
.\myri_local_install [--module]
```

Argument

See the following table for the available argument.

Argument	Meaning
--module	Copies modules to the driver directory and generates the dependency and map files.

***Table 26:** myri_local_install Argument*

Example

Create the devices and load the files needed for the driver without copying the modules to the directory.

```
./myri_local_install
```

8.4.5 myri_mdio_rw

Examines PHY/transceiver registers (TCVR).

Usage

Linux

```
./myri_mdio_rw [args] <dev>.<reg> <val>
```

Windows

```
.\myri_mdio_rw [args] <dev>.<reg> <val>
```

Arguments

See the following table for the available options.

Argument	Meaning
<dev>	Specifies the device part of a PHY register. This is used with <reg>.
<reg>	Indicates the register of the device specified in <dev>. This is interpreted as a decimal value unless 0x to indicate hexadecimal is used.
<val>	Specifies the value to write to the <dev>.<reg> location. If omitted, the data in the register is read.
-b <board_num>	Specifies the board number. By default, this is 0.
-s	Uses single-byte I2C operations to access TCVR.
-S	Uses multi-byte I2C operations to access TCVR.
-xx	Dumps the entire 256-byte TCVR table.
-N	Uses bar2 MMIO to access the MDIO bus.

***Table 27:** myri_mdio_rw Arguments*

Examples

Display TCVR status.

```
./myri_mdio_rw
```

```
Found 10G-PCIE2-8B2-2S: XFI=Firmware, PLUG=SFP+
Using method: Firmware
SFP+ info:
               Current status:

               Temp = 42.0781 Celsius
               Vcc = 3.2749 V
               TX bias = 9.178 mA
               TX power = 0.5478 mW
               RX power = 0.5347 mW
               Type = SFP (3)
               Connector = LC (7)
               Compliance = 10GBASE-SR (16)
               Wavelength = 850 nm
```

```
                Vendor = FINISAR CORP.
                PN = FTLX8574D3BCV
                SN = UW51UE0
                Date = 2016/08/02 (  )
```

Display laser-enable register.

```
./myri_mdio_rw 1.9
```

```
Found 10G-PCIE2-8B2-2S: XFI=Firmware, PLUG=SFP+
Using method: Firmware
0x1.9h = 0x0
```

Display TCVR status pins (QT2022-only).

```
./myri_mdio_rw 1.0xc200
```

```
Found 10G-PCIE2-8B2-2S: XFI=Firmware, PLUG=SFP+
Using method: Firmware
0x1.c200h = 0xaaaa
```

Turn laser on.

```
./myri_mdio_rw 1.9 0
```

```
Found 10G-PCIE2-8B2-2S: XFI=Firmware, PLUG=SFP+
Using method: Firmware
Setting 1.9h = 0x0
```

Turn laser off.

```
./myri_mdio_rw 1.9 1
```

```
Found 10G-PCIE2-8B2-2S: XFI=Firmware, PLUG=SFP+
Using method: Firmware
Setting 1.9h = 0x1
```

8.4.6 myri_port_failover

Controls failover behavior on NICs that have failover support. If no arguments, other than -b, are provided the failover status and settings are returned.

Usage

Linux

```
./myri_port_failover [args]
```

Windows

```
.\myri_port_failover [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <unit>	Specifies the board to use. By default, this is 0.
-0	Sets P0 as the primary link for failover purposes (P1 is backup).
-1	Sets P1 as the primary link for failover purposes (P0 is backup).
-s	Enables symmetric failover (only switch link if current port is down).
-r <0|1>	Disables/enables sending an RARP packet on failover (to update switches tables).

Table 28: myri_port_failover Arguments

Examples

Set P1 as the primary link for failovers on board 0.

```
./myri_port_failover -b 0 -1
```

Disable sending an RARP packet on failover on board 0.

```
./myri_port_failover -b 0 -r 0
```

8.4.7 myri_sram_dump

Examines the LANai SRAM.

Usage

Linux

```
./myri_sram_dump [args]
```

Windows

```
.\myri_sram_dump [args]
```

Arguments

See the following table for the available options.

Argument	Meaning
-b <num>	Specifies the board to use. By default, this is 0.
-d	Uses the cfg-pio access method.
-D	Uses the MMIO access method.
-x	Selects only LX-based card (D, E, or F).
-z	Selects only 10G cards.
-g <mon-ip>:<slot>:<2z_ rank>	Selects a LANai on a mon-card.

Argument	Meaning
-r	Outputs raw binary bytes (defaults to lxgdb core).
-e	Enables LANai-card mcp0 sram-shuffling but does not dump SRAM data.
-s <size>	Instructs mcp0 to save <size> KB (default 512) when used with -e.
-n	Ignores the next reset when used with -e (for “hard=" mcp-switching).
-c	Dumps the SRAM, reconstituting shuffled parts, for LANai-cards.
-f <file>	Uses RAM SRAM file image rather than the real NIC.
-F <file>	Uses coredump file image rather than the real NIC.

Table 29: myri_sram_dump Arguments

Example

Display the SRAM on 10G cards.

```
sbin/myri_sram_dump -z
```

```
ramsize = 2048 Kbyte
0x95ba165e
0xc5b846fe
0xcf592799
0x5f245a6d
0x55ca657d
...
```

8.4.8 myri_ze_scan

Displays LANai registers.

Usage

Linux

```
./myri_ze_scan -b <unit>
```

Windows

```
.\myri_ze_scan -b <unit>
```

Argument

See the following table for the available option.

Argument	Meaning
-b <unit>	Specifies the board to use. By default, this is 0.

Table 30: myri_ze_scan Argument

Example

Display the LANai registers on board 0.

```
./myri_ze_scan -b 0
```

```
P0_TX_DATA_RECEIVED=0x0
P0_TX_TAIL_RECEIVED=0x0
P0_TX_Q_SENT=0x1
P0_TX_I_SENT=0x1
P0_TX_S_SENT=0x0
P0_TX_D_SENT=0x0
P0_TX_T_SENT=0x0
P0_TX_FLOW_CONTROL_ON=0x1
P0_RX_FLOW_CONTROL_ON=0x1
P0_TX_TIMEOUT=0x0
...
```

MVA Counters

The MVA hardware and software provides various counters that allow you to review metrics based on packets and traffic on the PCIe bus.

To retrieve these counters, use the myri_counters program.

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:

Linux

```
/opt/mva/bin/myri_counters -p 0
/opt/mva/bin/myri_counters -p 1
```

Windows

```
C:\MVA_Myri-10G\bin\myri_counters -p 0
C:\MVA_Myri-10G\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.

> For more information about the command line options, see *myri_counters on page 92*.

This chapter describes each available counter.

A.1 Understanding Counters

The following table lists the different counters and the values they return.

Name	Returns
Lanai uptime (seconds)	The time (in seconds) since the MVA driver was loaded.
Counters uptime (seconds)	The time (in seconds) since the MVA counters were cleared.
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 Small recv	The number of small (typically 256 bytes) Ethernet packets passed through the OS driver.

Name	Returns
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 configured the Ethernet interface.
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.
MVA recv complete block	The total number of complete blocks the adapter placed in the user space for the host application to consume.
Drop endpoint closed	The number of packets dropped because the user level endpoint that is supposed to receive it is closed at this time.
Drop parity recovery	The number of packets dropped while the user level endpoint is still recovering from a parity error.
MVA drop no pages	A non-zero value for this counter indicates an internal error occurred where the adapter is not able to fetch hardware addresses from the host quickly enough. This error should never occur.
MVA hostq prefetch race	A non-zero value for this counter indicates an internal error occurred where the software exhausted all of the address prefetch slots. This error should never occur.
MVA block drop no buffer	A non-zero value for this counter indicates that the adapter received the beginning of a GVSP block but there were no queued buffers in which to put the data packet payloads. This situation can occur if the application does not pre-queue enough buffers using the mva_queue_buffer() function or does not call the function at a rate equal to or greater than the block receive rate.
MVA recv header	The total number of leading packet headers the adapter sent to the host application.
MVA incomplete	The total number of blocks that contained an out-of-order packet. The adapter drops packets until the next expected packet ID is received.
MVA incomplete final	The total number of incomplete blocks the adapter sent to the host application.
MVA packet drop resend	The total number of packets the adapter dropped as it was waiting for the next expected packet ID after the resend request was issued.
MVA packet drop order	If MVA_OPEN_ZEROLOSS is not enabled (it is not enabled by default) and a GVSP packet is received out of order, the out of order packet and all subsequent packets until the next leader packet are dropped. This counter is incremented for each dropped packet.

Name	Returns
MVA packet drop non-GVSP	A non-zero value for this counter indicates that a non-GVSP packet was received on the UDP port associated with the stream. Such packets are dropped by the network adapter.
MVA buffer overflow	A non-zero value for this counter indicates a usage error occurred where a buffer posted with mva_queue_buffer was too small to accommodate an incoming GVSP block.
Flow-control Pause recv	Internal use only.
Net send Raw	The amount of raw data sent (in kilobytes).
Interrupts	The number of MSI hardware interrupts and legacy hardware interrupts.
Wake Interrupts	Internal use only.
Wake race	Internal use only.
Wake endpoint closed	Internal use only.
Net send queued	Internal 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	Internal use only.
Fragmented request	Internal 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.
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.
Net Recv PAUSEs	The number of Ethernet pause packets received.
Net recv alternate channel	Internal use only.
Net Alt drop	Internal 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	Internal use only.
User request type unknown	Internal use only.

Name	Returns
Spurious user request	Internal use only.
Drop resend	The number of packets that were initially dropped, but then resent.
Incomplete final	The number of blocks that were marked as incomplete-final.
Incomplete	The number of blocks that were marked as incomplete.

Table 31: myri_counters Counters

A.2 Clearing Counters

To clear the counters on a specific port of a network adapter, use the myri_counters program with the -c option.

```
sudo /opt/mva/bin/myri_counters -p <port_num> -c
```

> For more information about the myri_counters command, see *myri_counters on page 92*.

ARIA Technical Support

ARIA technical support, downloads, and user documentation are available from the ARIA Cybersecurity Solutions website.

ARIA Technical Support Customer Portal

http://www.ariacybersecurity.com/support

ARIA Email Support

ARIA_support@ariacybersecurity.com

ARIA Product Development
C/O CSP Inc.
175 Cabot Street, Suite 210
Lowell, MA 01854

Tel: (800) 325-3110

ARIA_support@ariacybersecurity.com

http://www.ariacybersecurity.com