SNFv5.3.2.7 User Guide

Version 5.3.2.7

February 21, 2020
PREFACE

The SNFv5.3.2.7 User Guide describes the run-time SNFv5.3.2.7 software package for the CSPi - Myricom ARC Series E-class of network adapters.

Intended Audience

The document is intended for system and networking architects looking for a tailored solution with a focus on packet capture and reduced CPU usage. In this context, the SNFv5.3.2.7 product provides a deployable solution by using a combination of advanced software stacks and 1- and 10-Gigabit network adapters. The document assumes that readers are familiar with C programming language, GNU development tools, and general computer maintenance.

Software developers interested in directly utilizing the advanced features of CSPi Myricom products through the SNF interfaces, should refer to the SNFv5.3.2.7 API Reference Manual.

A Note on Handling Network Adapters

Follow industry-standard ESD anti-static procedures when handling network adapters to avoid accidently damaging integrated circuits.

For more information on ESD anti-static procedures, go to: https://www.esda.org/about-esd/esd-fundamentals/part-3-basic-esd-control-procedures-and-materials/

Notices

The following notices are used in this document:

<table>
<thead>
<tr>
<th>Notice</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE:</td>
<td>These notices provide important tips, guidance, or advice.</td>
</tr>
<tr>
<td>ATTENTION:</td>
<td>These notices indicate potential damage to programs, devices, or data.</td>
</tr>
</tbody>
</table>
Nomenclature

The following terms can be used interchangeably in the SNFv5.3.2.7 User Guide document, unless otherwise noted.

<table>
<thead>
<tr>
<th>Common terms</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sniffer, Sniffer 5.3.2.7, SNF, SNFv5.3.2.7, SNF version 5.3.2.7</td>
<td>SNFv5.3.2.7</td>
</tr>
<tr>
<td>NIC, adapter, card, network adapter, network card, device, Ethernet card, quad-port, dual-port. (10G-PCIE3-8E-2S) (10G-PCIE3-8E-4S)</td>
<td>ARC Series E-Class network adapter Dual-port network adapter Quad-port network adapter</td>
</tr>
</tbody>
</table>

Typographic Conventions

This section describes typographic conventions used in this document.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boldface type</strong></td>
<td>Emphasizes heading levels, column headings, and the following literals when writing procedures: Names of options and elements that appear on screens. Keyboard keys. Menu choice(s) and command selection(s). User input for procedures. Notes and attentions.</td>
</tr>
<tr>
<td><strong>Italic type</strong></td>
<td>Accentuates words and phrases that have special meaning or are being defined. Chapter titles: Chapter 1 “Introduction” Section titles: Section 1.4 “SNFv5.3.2.7 Components”</td>
</tr>
<tr>
<td><strong>Italic underline type</strong></td>
<td>Emphasizes a term, feature, or action: Example: SNFv5.3.2.7 does not support Windows…</td>
</tr>
<tr>
<td><strong>Boldface Courier type</strong></td>
<td>Coding format: $ /opt/snf/bin/tests/snf_simple_recv -p 0 -v</td>
</tr>
<tr>
<td><strong>Hyperlink</strong></td>
<td>Provides quick and easy access to web pages and cross-referenced topics. Hyperlinks are highlighted in blue and may be underlined.</td>
</tr>
</tbody>
</table>
Technical Support

SNFv5.3.2.7 software documentation, technical support, and downloads are available from the CSPi website, as follows:

CSPi website
https://www.cspi.com/network-adapters

Contact CSPi Technical Support via the CSPi Customer Portal
https://www.cspi.com/cybersecurity-products/support/

CSPi email support address
support@cspi.com
1 Table of Contents

1.1 INTRODUCTION .......................................................................................................................... 1
1.2 SNFv5.3.2.7 COMPONENTS ....................................................................................................... 2
1.3 SNFv5.3.2.7 BENEFITS ............................................................................................................... 3
1.4 WHAT’S NEW IN SNFv5.3.2.7 SOFTWARE .............................................................................. 4
  1.4.1 Enhancements.......................................................................................................................... 4
1.5 SNFv5.3.2.7 AND THE ARC SERIES E NETWORK ADAPTER ............................................. 5
1.6 SUPPORT FOR OPEN-SOURCE SOFTWARE ............................................................................... 5
1.7 SNFv5.3.2.7 CHAPTER SUMMARIES ...................................................................................... 6

2 INSTALLING THE ARC SERIES E ADAPTER HARDWARE ...................................................... 8

2.1 CHECKING EXPANSION CARD SLOT CONFIGURATION ......................................................... 8
2.2 INSTALLING THE ARC SERIES E NETWORK ADAPTER ..................................................... 9

3 TESTING THE ARC SERIES E ADAPTER HARDWARE .......................................................... 11

3.1 TESTING THE ARC SERIES E NETWORK ADAPTER – DUAL-PORT .................................. 11
  3.1.1 Identifying network card port LEDs – dual-port................................................................. 11
  3.1.2 Testing the network card........................................................................................................ 12
  3.1.3 Running network card diagnostics: .................................................................................... 13
3.2 TESTING THE ARC SERIES E NETWORK ADAPTER – QUAD-PORT ................................ 14
  3.2.1 Identifying network card port LEDs – quad-port............................................................... 14
  3.2.2 Testing the network card........................................................................................................ 14
  3.2.3 Running network card diagnostics..................................................................................... 16
3.3 RUNNING A TEST SCRIPT .......................................................................................................... 17

4 INSTALLING SNFv5.3.2.7 SOFTWARE IN LINUX ................................................................. 18

4.1 DOWNLOADING SNFv5.3.2.7 RPM OR TGZ DRIVERS ......................................................... 18
4.2 INSTALLING THE SNFv5.3.2.7 RPM SOFTWARE PACKAGE FOR LINUX ....................... 19
  4.2.1 Uninstalling the SNFv5.3.2.7 Binary RPM driver.............................................................. 23
  4.2.2 Listing module load-time variables.................................................................................... 23
  4.2.3 Controlling kernel module behavior (optional)................................................................... 23
4.3 INSTALLING THE SNFv5.3.2.7 TGZ DRIVER FOR LINUX ................................................... 24
  4.3.1 Uninstalling the SNFv5.3.2.7 Tarball TGZ driver............................................................ 28
  4.3.2 Listing module load-time variables.................................................................................... 28
4.4 REBUILDING AFTER OS/KERNEL UPDATE ............................................................................. 29
  4.4.1 Updating the kernel............................................................................................................... 29
  4.4.2 Rebuilding after kernel update............................................................................................ 29

5 INSTALLING SNFv5.3.2.7 SOFTWARE IN VMWARE CLIENT ............................................. 30
5.1 TERMINOLOGY
5.2 PREREQUISITES
5.3 INSTALLING THE ARC SERIES E ADAPTER TO THE VMWARE SERVER
5.4 PREPARING THE ADAPTER HARDWARE FOR PASS-THROUGH MODE
5.5 ADDING PASS-THROUGH HARDWARE TO THE VIRTUAL MACHINE
5.6 INSTALLING THE SNFv5.3.2.7 SOFTWARE TO THE VMWARE SERVER
5.7 FLASHING ARC SERIES E ADAPTER HARDWARE

6 VERIFYING SNFV5.3.2.7 SOFTWARE INSTALLATION
6.1 BASIC TEST
  6.1.1 Before you start
  6.1.2 Running the test

7 PHX-TOOLS NETWORK ADAPTER TOOLKIT
7.1 PHX-TOOLS DESCRIPTION
7.2 FEATURES AND ENHANCEMENTS
7.3 NETWORK ADAPTER DIAGNOSTIC INFORMATION (MYRI_INFO)
7.4 UPGRADING FPGA FIRMWARE IN LINUX
  7.4.1 Upgrading FPGA firmware (phx-replace-eeprom)
7.5 TRACKING ENVIRONMENTAL DIAGNOSTICS ON ARC SERIES E NETWORK ADAPTERS (PHX-SCAN)
7.6 TRACKING SFP+ DIAGNOSTICS ON ARC SERIES E NETWORK ADAPTERS (MYRI_PHX_MDIO)

8 TESTING SNFV5.3.2.7 SOFTWARE
8.1 SUMMARY OF TEST PROGRAM COMMANDS
8.2 SNFv5.3.2.7 TEST PROGRAM REQUIREMENTS
  8.2.1 Adapter port nomenclature
8.3 SAMPLE TEST PROGRAMS
  8.3.1 snf_basic_diags
  8.3.2 snf_simple_recv
  8.3.3 snf_multi_recv
  8.3.4 snf_bridge
  8.3.5 snf_pktgen
  8.3.6 snf_replay

9 RUNNING SNFV5.3.2.7 DIAGNOSTIC TOOL PROGRAMS
9.1 SUMMARY OF DIAGNOSTIC TOOL PROGRAMS
  9.1.1 sbin/phx_bug_report (Linux only)
  9.1.2 sbin/myri_info
  9.1.3 bin/myri_counters
10 CONFIGURING SNFV5.3.2.7 .................................................................59

10.1 DEBUG VARIABLE (SNF_DEBUG_MASK) ...........................................60

10.2 RING MANAGEMENT VARIABLES ....................................................61

10.2.1 SNF_RING_ID ............................................................................61

10.2.2 SNF_NUM_RINGS .......................................................................62

10.2.3 SNF_DATAURING_SIZE ...............................................................63

10.2.4 SNF_DESCRING_SIZE .................................................................63

10.3 RSS HASHING/LOAD BALANCING VARIABLES (SNF_RSS_FLAGS) ..........64

10.3.1 SNF_RSS_FLAGS default settings ...............................................64

10.4 PORT AGGREGATION (MERGING) VARIABLES (SNF_FLAGS) ...............67

10.4.1 SNF_F_PSHARED 0x1 ...............................................................67

10.4.2 SNF_F_AGGREGATE_PORTMASK 0x2 .........................................67

10.4.3 SNF_F_RX_DUPLICATE 0x030 ....................................................68

10.5 APPLICATION ID VARIABLE (SNF_APP_ID) ......................................69

10.5.1 Running SNF_APP_ID with third-party tools .................................69

10.6 KERNEL ETHERNET INTERFACES ..........................................................72

11 LOAD BALANCING AND PORT MERGE FEATURES ................................73

11.1 SNF_NUM_RINGS .............................................................................73

11.2 SNF_FLAGS ....................................................................................74

11.2.1 Running multiple instances .......................................................74

11.2.2 Sharing/Load balancing .............................................................74

11.2.3 Port Listening on libpcap .............................................................75

11.2.4 Hardware assisted port merging ................................................76

12 LIBPCAP, AND PF_RING PACKET CAPTURE ......................................78

12.1 LIBPCAP AND SNFv5.3.2.7 ...........................................................78

12.1.1 Preamble ..................................................................................78

12.1.2 Verifying the libpcap link to SNFv5.3.2.7 ....................................78

12.2 PF_RING AND SNFv5.3.2.7 ............................................................79

12.2.1 Installing PF_RING from Linux RPM .........................................79

12.2.2 Installing PF_RING from source (ntop) ......................................81

12.2.3 Configuring the PF_RING library with SNFv5.3.2.7 .................81

12.3 DEMONSTRATING MULTI-PROCESS PF_RING FUNCTIONALITY ........83

12.3.1 Multi-process traffic duplication example ....................................83

12.3.2 Multi-process traffic sharing (RSS) example ...............................84
16  **TUNING SNFV5.3.2.7 SOFTWARE** ................................................................. 110

16.1  **RING PERFORMANCE** ................................................................. 110

16.2  **SNFv5.3.2.7 TUNING CHECK LIST** ............................................... 112

   16.2.1  Setting the system performance profile ........................................... 112
   16.2.2  PCIe expansion slot seating ......................................................... 112
   16.2.3  Interrupts and IRQ Affinity (Linux) .............................................. 113
   16.2.4  Iqbalance ................................................................................. 114
   16.2.5  Interrupt Balancing ..................................................................... 116
   16.2.6  CPU Frequency Scaling ............................................................... 116
   16.2.7  CPU Binding ............................................................................. 116
   16.2.8  PCI Bridging ............................................................................. 116
   16.2.9  Hyperthreading ......................................................................... 116
   16.2.10 NUMA Awareness ..................................................................... 116
   16.2.11 L3 Cache Awareness .................................................................. 117
   16.2.12 CPU isolation ........................................................................... 118
   16.2.13 Process Priority ........................................................................ 118
   16.2.14 Blocking Mode ......................................................................... 118

17  **TROUBLESHOOTING** ............................................................................. 120

17.1  **HARDWARE INSTALLATION AND PERFORMANCE ISSUES** .................. 120

   17.1.1  Sample lspci -vvv output ............................................................... 121
   17.1.2  Sample myri_info output .............................................................. 121
   17.1.3  LED Issues ................................................................................. 123

17.2  **SOFTWARE INSTALLATION & SYSTEM CONFIGURATION ISSUES** .......... 124

   17.2.1  Bug report scripts ........................................................................ 124
   17.2.2  Linux RPM-TGZ installation failures ............................................ 124
   17.2.3  Software Counters (myri_counters) .............................................. 125
   17.2.4  Numbering of snfX interfaces ...................................................... 126
   17.2.5  Performance Tuning and Packet Drops ....................................... 126
   17.2.6  Network Adapter Timestamps ....................................................... 127
   17.2.7  Synchronization ......................................................................... 128

**APPENDIX 1. SNFV5.3.2.7 COUNTERS** ................................................................. 1

**LIST OF SNFv5.3.2.7 COUNTERS** .................................................................. III

**APPENDIX 2. OPERATING SYSTEMS AND HARDWARE SUPPORT** ................. XIII
1.1 Introduction

The ARIA™ Cybersecurity Solutions-Myricom® Sniffer version 5.3.2.7 software, or SNFv5.3.2.7, powers the ARC Series E-class of network adapters to deliver pure packet capture, with the flexibility to configure advanced functions, leaving the vast majority of server cycles available for your application requirements.

SNFv5.3.2.7 is a tightly-integrated combination of FPGA firmware and user-level software libraries to enable sustained capture with merging of 1- and 10-Gigabit Ethernet traffic.

This chapter describes the following topics:

- SNFv5.3.2.7 Components
- SNFv5.3.2.7 Benefits
- What’s New in SNFv5.3.2.7 Software
- SNFv5.3.2.7 and the ARC Series E Network Adapter
- Support for Open-Source Software
- SNFv5.3.2.7 Chapter Summaries
1.2 SNFv5.3.2.7 Components

The SNFv5.3.2.7 driver contains two components:

1. **SNFv5.3.2.7 or “Sniffer”** refers to driver features and privileged operations that are necessary to allow an application to directly receive packets into user space. The driver includes optional interrupt generation as well (Figure 1.).

   **NOTE:** The SNFv5.3.2.7 components are only active when the device is opened for capture through SNF API, or indirectly from libpcap/SNF.

2. **Ethernet compatibility** refers to the driver-level functionality required to run an ARC Series E network adapter. At a minimum, it allows familiar OS-provided tools, such as `ethtool`, to interact with the capture device to receive packets on port 0 until it is enabled for SNFv5.3.2.7 capture (Figure 1).

   The enabled SNFv5.3.2.7 capture device receives all incoming packets – with no duplication possible – to the Ethernet driver until the capture device is closed. In other words, tools and utilities that tap into the OS stack to extract information will not see any traffic.

   To that end, SNFv5.3.2.7 includes its own set of tools and counters to analyze packets as they are being captured. In special cases, the driver's send functionality is available while the underlying SNF capture device is enabled, permitting the use of RAW sockets for sending packets.

**Libpcap**

SNFv5.3.2.7 packet capture capabilities can be leveraged through the `libpcap` library or through the SNFv5.3.2.7 API, as a set of C programming language functions. With a SNFv5.3.2.7-aware `libpcap`, users can reference an ARC Series E network adapter through its Ethernet interface name and run `libpcap`-dependent applications via the `libpcap` portable interface.
At extremely high packet rates, a single system call on every packet could cause packet drops. SNFv5.3.2.7 significantly reduces packet drops by bypassing the kernel altogether, thus offering unrestricted network traffic access to user space applications. With direct (SNFv5.3.2.7 API) and indirect (libpcap/SNFv5.3.2.7) access, applications gain full user space access to all incoming packets without any OS intervention—an important consideration when comparing SNFv5.3.2.7 to other packet capture solutions.

For more information on libpcap, go to the libpcap and SNFv5.3.2.7 section of Libpcap, and PF_RING Packet Capture

### 1.3 SNFv5.3.2.7 Benefits

By using our shared library, libpcap users can leverage SNFv5.3.2.7, gaining benefits for packet capture applications. To simplify implementations, the SNFv5.3.2.7-capable libpcap library is included in the SNFv5.3.2.7 software distribution. Application benefits include:

- **Reduced CPU usage** – SNFv5.3.2.7 seamlessly sends all packets to the application, completely bypassing the OS kernel and freeing up CPU cycles.
- **Lossless packet capture** – With SNFv5.3.2.7 user-definable ring sizes, applications can target a memory queue of any size. This flexibility eliminates packet drops typically caused by rate-matching challenges between hardware and software.
- **Applications running in parallel** – You can run multiple copies of libpcap against a single packet capture stream. This translates into multiple applications running simultaneously against the same packets, with zero copying in the background.
- **Exact time stamps** – ARC Series E network adapters running SNFv5.3.2.7 provide timestamps with up to ±3 nanosecond accuracy, using high-quality time signals attached via standard coax inputs.
- **Low overhead** – SNFv5.3.2.7 efficiency allows you to capture four, 1- and 10-Gigabit Ethernet ports into a single server with zero drops, at maximum packet rate, and still have enough CPU cycles available to run significant applications against these streams.
- **SNFv5.3.2.7 software supports Linux operating systems.**

The SNFv5.3.2.7-capable libpcap library is an open-source enhancement developed by the ARIA Cybersecurity Solutions Product Development Team and is part of the standard source trees. Both source and compiled versions of these software modules are provided, all interfacing seamlessly with SNFv5.3.2.7.

- **Large FPGA supports a stream of future enhancements from the ARIA Cybersecurity Solutions Product Development Team.**
1.4 What’s New in SNFv5.3.2.7 Software

- SNF version 5.3.2.7 is the GA release of the Linux SNFv5.3.2.7 software.
- SNFv5.3.2.7 software supports 1- and 10-Gigabit Ethernet network adapter environments.
- Supports ARC Series E network adapter dual-port and four-port models in SFP or SFP+ module configurations.
- Exact time stamps: ARC Series E network adapters with SNFv5.3.2.7 drivers provide timestamps with up to ±3 nanosecond accuracy, using high-quality time signals supplied via standard coax inputs.

1.4.1 Enhancements

- The driver now recognizes firmware up to version 2.1.5.
- Added hardware RSS hash. Removed the -a parameter from the test programs `snf_simple_recv` and `snf_multi_recv`.
- Added the -b parameter to `snf_simple_recv` to verify injection pacing.

```
NOTE: The -b parameter applies to every packet. Use only with controlled input streams.
```

- Added transmit hardware injection pacing. Added new API function calls `snf_inject_sched()` and `snf_inject_sched_v()`.
- Added a new API call, `snf_get_injection_speed()`. The `snf_pktgen` program now uses this function to determine link speed.
- Added support for software transmit timestamps.
- Added acceleration support for port pair merging. Acceleration is adapter-specific and offers a significant improvement over the ARC 8C- and ARC 8B-type adapters, which all do SW port merging only. The new acceleration mode can port merge two ports of sustained 60B traffic with no packet drops and only 50 percent CPU usage. Port merge acceleration is restricted to the following two-port pair configurations: 0 and 1, or 2 and 3. Both port pairs can run concurrently. Other port merging combinations (including across multiple adapters) is supported but with reduced performance.
1.5 SNFv5.3.2.7 and the ARC Series E Network Adapter

SNFv5.3.2.7 software introduces SNF support for the latest generation of Myricom ARC Series E network adapters. SNFv5.3.2.7 software is only compatible with the following hardware platforms:

- ARC Series E network adapter – dual-port (10G-PCIE3-8E-2S)
- ARC Series E network adapter – dual-port (1G-PCIE3-8E-2S)
- ARC Series E network adapter – quad-port (10G-PCIE3-8E-4S)
- ARC Series E network adapter – quad-port (1G-PCIE3-8E-4S)
- ARC Series E network adapter – quad-port (1G/10G-PCIE3-8E-4S)

DISCLAIMER: If you have any other adapter you wish to use, contact ARIA Support.

1.6 Support for Open-Source Software

SNFv5.3.2.7 is compatible with industry-standard open-source packet capture application tools. Examples of tested applications include:

- tcpdump (standard Linux utility)
- tcpreplay (standard Linux utility)
- Suricata network intrusion detection and security monitoring
- Wireshark network protocol analyzer
- Bro network Intrusion Detection System (IDS)
- Snort intrusion prevention systems
- PF_RING packet capture network socket, ported to run above the SNF API

SNFv5.3.2.7 can accommodate the multi-process Bro, multi-threaded Suricata, and tcpdump packet capture tools running in parallel - each tool collecting and splitting network traffic among their respective application threads.
### 1.7 SNFv5.3.2.7 Chapter Summaries

The SNFv5.3.2.7 User Guide contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>Introduction to SNFv5.3.2.7 software</em></td>
</tr>
<tr>
<td>2</td>
<td><em>Installing the ARC Series E Adapter Hardware.</em> Provides detailed adapter hardware installation instructions for ARC Series E network adapters.</td>
</tr>
<tr>
<td>3</td>
<td><em>Testing the ARC Series E Network Adapter Hardware</em> Describes testing procedures for dual- and quad-port ARC Series E network adapters.</td>
</tr>
<tr>
<td>4</td>
<td><em>Installing SNFv5.3.2.7 Software in Linux</em> Provides detailed instructions for downloading and installing SNFv5.3.2.7 RPM and TGZ drivers for Linux OS.</td>
</tr>
<tr>
<td>5</td>
<td><em>Installing SNFv5.3.2.7 Software with VMware Client</em> This chapter describes the steps necessary to install SNFv5.3.2.7 software and ARC Series E network adapters in VMware ESXi Virtual Machines (VMs).</td>
</tr>
<tr>
<td>6</td>
<td><em>Verifying SNFv5.3.2.7 Software Installation</em> Verifies proper SNFv5.3.2.7 software installation, network adapter hardware connectivity, and packet rate.</td>
</tr>
<tr>
<td>7</td>
<td><em>PHX-TOOLS Network Adapter Toolkit</em> The PHX-TOOLS network adapter toolkit allows users to run diagnostics on ARC Series E network adapter operation and flash memory FPGA firmware programming.</td>
</tr>
<tr>
<td>8</td>
<td><em>Testing SNFv5.3.2.7 Software</em> Describes SNFv5.3.2.7 software test programs that serve to familiarize the user with its basic operation.</td>
</tr>
<tr>
<td>9</td>
<td><em>Running SNFv5.3.2.7 Diagnostic Tool Programs</em> Describes essential SNFv5.3.2.7 diagnostic tools for error reporting purposes.</td>
</tr>
<tr>
<td>10</td>
<td><em>Configuring SNFv5.3.2.7</em> Provides instructions on configuring and debugging SNFv5.3.2.7 with an assortment of environment variables. Includes load balancing and port merge features.</td>
</tr>
</tbody>
</table>
| Chapter 11 | **Load Balancing and Port Merging Features**  
Describes variables responsible for port merging and load balancing across multiple application processes. |
| Chapter 12 | **Libpcap and PF_RING Packet Capture Applications**  
Describes how SNFv5.3.2.7 improves packet capture performance with **libpcap** and **PF_RING**. |
| Chapter 13 | **Open-Source Packet Capture Tools**  
Describes SNFv5.3.2.7 compatibilities with industry-standard open-source packet capture application tools. |
| Chapter 14 | **Linux PTP Host Clock Synchronization**  
Describes system clock synchronization to the ARC Series E Adapter Clock. |
| Chapter 15 | **SNFv5.3.2.7 Timestamping Support**  
Describes SNFv5.3.2.7 timestamping variables. Offers ways to view time source status.  
**SNFv5.3.2.7** does not support Arista timestamping in this release. |
| Chapter 16 | **Tuning SNFv5.3.2.7 Software**  
Identifies and resolves certain issues that may affect packet rate. |
| Chapter 17 | **Troubleshooting**  
Addresses issues pertaining to hardware and software installation, system configuration, and performance. |
| Appendix 1 | **SNFv5.3.2.7 Counters** |
| Appendix 2 | **Operating Systems & Hardware Support** |
| Appendix 3 | **SNFv5.3.2.7 Driver Restrictions and Limitations** |
| Appendix 4 | **SNFv5.3.2.7 Firmware** |
| Appendix 5 | **SNFv5.3.2.7 Supported 1G and 10G Transceivers** |
| Appendix 6: | **Network Adapter Toolkit – v. 1.40** |
2 Installing the ARC Series E Adapter Hardware

This chapter describes the following tasks for installing the ARC Series E-Class network adapter hardware (10G-PCIE3-8E-2S and 10G-PCIE3-8E-4S):

- Checking Expansion Card Slot Configuration
- Installing the ARC Series E Network Adapter

Myricom network hardware products are designed to be compatible with prevailing industry standards and typically install quickly without significant user effort. Nonetheless, it is beyond the scope of this manual to address all hardware installation issues specific to your networking site. For all issues regarding installing and configuring your hardware, contact:

- ARIA Technical Support through the ARIA Customer Portal to https://www.ariacybersecurity.com/support/downloads/ or
- Email ARIA Technical Support at ARIA_support@ariacybersecurity.com

2.1 Checking Expansion Card Slot Configuration

The ARC Series E network adapter has been qualified with PCIe server expansion slots with a minimum of x8-lanes. It is recommended that a PCIe Gen3 x8 expansion slot be located closest to the CPU to achieve best performance. See Table 1. to determine which expansion card slots can accommodate the network adapter hardware.

<table>
<thead>
<tr>
<th>Expansion card slot</th>
<th>Compatibility</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen3 x16 PCIe slot</td>
<td>Supports x8 card</td>
<td>Check motherboard specifications.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check mechanical fit to guarantee a secure electrical connection.</td>
</tr>
<tr>
<td>Gen3 x8 PCIe slot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gen1 PCIe slot</td>
<td>Not supported.</td>
<td></td>
</tr>
<tr>
<td>Gen2 PCIe slot</td>
<td>Not supported.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. PCIe expansion card slot characteristics
2.2 Installing the ARC Series E Network Adapter

Install the ARC Series E network adapter as follows:

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 case and locate the PCIe expansion card slots on the motherboard. Do not disturb the legacy PCI card slots, which are different in size and electrical specifications.

**NOTE:** For optimal performance, install the ARC Series E network adapter in a PCIe Gen3 x8 slot on the server. Minimal testing on PCIe Gen2 has been performed and is not recommended. Servers with PCIe Gen1 slots are not supported.

4. Check for a free slot, then remove the mounting screw from the protective bracket plate covering the selected slot, and set aside the plate. If there are no free slots then it may be necessary to remove a surplus adapter to make room for the network adapter.

5. Carefully remove the ARC Series E network adapter from its sealed protective sleeve without touching the gold PCIe connectors (Figures 2 and 3).

![ARC Series E Network Adapter](image)

Figure 2. ARC Series E dual-port network adapter (10G-PCIE3-8E-2S)
6. Line up the gold PCIe connectors and indexing tab with the empty PCIe Gen3 slot, ensuring that the ports and mounting bracket are facing the back panel of the computer.

**NOTE:** Place the ARC Series E adapter as close as possible to the CPU to achieve best performance.

7. Seat the network adapter firmly into the PCIe expansion slot until the card “clicks” into place.

8. Secure the network adapter to the computer chassis with a screw.

9. Close the computer case and re-connect the power plug.

10. Insert cabled transceivers in the network adapter ports. Do not kink the cables.

The ARC Series E network adapter is now installed.

**ATTENTION:** When swapping 1G transceivers, insert the transceiver first and reload the SNFv5.3.2.7 driver, running `myri_start_stop` to detect the link.

For more information on SNFv5.3.2.7 supported 1G and 10G transceivers, refer to *Appendix 5: SNFV5.3.2.7 Supported 1G and 10G Transceivers*
3 Testing the ARC Series E Adapter Hardware

This chapter describes the following topics:

- Testing the ARC Series E-Class Network Adapter – Dual-Port
- Testing the ARC Series E-Class Network Adapter – Quad-Port
- Running a Test Script

For more information on test programs, go to the Sample Test Programs section of Testing SNFv5.3.2.7 Software

3.1 Testing the ARC Series E Network Adapter – Dual-Port

Once you have installed the dual-port ARC Series E adapter (10G-PCIE3-8E-2S) into the PCIe expansion slot, we recommend testing the card and acquainting yourself with the following:

- Identifying network card port LEDs – dual-port
- Testing the network card
- Running network card diagnostics

3.1.1 Identifying network card port LEDs – dual-port

The network adapter (10G-PCIE3-8E-2S) has two ports: port 0 and port 1. Each port has corresponding LEDs, identified as LED 1, 2, 3, and 4.

- LED 1 represents the first port transmit. It is referenced in SNFv5.3.2.7 software and counters as ‘port 0’.
- LED 2 represents the first port receive. It is referenced in SNFv5.3.2.7 software and counters as ‘port 0’.
- LED 3 represents the second port transmit. It is referenced in SNFv5.3.2.7 software and counters as ‘port 1’.
- LED 4 represents the second port receive. It is referenced in SNFv5.3.2.7 software and counters as ‘port 1’.
3.1.2 Testing the network card

Preparing the network card for testing

1. Prepare the card for testing by connecting port 0 to port 1 in loopback.
2. Insert SFP+ modules into port 0 and port1 and insert a cable between the modules.
   The corresponding port LEDs turn green.

Running the test

Testing requires two terminal windows. One window receives data on port 0 and the other sends data on port 1.

1. Run **snf_simple_recv** in the first window (port 0):
   
   ```
   $ /opt/snf/bin/tests/snf_simple_recv -p 0 -v
   ```

2. Run **snf_pktgen** in the second window (port 1):

   ```
   $ /opt/snf/bin/tests/snf_pktgen -p 1:1 -v -r 1.0
   ```

   The `-p` option specifies the port assigned to send the data.

   The output for the **snf_simple_recv** command is as follows:

   ```
   $ /opt/snf/bin/tests/snf_simple_recv -p 0 -v
   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: 14605824577425118579 ts_host: 1460582457742627269 ts_diff:313950
   pkt: 60000, len: 60, ts_hw: 146058245774747118579 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
   ```

   The **snf_simple_recv** and **snf_pktgen** commands will run continuously until you exit the programs.
3. Enter `CTRL-C` to exit the programs:

The `snf_pktgen` command generates no output.

The `snf_simple_recv` command generates the following output:

```
Output:

pkt: 90000, len: 60, ts_hw: 1460582457761532689 ts_host: 1460582457761832804
ts_diff:300115
Packets received in HW: 10348661
Packets reinjected, app: 0
Packets reflected to netdev: 0
Total bytes received, app: 620919660 (592 MB)
Total bytes received + HW aligned: 496735728 (473 MB)
Average Packet Length: 60 bytes
Dropped, NIC overflow: 0
Dropped, ring overflow: 0
Dropped, bad: 0
```

The corresponding port LEDs turn orange while the tests are running.

### 3.1.3 Running network card diagnostics:

The `myri_nic_info` command provides adapter information such as the hardware serial number, MAC address, firmware version, and so on.

**Command line:**

```
$ sudo /opt/snf/bin/myri_nic_info
```

**Output:**

```
# Serial  MAC               ProductCode     Driver   Version     License
0 491942  00:60:dd:48:b3 10G-PCIE3-8E-2S myri_snf-5.3.2.7.54367 Valid
1 491943  00:60:dd:43:48:b4 10G-PCIE3-8E-2S myri_snf-5.3.2.7.54367 Valid
```
3.2 Testing the ARC Series E Network Adapter – Quad-Port

Once you have installed the quad-port ARC Series E adapter (10G-PCIE3-8E-4S) into the PCIe expansion slot, we recommend testing the card and acquainting yourself with the following:

- Identifying network card port LEDs – quad-port
- Testing the network card
- Running network card diagnostics

3.2.1 Identifying network card port LEDs – quad-port

The network adapter (10G-PCIE3-8E-4S) has four ports: ports 0, 1, 2, and 3. Each port has corresponding LEDs, identified (from left to right) as LED 0, 1, 2, and 3.

- LED 0 represents the first port. It is referenced in SNFv5.3.2.7 software and counters as ‘port 0’.
- LED 1 represents the second port. It is referenced in SNFv5.3.2.7 software and counters as ‘port 1’.
- LED 2 represents the third port. It is referenced in SNFv5.3.2.7 software and counters as ‘port 2’.
- LED 3 represents the fourth port. It is referenced in SNFv5.3.2.7 software and counters as ‘port 3’.

3.2.2 Testing the network card

Preparing the network card for testing

1. Prepare the card for testing by connecting port 0 to port 1, and port 2 to port 3 in loopback.
2. Insert cables into the loopbacks.

   The corresponding port LEDs turn green.

Running the test

Testing requires four terminal windows. Two windows receive data on ports 0 and 2, and two windows send data on ports 1 and 3.

1. Run `snf_simple_recv` in the first window (port 0):

   ```
   $ sudo /opt/snf/bin/tests/snf_simple_recv -p 0 -v
   ```

2. Run `snf_simple_recv` in the second window (port 2):

   ```
   $ sudo /opt/snf/bin/tests/snf_simple_recv -p 2 -v
   ```
3. Run **snf_pktgen** in the third window (port 1):

   `$ sudo /opt/snf/bin/tests/snf_pktgen -p 1:1 -r 1.0`

   The `-p` option specifies the port assigned to send the data.
   In this example the command is sending the data out of port 1.

4. Run **snf_pktgen** in the fourth window (port 3):

   `$ sudo /opt/snf/bin/tests/snf_pktgen -p 3:1 -r 1.0`

   The output for the **snf_simple_recv** command is as follows:

   ```
   Output:
   
   $> /opt/snf/bin/tests/snf_simple_recv -p 0 -v
   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
   ```

5. Enter **CTRL-C** to exit the programs:

   The **snf_pktgen** command generates no output.
The `snf_simple_recv` command generates the following output:

**Output:**

```
pkt: 90000, len: 60, ts_hw: 1460582457761532689 ts_host: 1460582457761832804
ts_diff:300115
Packets received in HW: 10348661
Packets reinjected, app: 0
Packets reflected to netdev: 0
Total bytes received, app: 620919660 (592 MB)
Total bytes received + HW aligned: 496735728 (473 MB)
Average Packet Length: 60 bytes
Dropped, NIC overflow: 0
Dropped, ring overflow: 0
Dropped, bad: 0
```

The corresponding port LEDs turn orange while the tests are running.

### 3.2.3 Running network card diagnostics

The `myri_nic_info` command provides adapter information such as the hardware serial number, MAC address, firmware version, and so on.

**Command line:**

```
$ sudo /opt/snf/bin/myri_nic_info -B
```

**Output:**

```
$ sudo /opt/snf/bin/myri_nic_info -B
# Serial MAC ProductCode Driver Version License
0 491942 00:60:dd:43:48:b3 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
1 491943 00:60:dd:43:48:b4 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
2 491944 00:60:dd:43:48:b5 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
3 491945 00:60:dd:43:48:b6 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
```
3.3 Running a Test Script

You can run a dual-port or quad-port ARC Series E adapter test from a script. The script launches four terminal windows by default, and initiates send and receive in each.

The quad-port adapter test script launches all four terminal windows and runs the test. The dual-port adapter test script closes two terminal windows (with an error message indicating that the ports do not exist) and runs the test.

Test script:

Command line:

$ cat test_script

Output:

```bash
#!/bin/bash
/usr/bin/xterm -e "`cat tests/snf_simple_recv -p 0 -v` " &
/usr/bin/xterm -e "`cat tests/snf_simple_recv -p 2 -v` " &
/usr/bin/xterm -e "`cat tests/snf_pktgen -p1:1 -v -r1.0` " &
/usr/bin/xterm -e "`cat tests/snf_pktgen -p3:1 -v -r1.0` " &
```

To run diagnostics on ARC Series E network adapter operations and FPGA firmware programming, go to PHX-TOOLS Network Adapter Toolkit.
4 Installing SNFv5.3.2.7 Software in Linux

This chapter describes the following topics:

- Downloading SNFv5.3.2.7 RPM or TGZ Drivers
- Installing the SNFv5.3.2.7 RPM Software Package for Linux
- Installing the SNFv5.3.2.7 TGZ Driver for Linux
- Rebuilding after OS/Kernel Update

Linux file formats

The downloaded file format may be either .rpm or .tgz for Linux. A different installation process is required for each format and is described in detail in this chapter.

Software package contents

SNFv5.3.2.7 software is a single package file containing:

- A special network adapter driver to replace the normal Ethernet driver
- A dynamic library of software modules
- Test programs to demonstrate SNFv5.3.2.7 functionality

4.1 Downloading SNFv5.3.2.7 RPM or TGZ Drivers

To download a copy of the SNFv5.3.2.7 RPM or SNFv5.3.2.7 TGZ driver, either download the file from https://www.ariacybersecurity.com/support/downloads/ or contact ARIA support (ARIA_support@ariacybersecurity.com). Then save the file to your designated system directory.
4.2 Installing the SNFv5.3.2.7 RPM Software Package for Linux

**NOTE:** The RPM package can be used with Fedora based distributions including RHEL and CentOS.

**Example:**
To install the SNFv5.3.2.7 software package to your Linux operating system, follow these steps:

1. Enter the following command to uninstall any previous versions of Myricom software:
   
   ```
   $ sudo yum remove myri_snf
   ```
   
   All previous versions of Myricom software are deleted.

2. Verify that the operating system detects the presence of the network adapter.
   
   ```
   $ sudo lspci -d 1c09:
   ```
   
   The output displays a list of network adapter device versions.
   
   **Output:**
   
   ```
   01:00.0 Ethernet controller: CSP, Inc. Device 4260 (rev 01)
   ```

3. Install the SNFv5.3.2.7 driver.
   
   **For CentOS 7.7 & 8.0 & RHEL 7 Distributions:**
   
   ```
   $ sudo yum -y install ./myri_snf-<version_info>*.x86_64.rpm
   ```
   
   **Output:**
   
   ```
   kernel      = 3.10.0-327.36.1.el7.x86_64
   destination = /opt/snf/sbin
   *** myri_snf.ko ... ok
   Created symlink from
   /etc/systemd/system/default.target.wants/myri_start_stop.service to
   /etc/systemd/system/myri_start_stop.service.
   ```
4. Enter the following command to confirm that the driver is loaded:

   $ lsmod |grep myri

   Console output while the driver is loading.

   Output:

   | myri_snf   | 177214 | 0 |

   OR

   Enter the following command if the driver does not load:

   $ sudo /opt/snf/sbin/myri_start_stop restart

5. Enter the following command to confirm that the SNF driver detects the network adapter and the Gen3 x8 or the Gen3 x16 expansion slot.

   $ sudo /opt/snf/sbin/myri_info
The output confirms that the SNF driver detects the network adapter.

Output:

```
pci-dev at 01:00.0 vendor:product(rev)=1c09:4260 (01)
    behind bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)
Myri-10G-PCIE-8E -- Link x8
    EEPROM String=spec:
        MAC=00:60:dd:43:2d:e8
        SN=495892
        PC=10G-PCIE3-8E-4S
        PN=09-04680
        BOM=A

Firmware:
    Version: 2.1.5
    Type: SNF
    Config: 4 Port x 10 Gb
    SHA1: 2d13f73ad9fe4bd4bda8d7b50dd0ad0b

External Inputs:
    PPS: Enabled, No Input
        Front Panel PPS: No Input
        Card Edge PPS: No Input
    10Mhz Clock: Disabled
    100Mhz Clock Locked: Locked
```

6. Enter the following command to track the interface names assigned to each port:

   ```$ sudo ip link show | grep -iB 1 00:60:dd```

Example:

In this example, the interface name is “enpls0f0” and the MAC network adapter addresses begin with “00:60:dd:”

Output:

```
30: enpls0f0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 00:60:dd:43:e8:b0 brd ff:ff:ff:ff:ff:ff
```
7. Enter the following commands to manually assign an IP address and a subnet to port 0 of the adapter. You can only assign an IP address to port 0. Ports 1, 2, and 3 are only available to SNF API. (You can also create `ifcfg` files in the `/etc/sysconfig/network-scripts` directory).

   $ sudo systemctl stop NetworkManager.service
   $ sudo systemctl disable NetworkManager.service
   $ sudo ip address add 10.0.0.1/24 dev enp1s0f0

8. Enter the following command to confirm that each link is functioning:

   $ sudo ip address show up

   The output confirms the link is functioning.

   **Output:**

   ```
   27: enp1s0f0:
   <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc pfifo_fast state UNKNOWN qlen 1000
     link/ether 00:60:dd:43:52:f0 brd ff:ff:ff:ff:ff:ff
     inet 10.0.0.1/24 brd 10.0.0.255 scope global p1p1
       valid_lft forever preferred_lft forever
     inet6 fe80::260:ddff:fe43:52f0/64 scope link
       valid_lft forever preferred_lft forever
   ```

9. Enter the following commands to verify contact with the remote host:

   $ sudo ping 10.0.0.2

   The SNFv5.3.2.7 RPM software installation is now complete.

To test your SNF software, go to **Testing SNFv5.3.2.7 Software**
4.2.1 Uninstalling the SNFv5.3.2.7 Binary RPM driver

To uninstall the SNFv5.3.2.7 Binary RPM driver, enter the following commands:

```bash
$ sudo /opt/snf/sbin/myri_start_stop stop
$ sudo yum remove myri_snf
```

The SNFv5.3.2.7 RPM driver is removed from the system.

4.2.2 Listing module load-time variables

To list module load-time variables, enter the following command:

```bash
$ sudo modinfo /opt/snf/sbin/myri_snf.ko
```

4.2.3 Controlling kernel module behavior (optional)

To manually stop or start the kernel module, enter the following command with the `myri_start_stop` script:

```bash
$ sudo /opt/snf/sbin/myri_start_stop start
```
4.3 Installing the SNFv5.3.2.7 TGZ Driver for Linux

Non-Fedora-based Linux distributions with Tarball TGZ (.tgz) drivers, offering support up to Linux kernel version 5.5, are provided. The TGZ package can be used with Debian-based distributions including Ubuntu.

**NOTE:**
This procedure involves uninstalling any previous versions of Myricom software, which depends on the manner in which it was installed. For example, if a package manager was used to install Myricom software (e.g. rpm or yum), then software should be removed using that method.

Install the SNFv5.3.2.7 TGZ driver for Linux as follows:

1. To uninstall any previous versions of Myricom software, enter the following command:

   ```
   $ sudo rm -r /opt/snf
   ```

   All previous versions of Myricom software are deleted.

2. Verify that the operating system detects the presence of the network adapter.

   ```
   $ sudo lspci -d 1c09:
   ```

   The output displays a list of device versions.

   **Output:**
   
<table>
<thead>
<tr>
<th>Ethernet controller: CSP, Inc. Device 4260 (rev 01)</th>
</tr>
</thead>
</table>

3. Enter the following commands to install the SNFv5.3.2.7 TGZ driver:

   **Debian Distributions (Ubuntu):**

   ```
   $ cd /opt
   $ sudo tar xzvf ./myri_snf-<version_info>*.x86_64.tgz
   $ mv myri_snf-<version_info>*.x86_64 snf
   $ cd /opt/snf
   $ sudo sbin/rebuild.sh
   ```

   **Output:**
   
<table>
<thead>
<tr>
<th>kernel</th>
<th>3.19.0-25-generic</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>/opt/snf/sbin</td>
</tr>
<tr>
<td></td>
<td>myri_snf.ko ... ok</td>
</tr>
</tbody>
</table>
For Ubuntu servers that use systemd (15.04+) you can also perform the following steps to start the driver automatically.

$ sudo cp /opt/snf/sbin/myri_start_stop.service /etc/systemd/system

$ sudo systemctl enable myri_start_stop.service

$ sudo systemctl start myri_start_stop.service

4. Enter the following command to confirm that the driver is loaded:

$ sudo lsmod |grep myri

The output confirms the driver is loading.

Output:

| myri_snf  | 177214 0 |

OR

Enter the following command if the driver does not load:

$ sudo /opt/snf/sbin/myri_start_stop restart

The output confirms the driver is loaded.

5. Enter the following command to confirm that the SNFv5.3.2.7 driver detects the network adapter and the Gen3 x8 or the Gen3 x16 expansion slot:

$ sudo /opt/snf/sbin/myri_info
The output detects the network adapter and the expansion slot.

**Output:**

```plaintext
pci-dev at 01:00.0 vendor:product(rev)=1c09:4260(01)
   behind bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)
Myri-10G-PCIE-8E -- Link x8

EEPROM String-spec:
   MAC=00:60:dd:43:2d:e8
   SN=495892
   PC=10G-PCIE3-8E-4S
   PN=09-04680
   BOM=A

Firmware:
   Version: 2.1.5
   Type   : SNF
   Config : 4 Port x 10 Gb
   SHA1   : 2d13f73ad9fe4bd4bda8d7b50dd0ad0b

External Inputs:
   PPS: Enabled, No Input
      Front Panel PPS: No Input
      Card Edge PPS: No Input
   10Mhz Clock: Disabled
   100Mhz Clock Locked: Locked
```

6. Enter the following command to track the interface names assigned to each port:

   ```
   $ sudo ip link show | grep -iB 1 00:60:dd
   ```

**Example:**

In this example, the interface name is “enp1s0f0” and “enp1s0f1”.
The MAC network adapter addresses begin with “00:60:dd:”

**Output:**

```plaintext
30: enp1s0f0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
   link/ether 00:60:dd:43:e8:b0 brd ff:ff:ff:ff:ff:ff
31: enp1s0f1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
   link/ether 00:60:dd:43:e8:b1 brd ff:ff:ff:ff:ff:ff
```
7. Enter the following commands to manually assign an IP address and a subnet to each network adapter port. (You can also create ifcfg files in the /etc/sysconfig/network-scripts directory).

   $ sudo systemctl stop NetworkManager.service
   $ sudo systemctl disable NetworkManager.service
   $ sudo ip address add 10.0.0.1/24 dev enp1s0f0
   $ sudo ip link set dev enp1s0f0 up
   $ sudo ip address add 10.1.0.1/24 dev enp1s0f1
   $ sudo ip link set dev enp1s0f1 up

8. Enter the following command to confirm that each link is functioning:

   $ sudo ip address show up

   The output confirms each link is functioning.

   **Output:**

   ```
   27: enp1s0f0:
   <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc pfifo_fast state UNKNOWN qlen 1000
     link/ether 00:60:dd:43:52:f0 brd ff:ff:ff:ff:ff:ff
     inet 10.0.0.1/24 brd 10.0.0.255 scope global p1p1
        valid_lft forever preferred_lft forever
     inet6 fe80::260:ddff:fe43:52f0/64 scope link
        valid_lft forever preferred_lft forever
   ```

9. Enter the following commands to verify you can contact the remote host:

   $ sudo ping 10.0.0.2
   $ sudo ping 10.1.0.2

   To test your software, go to Testing SNFv5.3.2.7 Software
4.3.1 Uninstalling the SNFv5.3.2.7 Tarball TGZ driver

This section describes how to uninstall the SNFv5.3.2.7 Tarball TGZ driver.

To uninstall the SNFv5.3.2.7 Tarball TGZ driver, enter the following commands:

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

The SNFv5.3.2.7 TGZ driver is removed from the system.

4.3.2 Listing module load-time variables

To list module load-time variables, enter the following command:

$ sudo modinfo /opt/snf/sbin/myri_snf.ko
4.4 Rebuilding after OS/Kernel Update

Updating the kernel on CentOS, Fedora, or Ubuntu Linux is a straightforward process. You update the kernel first and then rebuild after the kernel update.

4.4.1 Updating the kernel

There are several ways to update the kernel on, Fedora, Ubuntu, or CentOS.

Option 1:

Command line (Fedora):
$ sudo dnf upgrade

Option 2:

Command line (Ubuntu):
$ sudo apt-get update

Option 3:

Command line (CentOS):
$ sudo yum update

4.4.2 Rebuilding after kernel update

Following the kernel update, you update, or rebuild, the drivers.

To rebuild after kernel update, enter the following commands:

$ cd /opt/snf
$ sudo sbin/rebuild.sh
5 Installing SNFv5.3.2.7 Software in VMware Client

This chapter describes the steps necessary to install SNFv5.3.2.7 software and ARC Series E-Class network adapters in VMware ESXi Virtual Machines (VMs).

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

The chapter includes the following topics:

- References and Terminology
- Prerequisites
- Installing the ARC Series E Adapter to the VMware Server
- Preparing the Adapter Hardware for Pass-through Mode
- Adding Pass-through Hardware to the vSphere Client
- Installing the SNFv5.3.2.7 Software to the VMware Server
- Flashing ARC Series E Adapter Hardware

5.1 Terminology

Terminology used throughout this chapter is as follows:

- **Server**: VMware ESXi server version 6.0.0 3029758
- **Client**: vSphere Client version 6.0.0 3562874
- **Virtual Machine (VM)**: CentOS 7.5 x86_64

5.2 Prerequisites

Note the following prerequisites before you proceed:

- To enable DirectPath I/O, verify that the server has Intel Virtualization Technology for Directed I/O (VT-d) or AMD I/O Virtualization Technology (IOV) enabled in the BIOS.
- Verify that the virtual machine is running CentOS software version 7.7 or later.
- Verify that the ARC Series E adapters are connected to the host and marked as available for pass-through.

| NOTE: | If the ESXi host is configured to boot from a USB device, disabling the USB controller for pass-through is recommended. VMware does not support USB controller pass-through for ESXi hosts that boot from USB devices or SD cards connected through USB channels. For more information, go to [https://kb.vmware.com/s/article/2068645](https://kb.vmware.com/s/article/2068645). |

SNFv5.3.2.7 User Guide
5.3 Installing the ARC Series E Adapter to the VMware Server

Install the network adapter hardware to the VMware server as follows:

**NOTE:** The ARC Series E-Class network adapter (10G-PCIE3 -8E-2S) described in this example is Device 4260 and Vendor 1C09, running in VM server location 0000:07:00:0

1. Turn off the server.
2. Insert an ARC Series E network adapter into one of the PCIe Gen3 x8 expansion slots on the server. For more information on installing the ARC Series E adapter, go to *Installing the ARC Series E Adapter Hardware*.
3. Turn on the server.
4. Configure the BIOS system virtualization settings to ensure that the VMs accept the network adapter in pass-through mode.
5. Save your settings and reboot.

**NOTE:** These options will vary from one BIOS system or motherboard type to another. Nonetheless the following options, relevant to virtualization, must be enabled:

- Intel I Virtualization, and
- Intel I VT-d Directed I/O
5.4 Preparing the Adapter Hardware for Pass-through Mode

To prepare the ARC Series E network adapter for pass-through mode, follow this procedure:

1. Log on to the vSphere client server and click the **Configuration** tab.

2. Select **Hardware > Advanced Settings**.

   The **DirectPath I/O Configuration** window appears, displaying all devices selected for pass-through mode. Note that the device list may be empty.

3. Click **Edit**.

   The **Mark devices for pass-through**: window appears.

   In this example, the ARC Series E adapter lists as “**0000:07:00.0 | Unknown <class> Ethernet controller**”.


4. Check the box corresponding to the “0000:07:00.0 | Unknown <class> Ethernet controller” device that will pass through to the Guest Virtual Machine.

The Device Details section lists the Device ID and Vendor ID attributes as “4260” and “1C09” respectively that correlate with the ARC Series E network adapter hardware.

5. Click OK to close the dialog window.

The ARC Series E network adapter is now ready for pass-through mode and appears in the DirectPath I/O Configuration window.
5.5 Adding Pass-through Hardware to the Virtual Machine

To add the ARC Series E-Class adapter pass-through hardware to the VM, follow this procedure:

1. Log on to the VMWare server from the vSphere Client.
2. Double-click Create a New Virtual Machine to generate the new virtual machine.
3. Select the Getting Started tab and click Edit virtual machine settings.
   The <VM Name> - Virtual Machine Properties window appears.

4. Click the Add... button.
   The Add Hardware window appears.
5. Click **PCI Device** to select the desired ARC Series E adapter hardware.

6. Click **Next**.

7. Select the pass-through ARC Series E adapter to connect to the virtual machine from the drop-down list.

8. Click **Next**.

9. Click **Finish**.

10. Click **Getting Started > Edit virtual machine settings** to confirm that the adapter is connected to the VM.

    The `<VM name> - Virtual Machine Properties` window appears, confirming ARC Series E adapter pass-through to the VM.
5.6 Installing the SNFv5.3.2.7 Software to the VMware Server

Once the ARC Series E network adapter is installed and recognized in VM, install the SNFv5.3.2.7 software package as described in the Installing the SNFv5.3.2.7 RPM Software Package for Linux section of Installing SNFv5.3.2.7 Software in Linux.
5.7 Flashing ARC Series E Adapter Hardware

| ATTENTION: | It is possible to flash the ARC Series E adapter once it is installed in the VM server. However, we do not recommend this procedure. Proceed with caution. |

To flash the ARC Series E adapter once it is installed in the VM server, follow the steps:

1. Power-cycle the VM server once flashing is completed. Rebooting the VM is not sufficient.
2. Delete the adapter from the list of devices available for pass-through mode on the VM server operating system.
3. Add the adapter back to the list of devices available for pass-through mode on the VM server operating system.
4. Remove the adapter from the PCIe expansion slot.
5. Reinstall the adapter in the PCIe expansion slot.

For more information on installing the adapter, refer to the *Installing the ARC Series E Network Adapter* section of *Installing the ARC Series E Adapter Hardware*.
6 Verifying SNFv5.3.2.7 Software Installation

SNFv5.3.2.7 software installation test programs are available from:

- `/opt/snf/bin/tests` of the install directory in binary form
- `/opt/snf/share/examples`

6.1 Basic Test

To verify that the SNFv5.3.2.7 software is properly installed, run `snf_simple_recv` to generate the packets for injection followed by `snf_pktgen` to receive the packets. This test is sufficient to verify proper software installation, network adapter hardware connectivity, and the expected packet rate for the SNFv5.3.2.7 software.

6.1.1 Before you start

This test assumes the following:

- The `/opt/snf/bin/tests` directory has been added to the current `$PATH` variable.
- An ARC Series E network adapter has been installed.
- SNFv5.3.2.7 software has been loaded on both a “server” host and a “client” host.
- The default adapter port on the server is physically connected (via a cable) to the default adapter port on the client. In other words, the server host and the client host are connected in a point-to-point (switchless) configuration.
- If both the server adapter and the client adapter have two physical ports, the default adapter port is port 0 (the port closest to the PCI connector and also the port with the lower MAC address), and it is assumed that port 0 of the server adapter is physically connected to port 0 of the client adapter.
- Run the `snf_simple_recv` process on the client first before running the packet generator `snf_pktgen` on the server.
- Exit the `snf_pktgen` packet generator on the server first before exiting the `snf_simple_recv` process on the client.

NOTE: Reversing the process causes the overflow counter to increase. With no SNFv5.3.2.7 program to consume packets through SNF API, the packets are delivered instead to the Ethernet portion of the driver. The driver attempts to drop them as fast as possible, but cannot sustain the packet rate.
6.1.2 Running the test

To test that the SNFv5.3.2.7 software is properly installed, enter the following:

1. On the server host system, open a command window.
2. Enter the following command to generate one billion 60-byte packets:
   
   $ sudo snf_pktgen -s 60 -n 100000000

3. On the client host system, open a command window.
4. Enter the following command:
   
   $ sudo snf_simple_recv -t

The following output appears.

Output:

```
14843291 pkts (890597460B) in 1.001 secs (14827884 pps), Avg Pkt: 60, BW (Gbps): 7.117
14843663 pkts (890619780B) in 1.001 secs (14828242 pps), Avg Pkt: 60, BW (Gbps): 7.118
14841101 pkts (890466060B) in 1.001 secs (14825697 pps), Avg Pkt: 60, BW (Gbps): 7.116
14845703 pkts (890742180B) in 1.001 secs (14830250 pps), Avg Pkt: 60, BW (Gbps): 7.119
14842322 pkts (890539320B) in 1.001 secs (14826946 pps), Avg Pkt: 60, BW (Gbps): 7.117
14842292 pkts (890537520B) in 1.001 secs (14826902 pps), Avg Pkt: 60, BW (Gbps): 7.117
14843007 pkts (890580420B) in 1.001 secs (14827601 pps), Avg Pkt: 60, BW (Gbps): 7.117
14842392 pkts (890543520B) in 1.001 secs (14826942 pps), Avg Pkt: 60, BW (Gbps): 7.117
```

The test output generates the packet rate per second (pps). The expected packet rate is 14.8 Mpps, on a 60-byte packet line rate.

To test your SNF software, go to Testing SNFv5.3.2.7 Software
7 PHX-TOOLS Network Adapter Toolkit

This chapter includes the following topics:

- PHX-TOOLS Description
- Features and Enhancements
- General Information on the ARC Series E Network Adapter
- Upgrading FPGA Firmware in Linux
- Tracking Environmental Diagnostics on ARC Series E Network Adapters
- Tracking SFP+ Diagnostics on ARC Series E Network Adapters

The Linux diagnostic tools package is listed as the Toolkit – Phoenix Group or PHX-TOOLS (phx-tools-1.40.tar.gz) in the ARIA Customer Portal. It is similar to the myri-tools.tar.gz packages in previous hardware generations (Myri-10G) except under different tool names. Various tool options and outputs have also changed to reflect ARC Series E network adapter requirements.

7.1 PHX-TOOLS Description

The PHX-TOOLS network adapter toolkit allows users to run diagnostics on ARC Series E network adapter operation and flash memory FPGA firmware programming.

The PHX-TOOLS toolkit contains the following command-line tools:

<table>
<thead>
<tr>
<th>MYRI_INFO</th>
<th>Displays ARC series E adapter diagnostic information</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHX-REPLACE-EEPROM</td>
<td>Programs the FPGA firmware into flash memory</td>
</tr>
<tr>
<td>PHX-SCAN</td>
<td>Tracks and records environmental diagnostics from ARC Series E network adapters</td>
</tr>
<tr>
<td>MYRI_PHX_MDIO</td>
<td>Retrieves SFP+ transceiver modules diagnostics from ARC Series E network adapters</td>
</tr>
</tbody>
</table>

The toolkit currently supports the following network card format:

- 10G-PCIE3-8E-2S with two SFP+ cages
- 10G-PCIE3-8E-4S with four SFP+ cages
7.2 Features and Enhancements

The PHX-TOOLS version 1.40 release describes the toolkit’s individual features, bug fixes, and limitations. For a full description, refer to Appendix 6: Network Adapter Toolkit - v. 1.40

We recommend that users migrate to this release at their earliest convenience.

7.3 Network Adapter Diagnostic Information (myri_info)

The myri_info command displays 10G-PCIE3-8E-2S and 10G-PCIE3-8E-4S diagnostic information such as product code, serial number, MAC address, FPGA flash image version, and so on.

Linux

Command line:
$ sudo ./myri_info

Output:

pci-dev at 01:00.0 vendor:product(rev)=1c09:4260(01)
behind bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)
Myri-10G-PCIE-8E -- Link x8
EEPROM String-spec:
  MAC=00:60:dd:43:2d:e8
  SN=495892
  PC=10G-PCIE3-8E-4S
  PN=09-04680
  BOM=A

Firmware:
  Version: 2.1.5
  Type   : SNF
  Config : 4 Port x 10 Gb
  SHA1   : 2d13f73ad9fe4bd4bda8d7b50dd0ad0b

External Inputs:
  PPS: Enabled, No Input
    Front Panel PPS: No Input
    Card Edge PPS: No Input
  10Mhz Clock: Disabled
  100Mhz Clock Locked: Locked
### 7.4 Upgrading FPGA Firmware in Linux

SNF-5.3.2.7 supports a variety of firmware downloads for 1Gbit and 10Gbit adapter speeds. Run `lspci` to view all available firmware modules from the PHX-TOOLS package by adapter speed and configuration.

Every network adapter format has a different firmware image, which is not interchangeable. For example, a firmware image for the dual-port 10G-PCIE3-8E-2S network adapter is `fw-8E-2S-SNF_10G-2.1.5.bin` and `fw-8E-4S-SNF_10G-2.1.5.bin` for the quad-port 10G-PCIE3-8E-4S network adapter.

For a comprehensive list of 1G and 10G firmware modules, go to Appendix 4: SNF-5.3.2.7 Firmware.

#### 7.4.1 Upgrading FPGA firmware (phx-replace-eeprom)

**NOTE:** Before you upgrade the FPGA firmware in Linux, verify that the SNFv5.3.2.7 driver is not loaded, otherwise the FPGA firmware will not program.

1. To prevent the SNFv5.3.2.7 software driver from loading, enter the following command:

   ```
   $ sudo /etc/init.d/myri_start_stop stop
   ```

2. Enter the following command to program the FPGA firmware into flash memory:

   ```
   $ sudo ./bin/phx-replace-eeprom ./fw-8E-4S-SNF_10G-2.1.5.bin
   ```

   A progress bar appears displaying the programming and verification pass status.

   **Output:**
   ```
   Preparing to reprogram firmware on unit 0 (129:0.0)
   Programming EEPROM with fpga image contained in ./fw-8E-4S-SNF_10G-2.1.5.bin
   16006908
   Please do not turn off power while flash is being programmed.
   Do you want to continue (enter yes)?yes
   Loading.... ####################################| 100%
   Verifying.. ####################################| 100%
   Flash verification succeeded!
   Power cycle the system to enable the new firmware.
   ```
3. Power-cycle the machine to load the new firmware. A reboot/restart is not sufficient.

4. In the event there are repeated programming or verification errors, run the command again,
   
   OR
   
   Power-cycle the machine and run the command again.

5. Enter the following command to confirm that the new firmware is running:

   $ sudo ./bin/myri_info

   **Output:**

<table>
<thead>
<tr>
<th>Version: 2.1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type : SNF</td>
</tr>
<tr>
<td>Config : 4 Port x 10 Gb</td>
</tr>
<tr>
<td>SHA1 : 2d13f73ad9fe4bd4bda8d7b50dd0ad0b</td>
</tr>
</tbody>
</table>

### 7.5 Tracking Environmental Diagnostics on ARC Series E Network Adapters (phx-scan)

Enter the following command line to track and record environmental diagnostics from ARC Series E network adapters:

**Command line:**

$ sudo ./phx-scan

**Output: FPGA Revisions**

<table>
<thead>
<tr>
<th>K35 FPGA Revisions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev: 0x00350001</td>
</tr>
<tr>
<td>Branch: 0x00010010</td>
</tr>
</tbody>
</table>

**Output: Sensor Records**

<table>
<thead>
<tr>
<th>K35 Sensor Records:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor 0: FPGA Core Temperature</td>
</tr>
<tr>
<td>Sensor 1: FPGA Local Temperature</td>
</tr>
<tr>
<td>Sensor 2: Over Temperature Alert</td>
</tr>
<tr>
<td>Sensor 3: Fan Speed Percentage</td>
</tr>
<tr>
<td>Sensor 4: Fan RPMs</td>
</tr>
<tr>
<td>Sensor 5: Power Management Good</td>
</tr>
<tr>
<td>Sensor 6: Power Management Alert</td>
</tr>
</tbody>
</table>
Output: Sensor Readings:

<table>
<thead>
<tr>
<th>Sensor Readings:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA Core Temperature: VALUE=32.31 degrees C, STATE=OK</td>
</tr>
<tr>
<td>FPGA Local Temperature: VALUE=32.00 degrees C, STATE=OK</td>
</tr>
<tr>
<td>Over Temperature Alert: VALUE=1, STATE=OK</td>
</tr>
<tr>
<td>Fan Speed Percentage: VALUE=100% STATE=OK</td>
</tr>
<tr>
<td>Fan RPMs: VALUE=6064 STATE=OK</td>
</tr>
<tr>
<td>Power Management Good: VALUE=0, STATE=OK</td>
</tr>
<tr>
<td>Power Management Alert: VALUE=4, STATE=OK</td>
</tr>
</tbody>
</table>

The ARC Series E adapters have two temperature sensors: a FPGA Core Temperature sensor and a FPGA Local Temperature sensor.

The FPGA Local Temperature sensor default setting is 80 °C with a hysteresis of -5 °C. When the temperature exceeds 80 °C the sensor readings output is STATE=NOT OK. ARC Series E adapter functionality is not interrupted or affected when the temperature exceeds the desired range.
7.6 Tracking SFP+ Diagnostics on ARC Series E Network Adapters (myri_phx_mdio)

The MYRI_PHX_MDIO command retrieves SFP+ transceiver modules diagnostics from the 10G-PCIE3-8E-2S and 10G-PCIE3-8E-4S adapters.

**Linux**

**Command line:**

```
$ sudo ./bin/myri_phx_mdio -s
```

**Output:**

```
pci-dev at 05:00.0 vendor:product(rev)=1c09:4264(01) port 0

Reading SFP 0
SFP+ info:
  Type = SFP (0x3)
  Connector = LC (0x7)
  Compliance = 10GBASE-SR (0x10)
  Wavelength = 850 nm
  Vendor = [vendor name]
  PN = [part number]
  SN = [serial number]
  Date = 2016/03/28

  Current status:

    Temp = 28.5586 Celsius
    Vcc = 3.3306 V
    TX bias = 8.734 mA
    TX power = 0.5674 mW
    RX power = 0.6599 mW

10GBASE-R PHY Link Status UP for port 0.
```
8 Testing SNFv5.3.2.7 Software

After you have successfully installed SNFv5.3.2.7 software to your operating system, best practices recommend that you subject the SNFv5.3.2.7 software to a testing regimen. To that end, SNFv5.3.2.7 software offers several test programs to ensure optimum SNFv5.3.2.7 packet capture performance.

This chapter contains the following topics:
- Summary of Test Program Commands
- SNFv5.3.2.7 Test Program Requirements
- Sample Test Programs

8.1 Summary of Test Program Commands

- snf_basic_diags
- snf_simple_recv
- snf_multi_recv
- snf_bridge
- snf_pktgen
- snf_replay

8.2 SNFv5.3.2.7 Test Program Requirements

Review the following requirements to take full advantage of the SNFv5.3.2.7 test program features.

8.2.1 Adapter port nomenclature

Many SNFv5.3.2.7 example test programs require a -p <port> as a command line argument. The port number refers to the physical connector port on the network adapter.

All ARC Series E network adapter ports are numbered, beginning at zero. If the network adapter has two physical ports, the port closest to the PCI connector is assigned “port 0” and the second port is assigned “port 1”. Port 0 also corresponds to the lower MAC address of a dual-port adapter. Quad-port ARC Series E network adapters have four physical ports: “port 0”, “port 1”, “port 2”, and “port 3”, respectively.

For more information on port numbering, go to Testing the ARC Series E Adapter Hardware
For multiple network adapters installed on the server, the adapter port numbers are assigned in the order in which they are detected by the BIOS. Refer to the output of `myri_nic_info -B` to determine the port numbering sequence, as described in the Summary of Diagnostic Tool Programs section of Running SNFv5.3.2.7 Diagnostic Tool Programs.

### 8.3 Sample Test Programs

The SNFv5.3.2.7 sample test programs can be found in the `/opt/snf/bin/tests/` directory.

**Summary of SNFv5.3.2.7 test program commands**

- `snf_basic_diags`
- `snf_simple_recv`
- `snf_multi_recv`
- `snf_bridge`
- `snf_pktgen`
- `snf_replay`

#### 8.3.1 snf_basic_diags

A nice adapter smoke test. The program is not as detailed as `phx_bug_report`, which delves more into system information.

**Definition:**

`snf_basic_diags`

**Usage:**

```
./snf_basic_diags [options]
```

**Command line [options]:**

```
enp2s0   portnum=0 mac_addr=00:60:dd:43:2e:04 maxrings=32 maxinject=0
link_speed 10 Gbps

enp2s0-snf1 portnum=1 mac_addr=00:60:dd:43:2e:05 maxrings=32 maxinject=0
link_speed 10 Gbps

test_getifaddrs() found 2 devices

snf_open test: First port      portnum =       0 successful
snf_open test: First port      portnum =       0
snf_ring_open() successful

snf_open test: All valid ports portmask=     0x3 successful
snf_open test: All valid ports portmask=     0x3
snf_ring_open() successful

snf_open test: All ports with a link up Skipping since no ports are UP
```
snf_open test: Invalid port request           portnum = 31 failure=11:
Resource temporarily unavailable

snf_open test: Invalid port request           portnum = 31 does NOT have
to pass

port=0, link status is down timesource is local host (no external)
port=1, link status is down timesource is local host (no external)

test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs

snf timeout test on port 0 passed

test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs
test_recv_timeout for 500 msecs returned 11 after 500 msecs
test_recv_timeout for 500 msecs returned 11 after 499 msecs

snf timeout test on port mask 0x3 passed

test_inject@0 host sent 2 packets (wait for stats update)
test_inject@0 NIC sent 2 packets was a SUCCESS

test_inject@1 host sent 2 packets (wait for stats update)
test_inject@1 NIC sent 2 packets was a SUCCESS

test_cancel_recv via pthread_kill started over 4 rings and threads

test_cancel_recv via pthread_kill started over 4 rings and threads was a
SUCCESS

test_cancel_recv via pthread_kill started over 8 rings and threads

test_cancel_recv via pthread_kill started over 8 rings and threads was a
SUCCESS
8.3.2 \textit{snf\_simple\_recv}

The \textit{snf\_simple\_recv} test program demonstrates usage of \textit{snf\_ring\_recv}().

\textbf{Definition:}

\texttt{snf\_simple\_recv}

\textbf{Usage:}

\begin{verbatim}
\end{verbatim}

\textbf{Command line [options]:}

- \texttt{v}: verbose
- \texttt{t}: print periodic statistics, \texttt{-t -t}: also print timestamp mode
- \texttt{p <port number>}: Myri10G port number to use.
- \texttt{T <timeout>:} Wait for at most \texttt{<timeout>} milliseconds between calls, 0 never blocks
- \texttt{n <num pkts>:} number of packet to receive (default: 0 - infinite)
- \texttt{d <ring sz>:} size if the recieve ring in bytes
- \texttt{S <snap len>:} display first \texttt{<snap len>} bytes of each packet
- \texttt{D:} display nanosecond delta from timestamp of prior packet
- \texttt{R <port number>:} reinject every received packet on port \texttt{<portnum>}
- \texttt{N:} pass every received packet to netdev
- \texttt{O:} do not check out-of-order packet timestamps
- \texttt{m <pkts>:} print verbose output every \texttt{<pkts>} packets (need \texttt{-v})
- \texttt{--help:} Display this help, and exit

8.3.3 \textit{snf\_multi\_recv}

The test program \textit{snf\_multi\_recv} demonstrates \textit{snf\_ring\_recv\_many()} and \textit{snf\_ring\_recv()} function call usage.

\textbf{Definition:}

\texttt{snf\_multi\_recv}

\textbf{Usage:}

\begin{verbatim}
./snf_multi_recv -w <num_workers> [options]
\end{verbatim}
Command line [options]:

```
options:
- p <port number>: Myri10G port number to use.
- n <num pkts>: number of packet to receive (default: 0 - infinite)
- t: show periodic stats, every second
- V: validate incoming packets
- v: verbose, or -vv very verbose
- d <ring sz>: size of the receive ring in bytes, or megabyte
- D <nanosecs>: add <nanosecs> of synthetic processing delay in packet handling
- W <msecs>: timeout of <msecs> in blocking receive calls
- S <snap len>: display first <snap len> bytes of each packet
- R <port number>: reinject every received packet on port <portnum>
- A <app_id>: set application ID
- q <core>[,<core>]: thread ID to which assign the respective workers. Should match <num_workers>
- O: do not check out-of-order packet timestamps
--help: Display this help, and exit.
```

8.3.4 snf_bridge

A test program that creates a transparent bridge to analyze traffic on one device and to replay it on another.

**Definition:**

```
    snf_bridge
```

**Usage:**

```
./snf_bridge [options] -b <...> [ -b <...> ... ]
```
Command line [options]:

options:
-\b <board_source>:<board_dest>:<num_rings>[::<0xcpumask>]
  <board_source> is where to capture packets
  <board_dest> is where to forward packets
  <num_rings> is the number of rings/workers to dedicate to capture
  <cpumask> is an optional binding cpumask in hexadecimal

-\n <num_packets>: Number of packets to forward before exiting
-\N <tx\_try\_again>: Number of times to try injecting before dropping packet
-\W <tx\_wait\_msecs>: Number of milliseconds to wait in snf\_inject\_send
  R: Reflect non UDP and TCP packets to network device
--help: Display this help, and exit.
8.3.5  snf_pktgen

A test program that generates packets for injection.

**Definition:**

snf_pktgen

**Usage:**

```
./snf_pktgen [options]
```

**Command line [options]:**

- **-v:** Print verbose output
- **-p <config>:** Configure threads (max: 8), where
  - `<config>` = `<board>[:<nthreads>][:<cpumask>]>` (default: 0:1)
- **-q <coreID>:** CPU ID to assign respective workers
- **-n <count>:** Number of packets to send per thread (default: infinite)
- **-r <x.y>:** Replay at x.y Gbits/s (per thread, not aggregate)
- **-c <climb_spec>:** Replay starting at x.y Gbits/s (per thread, not aggregate),
  - `<climb_spec>` = `<x.y>:[climb_sec]` (climb_sec defaults to 3sec)
  - then increment by x.y Gbits after climb_sec expired
- **-C <climb_spec>:** Same as `-c` except that bandwidth starts decreasing once it hits 10Gbps
  - When it reaches the starting bandwidth, it increases again
  - then increment by x.y Gbits after climb_sec expired
- **-f <sec>:** Replay at random [0.1-10.0] Gbits/s (per thread, not aggregate)
  - Bandwidth changes every `<sec>` seconds
- **-F <x.y>:<max_sec>:** Same as `-C` except that bandwidth periods vary randomly
  - `<x.y>` is the starting bandwidth, and the period is random
  - `[1, <max_sec>]`
- **-R <x.y>:** Replay at x.y Mpps (per thread, not aggregate)
- **-d <delay_ns>:** Delay in nanoseconds between start of each packet
- **-W:** Wait instead of busy-poll when injecting packets
- **-E <src mac>-<dst mac>:** Source-destination MAC address
- **-I <src ip>-<dst ip>:** Source-destination IP address
- **-s <size>:** Packet size (default: 60)
- **-S <src port>:** Source port
- **-D <dst port>:** Destination port (-S/S/D only)
  - For rand/seq ranges, specify `<min value>-<max value>:{R,S}`
- **--help:** Display this help, and exit
### 8.3.6 snf_replay

A test program that uses SNF-level injection to replay a libpcap file.

**NOTE:**

You must have enough available memory to accommodate the libpcap file you are attempting to replay.

**Definition:**

**snf_replay**

**Usage:**

```bash
./snf_replay [options] <file.pcap>
```

**Command line [options]:**

- `-v`: Print verbose output
- `-p <port>`: Myri10G port number to use
- `-n <count>`: Maximum number of packets to send from pcap file
- `-t <nthreads>`: Number of threads to concurrently send (max: 16)
- `-i <iters>`: Number of times to replay the file on each thread
- `-T <vlan_id>`: Insert a vlan tag for packets without VLAN tags
- `-r <x.y>`: Replay at x.y Gbits/s (per thread, not aggregate)
- `-R <x.y>`: Replay at x.y Mpps (per thread, not aggregate)
- `-z`: Replay according to packet timestamp
- `-Z`: Use injection pacing according to packet timestamp
- `-N`: Packets are stored with nanosecond precision
- `-F`: Disable setting thread affinity
- `-m`: Read pcap file into memory first
  
  Note - Make sure your pcap file fits in available memory

- `-e <trace_id>`: Generate a packet trace and send per timestamp
  
  `<trace_id>` (0, 1, ...) specifies a trace.
  
  0 - one 60B packet every 100nsec
  1 - one 60B packet every 500nsec
  2 - one 60B packet every 1usec
  3 - one 60B packet every 2usec
  4 - one 60B packet every 5usec
  5 - one 60B packet every 10usec

- `-E <trace_id>`: Receive and compare timestamps against the generated trace

--help: Display this help, and exit

**NOTE:**

You must have enough available memory to accommodate the libpcap file you are attempting to replay.
The [-m] command line option reads the libpcap file into memory, and then replays it from memory instead of from a disk. This option allows it to achieve line rate (if your machine/memory is fast enough) for small packets.

The [-z] flag calculates the deltas between packets in the libpcap file and passes them to the snf_inject_sched() call.

The [-z] flag uses an alternate delay method to pass zero to the DELAY_NS parameter.

The [-N] flag detects the nanosecond timestamp in libpcap files and opens them in PCAP_STAMP_PRECISION_NANO.
9 Running SNFv5.3.2.7 Diagnostic Tool Programs

The SNFv5.3.2.7 software package includes diagnostic tool programs, found in the sbin/ and bin/ directories. Some tools may be required to run as root.

Most of the diagnostic tools generate diagnostic information for error reporting.

<table>
<thead>
<tr>
<th>NOTE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SNFv5.3.2.7 diagnostic tool programs contain different arguments and outputs than in previous hardware generations. For example, the myri_counters and myri_endpoint_info outputs are noticeably different, which are described in this chapter.</td>
</tr>
</tbody>
</table>

9.1 Summary of diagnostic tool programs

- sbin/phx_bug_report
- sbin/myri_info
- bin/myri_counters
- bin/myri_endpoint_info
- bin/myri_nic_info
9.1.1  sbin/phx_bug_report (Linux only)

Run the sbin/phx_bug_report diagnostic bug report as root. Creates output as a compressed file.

Description:
sbin/phx_bug_report

Command Line:
$ sudo /opt/snf/sbin/phx_bug_report

Sample output from phx_bug_report.

Output:

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 Technical Support via the case you have opened
https://www.ariacybersecurity.com/support/downloads/
or ARIA_support@ariacybersecurity.com
and include a description of your problem.

9.1.2  sbin/myri_info

Provides network adapter information, such as hardware serial number, MAC address, firmware version, and so on.

Description:
sbin/myri_info

Command line [options]:
$ sudo /opt/snf/sbin/myri_info [options]
### Usage:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-b &lt;board_num&gt;</code></td>
<td>-- only print info about a card instance <code>&lt;board_num&gt;</code>&lt;br&gt;<code>&lt;board_num&gt;</code> is rank if decimal, or pci_bus if hex</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>-- verbose: more details about the card(s)</td>
</tr>
<tr>
<td><code>-vv</code></td>
<td>-- very verbose: even more details</td>
</tr>
<tr>
<td><code>-x</code></td>
<td>-- select only LX-based card (D, E, F)</td>
</tr>
<tr>
<td><code>-z</code></td>
<td>-- select only LZ-based 10G cards</td>
</tr>
<tr>
<td><code>-s &lt;spec_file&gt;</code></td>
<td>-- save the string_specs into file <code>&lt;spec_file&gt;</code></td>
</tr>
<tr>
<td><code>-c</code></td>
<td>-- on z-cards, checks the MCP(s) CRCs</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>-- don't try to use MMIO (to be used when MMIO fails)</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>-- show this help screen</td>
</tr>
</tbody>
</table>

### 9.1.3 bin/myri_counters

Generates output for low-level network adapter counters.

#### Description:

`bin/myri_counters`

#### Command line [Help]

```
$ sudo /opt/snf/bin/myri_counters -h
```

#### Usage:

Usage: myri_counters [args]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-p N</code></td>
<td>Port number or Ethernet MAC</td>
</tr>
<tr>
<td><code>-b N</code></td>
<td>Port number or Ethernet MAC</td>
</tr>
<tr>
<td><code>-c</code></td>
<td>-- clear the counters</td>
</tr>
<tr>
<td><code>-q</code></td>
<td>-- quiet: show only nonzero counters</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>-- show host interrupt counters</td>
</tr>
<tr>
<td><code>-x</code></td>
<td>-- expert: show all counters</td>
</tr>
<tr>
<td><code>-o</code></td>
<td>-- show register offset</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>-- raw: show register contents</td>
</tr>
<tr>
<td><code>-e N</code></td>
<td>-- show counters for specified endpoint [0]</td>
</tr>
<tr>
<td><code>-a</code></td>
<td>-- show counters for all endpoints</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>-- show all counters</td>
</tr>
<tr>
<td><code>-F</code></td>
<td>-- show filter state including all registered filters</td>
</tr>
<tr>
<td><code>-M</code></td>
<td>-- show MAC filters, if available</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>-- help</td>
</tr>
</tbody>
</table>
Resetting the counters

To clear counters on a specific port of the network adapter, enter the following command line:

Command line:
$ sudo /opt/snf/bin/myri_counters -p <port_num> -c

Resetting counters requires root privileges.

For more information on myri_counters, go to Appendix 1: SNFv5.3.2.7 Counters

9.1.4 bin/myri_endpoint_info

Identifies which processes consume adapter-level resources, such as endpoints. It also tracks which process IDs are in use.

Description:
$ sudo bin/myri_endpoint_info

Command line [options]:
$ sudo /opt/snf/bin/myri_endpoint_info [options]

Usage:

- -b - Board mac or number [0]
- -h - help

9.1.5 bin/myri_nic_info

Provides diagnostic information on the number of adapters installed, which driver is loaded, and the status of the software license(s) for each network adapter.

Description:
$ sudo bin/myri_nic_info

Command line [options]:
$ sudo /opt/snf/bin/myri_nic_info [options]

Usage:

--machine : Comma separated output (for machine parsing)
--all     : Print all known nics. (may need root privileges)
--license : Print license state details
--help    : Print this help message
10 Configuring SNFv5.3.2.7

This chapter describes the following topics:

- Debug Variable
- Ring Management Variables
- RSS Hashing/Load Balancing Variables
- Port Aggregation and Merging Variables
- Traffic Management Variable
- Kernel Ethernet Interfaces

This chapter describes the SNFv5.3.2.7 environment variables available to configure advanced functions so that they match application needs and perform efficiently. The following SNFv5.3.2.7 environment variable categories are described as follows:

**Debug variable**

SNF_DEBUG_MASK

**Ring management variables**

SNF_RING_ID
SNF_NUM_RINGS
SNF_DATARING_SIZE
SNF_DESCRING_SIZE

**RSS hashing/load balancing variable**

SNF_RSS_FLAGS

**Port aggregation and merging variable**

SNF_FLAGS

**Traffic management variable**

SNF_APP_ID
10.1 Debug Variable (SNF_DEBUG_MASK)

The debug environment variable enables the SNFv5.3.2.7 library to print various debug information, including configurations, to the console.

Example 1:

To verify that a SNF-compatible libpcap library is correctly linked to the application, set SNF_DEBUG_MASK to 0x3 and open the snfX device, as follows:

```
$ SNF_DEBUG_MASK=0x3 /path/to/tcpdump -i snf0
```

Setting the SNF_DEBUG_MASK variable causes the SNF API to display SNFv5.3.2.7 configuration information when libpcap opens snfX. If no output appears, the application may be linked to a version of libpcap that is not compatible with SNF. Monitor myri_counters to verify that traffic is being received.

Example 2:

Similarly, you can run SNF_DEBUG_MASK=0x3 with RSS hashing flags to ensure that the SNFv5.3.2.7 application is correctly linked to the SNF library.

```
$ SNF_DEBUG_MASK=0x3 SNF_RSS_FLAGS=0x1 ./snf_simple_recv
```

Output:

```
snf.0.0 P (userset) SNF_PORTNUM = 0
snf.0.0 P (default) SNF_RING_ID = -1 (0xffffffff)
snf.0.0 P (default) SNF_NUM_RINGS = 1 (0x1)
snf.0.0 P (default) SNF_RSS_FLAGS = 1 (0x1)
snf.0.0 P (default) SNF_DATARING_SIZE = 268435456 (0x10000000) (256.0 MiB)
snf.0.0 P (default) SNF_DESCRING_SIZE = 67108864 (0x4000000) (64.0 MiB)
snf.0.0 P (default) SNF_FLAGS = 0
snf.0.0 P (environ) SNF_DEBUG_MASK = 3 (0x3)
snf.0.0 P (default) SNF_DEBUG_FILENAME = stderr
...```

For more information on snfX numbering and how it corresponds to host network adapters, go to the Numbering of snfX Interfaces section in Troubleshooting.
10.2 Ring Management Variables

The ring management variables are:

- SNF_RING_ID
- SNF_NUM_RINGS
- SNF_DATARING_SIZE
- SNF_DESCRING_SIZE

To view the RING management variable settings, run the SNF API application with the environment variable SNF_DEBUG_MASK=3. The snf_open() function call displays the current variable settings and how they were set (hard-coded by the application or the default value).

Example:

Command line:

```
$ SNF_DEBUG_MASK=3 ./snf_simple_recv
```

Output:

```
snf.0.-1 P (userset)     SNF_PORTNUM = 0
snf.0.-1 P (default)    SNF_RING_ID = -1 (0xffffffff)
snf.0.-1 P (default)    SNF_NUM_RINGS = 1 (0x1)
snf.0.-1 P (environ)    SNF_RSS_FLAGS = 1 (0x1)
snf.0.-1 P (default)    SNF_DATARING_SIZE = 16777216 (0x1000000) (16.0 MiB)
snf.0.-1 P (default)    SNF_DESCRING_SIZE = 4194304 (0x400000) (4.0 MiB)
```

10.2.1 SNF_RING_ID

When running multiple rings, each program can either open the next available ring, or a specific ring. By default SNF_RING_ID=-1, opens the next available ring. In a particular instance where an application wants to open a specific ring, SNF_RING_ID is set to a value between 0 and SNF_NUM_RINGS - 1.

For example, if we set SNF_NUM_RINGS=4, we would expect to run four instances of an application, such as Bro or tcpdump; each one getting roughly one-quarter of the packets. By specifying SNF_RING_ID=1, the first application will open the first ring (ring 0), the second application will open the second ring (ring 1), the third application will open the third ring (ring 2), and the fourth application will open the last ring (ring 3).

By specifying which ring to open, the application selects the ring in question, as opposed to the next available ring.
Example:
Specifying which ring to open:

$ SNF_RING_ID=0 tcpdump -i snf0 &
$ SNF_RING_ID=1 tcpdump -i snf0 &
$ SNF_RING_ID=2 tcpdump -i snf0 &
$ SNF_RING_ID=3 tcpdump -i snf0 &

If one of these applications should die, you can restart the correct ring to continue.

10.2.2 SNF_NUM_RINGS

SNF_NUM_RINGS represents the number of rings the SNFv5.3.2.7 application supports. The maximum number of rings that SNFv5.3.2.7 can support per adapter port is 32. While the SNF_NUM_RINGS ranges from 1 to 32, the SNF_RING_ID ranges from 0 to 31. By default, SNF_NUM_RINGS=1.

Command line (Default):

SNF_NUM_RINGS=1

Assigning more than 32 rings per adapter port prompts the following myri_snf WARN: message.

Output [WARN]:

```
myri_snf WARN: eth2: endpt XX, early enable failed
```

In Linux, the user may utilize `taskset` to explicitly bind ring threads to different processors.
10.2.3 SNF_DATARING_SIZE

The SNF_DATARING_SIZE variable represents the total available memory to store incoming packet data. If the value is set to zero or less than zero, the library opts for a practical default unless SNF_DATARING_SIZE is set in the environment. The library may also adjust the user’s request to satisfy alignment requirements (typically 2MB boundaries). By default, SNF_DATARING_SIZE=16MB. The value may be specified in megabytes or in bytes, depending on size.

**Command line (Default):**

```
SNF_DATARING_SIZE=16MB
```

Increasing the SNF_DATARING_SIZE (and its corresponding SNF_DESCRING_SIZE variable) allows the application to buffer a greater number of packets. It may be useful to handle bursty traffic while avoiding packet losses.

**NOTE:** Increasing the data ring size also increases process start-up time. For maximum performance limit the aggregate ring size for data and descriptor rings to the L3 cache size.

For more information on L3 cache size, go to the L3 Cache Awareness section in Tuning SNFv5.3.2.7 Software.

10.2.4 SNF_DESCRING_SIZE

The SNF_DESCRING_SIZE variable sets the size of the descriptor ring. The ring stores packet metadata such as timestamp and length in main memory. The descriptor ring is typically sized to be a quarter the size of the data ring, and allows for small packets (60 bytes). In most cases the descriptor ring will fill up before the data ring.

For example, if a user increases the size of the data ring to 32MB, we recommend adjusting the descriptor ring size to a quarter of 32MB or 8MB.

**Command line (Default):**

```
SNF_DESCRING_SIZE=4MB
```

**NOTE:** The maximum permissible level of configured memory assigned to all rings is 80 percent of physical memory. For maximum performance, limit the aggregate ring size for data and descriptor rings to the L3 cache size.

For more information on L3 cache size, go to the L3 Cache Awareness section in Tuning SNFv5.3.2.7 Software.
10.3 RSS Hashing/Load Balancing Variables  
(SNF_RSS_FLAGS)

RSS hashing flags are set via the environment variable SNF_RSS_FLAGS. To view SNF_RSS_FLAGS variable settings, run the SNF API application with the environment variable SNF_DEBUG_MASK=3. The snf_open() function call displays the current variable settings and how they were set (hard-coded by the application or by the default value).

**Example:**

```bash
$ SNF_DEBUG_MASK=3 SNF_RSS_FLAGS=0x1 ./snf_simple_recv
```

**Output:**

```
snf.0.-1 P (userset)    SNF_PORTNUM = 0
snf.0.-1 P (default)    SNF_RING_ID = -1 (0xffffffff)
snf.0.-1 P (default)    SNF_NUM_RINGS = 1 (0x1)
snf.0.-1 P (environ)    SNF_RSS_FLAGS = 1 (0x1)
snf.0.-1 P (default)    SNF_DATARING_SIZE = 16777216 (0x1000000) (16.0 MiB)
snf.0.-1 P (default)    SNF_DESCRING_SIZE = 4194304 (0x400000) (4.0 MiB)
snf.0.-1 P (default)    SNF_FLAGS = 0
snf.0.-1 P (environ)    SNF_DEBUG_MASK = 3 (0x3)
snf.0.-1 P (default)    SNF_DEBUG_FILENAME = stderr
```

### 10.3.1 SNF_RSS_FLAGS default settings

By default, SNF_RSS_FLAGS=0x31. It tells the SNF library to generate RSS hash values based on IP addresses and TCP/UDP ports.

**Command line (Default):**

```
SNF_RSS_FLAGS=0x31.
```

The SNF_RSS_FLAGS definitions are found in the `/opt/snf/include/snf.h` header file. See Table 3 for the various RSS variable settings.

**Output:**

```c
enum snf_rss_mode_flags {
    SNF_RSS_IP = 0x01, /**< Include IP (v4 or v6) SRC/DST addr in hash */
    SNF_RSS_SRC_PORT = 0x10, /**< Include TCP/UDP/SCTP SRC port in hash */
    SNF_RSS_DST_PORT = 0x20, /**< Include TCP/UDP/SCTP DST port in hash */
    SNF_RSS_GTP = 0x40, /**< Include GTP TEID in hash */
    SNF_RSS_GRE = 0x80, /**< Include GRE contents in hash */
};
```
Output description:

- **SNF_RSS_IP** - IP address
- **SNF_RSS_SRC_PORT** - Source port
- **SNF_RSS_DST_PORT** - Destination port
- **SNF_RSS_GTP** - GPRS Tunneling Protocol
- **SNF_RSS_GRE** - Generic Routing Encapsulation

To include GTP TEID in calculated hash:

Environment variable setting:

```
SNF_RSS_FLAGS=0x71
```

To include GRE in calculated hash:

Environment variable setting:

```
SNF_RSS_FLAGS=0xB1
```

<table>
<thead>
<tr>
<th>Packet type</th>
<th>Select RSS variable settings</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP packets</td>
<td>SNF_RSS_IP</td>
<td></td>
</tr>
<tr>
<td>IP packets (UDP, TCP or SCTP)</td>
<td>SNF_RSS_SRC_PORT, SNF_RSS_DST_PORT</td>
<td></td>
</tr>
<tr>
<td>Traffic tunneled through the GTP protocol</td>
<td>SNF_RSS_GTP</td>
<td>Assigns a Tunnel Endpoint Identifier (TEID) to the hash when the first payload byte indicates the packet is GTP version 1 or GTP version 2.</td>
</tr>
<tr>
<td>Traffic tunneled through the GRE protocol</td>
<td>SNF_RSS_GRE</td>
<td>Assigns the encapsulated GRE IP payload packet headers to the hash.</td>
</tr>
<tr>
<td>Ethernet frames bearing “non-hashable” traffic (MPLS or non-supported traffic)</td>
<td>No variable setting. Distributed to ring 0.</td>
<td>Distributed to ring 0.</td>
</tr>
</tbody>
</table>

Table 3. RSS variable settings.
Special Considerations

- RSS implementation works well for standard Ethernet traffic. However, if SNF-provided hashes do not perform well for the given network traffic, you can write a custom hash function in C language to direct the SNF library to compute hash values.

- You can also take advantage of the RSS multi-ring capabilities of load balancing across your application for network traffic types that are not handled well by the variables shown in Table 3.

- No packets are dropped regardless of packet type. Packet drops may indicate a bad cyclic redundancy check (CRC32) or an overflow, where either the adapter and/or the application cannot sustain the packet rate.
10.4 Port Aggregation (Merging) Variables (SNF_FLAGS)

The SNF_FLAGS environment variable controls process-sharing (0x1), port merging (0x2), and packet duplication (0x300). The following flags are found in the /opt/snf/include/snf.h header file:

- SNF_F_PSHARED 0x1
- SNF_F_AGGREGATE_PORTMASK 0x2
- SNF_F_RX_DUPLICATE 0x300

When port merging is enabled, SNFv5.3.2.7 merges traffic into a single logical receiver from:

- Two or more ports.
- From the same or different set of ARC Series E adapters.

10.4.1 SNF_F_PSHARED 0x1

The SNF_FLAGS=1 environment variable allows multiple independent processes to share rings on the capturing device. This option may be used to design a custom capture solution, but it can also be used in libpcap when multiple rings are requested. In this scenario, each libpcap application sees a fraction of the traffic if multiple rings are used unless the SNF_F_RX_DUPLICATE option is used, in which case each libpcap application sees the same incoming packets.

Definition:

```c
#define SNF_F_PSHARED 0x1
```

10.4.2 SNF_F_AGGREGATE_PORTMASK 0x2

ARC Series E adapters are amenable to port aggregation (merging). For example, if you set the SNF_FLAGS environment variable to two, (SNF_FLAGS=2), the portnum variable in snf_ring_open() call is interpreted as a bitmask, with a value of 1 assigned to port 0, 2 to port 1, and 3 to the merging of ports 0 and 1. The SNFv5.3.2 library then attempts to open every port specified in order to merge the incoming data from multiple ports. Subsequent calls to snf_ring_open() return a ring handle that internally opens a ring on all underlying ports.

Definition:

```c
#define SNF_F_AGGREGATE_PORTMASK 0x2
```
**Dual-port ARC Series E adapters:**

The dual-port ARC Series E adapter has ports 0 and 1. When using `libpcap`, these ports are named `snf0` and `snf1`. With aggregation turned on, `libpcap` shows three devices: `snf1` (port 0), `snf2` (port 1), and `snf3` (port 0 and 1).

**Quad-port ARC Series E adapters:**

The quad-port ARC Series E adapter has ports 0, 1, 2, and 3. When using `libpcap`, these ports are named `snf0`, `snf1`, `snf2`, and `snf3` respectively. When port merging is enabled, the number represents a bitmask of ports. For example, `snf12` represents merging of ports 2 and 3.

### 10.4.3 SNF_F_RX_DUPLICATE 0x300

The SNF device can duplicate packets to multiple rings instead of applying RSS in order to split incoming packets across rings. Users should be aware that with ‘n’ rings opened, ‘n’ times the host processing power is necessary to process incoming packets without drops.

The duplication occurs in the host rather than the SNF device, so while only up to 10Gbps per port of traffic crosses the PCIe, ‘n’ times that bandwidth is necessary on the host.

When duplication is enabled, RSS options are ignored since every packet is delivered to every ring.

**Definition:**

```
#define SNF_F_RX_DUPLICATE 0x300
```
10.5 Application ID Variable (SNF_APP_ID)

SNFv5.3.2.7 enables multiple independent applications to run in parallel, each receiving all the traffic and internally splitting the traffic among its application threads. For example, the user may run Bro, Suricata, and tcpdump at the same time.

The SNF_APP_ID environment variable assigns a unique application ID to each independent application to be run. With each application ID set, SNFv5.3.2.7 can then duplicate receive packets to multiple applications, with each application having a different number of rings. If the application has one process with multiple rings and threads, all rings share the same ID. If the application consists of multiple processes, these processes share the same application ID.

10.5.1 Running SNF_APP_ID with third-party tools

This feature has the unfortunate side effect of overriding anything you set in the SNF_NUM_RINGS and SNF_FLAGS variables. Proceed with caution.

The following example uses the SNF_APP_ID variable with Bro and tcpdump.

1. Ensure SNFv5.3.2.7 has been installed and LD_LIBRARY_PATH has been set correctly so that both Bro and tcpdump are linked to libpcap with SNFv5.3.2.7 support.

2. Set SNF_APP_ID to identify which processes are working together:
   - Set Bro to run SNF_APP_ID=1
   - Set tcpdump to run SNF_APP_ID=2

3. Run Bro.

4. Ensure that the node.cfg file is defined as follows:

   File contents:
   
   [worker-1]
   type=worker
   host=localhost
   interface=snf0
   lb_method=69inuxpt
   lb_procs=6
   pin_cpus=3,4,5,6,7,8
   env_vars=SNF_APP_ID=1
**Bro** creates six processes to consume packets and sets the **SNF_NUM_RINGS** and **SNF_FLAGS** variables appropriately.

| NOTE: | To allow another application to obtain a copy of the packets, export **SNF_APP_ID=1** before starting **Bro**. |

5. With the **node.cfg** file set as above, run the following command:

   ```
   $ sudo broctl
   ```

   **Output:**

   ```
   Welcome to BroControl 1.2
   
   Type “help” for help.
   
   BroControl] > start
   ```

6. Run two instances of **tcpdump**.

   ```
   $ sudo SNF_APP_ID=2 SNF_NUM_RINGS=2 SNF_FLAGS=0x1 \ tcpdump -i snf0 -w /mnt/ramdisk/cap1
   $ sudo SNF_APP_ID=2 SNF_NUM_RINGS=2 SNF_FLAGS=0x1 \ tcpdump -i snf0 -w /mnt/ramdisk/cap2
   ```

7. Run **myri_endpoint_info** to view the open endpoints:

   ```
   $ sudo ./myri_endpoint_info
   ```
**Output:**

The `myri_snf` driver is configured to support a maximum of:

160 endpoints per NIC, 32 NICs per host

<table>
<thead>
<tr>
<th>Board 00:60:dd:45:4f:5c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td>33</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>35</td>
</tr>
<tr>
<td>36</td>
</tr>
<tr>
<td>37</td>
</tr>
<tr>
<td>38</td>
</tr>
<tr>
<td>39</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td>67</td>
</tr>
<tr>
<td>68</td>
</tr>
<tr>
<td>69</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>71</td>
</tr>
</tbody>
</table>
10.6 Kernel Ethernet Interfaces

SNFv5.3.2.7 creates a kernel Ethernet interface for port 0. It appears when you run `ip addr` or `ifconfig -a` as one interface with a SNF MAC address. The other ports (1, 2, and 3) are not visible to the kernel, and do not appear in the kernel’s interface list.

Normally, the port 0 kernel interface remains down, and it cannot send or receive packets through the interface. But the user may configure it and use it for various purposes, such as administration.

When the user starts a SNF application on port 0, it takes over the port. All packets go to the SNF application, and the kernel interface becomes a black hole while the application is running. When the SNF application terminates, the kernel Ethernet interface takes back the port and can send and receive traffic.
11 Load Balancing and Port Merge Features

Two environment variables are used to invoke port merging and load balancing across many cores (RSS hashing):

- `export SNF_NUM_RINGS=2`
- `export SNF_FLAGS=3`

11.1 SNF_NUM_RINGS

The SNF_NUM_RINGS variable enables RSS Hashing. RSS Hashing does not deploy if only one ring is in use.

SNF_RSS_FLAGS specifies hash types. By default, SNF calculates hashes using IP addresses and TCP/UDP ports.

**Output:**

```
Default hash value -

mumur_hash(min(src_ip, dst_ip), max(src_ip, dst_ip),
src_port+dst_port)
```

For more information on hashing, refer to the RSS Hashing/Load Balancing Variables section of Configuring SNFv5.3.2.7 Software

The user deploys a custom RSS Hashing function and instructs SNFv5.3.2.7 to run the function (the `custom_hash` is defined in `snf.h`, where you pass a function pointer).

This feature is not available in off-the-shelf `libpcap` applications, as there is no way to provide a function pointer (unless you modify the initialization code to provide the custom hash function). Because the custom hash feature is invoked by a procedure call, the additional CPU overhead on every packet by the driver is significant, under high packet-per-second loads.
11.2 SNF_FLAGS

The variable `SNF_FLAGS` is a bitmask with the following properties:

- Sharing between applications is 0x01
- Port merging is 0x02
- Sharing and merging is 0x03

11.2.1 Running multiple instances

There are two ways to run multiple instances of your application:

**Method 1:**

Running a single process and creating multiple threads, with each thread tied to a specific CPU core and assigned 1/nth of the traffic, where “n” is the number of rings created.

**Method 2:**

Running multiple Instances (processes), with “n” processes tied to a specific CPU core, and assigned 1/nth of the traffic, where “n” is the number of rings created.

11.2.2 Sharing/Load balancing

The sharing flag allows multiple copies of an application to open a port or `libpcap` device. Each open port receives 1/nth of the traffic flow, based on a pre-defined RSS Hashing. If you do not enable the sharing flag, any subsequent application’s attempt to open the port returns an error message.

Before the SNFv5.3.2.7 application runs and opens a SNF port, the port behaves as a standard Ethernet adapter, and the packets pass to a kernel stack. No environment variables are used.

Once the first application opens a port or `libpcap` device in SNFv5.3.2.7 mode, the first instance of the application’s environment variables is invoked. When sharing is enabled, this allows more than one process to open the same SNFv5.3.2.7 device, for example, SNF port 0 (`snf0`). These processes have the potential to set different SNFv5.3.2.7 environment variable values.

For example, you could set the environment variable `SNF_NUM_RINGS` to 4 in one instance, and not specify a value in another, in which case the `SNF_NUM_RINGS` value in the non-specified instance is set to default.
Where SNF_APP_ID values are identical

In the case where the SNF_APP_ID values are identical for two application processes (that open the same `snf0`), the first application process configuration variable opens the SNFv5.3.2.7 device, with the second application process using the same configuration variable values as the first, to open the SNFv5.3.2.7 device.

This concept is important, particularly if the processes do not always start in a prescribed manner. For example, process A sometimes opens `snf0` first, or process B sometimes opens `snf0` first. Whichever process is the first to open `snf0` determines the values to be assigned to both process A and process B.

Where SNF_APP_ID values are different

In the case where the SNF_APP_ID values are different for two application processes (that open the same `snf0`), the first application process configuration variable opens `snf0` for each SNF_APP_ID. Accordingly SNF_APP_ID 1 may have different variables than SNF_APP_ID 2.

Setting different environment variables

When invoking a second instance of the application with different environment variable values such as SNF_NUM_RINGS or SNF_DATARING_SIZE, those values are ignored, as only the values specified by the first open are used to open `snf0`.

Anything that can be done with SNF API applies to `libpcap` as well:

- You can use RSS hashing to distribute packets to several `libpcap` applications that open the same SNFv5.3.2.7 interface.
- You can also distribute the entire ring to two or more `libpcap` applications running SNF_APP_ID.
- You can also mix `libpcap` applications with SNFv5.3.2.7 API applications.

11.2.3 Port Listening on `libpcap`

With a dual-port adapter, the `libpcap` devices are named `snf0`, corresponding to port 0, and `snf1`, corresponding to port 1.

Example:

```
SNF_FLAGS=1 SNF_NUM_RINGS=2 tcpdump -i snf0 >/tmp/ring1 &
SNF_FLAGS=1 SNF_NUM_RINGS=2 tcpdump -i snf0 >/tmp/ring2 &
```

By setting the sharing bit in the SNF_FLAGS variable, multiple applications can open the same interface. Because there is more than one ring, each instance of the application gets its own virtual ring of packets.
Ring Descriptions

The packets are distributed among the following rings using RSS Hashing:

- There is only one physical ring per port, so the size of the ring, specified by SNF_DATARING_SIZE, is only used for one ring per port.
- The packet descriptors are stored in a separate ring, defined by SNF_DESCRING_SIZE, which is typically one quarter the size of SNF_DATARING_SIZE, to accommodate bursts of 60-byte packets.

11.2.4 Hardware assisted port merging

With port merging enabled, the libpcap device name number becomes a bitmask of the ports you wish to merge. For example, a dual-port ARC Series E adapter, the valid interfaces are:

Example:

- snf1 – all the packets on port 0
- snf2 – all the packets on port 1
- snf3 – all the packets on ports 0 and 1 in timestamp order

Hardware accelerated port merging

The dual- and quad-port ARC Series E adapters support hardware acceleration for port merging. The SNF host library automatically runs hardware accelerated port merging depending upon the applications.

The ARC Series E adapters can only support port merging in specific pairs of ports. If the application chooses ports outside of these supported combinations, the SNF host library implements software port merging, with lower performance than hardware acceleration.

Dual-port support

The supported port combinations for hardware accelerated port merging for the dual-port ARC Series E adapter are:

- ports 0 and 1

Quad-port support

The supported port combinations for hardware accelerated port merging for the quad-port ARC Series E adapter are:

- ports 0 and 1
- ports 2 and 3
Merging ports between multiple ARC Series E adapters

The SNF host library does not support port merging among multiple ARC Series E adapters. In this case, and in this case it will use software as not hardware acceleration is supported in this case.

Merging ports and adding rings

To merge ports and add multiple rings, specify `SNF_NUM_RINGS>1`, and `SNF_FLAGS=3` (sharing and merging):

**Example:**

```
SNF_FLAGS=3 SNF_NUM_RINGS=2 tcpdump -i snf3 >/tmp/ring1 &
SNF_FLAGS=3 SNF_NUM_RINGS=2 tcpdump -i snf3 >/tmp/ring2 &
```

The example demonstrates port merging with the `snf3` interface, by running two instances of the application.
12 Libpcap, and PF_RING Packet Capture

This chapter describes the following topics:

- **libpcap** and SNF
- **PF_RING** and SNFv5.3.2.7
- Demonstrating PF_RING Functionality

12.1 libpcap and SNFv5.3.2.7

**Libpcap** provides a packet-capture application API (pcap) and filtering engine for many open-source and commercial network tools, including protocol analyzers (packet sniffers), network monitors, network intrusion detection systems (NIDS), traffic-generators, and network-testers.

**Libpcap version**

The **libpcap** library distributed as part of the SNFv5.3.2.7 software package is version 1.7.4.

12.1.1 Preamble

SNFv5.3.2.7 software interfaces with the **libpcap** library, (or directly through the SNF API) to capture packets travelling over a network. With a SNF-aware **libpcap** library, users reference the ARC Series E network adapter through its Ethernet interface name to run existing **libpcap**-dependent applications.

When **libpcap** encounters a SNF-capable device, it enables the SNF API to obtain user-level, zero-copy packets instead of the usual kernel-based approach.

SNFv5.3.2.7 improves packet capture performance in two ways:

- SNFv5.3.2.7 exerts more control over user-level receive mechanisms by bypassing the kernel.
- By re-linking existing **libpcap** applications to SNF-capable **libpcap** libraries.

12.1.2 Verifying the libpcap link to SNFv5.3.2.7

To ensure that a SNF-aware **libpcap** library is linked to the application, set the SNF_DEBUG_MASK variable to 3:

**Command line:**

```
SNF_DEBUG_MASK=3
```

Setting this variable triggers SNF API to output information when **libpcap** interfaces with the SNFv5.3.2.7 device.
12.2 PF_RING and SNFv5.3.2.7

PF_RING is an alternative network socket that dramatically improves packet capture speed and efficiency, thereby preserving CPU cycles.

Linux

PF_RING is available for Linux and provides support to the ARC Series E class of network adapters.

12.2.1 Installing PF_RING from Linux RPM

1. Enter the following command line to install PF_RING on CentOS as root user:

   **Command line:**
   
   $ sudo rpm -ivh pfring-dkms-6.5.0-718.noarch.rpm

   **Output:**
   
   warning: pfring-dkms-6.5.0-718.noarch.rpm: Header V4 RSA/SHA1 Signature, key ID d1eb60be: NOKEY
   Preparing...                           [100%]
   Updating / installing... 1:pfring-dkms-6.5.0-718
   [100%]
   WARNING: /usr/lib64/dkms/common.postinst does not exist.
   Loading new pfring-6.5.0 DKMS files...
   Building for 3.10.0-327.4.5.el7.x86_64
   Building initial module for 3.10.0-327.4.5.el7.x86_64
   Done.

   Pf_ring:
   Running module version sanity check.
   - Original module
   - No original module exists within this kernel
   - Installation
     - Installing to /lib/modules/3.10.0-327.4.5.el7.x86_64/extra/
   Adding any weak-modules
   depmod...
   DKMS: install completed.

2. Enter the following command line:

   **Command line:**
   
   Command line:
$ sudo rpm -ivh pfring-6.5.0-718.x86_64.rpm

Output:

```
warning: pfring-6.5.0-718.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID d1eb60be: NOKEY
Preparing...
################################################ [100%]
Updating / installing... 1:pfring-6.5.0-718
################################################ [100%]
```

**NOTE:** By default, PF_RING installs test applications to the `/usr/local/bin` directory
12.2.2 Installing PF_RING from source (ntop)

PF_RING contains the following components:

- The PF_RING user-space SDK.
- An enhanced version of the libpcap library that transparently takes advantage of PF_RING if installed, or falls back to the standard behavior if not installed.
- The PF_RING kernel module.
- PF_RING-aware drivers for different chips from various vendors.

For more information on installing PF_RING from source, go to: http://www.ntop.org/get-started/download/#PF_RING

Download and install PF_RING by Git or Ubuntu/CentOS repositories by entering the following:

```bash
git clone https://github.com/ntop/PF_RING.git
cd PF_RING/kernel
make
sudo insmod ./pf_ring.ko
cd ../userland
make
```

You can compile the entire tree typing make (as normal, non-root, user) from the main directory.

12.2.3 Configuring the PF_RING library with SNFv5.3.2.7

Configure the PF_RING library by entering the following commands:

```bash
$ cd /opt/snf
$ sudo ./sbin/rebuild.sh
$ sudo ./sbin/myri_start_stop start
$ sudo echo < “/opt/snf/lib” > /etc/ld.so.conf.d/snf.conf
$ sudo ldconfig

$ cd PF_RING/kernel
$ make
$ sudo insmod pf_ring.ko
$ cd ../userland/lib
$ sudo ./configure
$ make
$ cd ../libpcap
```
$ sudo ./configure
$ make
$ cd ../examples
$ make
$ sudo ./pfcount -i myri:0

**NOTE:** Specify myri:0 in order to open port 0
12.3 Demonstrating Multi-Process PF_RING Functionality

The PF_RING distribution provides examples that can be used to demonstrate functionality, including:

- Multi-process traffic duplication
- Multi-process traffic sharing (RSS)
- Ports aggregation example
- PF_RING over SNF example

By default, PF_RING applications are installed in the /usr/local/bin directory. To access the SNFv5.3.2.7 devices, run the PF_RING program with root privileges.

12.3.1 Multi-process traffic duplication example

The “classic” PF_RING is synonymous with the concept of cluster, which serves to balance ingress packets coming from one or more ingress interfaces.

For example, take the pfcount application that receives and counts ingress packets, and suppose you want to balance traffic coming from port 0 to two pfcount applications. To do this, enter the following commands:

**Example:**

```
$ SNF_APP_ID=1 ./pfcount -i myri:0
$ SNF_APP_ID=2 ./pfcount -i myri:0
```

OR

```
$ sudo ./pfcount -i myri:A1P0
$ sudo ./pfcount -i myri:A2P0
```

Where A1 refers to SNF_APP_ID=1 and P0 refers to port 0.

### 12.3.2 Multi-process traffic sharing (RSS) example

The PF_RING distribution can also scale applications to work on multiple cores (RSS) as seen in Figure 4.

- Supports multicore processor scalability, where the ARC Series E devices distribute incoming packets among “virtual rings”. **Bro**, for example, built its scaling strategy around this concept.
- Configures one virtual ring for every CPU core that an application leverages.
- Sends all the packets in a single TCP flow to the same virtual ring.

![Diagram of Myricom multicore scaling (RSS)](image)

**Figure 4.** Myricom multicore scaling (RSS)

To perform multicore scaling (RSS) with a PF_RING distribution, enter the following command lines:

**Example:**

```bash
$ SNF_APP_ID=1 SNF_NUM_RINGS=2 SNF_RING_ID=0 ./pfcount -i myri:0

$ SNF_APP_ID=1 SNF_NUM_RINGS=2 SNF_RING_ID=1 ./pfcount -i myri:0

OR

$ sudo ./pfcount -i myri:A1R2P0@0

$ sudo ./pfcount -i myri:A1R2P0@1
```

Where **A1** refers to SNF_APP_ID=1, **R2** refers to RSS with two rings, **P0** refers to port 0, and **@0** refers to Ring 0.
### 12.3.3 Ports aggregation example

To allow `pfcount` to capture traffic on more than one port, a comma-separated list of ports needs to be given.

To merge two ports into one, enter the following command:

```
$ SNF_FLAGS=0x2 ./pfcount -i myri:0,1
```

### 12.3.4 PF_RING over SNF example

In this example, packets are being received from port 1 by a single instance of `pfcount` pinned to a single CPU as chosen by `pfcount`.

1. Enter the following command line:

```
$ sudo /usr/local/bin/pfcount -i myri:1
```

(Note the license error, shown corrected below).

**Output:**

```
# ERROR: You do not seem to have a valid PF_RING-Myricom 6.3.0.160304
# license for port 1 [00:60:DD:20:16:02]
# ERROR: Please get one at http://shop.ntop.org/.
# We’re now working in demo mode with packet capture and
# transmission limited to 5 minutes
Using PF_RING v.6.5.0
Capturing from myri:1 [mac: unknown][if_index: 1][speed: 0Mb/s]
# Device RX channels: 1
# Polling threads:  1
Dumping statistics on /proc/net/pf_ring/stats/18261-none.37
Absolute Stats: [0 pkts total][0 pkts dropped][0.0% dropped]
[0 pkts rcvd][0 bytes rcvd]
```

---

**NOTE:**

The user should refer to PF_RING documentation for further information on PF_RING command line options at [http://www.ntop.org/products/packet-capture/pf_ring/](http://www.ntop.org/products/packet-capture/pf_ring/).
2. Start a packet generator, such as `pfsend` or `snf_pkgen`, on another system. Packets total count starts increasing.

**Output:**

```
Absolute Stats: [772'784 pkts total][0 pkts dropped][0.0% dropped]
[772'784 pkts rcvd][64'913'856 bytes rcvd][193'164.32 pkt/sec][129.81 Mbit/sec]
Actual Stats: [772'784 pkts rcvd][1'000.15 ms][772'665.78 pps][0.52 Gbps]
```

Packet sender at full line rate.

**Output:**

```
Absolute Stats: [90'074'559 pkts total][0 pkts dropped][0.0% dropped]
[90'074'559 pkts rcvd][7'566'262'956 bytes rcvd][9'005'887.07 pkt/sec][6'051.95 Mbit/sec]
Actual Stats: [14'883'823 pkts rcvd][1'000.17 ms][14'881'144.39 pps][10.00 Gbps]
```

Packet sender ends, rate decreases steadily to zero.

**Output:**

```
Absolute Stats: [100'000'000 pkts total][0 pkts dropped][0.0% dropped]
[100'000'000 pkts rcvd][8'400'000'000 bytes rcvd][9'089'301.11 pkt/sec][6'108.01 Mbit/sec]
Actual Stats: [9'925'441 pkts rcvd][1'000.20 ms][9'923'416.62 pps][6.67 Gbps]
```

Packet sender has finished.

3. Enter **CTRL-C** to close the PF_RING receiver application.

4. Verify that all packets sent were received.

Once the license file has been created in `/etc/pf_ring/<MAC_ADDRESS>` the error and time limitation are no longer an issue.
5. Enter the following command line:

```
$ sudo /usr/local/bin/pfcount -i myri:1
```

**Output:**

```
Using PF_RING v.6.5.0
Capturing from myri:1 [mac: unknown][if_index: 1][speed: 0Mb/s]
# Device RX channels: 1
# Polling threads: 1
Dumping statistics on /proc/net/pf_ring/stats/19417-none.39
```

6. Enter the following command line to start a traffic generator on a remote connected system. There is no need to run as root.

```
Hostname:anypath> /opt/snf/bin/tests/snf_pktgen -b 1 -n 100000000 -S 80-8000:R -D 80-8000:R
```

**Output:**

```
open_tx_endpoint: board_type=K35SNF
Final flush: packets=1279
Draining packets. Inflight=69808
Draining packets. Inflight=55536
Draining packets. Inflight=40645
Draining packets. Inflight=24265
Draining packets. Inflight=7879
[0] Size Mbps Mpps Efficiency
  60 7140.07 14.875 99.96%
```
13 Open-Source Packet Capture Tools

SNFv5.3.2.7 is compatible with industry-standard open-source packet capture application tools. Examples of tested applications include:

- Standard Linux utility **tcpdump**
- Standard Linux utility **tcpreplay**
- **Suricata** network intrusion detection and security monitoring
- **Bro** IDS network intrusion detection system
- **Wireshark** network protocol analyzer
- **Snort** intrusion prevention systems

NOTE: SNFv5.3.2.7 can accommodate the multi-process **Bro**, multi-threaded **Suricata**, and **tcpdump** packet capture tools running in parallel; each tool collecting and splitting the network traffic among their respective application threads.

Figure 5. describes the various open-source application tools and their relationship to **libpcap**.

Figure 5. Industry Standard Packet Capture Software

SNFv5.3 does not support Windows operating systems, including WinPcap.
13.1 Running tcpdump with SNFv5.3.2.7

You can run tcpdump a number of ways:

- Running tcpdump with no recompiling
- Building a new tcpdump tool with recompilation
- Running the tcpdump tool with LD_LIBRARY_PATH

13.1.1 Running tcpdump with no recompiling

Run tcpdump as follows:

1. Verify that tcpdump has a dynamically linked libpcap.

   $ sudo ldd `which tcpdump` | grep pcap

2. When tcpdump statically links libpcap, there is no output. Proceed to the following Building a new tcpdump with recompilation section.

   OR

   When tcpdump does not statically link to libpcap, output is generated. Proceed to the Running the tcpdump tool with LD_LIBRARY_PATH section.

   **Output:**

   ```
   libpcap.so.0.9.4 => /usr/lib64/libpcap.so.0.9.4
   (0x00000034c1400000)
   ```

13.1.2 Building a new tcpdump tool with recompilation

**NOTE:** While libpcap remains binary-compatible between versions 0.9.x to 1.1.x, we recommend recompiling tcpdump against the newest version of libpcap.

To build a new tcpdump with recompilation, enter the following:

1. Enter the following command to install libpcap-devel.

   $ sudo yum install libpcap-devel

2. Download and build tcpdump.

   $ wget http://www.tcpdump.org/release/tcpdump-4.7.4.tar.gz
   $ tar xzf tcpdump-4.7.4.tar.gz
   $ cd tcpdump-4.7.4
   $ ./configure
   $ make
13.1.3 Running the tcpdump tool with LD_LIBRARY_PATH

To run `tcpdump` with LD_LIBRARY_PATH, enter the following:

1. Verify that `tcpdump` has a dynamically linked `libpcap`.
   
   $ sudo ldd tcpdump | grep pcap
   
   **Output:**
   
   ```
   $ ldd tcpdump | grep pcap
   libpcap.so.1 => /lib64/libpcap.so.1 (0x00007ffa48cc8000)
   ```

2. Set LD_LIBRARY_PATH to the SNF-compatible `libpcap`.
   
   Csh/tcsh:
   
   $ sudo setenv LD_LIBRARY_PATH /opt/snf/lib
   
   bash:
   
   $ sudo export LD_LIBRARY_PATH=/opt/snf/lib

3. Verify that `ldd` is pointing to `/opt/snf/lib/libpcap`...
   
   $ sudo ldd tcpdump | grep pcap
   
   **Output:**
   
   ```
   libpcap.so.1 => /opt/snf/lib/libpcap.so.1 (0x00007f5b6e577000)
   ```

4. Set `SNF_DEBUG_MASK=3` to verify that incoming packets are going through SNFv5.3.2.7.
   
   Csh/tcsh:
   
   $ sudo setenv SNF_DEBUG_MASK 3
   
   bash:
   
   $ sudo export SNF_DEBUG_MASK=3

5. Run `tcpdump` on the `eth18` interface.
   
   $ sudo tcpdump -i eth18
### Output:

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
<th>SNF_PORTNUM</th>
<th>SNF_RING_ID</th>
<th>SNF_NUM_RINGS</th>
<th>SNF_RSS_FLAGS</th>
<th>SNF_DATARING_SIZE</th>
<th>SNF_DESCRING_SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5273</td>
<td>snf.0.-1 P (userset)</td>
<td>0</td>
<td>-1 (0xffffffff)</td>
<td>0</td>
<td>0x31</td>
<td>16777216 (0x1000000)</td>
<td>4194304 (0x400000)</td>
</tr>
</tbody>
</table>

**tcpdump**: WARNING: eth18: no Ipv4 address assigned

**tcpdump**: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth18, link-type EN10MB (Ethernet), capture size 65535 bytes

---

**NOTE 1:** Be aware that `tcpdump` prints to output or to file, which may severely limit achievable packet rates to a couple of gigabits per second.

**NOTE 2:** The SNFv5.3.2.7 packet sniffer interface diverts all traffic from the Ethernet interface to the SNFv5.3.2.7 application and may cause host interface response failures.

### 13.1.4 Available `tcpdump` interfaces

Run the `tcpdump -D` argument to display the available SNF port interfaces.

**Command line:**

```
$ sudo tcpdump -D
```

**Output:**

15.eth18 (Myricom snf0)
16.eth18-snf1 (Myricom snf1)
17.eth18-snf2 (Myricom snf2)
18.eth18-snf3 (Myricom snf3)
Table 4 describes the available SNF port interfaces.

<table>
<thead>
<tr>
<th>Ethernet interface</th>
<th>SNF Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth18 (or snf0)</td>
<td>corresponds to SNF port0</td>
</tr>
<tr>
<td>eth18-snf1 (or snf1)</td>
<td>corresponds to SNF port1</td>
</tr>
<tr>
<td>eth18-snf2 (or snf2)</td>
<td>corresponds to SNF port2</td>
</tr>
<tr>
<td>eth18-snf3 (or snf3)</td>
<td>corresponds to SNF port3</td>
</tr>
</tbody>
</table>

Table 4. Available SNF port interfaces.
13.2 Running tcpreplay with SNFv5.3.2.7

tcpreplay is an open source tool that reads and replays (send) previously captured packets in libpcap. The tool supports a number of methods to send packets. By default, it supports PF_PACKET, by sending packets through the kernel.

For more information on tcpreplay, go to: http://tcpreplay.appneta.com/wiki/installation.html#download-source

13.2.1 Building tcpreplay

In order to sustain low overhead SNFv5.3.2.7 packet injection, build tcpreplay and run it with SNFv5.3.2.7 libpcap as follows:

Command lines:
$ git clone https://github.com/appneta/tcpreplay
$ cd tcpreplay
$ ./autogen.sh
$ LD_LIBRARY_PATH=/opt/snf/lib ./configure \
   --prefix=/path/to/install \
   --with-libpcap=/opt/snf \
   --enable-force-inject
$ LD_LIBRARY_PATH=/opt/snf/lib make
$ make install

The -enable-force-inject command line argument forces tcpreplay to run the PCAP_INJECT method with SNFv5.3.2.7 libpcap. In this fashion you are running a SNFv5.3.2.7 packet injection method.

13.2.2 Listing available SNF packet injection interfaces

In this example, enter the following command line to list which interfaces are available for injection:

Command line:
$ LD_LIBRARY_PATH=/opt/snf/lib tcpreplay --listnics
Output:

<table>
<thead>
<tr>
<th>Ethernet interface</th>
<th>SNF Port</th>
<th>Ip link</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2p1</td>
<td>corresponds to SNF port 0</td>
<td>Appears in ip link</td>
</tr>
<tr>
<td>p2p1-snf1</td>
<td>corresponds to SNF port1</td>
<td>Does not appear</td>
</tr>
<tr>
<td>p2p1-snf2</td>
<td>corresponds to SNF port2</td>
<td>Does not appear</td>
</tr>
<tr>
<td>p2p1-snf3</td>
<td>corresponds to SNF port3</td>
<td>Does not appear</td>
</tr>
</tbody>
</table>

Table 5. Available SNF port interfaces.

13.2.3 Running tcpreplay through a specific port

To replay through a specific port, run `tcpreplay -i` with an interface name from Table 5 (p2p1-snf1):

Command line:

```
$ LD_LIBRARY_PATH=/opt/snf/lib tcpreplay -i p2p1-snf1 <pcap-file>
```

tcpreplay replays through port 1.
13.3 Running Suricata with SNFv5.3.2.7

**Suricata** ([http://www.suricata-ids.org/](http://www.suricata-ids.org/)) is an open-source-based Intrusion Detection System (IDS) that interfaces with SNFv5.3.2.7 through **libpcap**.

### 13.3.1 Configuring and building Suricata with libpcap

In order to run **Suricata**, you must first configure **Suricata libpcap** settings.

**Example:**

We are configuring four SNF ARC Series E adapter ports, where *eth18* corresponds to port 0, and *eth18-snf1*, *eth18-snf2*, and *eth18-snf3* correspond to ports 1, 2, and 3, respectively (Table 6).

**Command line:**

```bash
$ sudo ./configure --with-libpcap-includes=/opt/snf/include/ --with-libpcap-libraries=/opt/snf/lib/ --prefix=/usr --sysconfdir=/etc --localstatedir=/var

$ make

$ make install
```
Output:

pcap:
- interface: eth18
  threads: 8
  checksum-checks: no
  snaplen: 9018

- interface: eth18-snf1
  threads: 8
  checksum-checks: no
  snaplen: 9018

- interface: eth18-snf2
  threads: 8
  checksum-checks: no
  snaplen: 9018

- interface: eth18-snf3
  threads: 8
  checksum-checks: no
  snaplen: 9018

NOTE: Enable snaplen for Suricata to retrieve maximum transmission unit (MTU) information from SNF interfaces.

The threads: 8 setting creates eight reader threads for eth18. The Myricom driver ensures each thread is attached to its own ring buffer.

<table>
<thead>
<tr>
<th>Ethernet interface</th>
<th>SNF Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth18 (or snf0)</td>
<td>corresponds to SNF port 0</td>
</tr>
<tr>
<td>eth18-snf1 (or snf1)</td>
<td>corresponds to SNF port 1</td>
</tr>
<tr>
<td>eth18-snf2 (or snf2)</td>
<td>corresponds to SNF port 2</td>
</tr>
<tr>
<td>eth18-snf3 (or snf3)</td>
<td>corresponds to SNF port 3</td>
</tr>
</tbody>
</table>

Table 6. Available SNF port interfaces.
13.3.2 Running Suricata with libpcap

To run Suricata, enter the following:

Example:

**Command line:**

```
SNF_FLAGS=1 SNF_NUM_RINGS=8 LD_LIBRARY_PATH=/opt/snf/lib 97inuxxptp -c /etc/Suricata/97inuxxptp.yaml -pcap=eth18 -runmode=workers
```

You will need to use `-pcap` to specify the SNF ports.

The command runs `-pcap=eth18` on SNF port 0. To run it on SNF port 1, specify `-pcap=eth18-snfl`, and so on, as described in Table 7.

<table>
<thead>
<tr>
<th>Ethernet interface</th>
<th>SNF Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcap=eth18</td>
<td>corresponds to SNF port 0</td>
</tr>
<tr>
<td>pcap=eth18-snfl1</td>
<td>corresponds to SNF port1</td>
</tr>
<tr>
<td>pcap=eth18-snfl2</td>
<td>corresponds to SNF port2</td>
</tr>
<tr>
<td>pcap=eth18-snfl3</td>
<td>corresponds to SNF port3</td>
</tr>
</tbody>
</table>

Table 7. Available SNF port interfaces.

13.3.3 Known issue

Running Suricata on ports other than port 0 displays the following timestamped error messages:

**Output [Error]:**

```
21/7/2016 - 15:05:48 - <Warning>  - [ERRCODE: SC_ERR_SYSCALL(50)] - Failure when trying to get feature via ioctl for ’eth18-snfl’: No such device (19)
21/7/2016 - 15:05:48 - <Warning>  - [ERRCODE: SC_ERR_SYSCALL(50)] - Failure when trying to get MTU via ioctl for ’eth18-snfl’: No such device (19)
```

**Explanation:**

SNFv5.3.2.7 exposes only port 0 as a kernel interface, such as `eth18`. The remaining ports are hidden from the kernel. On ports 1 through 3, Suricata cannot query the interface MTU or offload capabilities through system calls, thus prompting the error messages. These are expected and do not disrupt normal Suricata operations.
13.4 Running Bro with SNFv5.3.2.7

Bro (http://www.bro.org/), which is now referred to as Zeek, is an open-source-based IDS package that interfaces with SNFv5.3.2.7 through libpcap.

13.4.1 Configuring Bro

1. To configure Bro to run with SNFv5.3.2.7, enter the following command:

   $ sudo ./configure --with-pcap=/opt/snf

2. Configure any worker using SNFv5.3.2.7 in the node.cfg file by entering the following command lines:

   [worker1]
   type=worker
   host=1.2.3.4
   interface=snf1
   lb_method=98inuxpt
   lb_procs=8
   pin_cpus=3,4,5,6,7,8,9,10

Flow-based load balancing is automatically enabled. It is important to pin the processing to specific cores to maximize performance.

SNFv5.3.2.7 can accommodate the multi-process Bro, multi-threaded Suricata, and tcpdump packet capture tools running in parallel; each tool collecting and splitting the network traffic among their respective application threads.

NOTE: If Bro was previously configured, compiled, and dynamically linked with libpcap, there is no need to reconfigure. You can instead reset LD_LIBRARY_PATH as described in the following section.
13.4.2 Running Bro with LD_LIBRARY_PATH

**NOTE:**

If Bro is built with SNF libpcap (--with-pcap=/opt/snf), setting LD_LIBRARY_PATH is not necessary as the executable file explicitly links to /opt/snf/lib/libpcap.so.

If Bro is built with the generic libpcap, set LD_LIBRARY_PATH so that Bro dynamically links with SNF libpcap.

To run Bro using the environment variable LD_LIBRARY_PATH, enter the following:

1. Verify that Bro has a dynamically linked libpcap.
   
   ```
   $ sudo ldd `which bro` | grep pcap
   ```

2. When Bro statically links libpcap, there is no output.
   
   OR

   When Bro does *not* statically link to libpcap, the following output is generated.

   **Output:**

   ```
   libpcap.so.0.9.4 => /usr/lib64/libpcap.so.0.9.4
   (0x00000034c1400000)
   ```

3. Create a symlink between Bro and libpcap in the /opt/snf/lib directory.
   
   ```
   $ sudo ln -s /opt/snf/lib/libpcap.so
   /opt/snf/lib/libpcap.so.0.9.4
   ```

4. Set LD_LIBRARY_PATH to the SNF-compatible libpcap:
   
   **csh/tcsh:**
   
   ```
   $ setenv LD_LIBRARY_PATH /opt/snf/lib
   ```

   **bash:**
   
   ```
   $ export LD_LIBRARY_PATH=/opt/snf/lib
   ```
13.5 Running Wireshark with SNFv5.3.2.7

Wireshark version 2.0.5 (http://www.wireshark.org/) shares many characteristics with tcpdump, except that it supports a graphical user interface (GUI). Wireshark interfaces with SNFv5.3.2.7 software through libpcap.

13.5.1 Running Wireshark in Linux

NOTE: These examples are specifically for the Wireshark application; however, the procedure is similar for any application running libpcap.

To run Wireshark in Linux, follow these instructions:

2. Create a link from pcap.h to pcap/pcap.h in the /opt/snf/include directory.
   
   $ cd /opt/snf/include; sudo ln -sf pcap/pcap.h pcap.h
   $ sudo yum install bison flex gtk2 gtk2-devel
   $ sudo tar jxf wireshark-1.2.5.tar.bz2
   $ cd wireshark-1.2.5

   
   $ sudo ./configure -prefix=/opt/wireshark -with-pcap=/opt/snf
   $ make
   $ make install

13.5.2 Alternate approach to running Wireshark in Linux

While recompiling is the preferred and safest way to make use of SNFv5.3.2.7 functionality, the following approach also works:

1. Install the Wireshark and Wireshark-gnome packages on RHEL 5.4.
2. Run ldd to verify that Wireshark has a dynamically linked libpcap.

   $ sudo ldd /usr/sbin/wireshark | grep pcap

3. Once you install SNFv5.3.2.7 RPM, then symlink the libpcap library to the SNFv5.3.2.7 RPM distribution and change LD_LIBRARY_PATH.

   $ cd /opt/snf/lib
   $ sudo ln -s libpcap.so.1 libpcap.so.0.9.4
   $ sudo LD_LIBRARY_PATH=/opt/snf/lib /usr/sbin/wireshark

4. Rerun ldd to verify that the correct libpcap.so and libsnf.so have been selected.

   $ sudo LD_LIBRARY_PATH=/opt/snf/lib ldd /usr/sbin/wireshark | grep pcap
13.6 Running Snort with SNFv5.3.2.7 (Parallel Snort)

While libpcap is not thread-safe, it is possible to run multiple processes that use libpcap/SNF in parallel. Under this configuration, if multiple libpcap processes wish to process incoming data from a single device, they simply need to agree on the total number of rings by exporting the number of desired rings in the environment. This approach effectively translates the number of rings into an equal amount of “virtual” capture devices (Figure 6.).

![Diagram of Multi-Process Snort over libpcap/SNF](image)

**Figure 6. Multi-Process Snort over libpcap/SNF**

**NOTE:** While libpcap is not thread-safe to the extent that multiple threads cannot make calls to the same pcap_t and expect correct behavior. The goal is to have each thread open and access its own pcap_t.
13.6.1 Running Multi-Process Snort over libpcap/SNF

As an example, start eight parallel instances of Snort each bound to different cores, all using the same configuration file and the snf0 interface, as follows:

1. Start eight parallel instances of Snort.
   
   ```bash
   $ export SNF_NUM_RINGS=8
   ```

2. Duplicate incoming data to multiple Snort instances.
   
   `SNF_F_RX_DUPLICATE=0x300` and `SNF_F_PSHARED=0x1` flags
   
   ```bash
   $ export SNF_FLAGS=0x301
   
   $ i=0
   
   $ while [ $i -lt $SNF_NUM_RINGS ]; do
   
   /opt/snort/snort -c /opt/snort/snort.conf \ 
   -i snf0 &
   
   sleep 2
   
   let "i = i + 1"
   
   done
   ```
14 Linux PTP Host Clock Synchronization

This chapter contains the following topics:

- Verifying Adapter Clock Functionality
- Synchronizing the System Clock to the ARC Series E Adapter Clock
- Synchronizing the ARC Series E Clock

14.1 Verifying Adapter Clock Functionality

The ARC Series E network adapter clock displays as a **PTP Hardware Clock**, compatible with Linux kernel version 4.6 and later.

To verify adapter clock functionality, refer to the following procedure:

   OR
   Install through a package manager such as Yum, by entering the following command:
   ```bash
   $ sudo yum install linuxptp
   ```

2. Reference the hardware clock by its PTP device name (**/dev/ptpN**) or by its network name.
   
   All ports on a single network adapter share a single PTP clock. While there may be two or four network interfaces, there will only be one PTP clock. Access to the clock through different network devices references the same physical clock.

   **NOTE:** Make clock adjustments carefully as all timestamps originate from the interface.

14.1.1 Reading the clock by PTP device name:

   **Command line:**
   ```bash
   $ sudo phc_ctl /dev/ptp0 get
   ```
14.1.2  Reading the clock by network name:

   Command line:
   $ sudo phc_ctl enp2s0f0 get

14.1.3  Setting adapter clock to host time:

   Command line:
   $ sudo phc_ctl /dev/ptp0 set

14.1.4  Synchronizing the system clock with the network adapter clock (phc2sys):

   The linuxptp package synchronizes the system clock to the adapter clock on the network.

   Command line:
   $ sudo phc2sys -s /dev/ptp0 -O 0 -m

14.1.5  Maintaining clock synchronization (ptp4l):

   Linuxptp prompts the PTP daemon to send out periodic messages on the interface. To stop messages from affecting adapter speed, switch to a system interface that supports PTP.

   The SNF interface is limited to SW TX Timestamps, and requires a two-step process to synchronize the SNF PTP device over PTP.

   1. PTP must first be set up on the SNF interface, using TX Timestamps to update CLOCK_REALTIME.

      Command line:
      $ sudo ptp4l -i eth0 -p /dev/ptp1 -m -s -S

   Command descriptions:
   
   - The eth0 interface connected to the /dev/ptp1 device supports PTP.
   - The -m option instructs the application to direct output to the console.
   - The -s option forces the PTP daemon to run as a slave.
   - The -S option forces PTP to use SW TX Timestamps.
   - Omitting the -m option prevents ptp4l from directing output to the console if running as a daemon.
   - Omitting the -s option allows the daemon to assume the role of Grand Master if the linuxptp algorithm determines that it is the most accurate clock.
2. Synchronize the SNF PTP device to the CLOCK_REALTIME.

   **Command line:**
   
   $ sudo phc2sys -s CLOCK_REALTIME -c /dev/ptp1 -O 0 -m

   **Command descriptions:**

   - The `-s` option indicates the source clock
   - The clock being used as the source is CLOCK_REALTIME.
   - The `-c` option indicates the clock that will be changed.
   - The SNF clock is `/dev/ptp1`
   - The `-O` option indicates the offset from CLOCK_REALTIME.
   - The `-m` option instructs the application to direct output to the console,
   - Omitting the `-m` option prevents `ptp4l` from directing output to the console if running as a daemon.

   **NOTE:** PTP coordinates with the IAC (International Atomic Clock), which as of June 30, 2015, has a positive 36-second offset from UTC (Coordinated Universal Time). It may be necessary to specify this offset from the CLOCK_REALTIME when synchronizing the ARC Series E adapter PTP device.
14.2 Synchronizing the ARC Series E Clock to Another Clock

Synchronize the SNF PTP device to the CLOCK_REALTIME.

Command line:

```
$ sudo phc2sys -s CLOCK_REALTIME -c /dev/ptpl -O 0 -m
```

Command descriptions:

- The `-s` option indicates the source clock
- The clock being used as the source is `CLOCK_REALTIME`
- The `-c` option indicates the clock that will be changed.
- The SNF clock is `/dev/ptpl`
- The `-O` option indicates the offset from `CLOCK_REALTIME`
- The `-m` option instructs the application to direct output to the console,
- Omitting the `-m` option prevents `ptp4l` from directing output to the console if running as a daemon.

<table>
<thead>
<tr>
<th>NOTE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP coordinates with the IAC (International Atomic Clock), which as of June 30, 2015, has a positive 36-second offset from UTC (Coordinated Universal Time). It may be necessary to specify this offset from the <code>CLOCK_REALTIME</code> when synchronizing the ARC Series E adapter PTP device.</td>
</tr>
</tbody>
</table>
15 SNFv5.3.2.7 Timestamping Support

This chapter contains the following topics:

- Timestamping Module Variables
- Viewing the Time Source Status

SNFv5.3.2.7 in API mode supports SW TX, SW RX, and HW RX timestamping by default.

For more information on implementing timestamping, refer to the Network Adapter Timestamps section in Troubleshooting

15.1 Timestamping Module Variables

The SNFv5.3.2.7 driver has three timestamping module variables to specify the time source.

- MYRI_CLC_ENABLE_PPS
- MYRI_CLC_ENABLE_10MHZ
- MYRI_CLC_INVERT_PPS

Default support settings (disabled):

By default, we assume that there is nothing connected to the packets-per-second (PPS) input connector on the front of the card.

The driver sets both MYRI_CLC_ENABLE_PPS and MYRI_CLC_ENABLE_10MHZ to zero, which disables the PPS and 10MHZ inputs.

Command lines (Disabled):

myri_clk_enable_pps=0
myri_clk_enable_10mhz=0
Enabling PPS support:

To enable PPS support, set the MYRI_CLK_ENABLE_PPS module variable to 1 in /opt/snf/sbin/myri_start_stop and restart the driver.

Command line (Enabled):

```
myri_clk_enable_pps=1
```

Enabling 10MHz support:

To enable 10 MHz support, set the MYRI_CLK_ENABLE_10MH module variable to 1 in /opt/snf/sbin/myri_start_stop and restart the driver.

Command line (Enabled):

```
myri_clk_enable_10mhz=1
```

15.2 Viewing the Time Source Status

There are two ways to view the time source status.

- **dmesg command**
- **myri_info command**

15.2.1 dmesg

Enter the `dmesg` command.

**Output [example]:**

```
... 
[ 40.660764] myri_snf INFO: Disabling 10Mhz input 
[ 40.660766] myri_snf INFO: Disabling PPS input 
[ 40.660768] myri_snf INFO: PPS is inactive: wrote the time to the adapter 
  ts=1471521123.635242000  
... 
```

15.2.2 myri_info

The `myri_info` command displays one set of external time source connections per network adapter. Likewise, two network adapters create a duplicate output for each function.

**NOTE:**

Do not select the MYRI_CLK_ENABLE_10MHZ variable if it is not connected to an external 10MHz time source. Disable the variable first before disconnecting the 10MHz time source.
Command line:

$ sudo ./myri_info -b 0

Output [example]:

pci-dev at 01:00.0 vendor:product(rev)=1c09:4260 (01)
  behind bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)
Myri-10G-PCIE-8E -- Link x8
  EEPROM String-spec:
    MAC=00:60:dd:43:2d:e8
    SN=495892
    PC=10G-PCIE3-8E-4S
    PN=09-04680
    BOM=A

Firmware:
  Version: 2.1.5
  Type   : SNF
  Config : 4 Port x 10 Gb
  SHA1   : 2d13f73ad9fe4bd4bda8d7b50dd0ad0b

External Inputs:
  PPS: Enabled, No Input
    Front Panel PPS: No Input
    Card Edge PPS: No Input
  10Mhz Clock: Disabled
  100Mhz Clock Locked: Locked
16 Tuning SNFv5.3.2.7 Software

This chapter contains the following topics:

- Ring Performance
- Tuning Check List

Myricom high-speed, packet-capture networking solutions in real customer environments vary depending upon the particular details of the network configuration, end-user applications, and transaction workloads.

Critical applications often have internal measurements of effective transaction rates. To that end, we recommend running performance tests before and after installing the ARC Series E network adapter hardware to compare performance results.

Ideally one should use reliable and recurring workload data. If that is not possible, sample enough daily traffic to establish typical performance metrics. If there is no noticeable improvement after installing the hardware, verify that the network adapters and software are properly installed.

For a detailed description of SNFv5.3.2.7 software test packet performance rates, go to the Sample Test Programs section of the Testing SNFv5.3.2.7 Software chapter.

Contact ARIA Technical Support at https://www.ariacybersecurity.com/support/downloads/ for further assistance if performance results are below expectations.

16.1 Ring Performance

Single-ring Performance

Whereas most Internet traffic is usually bimodal in the distribution of packet sizes, SNFv5.3.2.7 has been designed to support a worst case scenario where all packets are at the minimum 10-Gigabit Ethernet packet size of 64 bytes.

When including the 8-byte preamble, the start byte, and the 12-byte inter-packet gap, a minimum-size packet of 64 bytes requires 84 byte times on the wire. Under a constant stream of minimum packet sizes, a packet arrives at every 67.2 nanosecond intervals, corresponding to a maximum packet rate of 14.88 Mpps.

On our reference platform, a Xeon X5570 at 2.93GHz, running SNFv5.3.2.7 in a single ring configuration demonstrates a library overhead of about 32 nanoseconds per packet on average for 64-byte packets. Minimizing library overhead is necessary to achieve high packet rate capture.
Multi-ring performance

The primary goal of using multiple rings is to leverage multiple cores in the packet analysis by effectively reducing the number of packets each ring has to process.

For example, if we assume that the incoming traffic can be fairly well balanced across eight cores, each core is responsible for processing one-eighth of a potential peak of 14.88 Mpps, for a worst case scenario of one packet every 537.6 nanoseconds. With the aforementioned library overhead, this leaves roughly 500 nanoseconds of analysis per core under a worst case scenario.
16.2 SNFv5.3.2.7 Tuning Check List

SNFv5.3.2.7 is a kernel-bypass driver. As such, it does not require much tuning; however, it helps to be aware of the following issues which may affect latency/packet rate.

In addition to these tuning suggestions, there are also a number of environment variables available in the SNFv5.3.2.7 software for debugging and for customizing the software configuration to the requirements of the application.

For more information on environment variables, go to Configuring SNFv5.3.2.7

Tuning is recommended to help reduce the performance hit that takes place when the SNFv5.3.2.7 application alerts the adapter (through a kernel call) that the ring memory has been freed. The SNFv5.3.2.7 application can be preempted by the OS when this call takes place. Tuning also helps reduce packet loss due to a drop in the number of RX data pages available or drops due to PCI backpressure (indicating there is no free memory available in the L3 cache).

For more information on the RX Data Pages Available Min counter, go to Appendix 1. SNFv5.3.2.7 Counters

16.2.1 Setting the system performance profile

We recommend setting the system performance profile to network-latency as described below:

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

This profile is chosen 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 recommendation refers to distribution Red Hat Enterprise Linux (RHEL) and its derivative, CentOS.

16.2.2 PCIe expansion slot seating

Verify that the ARC Series E network adapter is seated properly into a PCI-Express Gen3 expansion slot that is compatible with the correct PCI-Express link width (x8).

Run the myri_info command (included in SNFv5.3.2.7 in /opt/snf/sbin/ directory or the PHX-TOOLS package) to verify that the negotiated PCI-Express link is the correct x8 width.

For more information, refer to the Hardware Installation/Performance Issues section in Troubleshooting
16.2.3 Interrupts and IRQ Affinity (Linux)

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), and calls each ISR in turn. The ISR acknowledges the interrupt and ignores redundant interrupts from the same IRQ, then queues a deferred handler to finish processing the interrupt and stop the ISR from ignoring future interrupts.

IRQ Affinity

The affinity of an interrupt request (IRQ Affinity) is defined as the set of CPU cores that can service that interrupt.

Finding the interrupt number of the device

**Command line:**

grep myriC0-ep01 /proc/interrupts

**Usage:**

```
144: 0
```

Checking the current affinity:

**Command line:**

```
sudo cat /proc/irq/144/smp_affinity
```

**Usage:**

```
00000000,00000000,00000000,00000200
```
16.2.4 Irqbalance

Irqbalance is a Linux daemon that balances the CPU load generated by interrupts across all CPUs. The irqbalance daemon identifies the highest volume interrupt sources and isolates them to a single CPU, spreading the load as much as possible over an entire processor set, and minimizing cache hit rates for IRQ handlers.

Irqbalance is enabled by default.

The /proc/interrupts file includes the following:

- the number of interrupts per CPU per I/O device
- IRQ number
- the interrupt number handled by each CPU core
- interrupt type
- a comma-delimited list (CSV) of drivers that are registered to receive that interrupt.

Disabling irqbalance

By disabling irqbalance, you avoid hardware interrupts in your threads. In real-time deployments, applications are typically dedicated and bound to specific CPUs, so the irqbalance daemon is not required.

Example:

In this example, enter the following commands to disable irqbalance:

Command lines:

```bash
$ sudo systemctl stop irqbalance.service
$ sudo cat /proc/interrupts | grep myri | grep ke
```

Usage:

<table>
<thead>
<tr>
<th>CPU0</th>
<th>CPU1</th>
<th>CPU2</th>
<th>CPU3</th>
</tr>
</thead>
<tbody>
<tr>
<td>69:</td>
<td>9506696</td>
<td>768107</td>
<td>5662150</td>
</tr>
<tr>
<td>94:</td>
<td>7554013</td>
<td>13139806</td>
<td>11239993</td>
</tr>
</tbody>
</table>

Disabling irqbalance loads each interrupt to a separate CPU core, as follows:

Command line:

```
$ sudo cat /proc/irq/69/smp_affinity
```

Usage:

```bash
8
```
That gives us CPU core 3 (bit 3 set).

**Command line:**

```
$ sudo cat /proc/irq/94/smp_affinity
```

**Usage:**

```
4
```

That’s CPU core 2 (bit 2 set).

When all bits are set, the interrupt can run on any core even after rebooting the driver (irqbalance still disabled).

**Command line:**

```
$ sudo cat /proc/irq/69/smp_affinity
```

**Usage:**

```
f
```

The following forces the interrupts to two specific cores.

**Command lines:**

```
$ sudo echo 1 | sudo tee /proc/irq/69/smp_affinity
$ sudo echo 2 | sudo tee /proc/irq/94/smp_affinity
```

We can check that each interrupt is going to the specified core.

**Command line:**

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

**Usage:**

<table>
<thead>
<tr>
<th></th>
<th>CPU0</th>
<th>CPU1</th>
<th>CPU2</th>
<th>CPU3</th>
</tr>
</thead>
<tbody>
<tr>
<td>69:</td>
<td>23827111</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ep01</td>
<td></td>
<td></td>
<td></td>
<td>PCI-MSI-edge myriC0-ep01</td>
</tr>
<tr>
<td>94:</td>
<td>35</td>
<td>458887</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ep02</td>
<td></td>
<td></td>
<td></td>
<td>PCI-MSI-edge myriC0-ep02</td>
</tr>
</tbody>
</table>

Using irqbalance, we can also remove specific cores from the set of cores that receive interrupts.

For more information on disabling irqbalance, go to:

16.2.5 Interrupt Balancing

In some situations, performance may be impacted negatively by interrupt balancing. Disabling IRQ balancing (daemon irqbalance on RHEL), manually assigning the IRQ to a specific CPU, and then binding the tasks to a specific CPU (via `taskset` or `numactl`) may improve performance (see CPU Binding below).

16.2.6 CPU Frequency Scaling

In some cases, dynamic CPU frequency scaling may run CPUs at low frequencies. If so, disable dynamic scaling and always run CPUs at highest speeds.

16.2.7 CPU Binding

CPU binding (e.g., using `taskset` or `numactl` on Linux) can be very useful for optimizing performance for SNFv5.3.2.7.

On hosts with multiple CPU sockets, some CPUs are physically closer to the adapter and/or memory and perform better than others.

Some systems have multiple PCI-Express root ports. For example, multi-socket machines and some PCI Express slots can connect to one root port and are more easily accessible from one CPU socket, while another PCI-Express slot will be connected to another root port and are more easily accessible from another CPU socket.

16.2.8 PCI Bridging

Some PCIe slots on a machine may have deeper PCI bridging than others. Extra bridge chips between the CPU and the adapter will result in higher latency. If there seem to be lots of bridge chips, trying a different PCI-Express slot may improve performance for SNFv5.3.2.7. Run the `/opt/snf/sbin/myri_info` tool to detect the bridges between the CPU and adapter.

16.2.9 Hyperthreading

We do not recommend using more than one SNF receive thread (or process) on each physical core. Enable hyperthreading as long as the processes/threads are mapped correctly (see CPU Binding above).

16.2.10 NUMA Awareness

Execute the `snf_open()` function on the socket “closest” to the PCIe slot containing the ARC Series E adapter. Assign the `snf_ring_open()` function for each process/thread to its own core, preferably on the socket closest to the adapter. Latencies and contention across the QPI bridges between the sockets may arise in some cases.

For more information on NUMA zones, go to the L3 Cache Awareness section of this chapter.
16.2.11 L3 Cache Awareness

For optimal performance we recommend that the aggregate Rx data and descriptor ring size be smaller than the L3 cache. Ring sizes larger than the L3 cache can lead to L3 misses, which in turn can lead to stalls, writes to RAM instead of cache, and slower performance. Slower performance is more likely to lead to dropped packets when operating at line rate.

For more information on default data and descriptor ring sizes, go to the Ring Management Variables section of Configuring SNFv5.3.2.7

Running a single interface

When running a single interface, we recommend the application that executes snf_open() be opened in the NUMA zone closest to the adapter PCIe slot.

Running multiple interfaces

When running multiple interfaces, executing snf_open() for each interface in its own NUMA zone would be preferred.

Running in separate NUMA zones

Running the application in separate NUMA zones is more efficient for memory usage; however, the drawback is that DMA over QPI is slower than the L3 cache, but still more efficient than SW (the packets are written directly into RAM).

When running multiple interfaces in separate NUMA zones, the application with slower packet processing power may compete for the same NUMA zone as the adapter.

While it is acceptable to run SNFv5.3.2.7 applications outside the NUMA zone that the PCIe slot is in, we recommend running the SNFv5.3.2.7 threads on the same NUMA zone as the snf_open() call, otherwise performance will be degraded as the QPI read penalty will be larger. Three possible scenarios for application handling of packets are:

1. Fast packet processing - In this scenario, packets are read quickly out of the ring and immediately processed. This scenario would take advantage of the ring sizes fitting into the L3 cache size as there would be less probability of the rings getting full.

2. Hold packets - In this scenario, a larger ring size is preferred as the packets would be held in the ring longer as each application does other processing. Care must be taken to ensure that the rings don't fill to capacity, resulting in packet drops.
3. Hold packets with copy - In this scenario, the application would process the packets quickly, as in the Fast packet processing scenario; however, the packets would simply be copied out of the ring to allow other workers to process the data.

16.2.12 CPU isolation

We recommend running the SNFv5.3.2.7 application on a CPU that has been isolated from the kernel scheduler, otherwise the application may be preempted by other OS services, causing a loss of performance.

Isolating the specific CPU from the kernel scheduler binds the SNFv5.3.2.7 application to that CPU.

Benefits

- Prevents OS scheduling policy issues.
- Reduces host-based back pressure caused by packet drops (due to a lack of available data or descriptor pages).

For Grub based Linux systems the `isolcpu` option can be used to accomplish CPU isolation.

CPU isolation is especially important for smaller data and descriptor ring sizes.

For more information, go to the CPU Binding section in this chapter.

16.2.13 Process Priority

When the SNFv5.3.2.7 application cannot be run on a CPU that has been isolated from the kernel scheduler, setting the process to a higher priority can help limit host-based back pressure and give the SNFv5.3.2.7 application more run-time. Run the Linux `nice` or `renice` programs to adjust application priorities.

16.2.14 Blocking Mode

We recommend running the SNFv5.3.2.7 API in non-blocking mode whenever possible. In non-block mode, the application calling the SNFv5.3.2.7 API must poll the interface for the next packet. Calling the SNFv5.3.2.7 API with a receive timeout of zero enters non-blocking mode, thus increasing performance and reducing packet drops.

If you cannot execute the application in non-blocking mode, run the CPU isolation and/or Process Priority tuning steps.

For more information, go to the CPU isolation and/or Process Priority sections in this chapter.
We also recommend that you review the IRQ affinity steps to prevent the IRQs from running on a separate CPU socket from the SNFv5.3.2.7 application.

For more information on IRQ affinity, go to the *Interrupts and IRQ Affinity (Linux)* section in this chapter.
17 Troubleshooting

This chapter describes the following topics:

- Hardware Issues
- Software Installation & System Configuration Issues

Hardware

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

Software/System

If the network appears to be functioning correctly but application transaction rates have not significantly improved after installation of the 10-Gigabit Ethernet Network Adapters and SNFv5.3.2.7 software, it is possible that the proprietary software features are not properly enabled. Only those hosts with valid SNFv5.3.2.7 licenses loaded will demonstrate accelerated performance. It may be necessary to repeat the activation process on all suspect host machines using the latest customer site license file to insure proper status of all network adapters.

17.1 Hardware Installation and Performance Issues

The ARC Series E network adapter is a PCIe Gen3 x8 10-Gigabit Ethernet network adapter. For optimal performance, properly seat the adapter in a PCIe Gen3 x8 expansion slot on the server. The ARC Series E adapter auto-negotiates operation in the widest available mode (x8, x4, x2, or x1) supported by the expansion slot into which it is installed, and at the highest data rate (8, 5, or 2.5 GT/s).

For optimal performance, verify that the adapter reports Gen3 x8 (8 GT/s) PCIe link speed, once it is seated in the PCIe expansion slot on the server.
Two ways to check if the network adapter is properly seated in a Gen3 PCI Express slot:

- Sample `lspci -vvv` output
- Sample `myri_info` output

### 17.1.1 Sample `lspci -vvv` output

For operating systems with the `lspci` command, examine the `lspci -vvv` output to check link speed (\textit{Lnk Sta}).

**Description:**

`lspci -vvv`

**Command Line:**

```
$ sudo lspci -vvv
```

**Output:**

```
LnkSta: Speed 8GT/s, Width x8, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
```

### 17.1.2 Sample `myri_info` output

**Description:**

`myri_info`

**Command Line:**

```
$ sudo ./myri_info -b 0
```
Output:

$ sudo ./myri_info -b 0
pci-dev at 01:00.0 vendor:product(rev)=1c09:4260(01)
    behind bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)
Myri-10G-PCIE-8E -- Link x8
    EEPROM String-spec:
      MAC=00:60:dd:43:2d:e8
      SN=495892
      PC=10G-PCIE3-8E-4S
      PN=09-04680
      BOM=A

    Firmware:
      Version: 2.1.5
      Type : SNF
      Config : 4 Port x 10 Gb
      SHA1 : 2d13f73ad9fe4bd4bda8d7b50dd0ad0b

    External Inputs:
      PPS: Enabled, No Input
      Front Panel PPS: No Input
      Card Edge PPS: No Input
      10Mhz Clock: Disabled
      100Mhz Clock Locked: Locked

pci-dev output results:

- The “.3” notation from (x8.3/x16.3) refers to a PCIe 3.0 Gen3 slot.
- “behind the bridge root-port: 00:01.0 8086:0c01 (x8.3/x16.3)” indicates that the adapter is running at Gen3 x8 speed (maximum capability).
- The motherboard PCIe slot is x16-capable.
- “Myri-10G-PCIE-8E -- Link x8” indicates that the ARC Series E network adapter is running optimally at x8 speed.
17.1.3 LED Issues

Link LED behaviors are not well defined when ARC Series E adapters are disabled or when SNFv5.3.2.7 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 software installation, it is strongly recommended to ping the SNFv5.3.2.7 device before proceeding with testing.

For more information on LED behavior, go to Testing the ARC Series E Adapter Hardware
17.2 Software Installation & System Configuration Issues

Should you encounter problems with SNFv5.3.2.7 software installation, usage, or performance, send the bug report script to ARIA Technical Support. The script output contains the vital information we need to quickly resolve your software issues.

**NOTE:** It is very important that you retrieve the bug report script from the `/opt/snf/sbin` directory, otherwise important diagnostic information may not be collected.

### 17.2.1 Bug report scripts

**Linux**

The `/opt/snf/sbin/phx_bug_report` is a diagnostic script included in the Linux SNFv5.3.2.7 software distribution. The script collects diagnostic information about a customer’s system configuration, such as `uname` output, processor files such as `cpuinfo` and interrupts, `lspci`, kernel messages, `ethtool`, `myri_counters`, and so on. The script must be run as root.

### 17.2.2 Linux RPM-TGZ installation failures

**Linux RPM**

If you encounter errors during the Linux RPM installation process, send the complete output from the RPM command, the kernel log output, and the `/tmp/myri_snf.log` log to ARIA Technical Support.

**Linux TGZ**

If you encounter errors during the Linux TGZ installation, send the complete output from the `sbin/rebuild.sh` log to ARIA Technical Support.

**NOTE:** To build a SNFv5.3.2.7 kernel module, configure the source kernel tree to match the running kernel.

**RedHat example:**
You must install the `kernel-devel` package and the `kernel-headers` package from the RedHat distribution to build the SNFv5.3.2.7 kernel module.
17.2.3 Software Counters (myri_counters)

The myri_counters tool provides low-level SNFv5.3.2.7 hardware and software counters for traffic passing through the network adapter.

**Command line:**

```
$ sudo /opt/snf/bin/myri_counters
```

**For Dual port adapters**

By default, the myri_counters output only displays on dual-port adapters on port 0.

**For Quad-port adapters**

myri_counters output on quad-port adapters depends upon the port in question. If you have a quad-port network adapter installed in the host, you must specify the command line argument `-p <port_num>` to obtain the counters output for each port.

The environment variable `port_num` is an integer value from 0 to n-1, where “n” represents the number of network adapters installed in the host and running the SNFv5.3.2.7 driver, as follows:

**Command lines:**

```
$ sudo /opt/snf/bin/myri_counters -p 0
$ sudo /opt/snf/bin/myri_counters -p 1
```

The space between the “p” and the number is optional.

If a host has two quad-port adapters, use `-p0`, `-p1`, `-p2`, `-p3` to see the ports of the first adapter, and `-p4`, `-p5`, `-p6`, `-p7` to see the ports of the second adapter.

**Clearing the counters**

To clear and reset the counters on a specific port of a network adapter, enter the following:

**Command line:**

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

For a detailed list of command line arguments to myri_counters, go to the bin/myri_counters section of Running SNFv5.3.2.7 Diagnostic Tool Programs.
17.2.4 Numbering of snfX interfaces

If you have multiple quad-port network adapters per server, run the **myri_nic_info** utility to monitor the numbering and assigning of **snfX** interfaces to the installed adapters. The utility also lists the **snfX** interfaces' corresponding MAC addresses.

Refer to the following **myri_nic_info** output, where the board number corresponds to the **snfX** interface number. For example, adapter 4 is **snf4** in **libpcap**.

**Command lines:**

```
$ sudo /opt/snf/bin/myri_nic_info -B
$ sudo /opt/snf/bin/myri_nic_info
```

**Output:**

```
# Serial MAC     ProductCode     Driver     Version          License
0 495604 00:60:dd:43:2f:f8 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
1 495604 00:60:dd:43:2f:f9 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
2 495604 00:60:dd:43:2f:fa 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
3 495604 00:60:dd:43:2f:fb 10G-PCIE3-8E-4S myri_snf-5.3.2.7.54367 Valid
```

17.2.5 Performance Tuning and Packet Drops

If the network adapter has insufficient receive buffers on the host, all subsequent packets will be dropped until the application can provide more host buffers. SNFv5.3.2.7 supports very large receive rings but these large buffers are useful only if the application can, on average, sustain the incoming packet rate. While the network adapter can support a worst-case scenario of 14.88 Mpps, it can also be the source of dropped packets in some less common cases. The following points may help to address the issue.

**The libpcap library is not SNF-aware**

Ensure that the SNFv5.3.2.7 software is being used by setting **SNF_DEBUG_MASK=0x3** and opening the **snfX** device when invoking the application. This option causes the SNFv5.3.2.7 library to dump memory mapping information when the device is open. If no information outputs to the screen, it is likely that your application is linked against a version of **libpcap** that is not SNF-aware.

**PCIe expansion slot issues**

Verify that the network adapter is installed into a PCIe expansion slot that can sustain 10Gbit/s traffic.
Traffic issues

Monitor **myri_counters** to verify that traffic is in fact being received. For each network adapter, check the runtime counter values as follows:

1. First, examine the **Packets Received (all ports)** and **Packets Rx (this port)** counter values to verify that traffic is going directly to a consumer that uses the SNF API. If this count is zero, packets are probably being passed to the regular (much slower) kernel stack. Check to see if **libpcap** is SNF-aware with the **SNF_DEBUG_MASK=0x3** debugging option noted above.

2. Next, examine the **XMAC RX pktsAbort**, **XMAC RX pktsDropped**, **XMAC RX pktsNoMeta**, **XMAC RX pktsNoOutCtrl**, **XMAC RX pktsDropAbort**, and **PCIE FIFO * values** for traffic the network adapter and stack cannot sustain. Use **myri_counters -x** to see these low-level, detailed counters. These counters are zero in normal operation. If a non-zero value is reported for any of these counters, contact **ARIA Technical Support**.

**17.2.6 Network Adapter Timestamps**

SNFv5.3.2.7 supports socket timestamps on ARC Series E network adapters.

The timestamp is made available through the socket interface. The socket interface, running the **SO_TIMESTAMP[NS]** socket option, is a high-precision clock synchronized at startup to system host time (returned by the **gettimeofday()** function call). The timestamp automatically attaches to the packet upon arrival.

Linux users can access the clock as a POSIX clock or through the PTP protocol.

| ATTENTION: | Use caution when running NTP services (that rely on network time protocols) on the POSIX clock. NTP services cause variations in system host times that may exceed inter-packet arrival times. |
17.2.7 Synchronization

There are different levels of timestamp synchronization.
- Network Adapter-to-host synchronization (local sync)
- Host-to-host synchronization (global sync)

Network adapter-to-host synchronization (local sync)

Use the **phc2sys** Linux tool to run Network Adapter-to-host synchronization. Set the ARC Series E network adapter to “Master” clock, for higher resolution and accuracy (recommended).

Host-to-host synchronization (global sync)

Most systems use **NTPD** to run host-to-host synchronization (global sync). More precise options to **NTPD** exist; however, it only raises the accuracy level from milliseconds to tens of microseconds. Since both protocols are not typically run on dedicated networks and because there is typically a lot of host overhead in processing the protocol, the time can only be so accurate.

The ARC Series E network adapters connect externally to a PPS or a 10MHz input to improve adapter clock accuracy. When loading the driver, set the module variable inputs as follows:

```
"myri_clk_enable_pps=1"
"myri_clk_enable_10mhz=1"
```
Appendix 1. SNFv5.3.2.7 Counters

To download SNFv5.3.2.7 hardware and software counter values from the time the driver was loaded (or since the counters were reset), use the `myri_counters` tool.

**Definition:**

myri_counters

**Command line [Linux]:**

$ sudo /opt/snf/bin/myri_counters

Multi-port adapters appear in `myri_counters` as different ports. If you have a dual-port or quad port network adapter installed in the host, you must specify the command line argument `-p <port_num>` to obtain the counters output for only one port.

**Example:**

$ sudo /opt/snf/bin/myri_counters -p 0
$ sudo /opt/snf/bin/myri_counters -p 1

The space between the "p" and the number is optional.

**Command line [Help]:**

$ sudo /opt/snf/bin/myri_counters -h

**Usage:**

```
Usage: myri_counters [args]
-p N - Port number or Ethernet MAC
-b N - Port number or Ethernet MAC
-c - clear the counters
-q - quiet: show only nonzero counters
-i - show host interrupt counters
-x - expert: show all counters
-o - show register offset
-r - raw: show register contents
-e N - show counters for specified endpoint [0]
-a - show counters for all endpoints
-v - show all counters
-F - show filter state including all registered filters
-M - show MAC filters, if available
-h - help
```
Clearing counters

To clear the counters on a specific port of a network adapter, enter the following command line:

**Command line:**

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

To clear and/or reset the counters requires root privileges.
List of SNFv5.3.2.7 Counters

The following is a detailed list of SNFv5.3.2.7 software and hardware counters.

Cycle Count
The FPGA cycle count from the time the SNFv5.3.2.7 driver was loaded.

Packets Received
The number of packets received by the FPGA and transferred to the host memory.

Bytes Received
The number of bytes in the packets received by the FPGA and transferred to the host memory.

Packets Finalized
The number of packets received by the FPGA and the FPGA has transferred the descriptor to the host.

PCIE FIFO empty
The number of clock cycles for which the Rx PCIe FIFO was empty.

PCIE FIFO < ¼
The number of clock cycles for which the Rx PCIe FIFO was less than one quarter full.

PCIE FIFO ¼ - ½
The number of clock cycles for which the Rx PCIe FIFO was between one quarter and one half full.

PCIE FIFO ½ - ¾
The number of clock cycles for which the Rx PCIe FIFO was between one half and three quarters full.

PCIE FIFO > ¾
The number of clock cycles for which the Rx PCIe FIFO was between three quarters and full.

PCIE FIFO full
The number of clock cycles for which the Rx PCIe FIFO was full. These are cycles for which PCIe is blocking Rx.
RX Time Limit
The number of cycles that the RX hardware will wait for more packets to arrive before sending up partial TLP transactions.

APU version
Developer use only.

APU configuration
Developer use only.

RX Cycle Count
Continuous running count of cycles.

RX Data Pages
Number of pages in the receive data buffer.

RX Desc Pages
Number of pages in the receive descriptor buffer.

RX Data Pages Available
Number of free pages in the receive data buffer.

RX Desc Pages Available
Number of free pages in the receive data buffer.

RX Data Pages Available Min
RX Desc Pages Available Min
Indicates the minimum number of available data and descriptor pages. The lower these page values are, the more likely the adapter hardware will exhibit transmission problems.

RX Data Consumer Position
Host position in the receive data buffer.

RX Data Producer Position
FPGA position in the receive data buffer.

RX Desc Consumer Position
Host position in the receive descriptor buffer.
RX Desc Producer Position
FPGA position in the receive descriptor buffer.

RX Descriptors Created
Descriptors that have been processed by the host.

RX Wake Request
Used for interrupt generation.

RX Data TLP Q1
RX Data TLP Q2
RX Data TLP Q3
RX Data TLP Q4
RX Data TLP Full
Histogram of Rx Data DMA performance.

RX Desc TLP Q1
RX Desc TLP Q2
RX Desc TLP Q3
RX Desc TLP Q4
RX Desc TLP Full
Histogram of Rx Descriptor DMA performance.

RX Data MWr/s Min
RX Data MWr/s Max
RX Data Full MWr/s Min
RX Data Full MWr/s Max
RX Data MWr DW/s Min
RX Data MWr DW/s Max
RX Data MWr PLDW/s Min
RX Data MWr PLDW/s Max
Read Data buffer performance min/max statistics.
RX Desc MWr/s Min
RX Desc MWr/s Max
RX Desc Full MWr/s Min
RX Desc Full MWr/s Max
RX Desc MWr DW/s Min
RX Desc MWr DW/s Max
RX Desc MWr PLDW/s Min
RX Desc MWr PLDW/s Max

Read Descriptor buffer performance min/max statistics.

TX Time Limit
The number of cycles that the TX hardware will wait for more packets to be queued for sending, before starting partially full PCIe transactions.

TX Byte Count
The number of bytes in the packets sent by the FPGA.

TX Packet Count
The number of packets sent by the FPGA.

TX Completion Page LSW
TX Completion Page MSW
The two registers contain the page DMA address used for flow control on the TX portion of the SNF hardware.

TX Completion Count
Used to handshake with the hardware regarding TX completions.

TX Completion Count ACK
Used to handshake with the hardware regarding TX completions.

TX Data MRd Q1
TX Data MRd Q2
TX Data MRd Q3
TX Data MRd Q4
TX Data MRd Full
Histogram of TX Data DMA performance.
TX Desc MRd Q1
TX Desc MRd Q2
TX Desc MRd Q3
TX Desc MRd Q4
TX Desc MRd Full
Histogram of TX Descriptor DMA performance.

TX Data CplD Q1
TX Data CplD Q2
TX Data CplD Q3
TX Data CplD Q4
TX Data CplD Full
Histogram of TX Data Completion ID performance.

TX Desc CplD Q1
TX Desc CplD Q2
TX Desc CplD Q3
TX Desc CplD Q4
TX Desc CplD Full
Histogram of TX Descriptor Completion ID performance.

TX Data MRd/s Min
TX Data MRd/s Max
TX Data Full MRd/s Min
TX Data Full MRd/s Max
TX Data MRd DW/s Min
TX Data MRd DW/s Max
TX Data read performance min/max information.
TX Desc MRd/s Min
TX Desc MRd/s Max
TX Desc Full MRd/s Min
TX Desc Full MRd/s Max
TX Desc MRd DW/s Min
TX Desc MRd DW/s Max
TX Descriptor read performance min/max information.

TX Data CplD/s Min
TX Data CplD/s Max
TX Data Full CplD/s Min
TX Data Full CplD/s Max
TX Data CplD DW/s Min
TX Data CplD DW/s Max
TX Data completion read performance min/max information.

TX Desc CplD/s Min
TX Desc CplD/s Max
TX Desc Full CplD/s Min
TX Desc Full CplD/s Max
TX Desc CplD DW/s Min
TX Desc CplD DW/s Max
TX Descriptor completion read performance min/max information.

TX Data Ring Size (4 KB Pages)
The number of 4 KB pages for the host Transmit Data ring.

TX Data Ring ConsR Position
The position in the Data ring where the FPGA is pulling data.

TX Desc Ring Size (Descriptors)
The number of descriptors that can be held in the descriptor ring buffer.

TX Desc Ring ProdW Position
The descriptor index that is being written by the host.
**TX Desc Ring ConsR Position**
The descriptor index that is being read by the FPGA.

**XMAC RX pktsRcvd**
Incoming packet count at the XMAC RX core.

**XMAC RX pktsAbort**
Packets that get aborted in the XMAC RX core

**XMAC RX pktsDropped**
A count of receive packets dropped by the MAC.

**XMAC RX pktsNoMeta**
Packets dropped by XMAC RX due to no meta data.

**XMAC RX pktsNoOutCtrl**
Packets dropped by XMAC RX due to no output control.

**XMAC RX pktsDropAbort**
Packets dropped by XMAC RX due to other reasons.

**XMAC RX pktsSent**
Packets forwarded by the XMAC Rx core.

**XMAC RX bytesSent**
Bytes in packets forwarded by the XMAC Rx core.

**XMAC RX SendDup**
Should always be 0.

**XMAC RX SentWithErr**
Should always be 0.

**XMAC RX ExtraMeta**
Should always be 0.

**XMAC RX CtrlDropped**
Packets dropped due to control metadata FIFO full.
XMAC RX Discarded
Should always be 0.

XMAC RX Bytes
XMAC TX Bytes
The number of bytes in valid frames seen at the MAC.

XMAC RX Frames
XMAC TX Frames
The number of valid frames seen at the MAC.

XMAC RX Undersized Frames
Frames that were less than 64 bytes.

XMAC TX Fragmented Frames
Frames that were transmitted as runts due to underrun.

XMAC RX 64B Frames
XMAC RX 65-127B Frames
XMAC RX 128-255B Frames
XMAC RX 256-511B Frames
XMAC RX 512-1023B Frames
XMAC RX 1024-Max Frames
XMAC RX Oversized Frames
The number of received frames in various sizes as seen at the MAC.

XMAC TX 64B Frames
XMAC TX 65-127B Frames
XMAC TX 128-255B Frames
XMAC TX 256-511B Frames
XMAC TX 512-1023B Frames
XMAC TX 1024-Max Frames
XMAC TX Oversized Frames
The number of transmitted frames in various sizes as seen at the MAC.
**XMAC RX FCS Errors**
The count of frames received with FCS errors

**XMAC RX Broadcast Frames**
The count of correctly received broadcast frames

**XMAC RX Multicast Frames**
The count of correctly received multicast frames

**XMAC RX MAC CTRL Frames**
The count of correctly received MAC control frames

**XMAC RX Length/Type Range Error**
A count of error-free frames received that were at least 64 bytes in length where the length/type field contained a length value that did not match the number of MAC client data bytes received.

The counter also increments for frames in which the length/type field indicated that the frame contained padding but where the number of MAC client data bytes received was greater than 64 bytes

**XMAC RX VLAN Frames**
The count of correctly received VLAN frames

**XMAC RX Pause Frames**
The count of correctly received MAC control PAUSE frames

**XMAC RX MAC Unsupported Frames**
The count of correctly received MAC control frames that were not PAUSE frames

**XMAC TX Broadcast Frames**
The count of correctly transmitted broadcast frames

**XMAC TX Multicast Frames**
The count of correctly transmitted multicast frames

**XMAC TX Underrun Frames**
The count of transmitted frames where there was an underrun error at the MAC
RSS Mode

RSS Mask
These counters display the register setup values that control RSS distribution of packets to rings.

RSS Rx Packets
The number of packets that have arrived at the RSS core.

RSS Rx Bytes
The number of bytes in the packets that have arrived at the RSS core.

RSS Dropped Packets
The number of packets that have been dropped by the RSS core. Packets are dropped because they are arriving faster than the host is reading packets from the adapter.

RSS Dropped Bytes
The number of bytes in the packets that have been dropped by the RSS core.

RSS Error Packets
The number of packets that have errors and arrive at the RSS core. Packets are deemed to have errors if they are flagged by the MAC as having an FCS error. These packets are received and discarded by the SNF library.

RSS Error Bytes
The number of bytes in the packets that the RSS core has determined to have errors.

RSS Dispatcher Dropped Packets
The number of packets dropped out of the RSS before being dispatched to the next ring.

RSS Dispatcher Forwarded Packets
The number of packets forwarded out of the RSS to the next available ring, or distributed to different rings by way of FPGA port merging.
Appendix 2. Operating Systems and Hardware Support

Software Support

Linux support

- CentOS 8.0 is recommended.
- CentOS 7.7 is supported.
- For non-RPM-based Linux distributions, a .tgz driver is provided and supported up to Linux kernel version 5.5.

Hardware Support

Network Adapters:

ARC Series E (10G-PCIE3-8E-2S) network adapter
Part number: 09-04669

ARC Series E (10G-PCIE3-8E-4S) network adapter
Part number: 09-06480

Processors:

- Intel Haswell-E processors are recommended.
- Testing has been performed on i7-4790K, i7-5960X, i7-6700K, E5-2603, E5-2658, E5-2699, and E5-3699 processors.

Motherboards

- The following motherboards are compatible:
  - Asus Z97-Deluxe
  - Asus X99-WS/IPMI
  - Dell PowerEdgeR630
  - HP 2440 motherboards.
- Other HP Servers that are not recommended:
  - HP Z840 Workstation
  - HP G7 ProLiant servers
  - HP ProLiant Gen9 servers
Appendix 3. SNFv5.3.2.7 Driver Restrictions and Limitations

The SNFv5.3.2.7 driver restrictions and limitations are as follows:

- The SNFv5.3.2.7 release does not support Windows.
- SNF: Port merge combinations: Port merging is only permitted between two ports. You can merge ports 0 & 1, or ports 2 & 3. You cannot merge other port combinations such as ports 1 & 2.
- NUMA awareness: For best performance, all receive operations should be assigned to the single NUMA zone closest to the PCIe slot where the adapter is installed. Accessing from a socket CPU across QPI to a different NUMA zone may incur higher CPU utilization and dropped packets. The application must insure it runs from the NUMA zone CPUs where the rings/buffers are allocated to ensure no packet drops.
- The ARC Series E adapters only support DAC cables three meters in length or less. Fiber cabling and SFP+ transceivers are recommended for cables measuring more than three meters.
- Running `tcpdump -D` (lists available network devices) may not display the `snf0` device when `SNF_DEBUG_MASK=0`, even though the device is operational and can be referenced. Setting `SNF_DEBUG_MASK=3` does not display any devices, yet all are operational and can be referenced.
- SNFv5.3.2.7 FPGA firmware must match up to the proper 1G/10G transceiver. There are instances where 1G firmware may work with 10G transceivers but this practice is neither recommended nor supported. Run `myri_info` to verify firmware type and support. Refer to Appendix 4: SNFv5.3.2.7 Firmware of the SNFv5.3.2.7 User Guide for more information.
- The `snf/bin/tests/sniffex` test program is not supported. It will be removed in a future release.
Known Issues

- (ID# 174) PF_RING port aggregation is using only one CPU and may drop packets.
- (ID# 147, 153) Arista switch timestamping is not yet supported. There is no support for keyframes or timestamped packets.
- (ID# 181) `myri_endpoint_info` does not display the physical receive endpoint in use by the current port. It only displays physical endpoints in use by other ports.
- (ID# 387) FCS errors may occur on an adapter with 1G firmware when you `Ctrl-C` a receiving application.
- (ID# 388) The `snf_basic_diags` utility may intermittently fail due to port merging timing issues.
- (ID# 375) Entering `Ctrl-C` to terminate a Tx transmit application (`snf_inject_open()` function) may cause a server hang, with the warning, "myri_snf WARN: SnifferTx still not flushed after 30 msec". Reboot or power-cycle the server if the hang persists. We also suggest invoking `snf_inject_close` in the case of repeated lockups.
- (ID# 442) Port merging, on a quad-port card with 1G/10G firmware, fails for larger-sized packets (over 1500 bytes) and drops packets. Terminating the Rx receiver application prompts the warning, "myri_snf WARN: SnifferTX still not flushed after 30 msec", in `dmesg` or `/var/log/messages`. Reboot the server if the hang persists.
- (ID# 444) A segmentation fault may occur when port merging between multiple adapters on the same server.
- (ID# 374) `snf_replay -Z -N` (playback software pacing) runs slow (in microseconds). These two flags should not be used together.
## Appendix 4: SNFv5.3.2.7 Firmware

There are a variety of SNF-5.3.2.7 firmware downloads that support 1Gbit and 10Gbit speeds (Table 8).

<table>
<thead>
<tr>
<th>Firmware Prefix</th>
<th>Supported HW</th>
<th>Port 0 speed</th>
<th>Port 1 speed</th>
<th>Port 2 speed</th>
<th>Port 3 speed</th>
<th>Vendor Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>fw-8E-2S-SNF_1G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-2S</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>N/A</td>
<td>N/A</td>
<td>1c09:4265</td>
</tr>
<tr>
<td>fw-8E-2S-SNF_10G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-2S</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>N/A</td>
<td>N/A</td>
<td>1c09:4264</td>
</tr>
<tr>
<td>fw-8E-4S-SNF_1G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-4S</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>1c09:4263</td>
</tr>
<tr>
<td>fw-8E-4S-SNF_1G10G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-4S</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>1c09:4262</td>
</tr>
<tr>
<td>fw-8E-4S-SNF_10G1G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-4S</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>1 Gbit/sec</td>
<td>1c09:4261</td>
</tr>
<tr>
<td>fw-8E-4S-SNF_10G-&lt;x.x.x.&gt;</td>
<td>10G-PCIE3-8E-4S</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>10 Gbit/sec</td>
<td>1c09:4260</td>
</tr>
</tbody>
</table>

Table 8: SNFv5.3.2.7 firmware prefixes by adapter port configuration and speed.

**Legend:**

- **fw**: firmware
- **8E**: ARC Series E network adapter class
- **2S/4S**: dual-port or quad-port configuration
- **SNF**: product class
- **1G**: port speed in Gbit/sec
- **10G**: port speed in 10Gbit/sec
- **<x.x.x.>**: firmware version appended to the prefix
Appendix 5: SNFv5.3.2.7 Supported 1G and 10G Transceivers

The following 1G and 10G transceivers have been tested and are supported to run with ARC Series E (10G-PCIE3-8E-4S and 10G-PCIE3-8E-2S) network adapters.

Supported 1G transceiver modules

<table>
<thead>
<tr>
<th>Model Number</th>
<th>Speed</th>
<th>Connector</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTLX8574D3BCV</td>
<td>1G &amp; 10G</td>
<td>MM Fiber (fiber optic cable)</td>
<td>Can run at 1 Gb or 10 Gb</td>
</tr>
<tr>
<td>FTLF8519P3BNL</td>
<td>1G</td>
<td>MM Fiber (fiber optic cable)</td>
<td>1000Base-SX</td>
</tr>
<tr>
<td>FCLF8521P2BTL</td>
<td>1G</td>
<td>TXRX Copper SFP</td>
<td>1000BASE-T 100M</td>
</tr>
<tr>
<td>FCLF8522P2BTL</td>
<td>1G</td>
<td>TXRX Copper SFP</td>
<td>1000BASE-T</td>
</tr>
<tr>
<td>FTLF1318P2BTL</td>
<td>1G</td>
<td>SM Fiber (fiber optic cable)</td>
<td>1000BASE-LX Not tested.</td>
</tr>
<tr>
<td>ABCU-5730RZ</td>
<td>1G</td>
<td>TXRX RJ45 Copper SFP</td>
<td>Does not advertise connector type</td>
</tr>
<tr>
<td>ABCU-5740RZ</td>
<td>1G</td>
<td>TXRX RJ45 Copper SFP</td>
<td>Does not advertise connector type</td>
</tr>
<tr>
<td>FCLF-8520-3</td>
<td>1G</td>
<td>TXRX Copper SFP</td>
<td>Does not advertise connector type</td>
</tr>
</tbody>
</table>

Table 9: Supported 1G transceiver modules by speed and connector type.
## Supported 10G transceiver modules

<table>
<thead>
<tr>
<th>CSPI Model Number</th>
<th>Description</th>
<th>Software Releases verified in</th>
</tr>
</thead>
<tbody>
<tr>
<td>10G-SFP-SR</td>
<td>Optical-Fiber SFP+ transceiver for 10GBase-SR (850nm wavelength)</td>
<td>SNF Release Versions 2, 3, and 5</td>
</tr>
<tr>
<td>10G-SFP-LR</td>
<td>Optical-Fiber SFP+ transceiver for 10GBase-LR (1310nm wavelength)</td>
<td>SNF Release Versions 2, 3, and 5</td>
</tr>
<tr>
<td>10G-XFP-SR</td>
<td>Optical-Fiber XFP transceiver for 10GBase-SR (850nm wavelength)</td>
<td>SNF Release Versions 2, 3, and 5</td>
</tr>
<tr>
<td>10G-XFP-LR</td>
<td>Optical-Fiber XFP transceiver for 10GBase-LR (1310nm wavelength)</td>
<td>SNF Release Versions 2, 3, and 5</td>
</tr>
</tbody>
</table>

Table 10: Supported 10G transceiver modules.
Appendix 6: Network Adapter Toolkit - v. 1.40

The PHX-TOOLS Network Adapter Toolkit release (version 1.40) allows users to run diagnostics on ARC Series E network adapter operation and flash memory FPGA firmware programming.

The PHX-TOOLS toolkit version 1.40 describes the individual features, bug fixes, and limitations. We recommend that users migrate to this release at their earliest convenience.

Hardware Support

1. Refer to the README file in the PHX-TOOLS Network Adapter Toolkit for installation and configuration instructions.

2. The toolkit currently supports the following network adapter format:
   - 10G-PCIE3-8E-2S with two SFP+ cages
   - 10G-PCIE3-8E-4S with four SFP+ cages

3. The firmware must match the adapter model and transceiver type. For more information, refer to Appendix 4: SNFv5.3.2.7 Firmware.

4. Several models of 1G and 10G transceivers are supported. For more information, refer to Appendix 5: SNFv5.3.2.7 Supported 1G and 10G Transceivers.

5. The toolkit is compatible with Windows Server 2008R2 and later, and most Linux distributions.

New Features and Enhancements

1. (ID# 369) SNF: Release of firmware version 2.1.5 for ARC Series E adapters.

2. (ID# 74) SNF: ARC Series E- adapters have been updated with an improved oscillator and resistor. These hardware updates reduce FCS errors and improve timestamp accuracy. The updates also improve performance when using 1 Gb firmware/transceivers.

3. (ID# 166) SNF: Transmit injection pacing is now supported.

4. (ID# 282) SNF: Interrupts are now generated to detect the insertion or removal of SFP+ transceivers.

5. (ID# 211) SNF: Improved packet type coverage for RSS hash.
   a. (ID# 239) Packets are now spread among rings when using a varying destination port with a fixed source port.
   b. (ID# 248) GRE packets with fixed IPs and ports now land in one ring.
c. (ID# 249) GTP-U packets with varying TEIDs & fixed IPs and ports are now spread among rings.

6. (ID# 397) Added acceleration support for port pair merging. For more information, go to the Port Merging section of Load Balancing and Port Merging Features

Bug Fixes

1. (ID# 332) SNF: myri_info -v will now display the correct firmware build date.

2. (ID# 345) SNF: Fixed hang that occurred when entering CTRL-C while an application was transmitting data.

3. (ID# 257) SNF: myri_info -h will now display the correct version number of the toolkit release.

4. (ID# 364) SNF: Fixed slow throughput that occurred with quad-port 1G firmware.

5. (ID# 305) SNF: Fixed a case where drops would occur when receiving successive packets that were sequentially sized.

   Example: Receiving packet sizes: 64, 65, 66...

6. (ID# 379) SNF: Fixed problem that caused TX traffic on port 0 to stop when a cable was removed from port 1.

7. (ID# 380) SNF: Fixed problem that caused the first packet received to be dropped following a reboot or power-cycle.

Limitations

None

Known issues

1. (ID# 437) ARC Series E adapters running 1G firmware may run into a situation where they cannot receive packets, prompting the following dmesg error log: "myri_snf WARN: RX endpoint is not flushed". You must reboot to correct the problem.
Technical Support:

If there are any problems installing or using ARIA Cybersecurity Solutions products, or if any bugs or possible enhancements are noticed, do not hesitate to contact ARIA Technical Support.

Contact Technical Support via the ARIA Customer Portal *
https://www.ariacybersecurity.com/support/downloads/

ARIA Cybersecurity Solutions website:
https://www.ariacybersecurity.com/network-adapters/

ARIA email support at ARIA_support@ariacybersecurity.com

Before you contact our technical support staff, have the following information available:

- Your name, title, company name, phone number, and email address
- Operating system and version number
- Product name and release version
- Problem description

* Follow the instructions on the ARIA Customer Portal website to register for an ARIA Customer Support account
## Glossary

<p>| <strong>Bro</strong> | An open source based Intrusion Detection System (IDS) package that interfaces with SNFv5.3.2.7 through <strong>libpcap</strong>. |
| <strong>CentOS</strong> | A third-party Linux distribution. |
| <strong>Descriptor ring</strong> | Stores packet metadata such as timestamp and length in main memory. |
| <strong>DMA</strong> | Direct memory access. |
| <strong>DST</strong> | Destination address. |
| <strong>Dual-port</strong> | An adapter with two ports. |
| <strong>EEPROM</strong> | Electrically erasable programmable read-only memory |
| <strong>Endpoint</strong> | A connection point where HTML files or active server pages are exposed. An example of an endpoint is a PC. |
| <strong>ESD</strong> | Electrostatic Discharge. |
| <strong>Fedora</strong> | A third-party Linux distribution. |
| <strong>Firmware</strong> | Permanent software programmed into a read-only memory. |
| <strong>FPGA</strong> | Field Programmable Gate Array |
| <strong>Gen2</strong> | A second generation of PCIe standard. |
| <strong>Gen3 x16</strong> | A third generation of PCIe standard with 16 lanes. |
| <strong>Gen3 x8</strong> | A third generation of PCIe standard with 8 lanes. |
| <strong>GNU</strong> | Generally Not Unix |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPRS Tunneling Protocol</td>
<td>GPRS Tunneling Protocol (GTP) is a group of IP-based communications protocols used to carry General Packet Radio Service (GPRS) within cellular networks.</td>
</tr>
<tr>
<td>GRE Protocol</td>
<td>Generic Routing Encapsulation protocol, used to encapsulate packets of one network layer protocol over another network layer protocol.</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>ifcfg</td>
<td>A configuration file that defines parameters used to configure a network adapter.</td>
</tr>
<tr>
<td>isolcpus</td>
<td>Isolate CPUs from the kernel scheduler.</td>
</tr>
<tr>
<td>IRQ</td>
<td>Interrupt Request</td>
</tr>
<tr>
<td>IRQ Affinity</td>
<td>Defined as the set of CPU cores that can service an interrupt request.</td>
</tr>
<tr>
<td>Irqbalance</td>
<td>A Linux daemon that balances the CPU load generated by interrupts across all CPUs.</td>
</tr>
<tr>
<td>ISR</td>
<td>Interrupt Service Routine</td>
</tr>
<tr>
<td>ldd</td>
<td>List Dynamic Dependencies</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>Libpcap</td>
<td>Library used by programs to perform packet capture.</td>
</tr>
<tr>
<td>Linuxptp</td>
<td>Implementation of Precision Time Protocol on Linux.</td>
</tr>
<tr>
<td>MAC</td>
<td>A network adapter Media Access Controller address.</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multi-Protocol Label Switching</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>MSI</td>
<td>Message Signaled Interrupt</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>Ntpd</td>
<td>Network Time Protocol daemon. A program which uses Network Time Protocol (NTP) to keep system time synchronized with time servers.</td>
</tr>
<tr>
<td>NUMA</td>
<td>Non Uniform Memory Architecture</td>
</tr>
<tr>
<td>Numactl</td>
<td>A program used to monitor memory usage policy.</td>
</tr>
<tr>
<td>PATH</td>
<td>An environment variable used by a system to access programs without having to specify their path</td>
</tr>
<tr>
<td>Pcap</td>
<td>Packet capture</td>
</tr>
<tr>
<td>PCIe</td>
<td>Peripheral Component Interconnect (Express)</td>
</tr>
<tr>
<td>phc2sys</td>
<td>A Linux program used to run adapter-to-host synchronization.</td>
</tr>
<tr>
<td>PHX-TOOLS</td>
<td>A toolkit that runs diagnostics on ARC Series E network adapter operation and flash memory FPGA firmware programming.</td>
</tr>
<tr>
<td>POSIX clock</td>
<td>POSIX function used to determine CPU time consumed by a process</td>
</tr>
<tr>
<td>PPS</td>
<td>Pulse Per Second</td>
</tr>
<tr>
<td>PTP Hardware Clock</td>
<td>The ARC Series E network adapter clock.</td>
</tr>
<tr>
<td>PTP Protocol</td>
<td>Point-To-Point protocol</td>
</tr>
<tr>
<td>QPI</td>
<td>QuickPath Interconnect</td>
</tr>
<tr>
<td>Quad port</td>
<td>An adapter with four ports.</td>
</tr>
</tbody>
</table>
**RHEL**  
Red Hat Enterprise Linux (RHEL) is a Linux distribution developed by Red Hat.

**RMA**  
Return Merchandise Authorization

**RPM**  
RedHat Package Manager

**RSS**  
Receive Side Scaling

**SDK**  
Software Development Kit

**SFP+**  
Small Form-Factor Pluggable transceiver, “+” symbol indicates 10Gb.

**Snort**  
An open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks.

**SRC**  
Source address

**Suricata**  
An open source-based Intrusion Detection System (IDS) that interfaces with SNFv5.3.2.7 through libpcap

**SW**  
Software

**Taskset**  
A program used to set or retrieve a task’s CPU affinity

**TCP**  
Transmission Control Protocol

**Tcpdump**  
An open source packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

**Tcpreplay**  
An open source utility that edits and replays previously captured network traffic. Tcpreplay was originally designed to replay malicious traffic patterns to Intrusion Detection/Prevention Systems.
<table>
<thead>
<tr>
<th><strong>UDP</strong></th>
<th>User Datagram Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ubuntu</strong></td>
<td>An open source operating system based on the Debian Linux distribution.</td>
</tr>
<tr>
<td><strong>VM</strong></td>
<td>Virtual Machine</td>
</tr>
<tr>
<td><strong>Wireshark</strong></td>
<td>An open source packet analyzer, used for network troubleshooting, analysis, software and communications protocol development. Similar to <code>tcpdump</code> except that it supports a Graphical User Interface (GUI).</td>
</tr>
</tbody>
</table>