Dolphin eXpressWare
Installation and Reference Guide
For users of  SISCI , SuperSockets, IPoPCIe
Linux version

Dolphin Interconnect Solutions
This document describes the installation and usage of Dolphin PCI Express software stack (eXpressWare) version 5.4.x in combination with Curtiss-Wright PCIe enabled SBCs and backplanes running Linux. Please select the VxWorks guide for information on installing eXpressWare for VxWorks 6.9 and 7.

Publication date February 28th, 2017
Copyright © 2017 Dolphin Interconnect Solutions
Published under Gnu Public License v2
## Table of Contents

Abstract .................................................................................................................................................. ix

1. Introduction & Overview ................................................................................................................................. 1
   1. Who needs a PCI Express Network? ........................................................................................................... 1
   2. PCI Express networking basics .................................................................................................................... 1
   3. Contents of this Document .......................................................................................................................... 2
   4. Terminology .................................................................................................................................................. 2
   5. Support ......................................................................................................................................................... 4

2. Quick Installation Guide ................................................................................................................................ 5

3. Requirements and Planning .............................................................................................................................. 7
   1. Supported Platforms ................................................................................................................................... 7
      1.1. Application recommendations ............................................................................................................... 7
      1.2. Supported Platforms ................................................................................................................................ 7
      1.3. eXpressWare adapter names ................................................................................................................ 7
      1.4. Adapter serial numbers ......................................................................................................................... 8
      1.5. Recommended Cluster Node Hardware ............................................................................................... 8
      1.6. Recommended Cluster Management Hardware .................................................................................... 8
   2. Software Support ......................................................................................................................................... 8
      2.1. Linux ...................................................................................................................................................... 8
      2.2. VxWorks .............................................................................................................................................. 9
   3. Interconnect Planning ................................................................................................................................... 9
      3.1. Nodes to Equip with PCI Express Interconnect .................................................................................... 9
      3.2. PCI Express Interconnect Topologies ................................................................................................ 9

4. Initial Installation ........................................................................................................................................... 10
   1. Installation Overview .................................................................................................................................... 10
      1.1. Requirements ....................................................................................................................................... 10
      1.2. Installation Variants ............................................................................................................................. 11
         1.2.1. Live Installation ........................................................................................................................... 11
         1.2.2. Non-GUI Installation ..................................................................................................................... 11
      1.3. System installation overview ................................................................................................................ 12
      1.4. Installation Result .................................................................................................................................. 12
   2. Software Installation .................................................................................................................................... 13
      2.1. Overview ............................................................................................................................................... 13
      2.2. Starting the Software Installation ....................................................................................................... 13
      2.3. Post installation ................................................................................................................................... 17
      2.4. Working with the Dolphin Network Configurator, dis_netconfig ...................................................... 17
         2.4.1. Cluster Edit ................................................................................................................................. 17
         2.4.2. Node Arrangement ........................................................................................................................ 19
      2.5. Finalising the Software Installation ..................................................................................................... 22
         2.5.1. Static Connectivity Test .............................................................................................................. 22
         2.5.2. SuperSockets Configuration Test ............................................................................................... 23
         2.5.3. SuperSockets Performance Test ................................................................................................. 23
      2.6. Handling Installation Problems ........................................................................................................... 24
      2.7. Interconnect Validation using the management GUI .......................................................................... 24
         2.7.1. Installing dis_admin ..................................................................................................................... 24
         2.7.2. Starting dis_admin ....................................................................................................................... 24
         2.7.3. Cluster Overview ........................................................................................................................ 25
         2.7.4. PCIe connection Test ................................................................................................................... 26
      2.8. Interconnect Validation using the command line .................................................................................. 26
      2.9. Making Cluster Application use PCI Express ..................................................................................... 28
         2.9.1. Generic Socket Applications ..................................................................................................... 28
         2.9.2. Kernel Socket Services ............................................................................................................... 28
         2.9.3. Native SISCI Applications .......................................................................................................... 28

5. Update Installation ......................................................................................................................................... 29
   1. Complete Update ....................................................................................................................................... 29
   2. Rolling Update .......................................................................................................................................... 30
Dolphin eXpressWare Installation and Reference Guide

2.2.2. sockperf ................................................................................. 51
3. Troubleshooting ................................................................. 51
4. SuperSockets Utilization .................................................. 54
5. Kernel Socket Services ....................................................... 55
6. Command Reference ......................................................... 55
11. Advanced Topics ............................................................. 59
   1. Notification on Interconnect Status Changes ....................... 59
      1.1. Interconnect Status .................................................. 59
      1.2. Notification Interface .............................................. 59
      1.3. Setting Up and Controlling Notification ......................... 60
         1.3.1. Configure Notification via the dis_netconfig .............. 60
         1.3.2. Configure Notification Manually ......................... 61
         1.3.3. Verifying Notification ........................................ 61
         1.3.4. Disabling and Enabling Notification Temporarily .......... 61
   2. Managing IRM Resources .................................................. 61
      2.1. Updates with Modified IRM Configuration ..................... 62
   3. Using dis_diag .................................................................... 62
12. FAQ .......................................................................................... 65
   1. Software ............................................................................. 65
      A. Self-Installing Archive (SIA) Reference .......................... 68
         1. SIA Operating Modes ............................................... 68
            1.1. Full Cluster Installation .................................... 68
            1.2. Heterogenous Cluster Installation ......................... 68
            1.3. Node Installation ............................................. 68
            1.4. Cluster Management Node Installation .................... 68
            1.5. Installation of Configuration File Editor .................... 69
            1.6. Building RPM Packages Only .............................. 69
            1.7. Extraction of Source Archive .............................. 69
         2. SIA Options .................................................................. 69
            2.1. Node Specification ............................................. 69
            2.2. Installation Path Specification .............................. 69
            2.3. Installing from Binary RPMs ............................... 70
            2.4. Enforce Installation .......................................... 70
            2.5. Configuration File Specification ............................ 70
            2.6. PCIe Link width ............................................... 70
            2.7. eXpressWare CUDA® integration .......................... 70
            2.8. Batch Mode .................................................... 71
            2.9. Non-GUI Build Mode ........................................ 71
            2.10. Software Removal ........................................... 71
      B. dis_admin Reference ......................................................... 72
         1. Startup ........................................................................... 72
         2. Interconnect Status View ............................................. 72
            2.1. Icons .................................................................. 72
            2.2. Operation ................................................................ 73
               2.2.1. Cluster Status ............................................. 73
               2.2.2. Node Status .............................................. 76
         3. Node and Interconnect Control ....................................... 77
            3.1. Admin Menu ...................................................... 77
            3.2. Cluster Menu .................................................... 77
            3.3. Node Menu ....................................................... 78
            3.4. Cluster Settings ............................................... 78
            3.5. Adapter Settings .............................................. 79
         4. Interconnect Testing & Diagnosis .................................... 80
            4.1. Cable Test ......................................................... 80
            4.2. Fabric Test ...................................................... 81
         5. Troubleshooting Best Practice ........................................ 84
      C. Configuration Files ........................................................ 85
         1. Cluster Configuration ................................................ 85
1.1. dishosts.conf ................................................................. 85
  1.1.1. Basic settings ......................................................... 85
  1.1.2. SuperSockets settings ............................................. 86
1.2. networkmanager.conf .................................................. 87
1.3. cluster.conf ............................................................. 87
2. SuperSockets Configuration ............................................... 87
  2.1. supersockets_profiles.conf ........................................ 87
  2.2. supersockets_ports.conf ........................................... 88
3. Driver Configuration ...................................................... 89
  3.1. dis_irm.conf ............................................................ 89
    3.1.1. Resource Limitations .......................................... 89
    3.1.2. Real time behavior ............................................ 89
    3.1.3. Memory Preallocation ......................................... 90
    3.1.4. Logging and Messages ......................................... 91
  3.2. dis_ssocks.conf ..................................................... 92
D. Platform Issues and Software Limitations .............................. 93
  1. Platforms with Known Problems vs Dolphin PCI Express software 93
  2. IRM ............................................................................ 93
  3. SISCI ......................................................................... 93
  4. SuperSockets .................................................................. 93
List of Figures

4.1. Cluster Edit dialog of dis_netconfig ................................................................. 18
4.2. Main dialog of dis_netconfig ............................................................................ 20
4.3. Node dialog of dis_netconfig ........................................................................... 21
4.4. Connect to Dolphin Network Manager ............................................................. 25
B.1. Fabric is UP ......................................................................................................... 74
B.2. Fabric is DEGRADED ......................................................................................... 74
B.3. Fabric is REDUCED ......................................................................................... 75
B.4. Fabric has FAILED due to loss of connectivity .................................................. 75
B.5. Fabric has FAILED due to dead Cluster Nodes ................................................... 76
B.6. Options in the Admin menu ............................................................................. 77
B.7. Options in the Cluster menu ............................................................................. 77
B.8. Options in the Node menu ................................................................................. 78
B.9. Cluster configuration in dis_admin ..................................................................... 79
B.10. Advanced settings for a Cluster Node ............................................................. 80
B.11. Result of running cable test on a good cluster .................................................. 81
B.12. Result of cable test on a problematic cluster .................................................... 81
B.13. Result of fabric test without installing all the necessary rpms ......................... 83
B.14. Result of fabric test on a proper fabric ............................................................. 84
List of Tables

3.1. Curtiss-Wright SBC name to eXpressWare adapter mapping .......................................................... 7
B.1. Node or Adapter State .................................................................................................................... 72
B.2. Link State ....................................................................................................................................... 72
Abstract

This document describes the installation and management of the complete Dolphins PCI Express software stack, including SISCI, SuperSockets and Dolphins optimized IPoPCle TCP/IP driver, on single machines or on a cluster of machines.

This software stack is needed to use PCI Express for host to host communication and consists of drivers (kernel modules), user space libraries and applications, documentation and more.

SuperSockets drastically accelerate generic socket communication as used by clustered / networked applications.

The SISCI API provides an easy to use programming environment with direct access to PCI Express features like PIO, DMA, Interrupts and peer to peer transactions.

This document is for clusters built using Curtiss-Wright PCI Express enabled SBCs.
Chapter 1. Introduction & Overview

1. Who needs a PCI Express Network?

Clustered or networked applications running on multiple machines communicating via an Ethernet-based network (1G, 10G, 40G Ethernet) often suffer from the delays that occur when data needs to be exchanged between processes running on different machines. These delays caused by the communication protocols make processes wait for data when they otherwise could perform useful work. Dolphins PCI Express solution is a combination of a high-performance interconnect hardware to replace the Ethernet network together with a highly optimized software stack.

One important part of the Dolphin software stack is SuperSockets which implement a safe and reliable bypass of the TCP/UDP/IP protocol stack for standard socket-based inter-process communication. This bypass moves data directly via the high-performance interconnect and thereby reduces the minimal latency typically by a factor of 10 and more with 100% binary application compatibility, no special configuration and full plug and play.

The SuperSockets software was first released for critical applications in 2004 and has constantly been optimized and maintained to support new versions of Windows and Linux as well as new computer systems and interconnects.

The Dolphin SISCI Developers kit (Software Infrastructure Shared-Memory Cluster Interconnect) consists of driver and API software, tools, documentation and source needed to develop your own embedded application utilizing the low latency and high performance of a PCI Express Cluster. The development kit provides a C system call interface to ease customer integration to Dolphins cluster interconnect. SISCI enables customer application to easily bypass the limitations of traditional network solutions, avoiding time consuming operating system calls, and network protocol software overhead.

The SISCI Software enables applications to directly benefit from remote access time down to 0.54 microseconds and close to wire speed throughput. Direct mapped NTB PIO access gives extremely low transaction overhead and jitter for real-time systems. DMA capabilities provide very high throughput without using CPU for data transfers. Customers can combine PIO and DMA operations for maximum application benefits and performance. The SISCI API also support access to remote IO resources.

An optimized TCP/IP network driver was added to the stack in 2011, this enables customers to use PCI Express also for applications that is not compliant with the Berkeley Sockets API, e.g. NFS. The network driver supports standard TCP/IP functionality including ARP and routing to other networks.

2. PCI Express networking basics

The PCI Express network enables a process on one machine to safely write data directly into the address space of a process running on a remote machine. This can be done using either direct store operations of the CPU (for lowest latency), using the DMA engine of the PCI Express chipset or using a system DMA resource (for lowest CPU utilization).

The PCI Express network also enables standard PCI Express devices to be accessed from remote system. It is also possible to enable PCI Express devices to stream data to remote devices or memories.

PCI Express networking is made possible and very flexible by standard Non-Transparent PCI Express bridge functionality that exists in many standard PCI Express chipsets. It enables advanced device drivers to set up direct access to designated communication areas in remote system memory and PCI Express devices.

These hardware features are safely used by Dolphins software to implement several optimized software communication protocols:

The SISCI API implements an easy to use, but yet very powerful API for accessing PCI Express networking transport and functionality directly from applications. It provides a well defined C system call interface to ease customer integration to the PCI Express Network. SISCI enables customer application to easily bypass the limitations of traditional network solutions, avoiding time consuming operating system calls, and network protocol software overhead.
SuperSockets on Linux consists of both kernel modules and a user-space library. The implementation on kernel-level makes sure that the SuperSockets socket-implementation is fully compatible with the TCP/UDP/IP-based sockets provided by the operating system. By being explicitly preloaded, the user-space library operates between the unmodified binary of the applications and the operating system and intercepts socket-related function calls. Based on the system configuration and a potential user-provided configuration, the library makes a first decision if this function call will be processed by SuperSockets or the standard socket implementation and redirects it accordingly. The SuperSockets kernel module then performs the operation on the PCI Express interconnect. If a high speed network problem is detected (e.g. cable unplugged), it will fall back and forward communication to Ethernet transparently even when the socket is under load.

3. Contents of this Document

This document is the installation and reference guide for Dolphins software for PCI Express networking. It is structured as follows:

- Chapter 1, Introduction & Overview is what you are currently reading.
- For those with some experience in cluster management, the abbreviated installation instructions in Chapter 2, Quick Installation Guide might be sufficient. Otherwise, the following chapters cover all details.
- If you are installing the Dolphin PCI Express software stack for the first time, you should refer to Chapter 4, Initial Installation for a thorough guide. It explains the automatic cluster installation using the self installing archive (SIA). The Dolphin PCI Express software stack is distributed in this format.
- Updating an existing installation with SIA is covered in Chapter 5, Update Installation.
- Details of the installation process and procedures how to manually install the Dolphin PCI Express software stack, including platforms that do not support the RPM package format, are discussed in Chapter 6, Manual Installation.
- Once the cluster is installed, you will find procedures for typical maintenance tasks and for trouble shooting in Chapter 7, Interconnect Maintenance.
- A number features and configuration details that will be helpful for users with special requirements are explained in Chapter 11, Advanced Topics.
- Frequently asked questions are answered in Chapter 12, FAQ.
- The appendixes provide a reference of using the self installing archive (SIA), the GUI tool for interconnect management, the configuration files of the Dolphin PCI Express software stack, and a listing of known issues with few platforms and the default software limitations.

4. Terminology

We define some terms that will be used throughout this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter</td>
<td>A supported PCI Express cable adapter. This is the PCI Express hardware installed in the Cluster Nodes. Some Cluster Nodes may have an integrated PCIe Express chipset that connects the system to a PCIe enabled backplane. For the sake of this guide, we still in some cases refer to this as an adapter.</td>
</tr>
<tr>
<td>Adapter number</td>
<td>Each PCIe network connections in the SBC is identified by an adapter number. Most SBCs only support one connection</td>
</tr>
<tr>
<td>Cluster Node</td>
<td>A computer which is part of the PCI Express interconnect, which means it has a PCI Express network connection to other nodes. All Cluster Nodes together constitute the cluster.</td>
</tr>
<tr>
<td>SBC</td>
<td>Single Board Computer (SBC).</td>
</tr>
</tbody>
</table>
CPU architecture
The CPU architecture relevant in this guide is characterized by the addressing width of the CPU (32 or 64 bit) and the instruction set (x86, PowerPC, Sparc, ARM etc.). If these two characteristics are identical, the CPU architecture is identical for the scope of this guide.

Fabric
A fabric is an independent, closed communication network that connects a number of machines (here: all nodes in your cluster). Thus, with one adapter in each Cluster Node and all PCIe connections set up, the cluster is using a single fabric.

Link
The PCIe link between two SBCs or between a SBC and a PCIe switch card.

Cluster Management Node (frontend)
The single computer that is running software that monitors and configures the Cluster Nodes. The light weight cluster management service communicates with the Cluster Nodes out-of-band, which means via Ethernet.

Installation machine
The installation script is typically executed on the Cluster Management Node, but can also be executed on another machine that is neither a Cluster Node nor the Cluster Management Node, but has network (ssh) access to all Cluster Nodes and the Cluster Management Node. This machine is the installation machine.

Kernel build machine
The interconnect drivers are kernel modules and thus need to be built for the exact kernel running on the node (otherwise, the kernel will refuse to load them). To build kernel modules on a machine, the kernel-specific include files and kernel configuration have to be installed - these are not installed by default on most distributions. You will need to have one kernel build machine available which has these files installed (contained in the kernel-devel RPM that matches the installed kernel version) and that runs the exact same kernel version as the Cluster Nodes. Typically, the kernel build machine is one of the Cluster Nodes itself, but you can choose to build the kernel modules on any other machine that fulfills the requirements listed above.

Cluster
All Cluster Nodes constitute the cluster.

Network Manager
The Dolphin Network Manager is a daemon process named dis_networkmgr running on the Cluster Management Node. It is part of the Dolphin software stack and manages and controls the cluster using the Node Manager running on all Cluster Nodes. The Network Manager knows the interconnect status of all Cluster Nodes.

The service name of the Network Manager is dis_networkmgr.

Node Manager
The Node Manager is a daemon process that is running on each Cluster Node and provides remote access to the interconnect driver and other Cluster Node status to the Network Manager. It reports status and performs actions like configuring the installed adapter.

The service name of the Node Manager is dis_nodemgr.

self-installing archive (SIA)
A self-installing archive (SIA) is a single executable shell command file (for Linux and Solaris) that is used to compile and install the Dolphin software stack in all required variants. It largely simplifies the deployment and management of a PCI Express based cluster.

VxWorks Windows Installer (MSI)
The VxWorks Windows Installer is an engine for the installation, maintenance, and removal of the VxWorks software on modern Microsoft Windows systems. The installation information, and often the files themselves, are packaged in installation packages, loosely relational databases structured as OLE Structured Storage Files and commonly known as "MSI files", from their default file extension.
IX  The IX keyword is used for specifying a driver that supports PCI Express Gen2 chips from IDT.

PX  The PX keyword is used for specifying a driver that supports PCI Express Gen2 and Gen3 chips from Broadcom/Avago/PLX.

SuperSockets  SuperSockets is a Berkeley sockets compliant socket API provided by Dolphin. SuperSockets is currently supported on systems using Linux and Windows.

SISCI  SISCI (Software Infrastructure for Shared-Memory Cluster Interconnect) is the user-level API to create applications that make direct use of the low level PCI Express interconnect shared memory capabilities.

To run SISCI applications, a service named dis_sisci has to be running; it loads the required kernel module and sets up the SISCI devices.

NodeId  Each Cluster Node in a fabric is identified by an assigned NodeId

x1, x2, x4, x8, x16  PCI express combine multiple lanes (serial high-speed communication channels using few electrical connections) into communication paths with a higher bandwidth. With PCI Express Gen 1, each lane carries 2.5 Gbit/s of traffic, with PCI Express Gen 2, each lane carries 5.0 Gbit/s and with PCI Express Gen3, each lane carries 8.0 Gbit/s. Combining 8 lanes into a single communication path is called x8 and thus delivers 40 Gbit/s Bandwidth for Gen 2 or 64 Gbit/s Bandwidth for Gen 3, while x16 doubles this bandwidth using 16 lanes and delivers 128 Gbit/s for Gen3 in each direction.

5. Support

If you have any problems with the procedures described in this document, please contact Curtiss-Wright support https://www.curtisswrightds.com/support.html.

Please contact Dolphin via <sisci-support@dolphinics.com> if you need assistance on developing your own SISCI application. Dolphin currently offers several levels of SISCI Developer support programs.
Chapter 2. Quick Installation Guide

This chapter gives an overview of the steps necessary for a default installation of Dolphin eXpressWare for Linux using the self-installing archive format (SIA).

Please consult the detailed installation guide for more details, Chapter 4, Initial Installation.

1. **Installation requirements:** The SIA supports Linux 2.6, 3.x and 4.x kernels and distributions that support the RPM/DEB packages format. Please consult the release note for details.

   For details, see Chapter 3, Requirements and Planning for general platform requirements of the Dolphin PCI Express software stack and Chapter 4, Initial Installation, Section 1, “Installation Overview” for specific requirements for this installation procedure.

2. **Software download:** The Dolphin eXpressWare software for Curtiss-Wright systems is provided by Curtiss-Wright on a CD. Please copy the SIA to a directory on the machine that will serve as the Cluster Management Node. Make sure root has write access to this directory.

   Please contact Curtiss-Wright support if you need help to identify the required software: https://www.curtisswrightds.com/support.html

3. **Start software installation:** Become root and start the software installation by invoking the SIA as a shell script:

   ```
   # sh ./Dolphin_eXpressWare_<version>.sh
   ```

   In order to install the Cluster Nodes and the Cluster Management Node, the installation procedure will ask a number of questions including the acceptance of the software licensing terms.

   **Note**

   The Dolphin eXpressWare usage license requires the user to purchase a run time license for each system (PC, SBC etc) running the software. (e.g. To run a 4 node system, you need to purchase 4 licenses.)

   For details on this part of the installation, please see Chapter 4, Initial Installation, Section 2, “Software Installation”. For general information on the different ways to use the SIA, please see Appendix A, Self-Installing Archive (SIA) Reference.

4. **Interconnect configuration:** After the software has been installed, you need to set up the configuration files for the interconnect. The installer does this automatically, but for advanced options the GUI tool dis_netconfig can be started. In such a case the installation pauses until you quit dis_netconfig.

   For details on how to perform this task, see Chapter 4, Initial Installation, Section 2.4, “Working with the Dolphin Network Configurator, dis_netconfig”.

5. **Finish software installation:** Confirm the final questions of the installation script. The script will then finish the installation and perform some basic tests of the interconnect.

   For details on this part of the installation, see Chapter 4, Initial Installation, Section 2.5, “Finalising the Software Installation”.

6. **Verify the installation:** After the installation completes, start dis_admin and use the cabling test and the fabric test to verify the interconnect installation.

   **Warning**

   Please carefully perform this test. If you enable your application to use PCI Express without completing this step, you may be required to stop or restart your application to overcome possible instal-
lation issues at a later stage. All tests should pass before using the PCI Express cluster for production purposes. At this point, you can safely execute your tests without affecting the system.

For details on how to run these tests and handle the results, see Chapter 4, *Initial Installation*, Section 2.7, “Interconnect Validation using the management GUI”, Section 2.8, “Interconnect Validation using the command line”.

7. **Reboot the system**: It is recommended to reboot the system to verify the automatic restart of the drivers and to ensure optimal memory allocation for the PCI Express drivers.

8. **Run your application**: Enable your applications to make use of SuperSockets by using the `dis_ssocks_run` wrapper or just run your SISCI based application.

The Dolphin software installation comes with several example, demo and benchmark programs that can be executed to determine the actual PCI Express PIO, DMA and Interrupt performance.

For details on this step, please see Chapter 4, *Initial Installation*, Section 2.9, “Making Cluster Application use PCI Express”
Chapter 3. Requirements and Planning

Before you deploy a PCI Express cluster solution by either adding it to an existing system, or by planning it into a new system, some considerations on selection of products and the physical setup are necessary.

1. Supported Platforms

The Dolphin PCI Express software stack used with Curtiss-Wright systems is available for VxWorks 6.9 / 7 and Linux.

1.1. Application recommendations

PCI Express should normally be used between all systems that requires low latency or high throughput.

Most systems will provide low latency for small amount of data. Applications that needs high throughput will normally benefit from selecting a platform that provides DMA capabilities. PCI Express Gen3 based platforms are recommended for highest throughput.

1.2. Supported Platforms

The Dolphin PCI Express software is supported on a wide range of Curtiss-Wright Single Board Computers (SBCs) and Digital Signal Processor (DSP) modules, and supports any PCIe lane width and bus speed interconnect. For modules that support on-board DMA, the libraries have been optimized to make use of DMA to maximize data throughput and reduce CPU overhead. Supported Curtiss-Wright modules include:

- VPX3-1258: Intel 4th Gen Core i7 “Haswell” SBC
- VPX3-1259: Intel 5th Gen Core i7 “Broadwell” SBC
- VPX3-482 (Champ XD1): Intel Xeon D DSP Engine
- VPX3-131: NXP P4080 Power Architecture SBC
- VPX3-133: NXP T2080 Power Architecture SBC

Some combinations of CPU and chipset implementations offer sub-optimal performance which should be considered when planning a new system.

If you have questions about your specific hardware platform, please compare with the known issues listed in Appendix D, Platform Issues and Software Limitations or contact support.

1.3. eXpressWare adapter names

The table below gives the mapping between Curtiss-Wright SBC part numbers and the adapter names used by the eXpressWare. Adapter names are used by the eXpressWare drivers to identify and manage the underlying PCI Express chipset.

Table 3.1. Curtiss-Wright SBC name to eXpressWare adapter mapping

<table>
<thead>
<tr>
<th>Curtiss-Wright SBC name</th>
<th>eXpressWare Adapter name</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPX3-1258</td>
<td>PCH258</td>
</tr>
<tr>
<td>VPX3-1259</td>
<td>PCH259</td>
</tr>
<tr>
<td>VPX3-482</td>
<td>PCH482</td>
</tr>
<tr>
<td>VPX3-133</td>
<td>PCH133</td>
</tr>
</tbody>
</table>
1.4. Adapter serial numbers

The eXpressWare software will automatically assign each SBC a serial number that can be reached over PCIe to determine the remote partner.

The PCIe serial number for an SBC is calculated from the MAC address. If the MAC address is aa:bb:cc:dd:ee:ff the serial number is calculated as using the following formula:

\[
\text{serial\_no} = \text{ddee\textit{ff}(hex)} \mod 1000000\text{(dec)}
\]

For example:

MAC 00:1b:ac:00:90:6a
\[
\text{serial\_no} = 0x00906a \mod 1000000 = 36970
\]

1.5. Recommended Cluster Node Hardware

The hardware platform for the Cluster Nodes should be chosen from the Supported Platforms as described above. Next to the PCI Express specific requirements, you need to consult your application vendor expert / consultant on the recommended configuration for your application.

The PCI Express interconnect is fully inter-operable between all supported single board computers except when using the VPX3-131. (Please consult the release note for details.) As with all applications that communicate over a network, care must be taken by the applications if data with different endianess is communicated.

1.6. Recommended Cluster Management Hardware

The Cluster Management Node does only run a lightweight Network Manager service which does not impose special hardware requirements.

The Network Manager service is optional when using the SISCI API but mandatory if the SuperSockets or IPoPCIe software is used.

The Cluster Management Node requires a reliable Ethernet connection to all Cluster Nodes.

2. Software Support

The Dolphin PCI Express eXpressWare software stack running on Curtiss-Wright systems is currently available for Linux and VxWorks.

2.1. Linux

The complete Dolphin PCI Express software stack can be compiled for all popular Linux distributions (check the release note for details). A few extra packages (like the kernel include files and configuration) need to be installed for the compilation. Dolphin distributes an installer package that contains both binaries and source that needs to be compiled on the target system. Software stacks operating on different kernel versions are of course fully inter-operable for inter-node communication.

Dolphin PCI Express software fully supports native 32-bit and 64-bit platforms. On 64-bit platforms offering both 32-bit and 64-bit runtime environments, SuperSockets will support 32-bit applications if the compilation environment for 32-bit is also installed. Otherwise, only the native 64-bit runtime environment is supported.

For more information refer to Chapter 12, FAQ Q: 1.1.6
Please refer to the release notes of the software stack version you are about to install for the current list of tested Linux distributions and kernel versions. Installation and operation on Linux distributions and kernel versions that are not in this list will usually work as well, but especially the most recent versions may cause problems if it has not yet been qualified by Dolphin. Please notify Dolphin support at <pci-support@dolphinics.com> if you have a problem.

2.2. VxWorks

The Dolphin PCI Express SISCI software stack is available for VxWorks 6.9 and VxWorks 7. Please consult the software release note for details.

3. Interconnect Planning

This section discusses the decisions that are necessary when planning to install a PCI Express cluster.

3.1. Nodes to Equip with PCI Express Interconnect

Depending on the application that will run on the cluster, the choice of PCI Express Interconnect equipped machines differs.

3.2. PCI Express Interconnect Topologies

Two Cluster Nodes can be connected directly through a backplane.

Three Cluster Nodes can be connected directly through the backplane using the multi port topology.

Four and five Cluster Nodes can be connected directly through the backplane using the multi port topology, but only neighbour nodes will be able to communicate.

Up to 8 Cluster Nodes can be connected to one central PCIe switch.
Chapter 4. Initial Installation

This chapter guides you through the initial hardware and software installation of PCI Express software stack. This means that no Dolphin software is installed on the Cluster Nodes or the Cluster Management Node prior to these instructions. To update an existing installation, please refer to Chapter 5, Update Installation.

To add new Cluster Nodes to a cluster, please refer to Chapter 7, Interconnect Maintenance, “Adding Nodes”.

The recommended installation procedure, which is described in this chapter, uses the Self-Installing Archive (SIA) distribution of the Dolphin PCI Express software stack which can be used for all Linux distributions that support the RPM/DEB package formats. Installation on other Linux platforms is covered in Chapter 6, Manual Installation.

1. Installation Overview

1.1. Requirements

For the SIA-based installation of the full cluster and the Cluster Management Node, the following requirements have to be met:

• Homogeneous Cluster Nodes: All Cluster Nodes of the cluster are of the same CPU architecture and run the same kernel version. The Cluster Management Node may be of different CPU architecture and kernel version!

  Note
  
The installation of the Dolphin PCI Express software on a system that does not satisfy this requirement is described in Chapter 6, Manual Installation, Section 2, “Installation of a Heterogeneous Cluster”

• RPM support: The Linux distribution on the Cluster Nodes, the Cluster Management Node and the installation machine needs to support RPM or DEB packages. Both major distributions from Red Hat and (SUSE) use RPM packages.

  Note
  
  On platforms that do not support RPM packages, it is also possible to install the Dolphin PCI Express software. Please see Chapter 6, Manual Installation,?? for instructions.

• Installed RPM packages: To build the Dolphin PCI Express software stack, a few RPM packages that are often not installed by default are required:

  qt and qt-devel (> version 3.0.5), glibc-devel and libgcc (32- and 64-bit, depending on what binary formats should be supported), rpm-build, and the kernel header files and configuration (typically a kernel-devel or kernel-source RPM that exactly(!) matches the version of the installed kernel)

  Note
  
  The SIA will check for these packages, report what packages might be missing and will offer to install them if the yum RPM management system is supported on the affected machine. All required RPM packages are within the standard set of RPM packages offered for your Linux distribution, but may not be installed by default.

  If the qt-RPMs are not available, the Dolphin PCI Express software stack can be built nevertheless, but the GUI applications to configure and manage the cluster will not be available. Please see below (Section 1.2.2, “Non-GUI Installation”) on how to install the software stack in this case.

• GUI support: for the initial installation, the installation machine should be able to run GUI application via X.
Note

If the required configuration files are already available prior to the installation, a GUI is not required (see section Section 1.2.2, “Non-GUI Installation”).

• Disk space: To build the RPM packages, about 500MB free disk space in the system's temporary directory (typically /tmp on Linux) are required on the kernel build machine and the Cluster Management Node.

1.2. Installation Variants

1.2.1. Live Installation

If the Cluster Management Node does not support running GUI applications, but another machine in the network does, it is possible to run the installation on this machine. The only requirement is ssh-access from the installation machine towards the Cluster Management Node and all Cluster Nodes. This installation mode can be chosen by executing the SIA on the installation machine and specifying the Cluster Management Node name when being asked for it.

In this scenario, the dis_netconfig will be compiled, installed and executed on the installation machine, and the generated configuration files will be transferred to the Cluster Management Node by the installer. Optionally you can follow the instructions in the next section.

PCI Express can be installed into a cluster which is currently under operation without stopping the cluster application from working. This requires that the application running on the cluster can cope with single Cluster Nodes going down. It is only necessary to turn off each Cluster Node once to install the adapter. The software installation can be performed under load, although minor performance impacts are possible. For a description of this installation type, please proceed as described in Chapter 6, Manual Installation, Section 1, “Installation under Load”

1.2.2. Non-GUI Installation

If no machine in the network does have the capability to run GUI applications, you can still use the SIA-based installation. The SIA-based installation does not require that you use the GUI applications, and will use the commandline client, dis_mkconf to create dishosts.conf and networkmanager.conf.

The Dolphin software includes two GUI tools:

• dis_netconfig is a tool that is used to create the interconnect configuration file /etc/dis/dishosts.conf and the Network Manager configuration file /etc/dis/networkmanager.conf. It is needed once on the initial cluster installation, and each time Cluster Nodes are added or removed from the cluster.

• dis_admin is used to monitor and control the cluster interconnect.

These tools are not mandatory for the installation and operation of the cluster, but are very helpful. They can be run on machines separate from the cluster: the configuration files created by dis_netconfig can be copied over to the Cluster Management Node manually, and dis_admin can connect to the Network Manager on the Cluster Management Node via TCP sockets.

1.2.2.1. No X / GUI on Cluster Management Node

If the Cluster Management Node does not support running GUI applications, but another machine in the network does, it is possible to run the installation on this machine. The only requirement is ssh-access from the installation machine towards the Cluster Management Node and all Cluster Nodes. This installation mode can be chosen by executing the SIA on the installation machine and specifying the Cluster Management Node name when being asked for it.

In this scenario, the dis_netconfig will be compiled, installed and executed on the installation machine, and the generated configuration files will be transferred to the Cluster Management Node by the installer.
1.2.2. No X / GUI Anywhere

If no machine in the network does have the capability to run GUI applications, you can still use the SIA-based installation. The SIA-based installation does not require that you use the GUI applications, and will use the commandline client, dis_mkconf to create dishosts.conf and networkmanager.conf.

If you want to verify your settings in the Dolphin Network Configuration, it is possible to create the correct configuration files on another machine and store them in /etc/dis on the Cluster Management Node before executing the SIA on the Cluster Management Node (not on another machine). In this scenario, no GUI application is run at all during the installation. To create the configuration files on another machine, you can either run the SIA with the --install-editor option if it is a Linux machine, or install a binary version of the dis_netconfig if it is a Windows-based machine. Finally, you can send the necessary information to create the configuration files to Dolphin support which will then provide you with the matching configuration files and the cabling instructions. This information includes:

- External hostnames (or IP addresses) of all Cluster Nodes
- SBC type
- Hostnames (or IP addresses/subnet) which should be accelerated with SuperSockets (default is the list of hostnames provided above)
- Planned interconnect topology (default is derived from number of Cluster Nodes and adapter type)

1.3. System installation overview

To configure and operate a PCI Express cluster you need to select one computer to be the Cluster Management Node (frontend). The Cluster Management Node will automatically monitor and configure the PCI Express hardware based on configuration files created on the Cluster Management Node. The Cluster Management Node can be one of the Cluster Nodes in the cluster or it can be a separate machine. The Cluster Management Node will communicate with the Cluster Nodes through Ethernet. The burden of the Cluster Management Node is very modest. The Cluster Management Node machine needs to have a graphical display to be able to utilize the Dolphin Network Configurator and the Dolphin Admin.

Please note that there is no need to manually configure the adapter cards in the cluster as this will be done automatically by the Dolphin software.

1.4. Installation Result

The result of the installation will be that all necessary drivers and services are installed on the Cluster Nodes, that the Cluster Management Node has the Network Manager running and that the whole PCI Express interconnect is configured and functional. By default, the installation target of all Dolphin software is /opt/DIS.

In detail, the installation results on each Cluster Node are the following services:

- dis_kosif: This service loads the kosif module which manages OS dependent functionality.
- dis_ix: This service loads the dis_ix module which manages the IDT PCI Express hardware.
- dis_px: This service loads the dis_px module which manages the PLX PCI Express hardware.
- dis irm: This service loads the dis irm module which performs configuration, and session and resource management.
- dis_sisci: This service loads the dis_sisci module which provides abstractions and mappings for the SISCI user-level API.
- dis_supersockets: This service loads the dis_msq, dis_mbox and dis_ssocks modules which together provide the SuperSockets functionality.
- dis_nodemgr: This service controls the node manager daemon.
• dis_networkmgr (only on Cluster Management Node): This service controls the node manager daemon.

Note

Please note that you should only have one of the dis_ix or dis_px drivers installed on one node.

Next to this, a number of administration tools, benchmark and test programs are installed into /opt/DIS/bin and /opt/DIS/sbin.

On the Cluster Management Node, the service dis_networkmgr will be installed which controls the Network Manager daemon. Monitoring and test tools are installed to /opt/DIS/sbin.

2. Software Installation

On the Cluster Nodes, the hardware driver and additional kernel modules, user space libraries and the Node Manager will be installed. On the Cluster Management Node, the Network Manager and the cluster administration tool will be installed.

An additional package for SISCI development (SISCI-devel) will be created for both, Cluster Management Node and Cluster Nodes, but will not be installed. It can be installed as needed in case SISCI-based applications or libraries (like NMPI) need to be compiled from source.

2.1. Overview

The integrated cluster and Cluster Management Node installation is the default operation of SIA, but can be specified explicitly with the --install-all option. It works as follows:

• The SIA is executed on the installation machine with root permissions. The installation machine is typically the machine to serve as Cluster Management Node, but can be any other machine if necessary (see Section 1.2.2.1, “No X / GUI on Cluster Management Node”). The SIA controls the building, installation and test operations on the remote Cluster Nodes via ssh. Therefore, password-less ssh to all remote Cluster Nodes is required.

If password-less ssh access is not set up between the installation machine, Cluster Management Node and Cluster Nodes, SIA offers to set this up during the installation. The root passwords for all machines are required for this.

• The binary packages for the Cluster Nodes and the Cluster Management Node are built on the kernel build machine and the Cluster Management Node, respectively. The kernel build machine needs to have the kernel headers and configuration installed, while the Cluster Management Node and the installation machine only compile user-space applications.

• The Cluster Node packages with the kernel modules are installed on all Cluster Nodes, the kernel modules are loaded and the Cluster Node manager is started. At this stage, the interconnect is not yet configured.

• On an initial installation, the dis_netconfig is installed and executed on the installation machine to create the cluster configuration files. This requires user interaction.

• The cluster configuration files are transferred to the Cluster Management Node, and the Network Manager is installed and started on the Cluster Management Node. It will in turn configure all Cluster Nodes according to the configuration files. The cluster is now ready to utilize the PCI Express interconnect.

• A number of tests are executed to verify that the cluster is functional and to get basic performance numbers.

For other operation modes, such to install specific components on the local machine, please refer to Appendix A, Self-Installing Archive (SIA) Reference.

2.2. Starting the Software Installation

Log into the chosen installation machine, become root and make sure that the SIA file is stored in a directory with write access (/tmp is fine). Execute the script:
The script will ask questions to retrieve information for the installation. You will notice that all questions are Yes/no questions, and that the default answer is marked by a capital letter, which can be chosen by just pressing Enter. A typical installation looks like this:

```
[root@scimple tmp]# sh ./Dolphin_eXpressWare-Linux_x86-PX-v5_4_2_2017_Jan_20.sh
Verifying archive integrity... All good.
Uncompressing Dolphin DIS 5.0.0
** Logfile is /tmp/DIS_install.log_140 on tiger-0

* Dolphin ICS - Software installation (version: 1.52 $ of: 2015/06/30 16:31:32 $)
++
* Installing a full cluster (Cluster Nodes and Cluster Management Node).
* This script will install Dolphin PCI Express drivers, tools and services
++ on all Cluster Nodes of the cluster and on the Cluster Management Node.
++
* All available options of this script are shown with option '--help'
>>> OK to proceed with cluster installation? [Y/n] y
>>> Will the local machine <tiger-0> serve as Cluster Management Node? [Y/n] y
```

The default choice is to use the local machine as Cluster Management Node. If you answer n, the installer will ask you for the hostname of the designated Cluster Management Node machine. Each cluster needs its own Cluster Management Node machine.

Please note that the complete installation is logged to a file which is shown at the very top (here: /tmp/DIS_install.log_140). In case of installation problems, this file is very useful to Dolphin support.

```
* NOTE: Cluster configuration files can be specified now, or be generated
* ..... during the installation.
* >>> Do you have a 'dishosts.conf' file that you want to use for installation? [y/N] n
```

Because this is the initial installation, no installed configuration files could be found. If you have prepared or received configuration files, they can be specified now by answering y. In this case, no GUI application needs to run during the installation, allowing for a shell-only installation.

For the default answer, the hostnames of the Cluster Nodes need to be specified (see below), and the cluster configuration is created automatically.

```
* NOTE:
++ No cluster configuration file (dishosts.conf) available.
++ You can now specify the Cluster Nodes that are attached to the PCI
++ Express network. The necessary configuration files can then
++ be created based on this list of Cluster Nodes.
++
++ Please enter hostname or IP addresses of the Cluster Nodes one per line.
++ When done, enter a single full period ('.').
+++ (proposed hostname is given in [brackets])
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-1]
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-2]
-> tiger-2
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-3]
-> tiger-3
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-4]
-> tiger-4
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-5]
-> tiger-5
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-6]
-> tiger-6
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-7]
-> tiger-7
>>> Cluster Node hostname/IP address <full period '.' when done> [tiger-8]
```
Initial Installation

-> tiger-8
# >>> Cluster Node hostname/IP address <full period '.' when done> [tiger-9].

The hostnames or IP-addresses of all Cluster Nodes need to be entered. The installer suggests the hostnames if possible in brackets. To accept a suggestion, just press Enter. Otherwise, enter the hostname or IP address. The data entered is verified to represent an accessible hostname. If a Cluster Node has multiple IP addresses / hostnames, make sure you specify the one that is visible for the installation machine and the Cluster Management Node.

When all hostnames are entered, enter a single full period . to finish.

#* NOTE:
#+ The kernel modules need to be built on a machine with the same kernel
#* version and architecture of the interconnect Cluster Node. By default, the first
#* given interconnect Cluster Node is used for this. You can specify another build
#* machine now.
# >>> Build kernel modules on Cluster Node tiger-1 ? [Y/n]y

If you answer n at this point, you can enter the hostname of another machine on which the kernel modules are built. Make sure it matches the Cluster Nodes for CPU architecture and kernel version.

# >>> Can you access all machines (local and remote) via password-less ssh? [Y/n]y

The installer will later on verify if the password-less ssh access actually works. If you answer n, the installer will set up password-less ssh for you on all Cluster Nodes and the Cluster Management Node. You will need to enter the root password once for each Cluster Node and the password.

The password-less ssh access remain active after the installation. To disable it again, remove the file /root/.ssh/authorized_keys from all Cluster Nodes and the Cluster Management Node.

#* NOTE:
#+ It is recommended that interconnect Cluster Nodes are rebooted after the
#+ initial driver installation to ensure that large memory allocations will succeed.
#+ You can omit this reboot, or do it anytime later if necessary.
# >>> Reboot all interconnect Cluster Nodes
(tiger-1 tiger-2 tiger-3 tiger-4 tiger-5 tiger-6 tiger-7 tiger-8)? [y/N]n

For optimal performance, the low-level driver needs to allocate some amount of kernel memory. This allocation can fail on a system that has been under load for a long time. If you are not installing on a live system, rebooting the Cluster Nodes is therefore offered here. You can perform the reboot manually later on to achieve the same effect.

If chosen, the reboot will be performed by the installer without interrupting the installation procedure.

#* NOTE:
#+ About to INSTALL Dolphin PCI Express interconnect drivers on these Cluster Nodes:
... tiger-1
... tiger-2
... tiger-3
... tiger-4
... tiger-5
... tiger-6
... tiger-7
... tiger-8
#* About to BUILD Dolphin PCI Express interconnect drivers on this Cluster Node:
... tiger-1
#* About to install management and control services on the Cluster Management Node machine:
... tiger-0
#* Installing to default target path /opt/DIS on all machines
.. (or the current installation path if this is an update installation).
# >>> OK to proceed? [Y/n]y

The installer presents an installation summary and asks for confirmation. If you answer n at this point, the installer will exit and the installation needs to be restarted.

#* NOTE:
#+ Testing ssh-access to all Cluster Nodes and gathering configuration.
++
++ If you are asked for a password, the ssh access to this Cluster Node without
++ password is not working. In this case, you need to interrupt with CTRL-c
++ and restart the script answering 'no' to the initial question about ssh.
... testing ssh to tiger-1
... testing ssh to tiger-2
... testing ssh to tiger-3
... testing ssh to tiger-4
... testing ssh to tiger-5
... testing ssh to tiger-6
... testing ssh to tiger-7
... testing ssh to tiger-8
++ OK: ssh access is working
++ OK: Cluster Nodes are homogenous
++ OK: found 1 interconnect fabric(s).

* Testing ssh to other Cluster Nodes
... testing ssh to tiger-1
... testing ssh to tiger-0
... testing ssh to tiger-0
* OK.

The ssh-access is tested, and some basic information is gathered from the Cluster Nodes to verify that the Cluster
Nodes are homogeneous and equipped with at least one PCI Express adapter and meet the other requirements. If
a required RPM package was missing, it would be indicated here with the option to install it (if yum can be used),
or to fix the problem manually and retry.

If the test for homogeneous Cluster Nodes failed, please refer to section Section 2, “Installation of a Heterogeneous
Cluster” for information on how to install the software stack.

* Building Cluster Node RPM packages on tiger-1 in /tmp/tmp.AEgiO27908
* This will take some minutes...
* Logfile is /tmp/DIS_install.log_983 on tiger-1
* OK, Cluster Node RPMs have been built.

* Building Cluster Management Node RPM packages on scimple in /tmp/tmp.dQdwS17511
* This will take some minutes...
* Logfile is /tmp/DIS_install.log_607 on scimple
* OK, Cluster Management Node RPMs have been built.

* Copying RPMs that have been built:
/tmp/frontend_RPMS/Dolphin-NetworkAdmin-PX-5.0.0-1.x86_64.rpm
/tmp/frontend_RPMS/Dolphin-NetworkHosts-PX-5.0.0-1.x86_64.rpm
/tmp/frontend_RPMS/Dolphin-SISCI-PX-devel-5.0.0-1.x86_64.rpm
/tmp/frontend_RPMS/Dolphin-NetworkManager-PX-5.0.0-1.x86_64.rpm
/tmp/node_RPMS/Dolphin-SISCI-PX-5.0.0-1.x86_64.rpm
/tmp/node_RPMS/Dolphin-SISCI-PX-devel-5.0.0-1.x86_64.rpm
/tmp/node_RPMS/Dolphin-SuperSockets-PX-5.0.0-1.x86_64.rpm
/tmp/node_RPMS/Dolphin-PX-5.0.0-1.x86_64.rpm

The binary RPM packages matching the Cluster Nodes and Cluster Management Node are built and copied
to the directory from where the installer was invoked. They are placed into the subdirectories node_RPMS and
frontend_RPMS for later use (see the SIA option --use-rpms).

* To install/update the Dolphin PCI Express services like SuperSockets, all running
Dolphin PCI Express services needs to be stopped. This requires that all user
applications using SuperSockets (if any) need to be stopped NOW.
* >>> Stop all Dolphin PCI Express services (SuperSockets) NOW? [Y/n]y
* OK: all Dolphin PCI Express services (if any) stopped for upgrade.

On an initial installation, there will be no user applications using SuperSockets, so you can easily answer y right
away.

* Installing Cluster Node tiger-1
* OK.
2.3. Post installation

The Cluster Nodes get installed and drivers and the node manager are started. Then, the basic packages are installed on the Cluster Management Node, and the dis_netconfig application is launched to create the required configuration files /etc/dis/dishosts.conf and /etc/dis/networkmanager.conf if they do not already exist. The script will wait at this point until the configuration files have been created with dis_netconfig, and until you confirm that all cables have been connected according to the cabling instructions. This is described in the next section.

For typical problems at this point of the installation, please refer to Chapter 12, FAQ.

2.4. Working with the Dolphin Network Configurator, dis_netconfig

The Dolphin Network Configurator, dis_netconfig is a GUI tool that helps gathering the cluster configuration (and is used to create the cluster configuration file /etc/dis/dishosts.conf and the Network Manager configuration file /etc/dis/networkmanager.conf). A few global interconnect properties need to be set, and the position of each Cluster Node within the interconnect topology needs to be specified.

2.4.1. Cluster Edit

When dis_netconfig is launched, it first displays a dialog box where the global interconnect properties need to be specified (see Figure 4.1, “Cluster Edit dialog of dis_netconfig”).
2.4.1.1. Interconnect Topology

In the upper half of the Cluster Edit dialog, you need to specify the interconnect topology that you will be using with your cluster. If dis_netconfig is launched by the installation script, the script tries to set these values correctly, but you need to verify the settings.

First, select the Topology of your cluster: either you use a single PCI Express switch for 2-8 Cluster Nodes, or 2 Cluster Nodes with direct connection.

Then, specify the Number of Cluster Nodes in your cluster.

The Socketadapter setting determines which of the available adapter is used for SuperSockets:

- SINGLE 0: adapter 0 is used
- NONE: SuperSockets should not be used.

You then need to Set Link Widths for each Cluster Node. This can be set to x8 or x4. The default is x8. The value must be set to x4 if a x4 cable is used, otherwise the low level driver will try to reset the link to establish a x8.

The Advanced Edit option does not need to be changed: the session between the Cluster Nodes should typically always be set up automatically.

2.4.1.2. SuperSockets Network Address

If your cluster operates within its own subnet and you want all Cluster Nodes within this subnet to use SuperSockets (having PCI Express installed), you can simplify the configuration by specifying the address of this subnet in
this dialog. To do so, activate the Network Address field and enter the cluster IP subnet address including the mask. I.e., if all your Cluster Node communicate via an IP interface with the address 192.168.4.*, you would enter 192.168.4.0/8 here.

SuperSockets will try to use the PCI Express for any Cluster Node in this subnet when it connects to another Cluster Node of this subnet. If using PCI Express is not possible, i.e. because one or both Cluster Nodes are only equipped with an Ethernet interface, SuperSockets will automatically fall back to Ethernet. Also, if a Cluster Node gets assigned a new IP address within this subnet, you don’t need to change the SuperSockets configuration. Assigning more than one subnet to SuperSockets is also possible, but this type of configuration is not yet supported by dis_netconfig. See section Section 1.1, “dishosts.conf” on how to edit dishosts.conf accordingly.

This type of configuration is required if the same Cluster Node can be assigned varying IP addresses over time, as it is done for fail-over purposes where one machine takes over the identity of a machine that has failed. For standard setups where the assignment of IP addresses to Cluster Nodes is static, it is recommended to not use this type of configuration, but instead use the default static SuperSockets configuration type.

2.4.1.3. Status Notification

In case you want to be informed on any change of the interconnect status (i.e. an interconnect link was disabled due to errors, or a Cluster Node has gone down), active the check box Alert target and enter the alert target and the alert script to be executed. The default alert script is alert.sh and will send an e-mail to the address specified as alert target.

Other alert scripts can be created and used, which may require another type of alert target (i.e. a cell phone number to send an SMS). For more information on using status notification, please refer to Chapter 11, Advanced Topics, Section 1, “Notification on Interconnect Status Changes”.

2.4.2. Node Arrangement

In the next step, the main pane of the dis_netconfig will present the Cluster Nodes in the cluster arranged in the topology that was selected in the previous dialog. To change this topology and other general interconnect settings, you can always click Edit in the Cluster Configuration area which will bring up the Cluster Edit dialog again.

If the font settings of your X server cause dis_netconfig to print unreadable characters, you can change the font size and the type with the drop-down box at the top of the windows, next to the floppy disk icon.
Figure 4.2. Main dialog of dis_netconfig

[Image: dishosteditor_for_DX.png]
In the Node dialog you specify if you want to use 4 or 8 PCI Express lanes.

After you have assigned the correct hostname to this machine, you may need to configure SuperSockets on this Cluster Node. If you selected the Network Address in the cluster configuration dialog (see above), then SuperSockets will use this subnet address and will not allow for editing this property on the Cluster Nodes. Otherwise, you can choose between 3 different options for each of the currently supported 2 SuperSockets accelerated IP interfaces per Cluster Node:
disable
Do not use SuperSockets. If you set this option for both fields, SuperSockets can not be used with this Cluster Node, although the related kernel modules will still be loaded.

static
Enter the hostname or IP address for which SuperSockets should be used. This hostname or IP address will be statically assigned to this physical Cluster Node (its PCI Express interconnect adapter).

Choosing a static socket means that the mapping between the Cluster Node (its adapters) and the specified hostname/IP address is static and will be specified within the configuration file dishosts.conf. All Cluster Nodes will use this identical file (which is automatically distributed from the Cluster Management Node to the Cluster Nodes by the Network Manager) to perform this mapping.

This option works fine if the Cluster Nodes in your cluster don’t change their IP addresses over time and is recommend as it does not incur any name resolution overhead.

dynamic
Enter the hostname or IP address for which SuperSockets should be used. This hostname or IP address will be dynamically resolved to the PCI Express interconnect adapter that is installed in the machine with this hostname/IP address. SuperSockets will therefore resolve the mapping between adapters and hostnames/IP addresses dynamically. This incurs a certain initial overhead when the first connection between two Cluster Nodes is set up and in some other specific cases.

This option is similar to using a subnet (see Section 2.4.1.2, “SuperSockets Network Address”), but resolves only the explicitly specified IP addresses (for all Cluster Nodes) and not all possible IP addresses of a subnet. Use this option if Cluster Nodes change their IP addresses i.e. in a fail-over setup.

2.5. Finalising the Software Installation

Once the cables are connected, no more user interaction is required. Please confirm that all cables are connected and all LEDs are green, and the installation will proceed. The Network Manager will be started on the Cluster Management Node, configuring all cluster Cluster Nodes according to the configuration specified in dishosts.conf. After this, a number of tests are run on the cluster to verify that the interconnect was set up correctly and delivers the expected performance. You will see output like this:

```
#* NOTE: checking for cluster configuration to take effect:
... node tiger-1:
... node tiger-2:
... node tiger-3:
... node tiger-4:
... node tiger-5:
... node tiger-6:
... node tiger-7:
... node tiger-8:
#* OK.

#* Installing remaining Cluster Management Node packages

#* NOTE:
#* To compile SISCI applications (like NMPI), the SISCI-devel RPM needs to be
#* installed. It is located in the frontend_RPMS and node_RPMS directories.
#* OK.
```

If no problems are reported (like in the example above), you are done with the installation and can start to use your DolphinExpress accelerated cluster. Otherwise, refer to the next subsections and Section 2.7, “Interconnect Validation using the management GUI” or Section 2.8, “Interconnect Validation using the command line” to learn about the individual tests and how to fix problems reported by each test.

2.5.1. Static Connectivity Test

The Static Connectivity Test verifies that links are up and all Cluster Nodes can see each other via the interconnect. Success in this test means that all adapters have been configured correctly, and that the cables are inserted properly. It should report TEST RESULT: *PASSED* for all Cluster Nodes:
If this test reports errors or warning, you are offered to re-run dis_netconfig to validate and possibly fix the interconnect configuration. If the problems persist, you should let the installer continue and analyse the problems using disadmin after the installation finishes (see Section 2.7, “Interconnect Validation using the management GUI”).

2.5.2. SuperSockets Configuration Test

The SuperSockets Configuration Test verifies that all Cluster Nodes have the same valid SuperSockets configuration (as shown by /proc/net/sf_ssocks/socket_maps).

* NOTE: Verifying SuperSockets configuration on all Cluster Nodes.

*# No SuperSocket configuration problems found.

Success in this test means that the SuperSockets service dis_supersockets is running and is configured identically on all Cluster Nodes. If a failure is reported, it means the interconnect configuration did not propagate correctly to this Cluster Node. You should check if the dis_nodemgr service is running on this Cluster Node. If not, start it, wait for a minute, and then configure SuperSockets by calling dis_ssocks_cfg.

2.5.3. SuperSockets Performance Test

The SuperSockets Performance Test runs a simple socket benchmark between two of the Cluster Nodes. The benchmark is run once via Ethernet and once via SuperSockets, and performance is reported for both cases.

* NOTE:

*# Verifying SuperSockets performance for tiger-2 (testing via tiger-1).
*# Checking Ethernet performance
... single-byte latency: 56.63 us
*# Checking Dolphin PCI Express SuperSockets performance
... single-byte latency: 0.99 us
... Latency rating: Very good. SuperSockets are working well.
*# SuperSockets performance tests done.

The SuperSockets latency is rated based on our platform validation experience. If the rating indicates that SuperSockets are not performing as expected, or if it shows that a fall-back to Ethernet has occurred, please contact Dolphin Support. In this case, it is important that you supply the installation log (see above).

The installation finishes with the option to start the administration GUI tool dis_admin, a hint to use LD_PRELOAD to make use of SuperSockets and a pointer to the binary RPMs that have been used for the installation.

*# OK: Cluster installation completed.

** Remember to use LD_PRELOAD=libksupersockets.so for all applications that
** should use Dolphin PCI Express SuperSockets.
** Do you want to start the GUI tool for interconnect administration (GUI admin)? [y/N]n

* RPM packages that were used for installation are stored in
** /tmp/node_PRMS and /tmp/frontend_PRMS.
2.6. Handling Installation Problems

If for some reason the installation was not successful, you can easily and safely repeat it by simply invoking the SIA again. Please consider:

- By default, existing RPM packages of the same or even more recent version will not be replaced. To enforce re-installation with the version provided by the SIA, you need to specify `--enforce`.

- To avoid that the binary RPMs are built again, use the option `--use-rpms` or simply run the SIA in the same directory as before where it can find the RPMs in the `node_RPMS` and `frontend_RPMS` subdirectories.

- To start an installation from scratch, you can run the SIA on each Cluster Node and the Cluster Management Node using the option `--wipe` to remove all traces of the Dolphin PCI Express software stack and start again.

If you still fail to install the software successfully, refer to Chapter 7, *Interconnect Maintenance*.

Every installation attempt creates a differently named log file; it's name is printed at the very beginning of the installation. Please also include the configuration files that can be found in `/etc/dis` on the Cluster Management Node.

2.7. Interconnect Validation using the management GUI

Dolphin provides a graphical tool named dis_admin. dis_admin serves as a tool to visually view the status and run diagnostics on your PCIe network. It shows an overview of the status of all adapters and links of a cluster and allows to perform detailed status queries. It also provides means to manually control the interconnect, inspect and set options and perform interconnect tests. For a complete description of dis_admin, please refer to Appendix B, *dis_admin Reference*. Here, we will only describe how to use dis_admin to verify the newly installed PCI Express network.

2.7.1. Installing dis_admin

dis_admin will be installed on the Cluster Management Node machine by the SIA if this machine is capable to run X applications and has the Qt toolkit installed. If the Cluster Management Node does not have these capabilities, you can install it on any other machine that has these capabilities using SIA with the `--install-frontend` option, or use the Dolphin-NetworkAdmin RPM package from the `frontend_RPMS` directory (this RPM will only be there if it could be built for the Cluster Management Node).

It is also possible to download a binary version for Windows that runs without the need for extra compilation or installation.

You can use dis_admin on any machine that can connect to the Network Manager on the Cluster Management Node via a standard TCP/IP socket. You have to make sure that connections towards the Cluster Management Node using the ports 3444 (Network Manager) and 3443 (node manager) are possible (potentially firewall settings need to be changed).

2.7.2. Starting dis_admin

dis_admin will be installed in the `sbin` directory of the installation path (default: `/opt/DIS/sbin/dis_admin`). It will be within the `PATH` after you login as root, but can also be run by non-root users.

After it has been started, you will need to connect to the Network Manager controlling your cluster. Click the Connect button in the tool bar and enter the appropriate hostname or IP address of the Network Manager.
Figure 4.4. Connect to Dolphin Network Manager

Figure 4.4. Connect to Dolphin Network Manager

2.7.3. Cluster Overview

Normally, all Cluster Nodes and interconnect links should be shown green, meaning that their status is OK. This is a requirement for a correctly installed and configured cluster and you may proceed to Section 2.7.4, “PCIe connection Test”.

If a Cluster Node is plotted red, it means that the Network Manager can not connect to the node manager on this Cluster Node. To solve this problem:

1. Make sure that the Cluster Node is powered and has booted the operating system.

2. Verify that the node manager service is running:

   On Red Hat:
   
   # service dis_nodemgr status

   On other Linux variants:
   
   # /etc/init.d/dis_nodemgr status

   The command
   
   # svcs dis_nodemgr

   should tell you that the node manager is running. If this is not the case:

   a. Try to start the node manager:
On Red Hat:

# service dis_nodemgr start

On other Linux variants:

#/etc/init.d/dis_nodemgr start

b. If the node manager fails to start, please see /var/log/dis_nodemgr.log

c. Make sure that the service is configured to start in the correct runlevel (Dolphin installation makes sure this is the case).

On Red Hat:

# chkconfig --add 2345 dis_nodemgr on

On other Linux variants, please refer to the system documentation to determine the required steps.

2.7.4. PCIe connection Test

dis_admin can validate that all PCIe connections are connected according to the configuration that was specified in the dis_netconfig, and which is now stored in /etc/dis/dishosts.conf on all Cluster Nodes and the Cluster Management Node. To perform the connection test, select Cluster -> Test PCIe Connections. This test runs for only a few seconds and will verify that the Cluster Nodes are connected according to the configuration provided by the dis_netconfig.

Warning

Running this test may stop the normal traffic over the interconnect as the routing may be changed to fully test the network.

If you run this test while your cluster is in production, you might experience communication delays. SuperSockets in operation will fall back to Ethernet during this test, which leads to increased communication latency and lower throughput.

If the test detects a problem, it will inform you that Cluster Node A can not communicate with Cluster Node B. You will get more than one error message if the problem is caused by problem with a central switch etc.:  

2.8. Interconnect Validation using the command line

Dolphin provides a console tool named dis_admin_commandline. dis_admin_commandline shows an overview of the status of all nodes and links of a cluster and allows to perform a limited set of queries.

dis_admin_commandline will be installed on the Cluster Management Node by the SIA with the --install-frontend option, or use the Dolphin-NetworkAdmin RPM package from the frontend_RPMS directory.

Any command that is passed to dis_admin_commandline must be preceded by -cluster <frontend_machine_name> -fabric <adapter_number>. The full list of commands is printed when the program is started with no command line arguments.

Here are the most common commands used for inspection:

- get-cluster-info shows the cluster status and the topology dimensions.

- get-fabric-state prints on one line the cluster status.

- application <name_of_application> <node1> <node2> <node...> runs one of the predefined applications on the specified Cluster Nodes. The supported applications are: alltoall, sciconntest, latency_bench, scipp, scibench2, dma_bench, intr_bench

dis_admin_commandline -cluster node-3 -fabric 0 application sciconntest node-3 node-4

Feb 10 19:17:18 Started an application on node-3,node-4.
Feb 10 19:17:18 This might take some time. Timeout is 55 seconds.

[RESULT FROM RUNNING CMD ON SELECTED NODES]

[node-3]:
C:\Program Files\Dolphin Express\Util\..\demo\sciconntest compiled Feb 10 2012 : 00:08:07

Response from remote node 8
-------------------------------------
Local node-id : 4
Local adapter no. : 0
Segment size : 8192
MinSize : 4
Time to run (sec) : 10
Idelay : 0
No Write : 0
Loopdelay : 0
Delay : 0
Bad : 0
Check : 0
Mcheck : 0
Max nodes : 256
rnl : 1
Callbacks : Yes
-------------------------------------
Local segment (id=8, size=8192) is created.
Local segment (id=8, size=8192) is shared.
Connecting to 1 nodes
Connect to remote segment, node 8
Remote segment on node 8 is connected.
SCICONNTEST_REPORT
NUM_TESTLOOPS_EXECUTED 155
NUM_NODES_FOUND 1
NUM_ERRORS_DETECTED 0
node 8 : Found
node 8 : Number of failures : 0
node 8 : Longest failure : 0.00 (ms)
SCICONNTEST_REPORT_END

SCI_CB_DISCONNECT:Segment removed on the other node disconnecting.....
The local segment is set to unavailable
The local segment is removed

[node-4]:
C:\Program Files\Dolphin Express\Util\..\demo\sciconntest compiled Feb 10 2012 : 00:08:07

Response from remote node 4
-------------------------------------
Local node-id : 8
Local adapter no. : 0
Segment size : 8192
MinSize : 4
Time to run (sec) : 10
Idelay : 0
No Write : 0
Loopdelay : 0
Delay : 0
Bad : 0
Check : 0
Mcheck : 0
Max nodes : 256
rnl : 1
Callbacks : Yes
-------------------------------------
2.9. Making Cluster Application use PCI Express

After the PCI Express hard- and software has been installed and tested, you can configure your cluster application to make use of the increased performance.

2.9.1. Generic Socket Applications

All applications that use generic BSD sockets for communication can be accelerated by SuperSockets without modifying the application binaries. For details, please refer to Section 1, “Make Generic Linux Applications use SuperSockets”.

2.9.2. Kernel Socket Services

SuperSockets can also be used to accelerate kernel services that communicate via the kernel socket API. This requires the kernel modules to be aware of the SuperSockets kernel service. For details, please refer to Section 5, “Kernel Socket Services”

2.9.3. Native SISCI Applications

Native SISCI applications use the SISCI API to use the PCI Express hardware features like transparent remote memory access, DMA transfers or remote interrupts. The SISCI library `libsisci.so` is installed on all Cluster Nodes by default. Any application that uses the SISCI API will be able use the PCI Express interconnect immediately when the network is configured and operational.

**Note**

The SISCI library can be compiled both as a 32-bit version and a 64-bit version on x86_64 Linux platforms.

To compile and link SISCI applications like the MPI-Implementation NMPI, the SISCI-devel RPM needs to be installed on the respective machine. This RPM is built during installation and placed in the node_RPMS and frontend_RPMS directory, respectively.
Chapter 5. Update Installation

This chapter describes how an existing Dolphin PCI Express software stack is to be updated to a new version using the SIA. Dolphin PCI Express software supports "rolling upgrades" to the next release unless explicitly noted otherwise in the release notes.

1. Complete Update

The update installation can be performed in a fully automatic manner without manual intervention. Therefore, this convenient update method is recommended if you can afford some downtime of the whole cluster. Typically, the update of a 16-node cluster takes about 30 minutes.

A complete update is also required in case of protocol incompatibilities between the installed version and the version to be installed. Such incompatibilities are rare and will be described in the release notes. If this is applies, a rolling update is not possible, but you will need to update the system completely in one operation. This will make PCI Express functionality unavailable for the duration of this update.

Proceed as follows to perform the complete update installation:

1. Stop all applications using PCI Express on all Cluster Nodes. This step can be omitted if you choose the --reboot option below.
3. Run the SIA on the Cluster Management Node with any combination of the following options:

   --install-all
   This is the default installation variant and will update all Cluster Nodes and the Cluster Management Node.

   You can specify --install-node or --install-frontend here to update only the current Cluster Node or the Cluster Management Node (you need to execute the SIA on the respective Cluster Node in these cases!)

   --batch
   Using this option, the script will run without any user interaction, assuming the default answers to all questions which would otherwise be posed to the user. This option can safely be used if no configuration changes are needed, and if you know that all services/applications using PCI Express are stopped on the Cluster Nodes.

   --reboot
   Rebooting the Cluster Nodes in the course of the installation will avoid any problems when loading the updated drivers. Such problems can occur because the drivers are currently in use, or due to resource problems. This option is recommended.

   --enforce
   By default, packages on a Cluster Node or the Cluster Management Node will only be updated if the new package has a more recent version than the installed package. This option will enforce the uninstallation of the installed package, followed by the installation of the new package. This option is recommended if you are unsure about the state of the installation.

   As an example, the complete, non-interactive and enforced installation of a specific driver version (provided via the SIA) with a reboot of all Cluster Nodes will be invoked as follows:

   ```
   # sh Dolphin_eXpressWare-<version>.sh --install-all --batch --reboot --enforce
   ```

4. Wait for the SIA to complete. The updated Dolphin PCI Express services will be running on the Cluster Nodes and the Cluster Management Node.
2. Rolling Update

A rolling update will keep your cluster and all its services available on all but one Cluster Node at a time. This kind of update needs to be performed Cluster Node by Cluster Node. It requires that you stop all applications which use the Dolphin PCI Express software stack (like a database server using SuperSockets) on the Cluster Node you intend to update. This means your systems needs to tolerate applications going down on a single Cluster Node.

Before performing a rolling update, please refer to the release notes of the new version to be installed if it supports a rolling update of the version currently installed. If this is not the case, you need to perform a complete update (see previous section).

**Note**

It is possible to install the updated files while the applications are still using PCI Express services. However, in this case the updated PCI Express services will not become active until you restart them (or reboot the machine).

Perform the following steps on each Cluster Node:

1. Log into the Cluster Node and become superuser (root).
2. Build the new binary RPM packages for this Cluster Node:
   
   ```
   # sh Dolphin_eXpressWare-<version>.sh --build-rpm
   ```
   
   The created binary RPM packages will be stored in the subdirectories `node_RPMS` and `frontend_RPMS` which will be created in the current working directory.

   **Tip**

   To save a lot of time, you can use the binary RPM packages built on the first Cluster Node that is updated on all other Cluster Nodes (if they have the same CPU architecture and Linux version). Please see Section 2.3, “Installing from Binary RPMs” for more information.

3. Stop all applications on this Cluster Node that use Dolphin PCI Express services.
4. Stop all Dolphin PCI Express services on this Cluster Node using the `dis_services` command:

   ```
   # dis_services stop
   ```

   **Note**

   The SIA will also try to stop all services when doing an update installation. Performing this step explicitly will just assure that the services can be stopped, and that the applications are shut down properly.

   If you run `dis_admin`, you will notice that this Cluster Node will show up as disabled (not active).

5. Run the SIA with the `--install-node --use-rpms <path>` options to install and updated RPM packages and start the updated drivers and services. The `<path>` parameter to the `--use-rpms` option has to point to the directory where the binary RPM packages have been built (see step 1). If you had run the SIA in `/tmp` in step 1, you would issue the following command:
Adding the option --reboot will reboot the Cluster Node after the installation has been successful. A reboot is not required if the services were shut down successfully in step 4, but recommend to allow the low-level driver the allocation of sufficient memory resources for remote-memory access communication.

**Important**

If the services could not be stopped in step 4, a reboot is required to allow the updated drivers to be loaded. Otherwise, the new drivers will only be installed on disk, but will not be loaded and used.

If for some reason you want to re-install the same version, or even an older version of the Dolphin PCI Express software stack than is currently installed, you need to use the --enforce option.

6. The updated services will be started by the installation and are available for use by the applications. Make sure that Cluster Node has shown up as active (green) in dis_admin again before updating the next Cluster Node.

If the services failed to start, a reboot of the Cluster Node will fix the problem. This can be caused by situations where the memory is too fragmented for the low-level driver (see above).
Chapter 6. Manual Installation

This chapter explains how to manually install the different software packages on the Cluster Nodes and the Cluster Management Node, and how to install the software if the native package format of a platform is not supported by the Dolphin installer SIA.

1. Installation under Load

This section describes how to perform the initial Dolphin PCI Express installation on a cluster in operation without the requirement to stop the whole cluster from operating.

This type of installation does not require more than one Cluster Node at a time being offline.

1. Installing the drivers on the Cluster Nodes

On all Cluster Nodes, run the SIA with the option `--install-node`. This is a local operation which will build and install the drivers on the local machine only. Do not reboot the Cluster Nodes now!

Tip

You can speed up this Cluster Node installation by re-using binary RPMs that have been build on another Cluster Node with the same kernel version and the same CPU architecture. To do so, proceed as follows:

1. After the first installation on a Cluster Node, the binary RPMs are located in the directories `node_RPMS` and `frontend_RPMS`, located in the directory where you launched the SIA. Copy these sub-directories to a path that is accessible from the other Cluster Nodes.

2. When installing on another Cluster Node with the same Linux kernel version and CPU architecture, use the `--use-rpms` option to tell SIA where it can find matching RPMs for this Cluster Node, so it does not have to build them once more.

2. Updating the PCI Express firmware

For an installation under load, perform the following steps for each Cluster Node one by one:

1. Shut down your application processes on the current Cluster Node.

2. Power off the Cluster Node, and install the PCI Express firmware that matches your desired topology and link speed.

3. Power on the Cluster Node and boot it up. The Dolphin PCI Express drivers should load successfully now, although the SuperSockets service will not be configured. Verify this via `dis_services`:

```
# dis_services status
Dolphin kOSIF 5.0.0 is running
Dolphin PX 5.0.0 is running
Dolphin IRM (GX) 5.0.0 ( June 30th 2015 ) is running.
Dolphin Node Manager is running (pid 3172).
Dolphin SISCI 5.0.0 ( June 30th 2015 ) is running.
Dolphin SuperSockets 5.0.0 "St.Martin", June 30th 2015 (built June 30 2015) loaded, but not configured.
```

4. Stop the SuperSockets service:

```
# service dis_supersockets stop
Stopping Dolphin SuperSockets drivers [ OK ]
```

5. Start all your own applications on the current Cluster Node and make sure the whole cluster operates normally.

6. Repeat the same procedure for all Cluster Nodes.
3. **Creating the cluster configuration files**

If you have a Linux machine with X available which can run GUI applications, run the SIA with the `--install-editor` option to install the tool `dis_netconfig`. Ideally, this step is performed on the Cluster Management Node. If this is the case, you should create the directory `/etc/dis` and make it writable for root:

```
# mkdir /etc/dis
# chmod 755 /etc/dis
```

After the SIA has completed the installation, start the tool `dis_mkconf` (default installation location is `/opt/DIS/sbin`):

```
# /opt/DIS/sbin/dis_mkconf
```

or `dis_netconfig` (default installation location is `/opt/DIS/sbin`) for GUI-based installation:

```
# /opt/DIS/sbin/dis_netconfig
```

Information on how to work with this tool can be found in Chapter 4, *Initial Installation*, Section 2.4, “Working with the Dolphin Network Configurator, `dis_netconfig`”. Make sure you create the cabling instructions needed in the next step.

If the `dis_netconfig` was run as root on the Cluster Management Node, proceed with the next step. Otherwise, copy the configuration files `dishosts.conf` and `networkmanager.conf` which you have just created to the Cluster Management Node and place it there under `/etc/dis` (you may need to create this directory, see above).

4. **Cluster Management Node Installation**

On the Cluster Management Node, run the SIA with the `--install-frontend` option. This will start the Network Manager, which will then configure the whole cluster according to the configuration files created in the previous steps.

5. **Start all services on all the Cluster Nodes:**

```
# dis_services start
Starting Dolphin KOSIF 5.0.0 [ OK ]
Starting Dolphin PX 5.0.0 [ OK ]
Starting Dolphin IRM 5.0.0 (June 30th 2015) [ OK ]
Starting Dolphin Node Manager [ OK ]
Starting Dolphin SISCI 5.0.0 (June 30th 2015) [ OK ]
Starting Dolphin SuperSockets drivers [ OK ]
```

6. **Verify the functionality and performance according to Chapter 4, *Initial Installation*, Section 1, “Verifying Functionality and Performance”.

7. **At this point, PCI Express and SuperSockets are ready to use, but your application is still running on Ethernet. To make your application use SuperSockets, you need to perform the following steps on each Cluster Node one-by-one:**

   1. Shut down your application processes on the current Cluster Node.

   2. Refer to Chapter 4, *Initial Installation*, Section 2.9, “Making Cluster Application use PCI Express” to determine the best way to have you application use SuperSockets. Typically, this can be achieved by simply starting the process via the `dis_ssocks_run` wrapper script (located in `/opt/DIS/bin` by default), like:

   3. Start all your own applications on the current Cluster Node and make sure the whole cluster operates normally. Because SuperSockets fall back to Ethernet transparently, your applications will start up normally independently from applications on the other Cluster Nodes already using SuperSockets or not.

After you have performed these steps on all Cluster Nodes, all applications that have been started accordingly will now communicate via SuperSockets.
Note

This single-node installation mode will not adapt the driver configuration `dis_irm.conf` to optimally fit your cluster. This might be necessary for clusters with more than 4 Cluster Nodes. Please refer to Appendix C, Configuration Files, Section 3.1, "dis_irm.conf" to perform recommended changes, or contact Dolphin support.

2. Installation of a Heterogeneous Cluster

This section describes how to perform the initial installation on with heterogeneous Cluster Nodes (different CPU architecture or different operating systems).

Note

Please note that the SIA installer supports installation on a Linux cluster with different Linux kernel versions as long as the architecture is homogeneous. To automatically install on a heterogeneous Linux cluster, please use the SIA option --install-all-hetero. More information can be found in Appendix A, Self-Installing Archive (SIA) Reference.

Note

This single-node installation mode will not adapt the driver configuration `dis_irm.conf` to optimally fit your cluster. This might be necessary for clusters with more than 4 Cluster Nodes. Please refer to Appendix C, Configuration Files, Section 3.1, "dis_irm.conf" to perform recommended changes, or contact Dolphin support.

1. Installing the PCI Express hardware

   Install the SBCs with the correct firmware matching the desired topology.

2. Installing the drivers on the Cluster Nodes

   1. On all Cluster Nodes, run the SIA with the option --install-node. This is a local operation which will build and install the drivers on the local machine only.

   Tip

   You can speed up this Cluster Node installation by re-using binary RPMs that have been build on another Cluster Node with the same kernel version and the same CPU architecture. To do so, proceed as follows:

   1. After the first installation on a Cluster Node, the binary RPMs are located in the directories `node_RPMS` and `frontend_RPMS`, located in the directory where you launched the SIA. Copy these sub-directories to a path that is accessible from the other Cluster Nodes.

   2. When installing on another Cluster Node with the same Linux kernel version and CPU architecture, use the --use-rpms option to tell SIA where it can find matching RPMs for this Cluster Node, so it does not have to build them once more.

   2. The Dolphin PCI Express drivers should load successfully now, although the SuperSockets service will not be configured. Verify this via `dis_services`:

   ```
   # dis_services status
   Dolphin kOSIF 5.0.0 is running
   Dolphin PX 5.0.0 is running
   Dolphin IRM (GX) 5.0.0 (June 30th 2015) is running.
   Dolphin Node Manager is running (pid 3172).
   Dolphin SISCI 5.0.0 (June 30th 2015) is running.
   Dolphin SuperSockets 5.0.0 "St.Martin", June 30th 2015 (built June 30th 2015) loaded, but not configured.
   ```

   3. Stop the SuperSockets service:
3. **Creating the cluster configuration files**

If you have a Linux machine with X available which can run GUI applications, run the SIA with the --install-editor option to install the tool dis_netconfig. Ideally, this step is performed on the Cluster Management Node. If this is the case, you should create the directory /etc/dis and make it writable for root:

```
# mkdir /etc/dis
# chmod 755 /etc/dis
```

After the SIA has completed the installation, start the tool `dis_mkconf` (default installation location is /opt/DIS/sbin):

```
#/opt/DIS/sbin/dis_mkconf
```

or `dis_netconfig` (default installation location is /opt/DIS/sbin) for GUI-based installation:

```
#/opt/DIS/sbin/dis_netconfig
```

Information on how to work with this tool can be found in Chapter 4, *Initial Installation*, Section 2.4, “Working with the Dolphin Network Configurator, dis_netconfig”. Make sure you create the cabling instructions needed in the next step.

If the dis_netconfig or dis_mkconf was run as root on the Cluster Management Node, proceed with the next step. Otherwise, copy the configuration files dishosts.conf and networkmanager.conf which you have just created to the Cluster Management Node and place it there under /etc/dis (you may need to create this directory).

4. On the Cluster Management Node, run the SIA with the --install-frontend option. This will start the Network Manager, which will then configure the whole cluster according to the configuration files created in the previous steps.

5. Start all services on all the Cluster Nodes:

```
# dis_services start
Starting Dolphin kOSIF 5.0.0 [ OK ]
Starting Dolphin PX 5.0.0 [ OK ]
Starting Dolphin IRM 5.0.0 ( June 30th 2015 ) [ OK ]
Starting Dolphin Node Manager [ OK ]
Starting Dolphin SISCI 5.0.0 ( June 30th 2015 ) [ OK ]
Starting Dolphin SuperSockets drivers [ OK ]
```

6. Verify the functionality and performance according to Chapter 7, *Interconnect Maintenance*, Section 1, “Verifying Functionality and Performance”.

### 3. Manual RPM Installation

It is possible to manually install the RPM packages on the Cluster Nodes and the Cluster Management Node. This section describes how to do this if it should be necessary.

#### 3.1. RPM Package Structure

The Dolphin PCI Express software stack is organized into a number of RPM packages. Some of these packages have inter-dependencies.

**Dolphin-PX**

Low-level hardware driver for the adapter. Installs the `dis_irm` kernel module, the node manager daemon and the `dis_irm` and `dis_nodemgr` services on a Cluster Node.

To be installed on all Cluster Nodes.
Dolphin-SISCI
User-level access to the adapter capabilities via the SISCI API. Installs the dis_sisci kernel module and the dis_sisci service, plus the run-time I library and header files for the SISCI API on a Cluster Node. Required by Dolphin benchmark and diagnostic tools.

To be installed on all Cluster Nodes. Depends on Dolphin-PX.

Dolphin-SuperSockets
Standard Berkeley sockets with low latency and high bandwidth. Installs the dis_mbox, dis_msq and dis_ssocks kernel modules and the dis_supersockets service, and the redirection library for preloading on a Cluster Node.

To be installed on all Cluster Nodes. Depends on Dolphin-PX.

Dolphin-NetworkHosts
Installs the GUI application dis_netconfig for creating the cluster configuration files on the Cluster Management Node. Also installs some template configuration files for manual editing.

To be installed on the Cluster Management Node (and additionally other machines that should run dis_netconfig).

Dolphin-NetworkManager
Contains the Network Manager on the Cluster Management Node, which talks to all Cluster Node managers on the Cluster Nodes. Installs the service dis_networkmgr.

To be installed on the Cluster Management Node. Depends on Dolphin-NetworkHosts.

Dolphin-NetworkAdmin
Contains the GUI application dis_admin for managing and monitoring the interconnect. The dis_admin communicates to the Network Manager and can be installed on any machine that has connection to the Cluster Management Node.

To be installed on the Cluster Management Node (or any other machine).

Dolphin-SISCI-devel
Contains the SISCI Development Kit required to compile and link applications that use the SISCI API. This RPM installs the header files and library plus examples and documentation on any machine. The Dolphin-SISCI package is required to run the compiled applications on a Cluster Node.

To be optionally installed on the Cluster Management Node, or any other machine on which SISCI applications should be compiled and linked.

3.2. RPM Build and Installation

On each machine, the matching binary RPM packages need to be built by calling the SIA with the --build-rpm option. This will take some minutes, and the resulting RPMs are stored in three directories:

node_RPMS
Contains the binary RPM packages for the driver and kernel modules to be installed on each Cluster Node. These RPM packages can be installed on every Cluster Node with the same kernel version.

frontend_RPMS
Contains the binary RPM packages for the user-level managing software to be installed on the Cluster Management Node. The Dolphin-SISCI-devel and Dolphin-NetworkAdmin packages can also be installed on other Cluster Nodes, not being the Cluster Management Node or any of the Cluster Nodes, for development and administration, respectively.

source_RPMS
The source RPM packages contained in this directory can be used to build binary RPMs on other machines using the standard rpmbuild command.
To install the packages from one directory, just enter the directory and install them all with a single call of the `rpm` command, like:

```
# cd node_RPMS
# rpm -Uhv *.rpm
```
Chapter 7. Interconnect Maintenance

This chapter describes how to perform a number of typical tasks related to the PCI Express interconnect.

1. Verifying Functionality and Performance

When installing the Dolphin PCI Express software stack (which includes SISCI, SuperSockets and the IPoPCIe (TCP/IP driver) via the SIA, the basic functionality and performance is verified at the end of the installation process by some of the same tests that are described in the following sections. This means, that if the tests performed by the SIA did not report any errors, it is very likely that both, the software and hardware work correctly.

The following sections describe how to verify that the interconnect is setup correctly, which means that all Cluster Nodes can communicate with all other Cluster Nodes via the PCI Express interconnect by sending low-level control packets and performing remote memory access.

1.1. Availability of Drivers and Services

The Cluster Management Node functionality is optional for SISCI based applications but currently mandatory for SuperSockets operation. The Cluster Management Node will automatically distribute configuration files and simplify diagnostic of the cluster. On the Cluster Management Node, only the user-space service dis_networkmgr (the central Network Manager) needs to be running.

Without the required drivers and services running on a Cluster Node, the node will fail to communicate with other nodes. On the Cluster Nodes, the kernel module dis_kosif (providing operating system dependent functionality to all other kernel modules), the kernel services dis_irm (interconnect resources driver) and dis_sisci (upper level hardware services) and dis_ssocks need to be running. Next to these kernel drivers, the user-space service dis_nodemgr (node manager, which talks to the central Network Manager) needs to be active for configuration and monitoring.

Because the drivers do also appear as services, you can query their status with the usual tools of the installed operating system distribution. I.e., for Red Hat-based Linux distributions, you can do

```
# service dis_irm status
Dolphin IRM 5.0.0 ( June 30th 2015 ) is running.
```

Dolphin provides a script dis_services that performs this task for all Dolphin services installed on a machine. It is used in the same way as the individual service command provided by the distribution:

```
# dis_services status
Dolphin KOSIF 5.0.0 is running
Dolphin PX 5.0.0 ( January 13th 2015 ) is running.
Dolphin IRM 5.0.0 ( January 13th 2015 ) is running.
Dolphin Node Manager is running (pid 3172).
Dolphin SISCI 5.0.0 ( June 30th 2015 ) is running.
```

If any of the required services is not running, you will find more information on the problem that may have occurred in the system log facilities. Call dmesg to inspect the kernel messages, and check /var/log/messages for related messages.

1.2. PCIe Connection Test

To ensure that the backplane operates correctly, please perform the PCIe connection test as described in Chapter 4, Initial Installation, Section 2.7.4, “PCIe connection Test”.

1.3. Static PCIe Interconnect Test - dis_diag

The static interconnect test makes sure that all PCIe communication hardware are working correctly by performing a self-test, and determines if the setup and the PCIe routing is correct (matches the actual hardware topology). It will also check all PCIe connections, but this has already been done in the PCIe Connection Test. The tool to perform this test is dis_diag (default location /opt/DIS/sbin/dis_diag).
Running dis_diag on a Cluster Node will perform a self test on the local adapter(s) and list all remote adapters that this adapter can see via the PCI Express interconnect. This means, to perform the static interconnect test on a full cluster, you will basically need to run dis_diag on each Cluster Node and see if any problems with the adapter are reported, and if the adapters in each Cluster Node can see all remote adapters installed in the other Cluster Nodes.

Normally you should invoke dis_diag with no arguments, and it will do a general test and only show the most interesting information. Advanced users may want to enable the full verbose mode by using the -V 9 command line option:

```
dis_diag -V 9
```

The -V 9 option will generate a lot of information, some parts of the information requires knowledge about the PCIe chipset and the PCIe specification in general. The diagnostic module will collect various usage and error information over time. This can be cleared by using the -clear command line option:

```
dis_diag -clear
```

An example output of dis_diag for a Cluster Node which is part of a 3 node Asymmetric cluster topology looks like this:

```
================================================================================
Dolphin diagnostic tool -- dis_diag version 5.4.2 ( Mon Jan 23 18:04:18 CET 2017 )
================================================================================
dis_diag compiled in 64 bit mode
Driver : Dolphin IRM (GX) 5.4.2.0 Jan 20th 2017 (rev 828884a)
Date   : Not available
System : Not available
Number of configured local adapters found: 1

Adapter 0 > Type                       : PCH133
      NodeId                     : 8
      Serial number              : PCH133-AA-039113
      PXH chip family            : PLX_SIRIUS
      PXH chip vendorId          : 0x10b5
      PXH chip device            : 0x8619
      PXH chip revision          : 0xBA
      EEPROM version NTB mode    : 02
      EEPROM vendor info         : 0x0000
      Card revision              : AA
      Topology type              : Asymmetric
      Topology Autodetect        : No
      Number of enabled links    : 2
      Max payload size (MPS)     : 128
      Multicast group size       : 2 MB
      Prefetchable memory size   : 256 MB (BAR2)
      Non-prefetchable size      : 64 MB (BAR4)
      Clock mode upstream        : Port
      Clock mode link            : Global
      PCIe upstream state        : x8, Gen2 (5 GT/s)
      PCIe upstream capabilities : x8, Gen2 (5 GT/s)

******************************************************************************
PXH ADAPTER 0 PCIe0 STATE ******************************************************************************
      PCIe0 uptime               : 98 seconds
      PCIe0 state                : ENABLED
      PCIe0 state                : x4, Gen2 (5 GT/s)
      PCIe0 configured           : x4, Gen2 (5 GT/s)
      PCIe0 capability           : NA
      PCIe0 active               : 1
      PCIe0 configuration        : TRANSPARENT
```

39
The static interconnect test passes if dis_diag delivers **TEST RESULT: *PASSED* and reports the same topology (remote adapters) on all Cluster Nodes.**

### 1.4. Interconnect Load Test

While the static interconnect test sends very a few packets over the links to probe remote nodes, the *Interconnect Load Test* puts significant stress on the interconnect and observes if any data transmissions have to be retried due to link errors. This can happen if the SBC or central switch is not correctly installed or defect. Before running this test, make sure your cluster is connected and configured correctly by running the tests described in the previous sections.

**1.4.1. Test Execution from Dolphin dis_admin GUI**

This test can be performed from within the Dolphin dis_admin GUI tool. Please refer to Appendix B, *dis_admin Reference* for details.

### 1.5. Interconnect Performance Test

Once the correct installation and setup and the basic functionality of the interconnect have been verified, it is possible to perform a set of low-level benchmarks to determine the base-line performance of the interconnect without any additional software layers. The tests that are relevant for this are *scibench2* (streaming remote memory PIO
access performance), scipp (request-response remote memory PIO write performance), dma_bench (streaming remote memory DMA access performance) and intr_bench (remote interrupt performance).

All these tests need to run on two Cluster Nodes (A and B) and are started in the same manner:

1. Determine the NodId of both Cluster Nodes using the query command (default path /opt/DIS/bin/query). The NodId is reported as "Local node-id".

2. On node A, start the server-side benchmark with the options -server and -rn <NodeId of B>, like:
   
   ```sh
   $ scibench2 -server -rn 8
   ```

3. On Cluster Node B, start the client-side benchmark with the options -client and -rn <NodeId of A>, like:

   ```sh
   $ scibench2 -client -rn 4
   ```

4. The test results are reported by the client.

scibench2

Scibench2 measures the streaming bandwidth using CPU based PIO transfers (memcpy)

The following results are measured using a PCI Express Gen3 x8 link

---

<table>
<thead>
<tr>
<th>Segment Size</th>
<th>Average Send Latency</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.07 us</td>
<td>58.31 MBytes/s</td>
</tr>
<tr>
<td>8</td>
<td>0.07 us</td>
<td>117.14 MBytes/s</td>
</tr>
<tr>
<td>16</td>
<td>0.07 us</td>
<td>231.06 MBytes/s</td>
</tr>
<tr>
<td>32</td>
<td>0.07 us</td>
<td>445.08 MBytes/s</td>
</tr>
<tr>
<td>64</td>
<td>0.08 us</td>
<td>838.84 MBytes/s</td>
</tr>
<tr>
<td>128</td>
<td>0.09 us</td>
<td>1483.27 MBytes/s</td>
</tr>
<tr>
<td>256</td>
<td>0.11 us</td>
<td>2408.40 MBytes/s</td>
</tr>
<tr>
<td>512</td>
<td>0.15 us</td>
<td>3497.44 MBytes/s</td>
</tr>
<tr>
<td>1024</td>
<td>0.23 us</td>
<td>4530.20 MBytes/s</td>
</tr>
<tr>
<td>2048</td>
<td>0.39 us</td>
<td>5294.99 MBytes/s</td>
</tr>
<tr>
<td>4096</td>
<td>0.77 us</td>
<td>5308.03 MBytes/s</td>
</tr>
<tr>
<td>8192</td>
<td>1.54 us</td>
<td>5306.65 MBytes/s</td>
</tr>
<tr>
<td>16384</td>
<td>3.10 us</td>
<td>5291.49 MBytes/s</td>
</tr>
<tr>
<td>32768</td>
<td>6.19 us</td>
<td>5294.48 MBytes/s</td>
</tr>
<tr>
<td>65536</td>
<td>12.39 us</td>
<td>5289.90 MBytes/s</td>
</tr>
</tbody>
</table>

Average Send latency is the wall time to write 4 bytes to remote memory

Throughput is the streaming performance using PIO writes to remote memory.

The actual performance will depend on your PCIe link capabilities, x4, x8, Gen1, Gen2, Gen3 etc.

dma_bench

dma_bench measures the streaming DMA bandwidth available through the SISCI API.

The following results are measured using a PCI Express Gen3 x16 link and a PLX chip DMA engine.

---

<table>
<thead>
<tr>
<th>Message size</th>
<th>Total size</th>
<th>Vector length</th>
<th>Transfer time</th>
<th>Latency per message</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>16384</td>
<td>256</td>
<td>35.76 us</td>
<td>0.14 us</td>
<td>458.18 MBytes/s</td>
</tr>
<tr>
<td>128</td>
<td>32768</td>
<td>256</td>
<td>36.81 us</td>
<td>0.14 us</td>
<td>890.24 MBytes/s</td>
</tr>
<tr>
<td>256</td>
<td>65536</td>
<td>256</td>
<td>37.16 us</td>
<td>0.15 us</td>
<td>1763.43 MBytes/s</td>
</tr>
<tr>
<td>512</td>
<td>131072</td>
<td>256</td>
<td>39.36 us</td>
<td>0.15 us</td>
<td>3329.83 MBytes/s</td>
</tr>
<tr>
<td>1024</td>
<td>262144</td>
<td>256</td>
<td>41.34 us</td>
<td>0.16 us</td>
<td>6340.40 MBytes/s</td>
</tr>
<tr>
<td>2048</td>
<td>524288</td>
<td>256</td>
<td>54.75 us</td>
<td>0.21 us</td>
<td>9576.21 MBytes/s</td>
</tr>
<tr>
<td>4096</td>
<td>524288</td>
<td>128</td>
<td>51.83 us</td>
<td>0.40 us</td>
<td>10116.51 MBytes/s</td>
</tr>
</tbody>
</table>
Interconnect Maintenance

<table>
<thead>
<tr>
<th>size</th>
<th>retries</th>
<th>latency (usec)</th>
<th>latency/2 (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2486</td>
<td>1.079</td>
<td>0.539</td>
</tr>
<tr>
<td>4</td>
<td>2406</td>
<td>1.078</td>
<td>0.539</td>
</tr>
<tr>
<td>8</td>
<td>2442</td>
<td>1.090</td>
<td>0.545</td>
</tr>
<tr>
<td>16</td>
<td>2454</td>
<td>1.098</td>
<td>0.549</td>
</tr>
<tr>
<td>32</td>
<td>2482</td>
<td>1.117</td>
<td>0.558</td>
</tr>
<tr>
<td>64</td>
<td>2562</td>
<td>1.151</td>
<td>0.575</td>
</tr>
<tr>
<td>128</td>
<td>2608</td>
<td>1.176</td>
<td>0.588</td>
</tr>
<tr>
<td>256</td>
<td>2667</td>
<td>1.247</td>
<td>0.624</td>
</tr>
<tr>
<td>512</td>
<td>2866</td>
<td>1.331</td>
<td>0.666</td>
</tr>
<tr>
<td>1024</td>
<td>3064</td>
<td>1.492</td>
<td>0.746</td>
</tr>
<tr>
<td>2048</td>
<td>3773</td>
<td>1.880</td>
<td>0.940</td>
</tr>
<tr>
<td>4096</td>
<td>4850</td>
<td>2.659</td>
<td>1.330</td>
</tr>
<tr>
<td>8192</td>
<td>7364</td>
<td>4.247</td>
<td>2.123</td>
</tr>
</tbody>
</table>

The actual performance will depend on your PCIe link capabilities, x4, x8, Gen1, Gen2, Gen3 etc, type of DMA (PCIe chip or system DMA).

DMA is not supported by all SBCs. Please consult the SBC documentation for more details.

scipp

The scipp SISCI benchmark sends a message of the specified size to the remote system. The remote system is polling for incoming data and will send a similar message back to the first node.

The minimal round-trip latency for writing to remote memory is extremely low using PCI Express networks.

The following results are typical for a PCI Express Gen3 x8 link

intr_bench

The interrupt latency is affected by the operating system and can therefore vary.

Average unidirectional interrupt time : 2.515 us.
Average round trip interrupt time : 5.030 us.

To simply gather all relevant low-level performance data, the script `sisci_benchmarks.sh` can be called in the same way. It will run all of the described tests.

2. Replacing a Node

In case a Cluster Node needs to be replaced, proceed as follows concerning the PCIe interconnect:

1. Power down the Cluster Node. When you run `dis_admin` on the Cluster Management Node, you will see the icon of the Cluster Node turn red within the GUI representation of the Cluster Nodes.
2. Replace the selected SBC and make sure the correct firmware is installed.
3. Power up the SBC.
4. Run the SIA with the option `--install-node`. To verify the installation after the SIA has finished:
   1. The icon of the Cluster Node in the `dis_admin` GUI must have turned green again.
   2. The output of the `dis_services` script should list all services as running.
5. Perform the PCIe connection test from within dis_admin to ensure that the cabling is correct (see Chapter 4, Initial Installation, Section 2.7.4, “PCIe connection Test”).

**Warning**

Running the PCIe connection test may stop other traffic on the interconnect for the time the test is running, which can be up to a minute. If this is not an option, please use dis_diag from the commandline to verify the functionality of the interconnect (see Chapter 7, Interconnect Maintenance, Section 1.3, “Static PCIe Interconnect Test - dis_diag”).

Communication between all other Cluster Nodes will continue uninterrupted during this procedure.

### 3. Adding Nodes

To add new Cluster Nodes to the cluster, please proceed as follows:

1. Install the new SBC in the backplane and power it up. Ensure the appropriate PCIe switch firmware is installed on the SBC.

2. Install the Dolphin eXpressWare software stack on all Cluster Nodes via the --install-node option of the SIA.

**Important**

Do not yet start the applications on the new Cluster Nodes that should use SuperSockets, as they would connect to the old Cluster Nodes via Ethernet, and would remain in this mode even when SuperSockets are available.

3. Change the cluster configuration using dis_netconfig:
   1. Load the existing configuration.
   2. In the cluster settings, change the topology to match the topology with all new Cluster Nodes added.
   3. Change the hostnames of the newly added Cluster Nodes via the Cluster Node settings of each Cluster Node. Also make sure that the socket configuration matches those of the existing Cluster Nodes.
   4. Save the new cluster configuration. If desired, create and save or print the cabling instructions for the extended cluster.
   5. If you are not running dis_netconfig or dis_mkconf on the Cluster Management Node, transfer the saved files dishosts.conf and cluster.conf to the directory /etc/dis on the Cluster Management Node.

4. Restart the Network Manager on the Cluster Management Node. If you run dis_admin, the new Cluster Nodes should show up as red icons. All other Cluster Nodes should continue to stay green.

5. Perform the PCIe connection test from within dis_admin to ensure that the PCIe connections are correct (see Chapter 4, Initial Installation, Section 2.7.4, “PCIe connection Test”).

6. Now you can start the applications on the new Cluster Nodes as you did on the old Cluster Nodes, and they will be able to connect to the old Cluster Nodes.

### 4. Removing Nodes

To permanently remove Cluster Nodes from the cluster, please proceed as follows:

1. Change the cluster configuration using dis_netconfig:
   1. Load the existing configuration.
2. In the cluster settings, change the topology to match the topology with all Cluster Nodes removed.

3. The topology change might cut out Cluster Nodes from the cluster at the "wrong" end, you have to make sure that the hostnames and the placement within the new topology for the remaining Cluster Nodes is correct. To do this, change the hostnames of Cluster Nodes by double-clicking their icon and changing the hostname in the displayed dialog box. If the SuperSockets configuration is based on the hostnames (not on the subnet addresses), make sure that the name of the socket interface matches a modified hostname.

4. Save the new cluster configuration. If desired, create and save or print the cabling instructions for the reduced cluster.

5. If you are not running dis_netconfig or dis_mkconf on the Cluster Management Node, transfer the saved files dishosts.conf and cluster.conf to the directory /etc/dis on the Cluster Management Node.

2. Restart the Network Manager on the Cluster Management Node. If you run dis_admin, the removed Cluster Nodes should no longer show up. All other Cluster Nodes should continue to stay green.

3. Disconnect the Cluster Nodes to be removed one by one, making sure that the remaining Cluster Nodes are cabled according to the cabling instructions generated above.

4. On the Cluster Nodes that have been removed from the cluster, the Dolphin eXpressWare software can easily be removed using the SIA option --wipe, like:

   # sh Dolphin_eXpressWare-<version>.sh --wipe

   This will remove all Dolphin software packages, services and configuration data from the Cluster Node.

   If no SIA is available, the same effect can be achieved by manually uninstalling all packages that start with Dolphin-, remove potentially remaining installation directories (like /opt/DIS), and remove the configuration directory /etc/dis.

5. To remove all Dolphin software from the Cluster Node, use the standard package commands to determine which packages are installed, and remove them:

   # rpm -qa | grep Dol
   ...
   # rpm -e <package name>
   ...

6. Perform the cable test from within dis_admin to ensure that the cabling is correct (see Chapter 4, *Initial Installation*, Section 2.7.4, “PCIe connection Test”) for the remaining Cluster Nodes.

**Warning**

Running the cable test will stop other traffic on the interconnect for the time the test is running, which can be up to a minute. If this is not an option, please use dis_diag from the commandline to verify the functionality of the interconnect (see Chapter 7, *Interconnect Maintenance*, Section 1.3, “Static PCIe Interconnect Test - dis_diag”).
Chapter 8. SISCI API

This chapter explains how to use the SISCI API.

SISCI is a powerful remote/shared memory API that makes it easy to write an application that can make local resources available to remote systems and access remote resources using PCI Express. The SISCI API can be used to implement various types of interprocess communication and message passing.

1. SISCI Documentation and resources

The SISCI API specification and the SISCI Users Guide can be found at http://www.dolphinics.com/products/embedded-sisci-developers-kit.html

The eXpressWare installation for all supported operating systems comes with several SISCI example, demo and benchmark programs. The source code is included.

2. Enable applications to use the SISCI API

Applications needs to be written following the SISCI API specification to be able to use SISCI enabled communication.

SISCI uses the PCI Express interconnect for low-latency, high-bandwidth communication inside the cluster.

To run a SISCI application on a Cluster Node, the SISCI library libsisci.so must be installed. This is installed on all Cluster Nodes by default.

3. How to compile your own SISCI application

To compile and link SISCI applications, the SISCI-devel RPM needs to be installed on the development machine. This RPM is built during installation and placed in the node_RPMS and frontend_RPMS directory, respectively.

If you are installing the SISCI-devel package on a computer that also is a Cluster Node, you should use the SISCI-devel package found in the node_RPMS directory. If you are installing the SISCI-devel on the Cluster Management node or a standalone computer, you should use the SISCI-devel package found in the frontend_RPMS directory.

SISCI-devel will by default install the required header files in the directory

/opt/DIS/include/

This can be changed by modifying the rpm installation directory using the --prefix parameter to rpm.

4. SISCI API Demo and Example programs

The eXpressWare development package comes with several example and benchmark programs. It is recommended to study these before designing your own application.

When you install the SISCI-devel RPM, it will place the available SISCI Demo and Example program source in /opt/DIS/src directory. The Cluster Node installation will by default also install the pre-compiled corresponding binaries in /opt/DIS/bin. More information in the next sections.

All example code can be compiled using the Makefile.demo makefile found in /opt/DIS/src.

```
cd /opt/DIS/src
make -f Makefile.demo
```
All SISCI example, demo and benchmark programs supports various command line options, details will be provided during runtime if you start each application with the -help option.

4.1. SISCI API Example programs

The purpose of the example programs is to demonstrate the basic usage of selected SISCI API functionality.

When you install the SISCI-devel RPM, it will place the SISCI example program source in /opt/DIS/src/examples/. The Cluster Node installation will install a precompiled set of these applications into the /opt/DIS/bin directory.

4.1.1. shmem

The shmem program code demonstrates how to create a basic SISCI program and exchange data using PIO. An interrupt is created and signalled when the data exchange is completed.

4.1.2. memcopy

The memcopy program code demonstrates how to create a basic SISCI program and exchange data using PIO. An interrupt is created and signalled when the data exchange is completed.

4.1.3. interrupt

The interrupt program code demonstrates how to trigger an interrupt on a remote system using the SISCI API. The receiver thread is blocking, waiting for the interrupt to arrive.

4.1.4. data_interrupt

The data_interrupt program code demonstrates how to trigger an interrupt with data on a remote system.

4.1.5. intcb

The intcb program code demonstrates how to trigger an interrupt on a remote system. The receiver thread is notified using an interrupt callback function.

4.1.6. lsegcb

The lsegcb program code demonstrates the use of local segment callbacks.

4.1.7. rsegcb

The rsegcb program code demonstrates the use of remote segment callbacks.

4.1.8. dma

The dma program code demonstrates the basic use of DMA operations to move data between segments.

4.1.9. dmacb

The dmacb program code demonstrates the basic use of DMA operations to move data between segments using the completion callback mechanism.

4.1.10. dmavec

The dmavec program code demonstrates how to set up a vectorized DMA operations.

4.1.11. rpcia

The rpcia program code demonstrates how to use the PCIe peer to peer functionality to enable remote systems to access a local PCIe resource / Physical address within the system.
4.1.12. reflective_memory

The reflective_memory program code demonstrates how to use PCIe multicast / reflective memory functionality.

Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.13. reflective_dma

The reflective_dma program code demonstrates how to use multicast / reflective memory with DMA transfers.

Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.14. reflective_device

The reflective_device program code demonstrates how to use the SISCI API to enable a PCIe device to directly utilize the PCIe multicast / reflective memory functionality.

Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.15. reflective_write

The reflective_write program code demonstrates how to use PCIe multicast / reflective memory functionality.

Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.16. probe

The probe program code demonstrates how to determine if a remote system is accessible via the PCIe network.

4.1.17. query

The query program code demonstrates how to identify various system properties and status settings.

4.1.18. cuda

The cuda program demonstrates basic integration with the NVIDIA CUDA® programming environment for GPUs.

The program code demonstrates how to use the SISCI API to attach and access a CUDA GPU buffer.

To use this program, you need to install the CUDA programming environment from NVIDIA.

The Dolphin eXpressWare must also be installed using the --cuda-support option. More information can be found in Section 2.7, “eXpressWare CUDA® integration”

The system having the GPU installed must also support PCI Express Peer 2 Peer transactions (P2P).

4.2. SISCI API demo and benchmarks programs

The purpose of the benchmark and demo programs is to demonstrate how to measure the actual communication performance over the PCIe network.

When you install the SISCI Devel RPM, it will place the SISCI benchmark and demo program source in /opt/DIS/src/demo/. The Cluster Node installation will install a precompiled set of these applications into the /opt/DIS/bin directory.

4.2.1. scibench2

The scibench2 program can be used to determine the actual CPU load/store performance to a remote or local segment.

The program copies data to the remote segment without any synchronization between the client and server side during the benchmark.
The send latency displayed by the application is the wall clock time to send the data once.

4.2.2. scipp

The scipp program can be used to determine the actual CPU store latency to a remote or local segment.

The program will sends data to the remote system. The remote system is polling for new data and will send a similar amount of data back when it detects the incoming message.

4.2.3. dma_bench

The dma_bench program can be used to determine the actual DMA performance to a remote or local segment.

The program connects to a remote segment and executes a series of single sided DMA operations copying data from a local segment to a remote segment. There is no synchronization between the client and server side during the benchmark.

DMA operations is not supported by all systems. The program will fail if the local system does not support PCIe chip or System DMA.

4.2.4. intr_bench

The intr_bench program can be used to determine the actual latency for sending a remote interrupt.

The program implements a interrupt ping - pong benchmark where the client and server sides exchanges interrupts and measures the full round trip latency. The interrupt latency measured by the program will be the average of both systems. The interrupt latency measured is the full application to application latency.

4.2.5. reflective_bench

The reflective_bench program can be used to benchmark the reflective memory / multicast functionality enabled by PCI Express networks.

The program implements a multicast data ping - pong benchmark where the client and server sides exchanges multicast data.

Reflective memory functionality is fully supported in two node configurations and with a central switch.

Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.
Chapter 9. Dolphin PCI Express TCP/IP Driver

1. Who should use the TCP/IP driver

Version 4.4.0 and newer of the Dolphin software stack includes a legacy TCP/IP driver (IPoPCIe) for the PCI Express interconnect. Most applications should NOT use this driver as SuperSockets normally will give much lower latency and higher throughput. However, some applications and services are not supported by SuperSockets and needs to use the TCP/IP driver to benefit from PCI Express. The following applications are known to benefit from the TCP/IP driver:

- Applications that does not need low latency, high throughput.
- NFS, iSCSI and similar very kernel-level services.
- Other Linux kernel services that can't be modified to use SuperSockets

1.1. Enable Linux applications to use the Dolphin PCI Express TCP/IP driver

This section explains how to use the Dolphin PCI Express TCP/IP (IPoPCIe) driver.

DISIP is part of the SuperSockets driver and require that SuperSockets has been enabled and properly configured. Please refer to separate installation/configuration descriptions for SuperSockets in this document.

Assuming SuperSockets is properly configured you should have a 'dis0' network interface available:

```
# ifconfig dis0
```

```
  dis0      Link encap:Ethernet HWaddr 00:00:00:00:00:00
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

To use the adapter you have to configure it with an IP address. You may use any valid IP address.

The dis0-interface supports the common ifconfig configuration parameters like this:

```
# ifconfig dis0 1.2.3.4 netmask 255.255.127.0 (example)
```

Repeat this for all Cluster Nodes which you want to communicate with using DISIP.

You can verify connectivity by using ping against remote Cluster Nodes:

```
# ping 1.2.3.4
```

DISIP should now be ready to be used.
Chapter 10. SuperSockets

This chapter explains how to make any application use SuperSockets, how SuperSockets can be configured, and how to verify SuperSockets usage and gather usage statistics.

1. Make Generic Linux Applications use SuperSockets

All applications that use generic BSD sockets for communication will be accelerated by SuperSockets. No configuration change is required for the application as the same host names/IP v4 addresses can be used.

All relevant socket types are supported by SuperSockets: TCP stream sockets as well as UDP and RDS datagram sockets. Please check the release note for more details.

SuperSockets will use the PCI Express interconnect for low-latency, high-bandwidth communication inside the cluster, and will transparently fall back to Ethernet when connecting to Cluster Nodes outside the cluster, or if a PCI Express connection inside the cluster should become unavailable.

To make an application use SuperSockets, you need to preload the dynamic library libksupersockets.so on application start. This can be achieved by two means as described in the next two sections.

1.1. Launch via wrapper script

To let generic socket applications use SuperSockets, you just need to run them via the wrapper script dis_ssocks_run which sets the LD_PRELOAD environment variable. This script is installed to the bin directory of the installation (default is /opt/DIS/bin) which is added to the default PATH environment variable.

To have i.e. the socket benchmark netperf run via SuperSockets, start the server process on Cluster Node server_name like

\[ \text{dis_ssocks_run netperf} \]

and the client process on any other Cluster Node in the cluster like

\[ \text{dis_ssocks_run netperf -h server_name} \]

1.2. Launch with LD_PRELOAD

As an alternative to using this wrapper script, you can also make sure to set LD_PRELOAD correctly to preload the SuperSockets library, i.e. for sh-style shells such as bash:

\[ \text{export LD_PRELOAD=libksupersockets.so} \]

**Warning**

Do not set LD_PRELOAD to libksupersockets.so globally for a user or even the whole system. This can be a waste of resources, as SuperSockets require additional memory and interconnect resources for each socket. Instead, the preferred way to use SuperSockets is to prepend the dis_ssocks_run command to the invocation of just these processes / applications which are to benefit from SuperSockets.

2. SuperSockets Functionality and Performance

This section describes how to verify that SuperSockets are working correctly on a cluster.

2.1. SuperSockets Status

The general status of SuperSockets can be retrieved via the SuperSockets init script that controls the service dis_supersockets. On Red Hat systems, this can be done like

\[ \text{# service dis_supersockets status} \]
which should show a status of running. If the status shown here is loaded, but not configured, it means that the SuperSockets configuration failed for some reason. Typically, it means that a configuration file could not be parsed correctly. The configuration can be performed manually like

```
# /opt/DIS/sbin/dis_ssocks_cfg
```

If this indicates that a configuration file is corrupted, you can verify them according to the reference in Appendix C, Configuration Files, Section 2, “SuperSockets Configuration”. At any time, you can re-create dishosts.conf using the GUI dis_netconfig or the command line tool dis_mkconf and restore modified SuperSockets configuration files (supersockets_ports.conf and supersockets_profiles.conf) from the default versions that have been installed in /opt/DIS/etc/dis.

Once the status of SuperSockets is running, you can verify their actual configuration via the command dis_ssocks_adm. The output of dis_ssocks_adm -m shows you, which IP address (or network mask) the local Cluster Node’s SuperSockets know about. The output should be non-empty and identical on all Cluster Nodes.

### 2.2. SuperSockets Benchmarks

Several benchmarks that can be used to validate the functionality and performance of SuperSockets is are installed in /opt/DIS/bin/socket.

#### 2.2.1. latency_bench

A benchmark that can be used to validate the functionality and performance of SuperSockets is latency_bench. The basic usage requires two machines (n1 and n2). Start the server process on Cluster Node n1 as server:

```
$ dis_ssocks_run latency_bench -server
```

On Cluster Node n2, run the client side of the benchmark like:

```
$ dis_ssocks_run latency_bench -client n2
```

The latency reported by latency_bench depends on your system, but should starts around 1µs using a modern machine where the PCIe slot is directly attached to the CPU. Using older machines, the latency may be higher. Latencies above 5µs indicate a problem; typical Ethernet latencies start at 20µs and more.

#### 2.2.2. sockperf

A benchmark that can be used to validate the functionality and performance of SuperSockets is installed as /opt/DIS/bin/socket/sockperf. The basic usage requires two machines (n1 and n2). Start the server process on Cluster Node n1 without any parameters:

```
$ dis_ssocks_run sockperf
```

On Cluster Node n2, run the client side of the benchmark like:

```
$ dis_ssocks_run sockperf -h n1
```

The output for a working setup should look like this:

In case of latencies being to high, please verify if SuperSockets are running and configured as described in the previous section. Also, verify that the environment variable LD_PRELOAD is set to libksupersockets.so. This is reported for the client in the second line of the output (see above), but LD_PRELOAD also needs to be set correctly on the server side. See Chapter 4, Initial Installation, Section 2.9, “Making Cluster Application use PCI Express” for more information on how to make generic socket applications (like sockperf) use SuperSockets.

### 3. Troubleshooting

If the applications you are using do not show increased performance, please verify that they use SuperSockets as follows:

1. To verify that the preloading works, use the ldd command on any executable, i.e. the netperf binary mentioned above:
$ export LD_PRELOAD=libksupersockets.so
$ ldd netperf
    libksupersockets.so => /opt/DIS/lib64/libksupersockets.so (0x0000002a95577000)
    libpthread.so.0 => /lib64/tls/libpthread.so.0 (0x00000033ed300000)
    libc.so.6 => /lib64/tls/libc.so.6 (0x00000033ec800000)
    libdl.so.2 => /lib64/libdl.so.2 (0x00000033ecb00000)
    /lib64/ld-linux-x86-64.so.2 (0x00000033ec600000)

The library libksupersockets.so has to be listed at the top position. If this is not the case, make sure the
library file actually exists. The default locations are /opt/DIS/lib/libksupersockets.so and /opt/DIS/
lbin64/libksupersockets.so on 64-bit platforms, and libksupersockets.so actually is a symbolic link
on a library with the same name and a version suffix:

$ ls -lR /opt/DIS/lib*/*ksupersockets*
    -rw-r--r--  1 root root 29498 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.a
    -rw-r--r--  1 root root 901 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.1a
    lrwxrwxrwx  1 root root 25 Nov 14 12:50 /opt/DIS/lib64/libksupersockets.so -> libksupersockets.so.3
    lrwxrwxrwx  1 root root 25 Nov 14 12:50 /opt/DIS/lib64/libksupersockets.so.3 -> libksupersockets.so.3.3.0
    -rw-r--r--  1 root root 65160 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.a
    -rw-r--r--  1 root root 19746 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.a
    -rw-r--r--  1 root root 899 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.1a
    lrwxrwxrwx  1 root root 25 Nov 14 12:50 /opt/DIS/lib64/libksupersockets.so -> libksupersockets.so.3
    lrwxrwxrwx  1 root root 25 Nov 14 12:50 /opt/DIS/lib64/libksupersockets.so.3 -> libksupersockets.so.3.3.0
    -rw-r--r--  1 root root 48731 Nov 14 12:43 /opt/DIS/lib64/libksupersockets.so.3.3.0

Also, make sure that the dynamic linker is configured to find it in this place. The dynamic linker is configured
accordingly on installation of the RPM; if you did not install via RPM, you need to configure the dynamic
linker manually. To verify that the dynamic linking is the problem, set LD_LIBRARY_PATH to include the path
to libksupersockets.so and verify again with ldd:

$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/DIS/lib:/opt/DIS/lib64
$ echo $LD_PRELOAD
libksupersockets.so
$ ldd netperf
    ...

A better solution than setting LD_LIBRARY_PATH is to configure the dynamic linker ld to include these
directories in its search path. Use man ldconfig to learn how to achieve this.

2. You need to make sure that the preloading of the SuperSockets library described above is effective on both
Cluster Nodes, for both applications that should communicate via SuperSockets.

3. Make sure that the SuperSockets kernel module (and the kernel modules it depends on) are loaded and con-
figured correctly on both Cluster Nodes.

1. Check the status of all Dolphin kernel modules via the dis_services script (default location /opt/DIS/
/sbin):

    # dis_services status
    Dolphin kOSIF 5.0.0 is running
    Dolphin FX 5.0.0 is running
    Dolphin IRM 5.0.0 ( June 30th 2015 ) is running.
    Dolphin Node Manager is running (pid 3172).
    Dolphin SISCI 5.0.0 ( June 30th 2015 ) is running.

    At least the services dis_irm and dis_supersockets need to be running, and you should not see a message
about SuperSockets not being configured.

2. Verify that SuperSockets have the correct view of the PCI Express adapters within the cluster. Call dis_ssocks_adm with the option --n:

    root@d0 # dis_ssocks_adm --n
    Local node ID list
    -------------------------------------
    X  4  0
Running this command on all Cluster Nodes should give identical output apart from the marker X which indicates the current Cluster Node.

If this is not the case, the affected Cluster Node has an invalid /etc/dis/dishosts.conf. Make sure that the dis_nodemgr service is up on this Cluster Node, and that the Cluster Node is shown active in the dis_admin GUI.

3. Verify the SuperSockets routing configuration if all cluster Cluster Nodes will connect and communicate via SuperSockets using the right IP addresses. The active configuration can be retrieved via `dis_ssocks_adm -m`:

```
# dis_ssocks_adm -m
IP/net             Adapter    NodeId List
-----------------------------------------------
172.16.5.1/32      0x0000        4    0    0
172.16.5.2/32      0x0000        8    0    0
172.16.5.3/32      0x0000       68    0    0
172.16.5.4/32      0x0000       72    0    0
```

Depending on the configuration variant you used to set up SuperSockets, the content of this file may look different, but it must never be empty and should be identical on all Cluster Nodes. The example above shows a four-node cluster with a single fabric and a static SuperSockets configuration, which will accelerate one socket interface per Cluster Node.

For more information on the configuration of SuperSockets, please refer to Section 1.1, “dishosts.conf”.

4. Make sure that the host names/IP addresses used effectively by the application are the ones that are configured for SuperSockets, especially if the Cluster Nodes have multiple Ethernet interfaces configured.

4. SuperSockets provide an internal event log, which can be accesses via `dis_ssocks_diag`. To attach to the event log and get all events printed to the terminal as they occur, use `dis_ssocks_diag -Ev`. If you then run the application, you will see all connection attempts and their results.

A successful connection attempt of a client towards a server via the PCI Express interconnect will look like this:

```
[Jul 14 14:08:36] TRACE: new SuperSocket created
local:0.0.0.0:0 peer:0.0.0.0:0 pid:3293 obj:0x0ffff880259440800
[Jul 14 14:08:36] TRACE: SuperSockets connection established
local:172.16.6.15:35394 peer:172.16.6.16:5432 pid:3293 obj:0x0ffff88259440800
[Jul 14 14:08:37] TRACE: releasing stream socket
local:172.16.6.15:35394 peer:172.16.6.16:5432 pid:3293 obj:0x0ffff880259440800
```

The server will report the accepted SuperSockets connection like this:

```
[Jul 14 14:10:35] TRACE: native accept succeeded
local:0.0.0.0:5432 peer:172.16.6.15:55215 pid:21472 obj:0x0ffff880257454800
[Jul 14 14:10:35] TRACE: SuperSockets connection accepted
local:172.16.6.16:5432 peer:172.16.6.15:55215 pid:21472 obj:0x0ffff880257454c00
[Jul 14 14:10:35] TRACE: releasing stream socket
local:0.0.0.0:0 peer:0.0.0.0:0 pid:21472 obj:0x0ffff880257454800
[Jul 14 14:10:36] TRACE: releasing stream socket
local:172.16.6.16:5432 peer:172.16.6.15:55215 pid:21472 obj:0x0ffff880257454c00
```

A client's connection towards a server that (the client thinks) is not configured to use SuperSockets is performed via Ethernet and reported as follows:
If a client tries to connect via SuperSockets, but fails to do, it falls back to Ethernet by default. This fall-back capability can be disabled to ensure that SuperSockets and nothing else are actually used if they are to be used. The event log will look like this:

The server may not report this event at all, as it may not got notice of it.

For an explanation of typical error messages, please refer to Section 1, “Software”.

5. Don't forget to check if the port numbers used by this application, or the application itself have been explicitly been excluded from using SuperSockets. By default, only the system port numbers below 1024 are excluded from using SuperSockets, but you should verify the current configuration using `dis_ssocks_adm -p` (see Section 2, “SuperSockets Configuration”).

6. If you can't solve the problem, please contact Dolphin Support. When doing so, please attach

   • the output of `dis_status`
   • the output of `dis_ssocks_diag -Ev` for the connection tries.

4. SuperSockets Utilization

To verify if and how SuperSockets are used on a Cluster Node in operation, the command `dis_ssocks_stat` can be used. The global

```
# dis_ssocks_stat -G
...```

The first line shows the number of open TCP (STREAM) and UDP (DGRAM) sockets that are using SuperSockets.

For more detailed information, the extended statistics need to be enabled. Only the root user can do this:

```
# dis_ssocks_stat -x```

With enabled statistics, `dis_ssocks_stat -G` will display a message size histogram (next to some internal information). When looking at this histogram, please keep in mind that the listed receive sizes (RX) may be incorrect as it refers to the maximal number of bytes that a process wanted to receive when calling the related socket function. Many applications use larger buffers than actually required. Thus, only the send (TX) values are reliable.

To observe the current throughput on all SuperSockets-driven sockets, the tool `dis_ssocks_rate` can be used (see `dis_ssocks_rate`).
5. Kernel Socket Services

SuperSockets can also be used to accelerate kernel services that communicate via sockets. However, such services need to be adapted to actually use SuperSockets. This adaption is typically a trivial modification to make the service use a different address family when opening new sockets. Either the service can be configured to do so, or it needs to be patched and recompiled.

The Distributed Replicated Block Device (DRBD™, see http://www.linbit.com) for Linux has integrated Dolphin PCI Express software support.

If you are interested in accelerated kernel services like iSCSI, GNBD or others, please contact Dolphin Support.

6. Command Reference

dis_ssocks_cfg

dis_ssocks_cfg is run on SuperSockets startup and passes the configuration data from /etc/dis/dishosts.conf, /etc/dis/supersockets_profiles.conf and /etc/dis/supersockets_ports.conf to the SuperSockets kernel module dis_ssocks. Whenever these files are changed, dis_ssocks_cfg needs to be run to propagate these changes into the kernel.

Supported options:

- -v
  verbose output

dis_ssocks_adm

dis_ssocks_adm allows to retrieve current configuration data from the kernel module dis_ssocks, and to perform certain administrative tasks.

Only one operation per invocation is supported:

- -i
  Show the current configuration status.

- -m
  List the mapping of IP addresses, or address masks, to PCI Express NodeIds.

- -n
  List the PCI Express NodeIds that the local node has recognized, and marks the local NodeId.

- -p
  Print the status of the configuration for enabled and disabled ports for SuperSockets.

- -d
  Print the status of the dynamic IP resolution. By default, SuperSockets are configured for static IP resolution.

- -l
  Print the local IP interfaces that SuperSockets has recognized.

Output verbosity:

- -v
  verbose output

Options not useful for typical user:

- -u [0|1]
  To keep configuration data persistent, the kernel module dis_ssocks is blocked from being unloaded if not in use. To manually unload the module, it needs to be unlocked via the option -u 0.
**Note**

The recommended way to control the SuperSockets service, including loading and unloading of the respective kernel modules, is to use the standard SMF for the service `dis_supersockets`.

-`f`
  Print the index value for the AF_SSOCKS address family to `stdout`. The default value is 27, but SuperSockets will choose another available value if 27 is already in use.

-`c [0][1]`
  Enable or disable kernel socket callbacks (default: disabled).

`dis_ssocks_stat`

`dis_ssocks_stat` allows to retrieve SuperSockets usage information and statistics.

-`G`
  Show global SuperSockets usage counters for the different socket types STREAM, DGRAM and RDS. Every open socket is counted, no matter if it is connected or not.

-`t`
  List details of open STREAM sockets.

-`u`
  List details of open DGRAM and RDS sockets.

-`x [0][1]`
  Enable or disable extended statistics. Enabling extended statistics has minor performance impacts (in the order of 100ns per operation).

-`H`
  Generate a histogram for send() and receive() operations, accumulated for all SuperSockets sockets.

-`r`
  Reset all statistics counters.

-`v`
  More verbose output

`dis_ssocks_stat`

`dis_ssocks_stat` allows to retrieve SuperSockets usage information and statistics.

-`G`
  Show global SuperSockets usage counters for the different socket types STREAM, DGRAM and RDS. Every open socket is counted, no matter if it is connected or not.

-`t`
  List details of open STREAM sockets.

-`u`
  List details of open DGRAM and RDS sockets.

-`x [0][1]`
  Enable or disable extended statistics. Enabling extended statistics has minor performance impacts (in the order of 100ns per operation).

-`H`
  Generate a histogram for send() and receive() operations, accumulated for all SuperSockets sockets.

-`r`
  Reset all statistics counters.
dis_ssocks_diag

**dis_ssocks_diag** allows to retrieve SuperSockets events and to perform internal diagnostics.

- **-E**
  Turn on event logging on all levels, and print all events to the console. Stopping the command via `CTRL-C` will turn off event logging.

- **-l [n|a|w|e]**
  Control event logging. Available logging levels are trace (socket creation, connection, accept and release), warning (fall-back connections and fail-over to Ethernet) and error (resource outage and other failures).

  - **n** no logging
  - **a** logging on all levels
  - **w** logging of warnings and errors
  - **e** logging of errors

- **-e**
  Get the next pending event. Will return immediately if no event is pending unless a timeout is specified (`-t`)

- **-t <timeout>**
  Specify timeout in seconds to wait for next event (`-e`) and timeout value of 0 specifies indefinite timeout.

- **-v**
  verbose output

The following options are not useful for the typical user, but can be useful in cooperation with Dolphin support or for developers:

- **-s [0|1|2]**
  Dump the status of STREAM sockets to the kernel log with different verbosity levels.

- **-w**
  Wakeup all connections.

- **-R [0|1|s]**
  Dump the status of DGRAM/RDS receive sockets to the kernel log.

- **-T [0|1]**
  Dump the status of DGRAM/RDS send sockets to the kernel log.

- **-d [0|c|t]**
  Control debug tracing of SuperSockets operation (off, common, transfer)

- **-f [n|s]**
  Explicit control of fall back (n) and fall forward (s) of all connected STREAM sockets.

**dis_ssocks_rate**

**dis_ssocks_rate** is a script that uses **dis_ssocks_stat** to continuously report on the receive (RX) and send (TX) data transfer rate on SuperSockets via PCI Express (and Ethernet for fall-back connections).
-d <seconds>  
  Delay in seconds between measurements. This will cause **dis_ssocks_stat** to loop until interrupted.

-t  
  Print time stamp next to measurement point.

-w  
  Print all output to a single line.

Example:

```bash
# dis_ssocks_stat -d 1 -t
(1 s) RX: 162.82 MB/s  TX: 165.43 MB/s  ( 0 B/s 0 B/s )  Mon Nov 12 17:59:33 CET 2007
(1 s) RX: 149.83 MB/s  TX: 168.65 MB/s  ( 0 B/s 0 B/s )  Mon Nov 12 17:59:34 CET 2007
...
```

The first two pairs show the receive (RX) and send (TX) throughput via PCI Express of all sockets. The number pair in parentheses shows the throughput of sockets that operated by SuperSockets, but are currently in fallback (Ethernet) mode. Typically, there will be no fallback traffic.

**dis_ssocks_run**  
**dis_ssocks_run** is a wrapper script to start applications that should use SuperSockets to accelerate its socket operations.
Chapter 11. Advanced Topics

This chapter deals with techniques like performance analysis and tuning, irregular topologies and debug tools. For most installations, the content of this chapter is not relevant.

1. Notification on Interconnect Status Changes

The Network Manager provides a mechanism to trigger actions when the state of the interconnect changes. The action to be triggered is a user-definable script or executable that is run by the Network Manager when the interconnect status changes.

1.1. Interconnect Status

The interconnect can be in any of these externally visible states:

**UP**
All Cluster Nodes and interconnect links are functional.

**REDUCED**
All Cluster Nodes and interconnect links that can be reached by the Interconnect Manager are functional, but there are some Cluster Nodes where the node manager is not responding (due to a missing Ethernet connection, or that the Dolphin Software has not been installed on the Cluster Node).

**DEGRADED**
All Cluster Nodes are up, but one or more interconnect links have been disabled. Disabling links can either happen manually via dis_admin, or through the Network Manager because of problems reported by the node managers. In status DEGRADED, all Cluster Nodes can still communicate via PCI Express, but the overall performance of the interconnect may be reduced.

**FAILED**
One or more Cluster Nodes are down (the node manager is not reachable via Ethernet), and/or a high number of links has been disabled which isolates one or more Cluster Nodes from the interconnect. These Cluster Nodes can not communicate via PCI Express, but i.e. SuperSockets will fall back to communicate via Ethernet if it is available.

**UNSTABLE**
UNSTABLE is a state which is only visibly externally. If the interconnect is changing states frequently (i.e. because Cluster Nodes are rebooted one after the other), the interconnect will enter the state UNSTABLE. After a certain period of less frequent internal status changes (which are continuously recorded by Network Manager), the external state will again be set to either UP, REDUCED, DEGRADED or FAILED (The first 60 seconds of operation Network Manager will not consider the unstable state).

It is possible for the user to set the "- unstableinterval <interval in minutes>" in networkmanager.conf. If the cluster changes state more than 5 times in the <interval in minutes> then the state will be UNSTABLE. If the <interval in minutes> is set to 0, then this state will never be set. We exit the UNSTABLE state when the above requirement no longer applies. If the user sets the

While in status UNSTABLE, the Network Manager will enable verbose logging (to /var/log/dis_networkmgr.log) to make sure that no internal events are lost.

1.2. Notification Interface

When the Network Manager invokes the specified script or executable, it hands over a number of parameters by setting environment variables. The content of these variables can be evaluated by the script or executable. The following variables are set:

**DIS_FABRIC**
The number of the fabric for which this notification is generated. Can be 0, 1 or 2.
DIS_STATE
The new state of the fabric. Can be either UP, REDUCED, DEGRADED, FAILED or UNSTABLE.

DIS_OLDSTATE
The previous state of the fabric. Can be either UP, REDUCED, DEGRADED, FAILED or UNSTABLE.

DIS_ALERT_TARGET
This variable contains the target address for the notification. This target address is provided by the user when the notification is enabled (see below), and the user needs to make sure that the content of this variable is useful for the chosen alert script. I.e., if the alert script should send an email, the content of this variable needs to be an email address.

DIS_ALERT_VERSION
The version number of this interface (currently 1). It will be increased if incompatible changes to the interface need to be introduced, which could be a change in the possible content of an existing environment variable, or the removal of an environment variable. This is unlikely and does not necessarily make an alert script fail, but a script that relies on this interface in a way where this matters needs to verify the content of this variable.

1.3. Setting Up and Controlling Notification

1.3.1. Configure Notification via the dis_netconfig

Notification on interconnect status changes is done via the dis_netconfig. In the Cluster Edit dialog, tick the check box above Alert target as shown in the screen shot below.

Then enter the alert target and choose the alert script by pressing the button and selecting the script in the file dialog. Dolphin provides an alert script /opt/DIS/etc/dis/alert.sh (for the default installation path) which sends out an email to the specified alert target. Any other executable can be specified here. Please consider that this script will be executed in the context of the user running the Network Manager (typically root), so path settings and permissions should be managed accordingly.

To make the changes done in this dialog effective, you need to save the configuration files (to /etc/dis on the Cluster Management Node) and then restart the Network Manager:

# service dis_networkmgr restart
1.3.2. Configure Notification Manually

If the dis_netconfig cannot be used, it is also possible to configure the notification by editing /etc/dis/networkmanager.conf. Notification is controlled by two options in this file:

-alert_script <file>
   This parameter specifies the alert script `<file>` to be executed.

-alert_target <target>
   This parameter specifies the alert target `<target>` which is passed to the chosen alert script.

To disable notification, these lines can be commented out (prefix them with a `#`).

After the file has been edited, the Network Manager needs to be restarted to make the changes effective:

```bash
# service dis_networkmgr restart
```

1.3.3. Verifying Notification

To verify that notification is actually working, you should provoke a interconnect status change manually. This can easily be done from dis_diag by disabling any link via the Node Settings dialog of any Cluster Node.

1.3.4. Disabling and Enabling Notification Temporarily

Once the notification has been configured, it can be controlled via dis_diag. This is useful if the alerts should be stopped for some time. To disable alerts, open the Cluster Settings dialog and switch the setting next to Alert script as needed.

![Cluster Settings Dialog]

This is a per-session setting and will be lost if the Network Manager is restarted.

**Warning**

Make sure that the messages are enabled again before you quit dis_diag. Otherwise, interconnect status changes will not be notified until the Network Manager is restarted.

2. Managing IRM Resources

A number of resources in the low-level driver IRM (service dis_irm) are run-time limited by parameter in the driver configuration file `/opt/DIS/lib/modules/<kernel version>/dis_irm.conf` (for the default installation...
Advanced Topics

This file contains numerous parameter settings; for those parameters that are relevant for changes by the user, please refer to Appendix C, Configuration Files, Section 3.1, “dis_irm.conf”.

Generally, to change (increase) default limits, dis_irm.conf file needs to be changed on each Cluster Node. Typically, you should edit and test the changes on one Cluster Node, and then copy the file over to all other Cluster Nodes. To make changes in the configuration file effective, you need to restart the dis_irm driver. Because all other drivers depend on it, it is necessary to restart the complete software stack on the Cluster Nodes:

```
# dis_services restart
```

### 2.1. Updates with Modified IRM Configuration

You need to be careful when updating RPMs on the Cluster Nodes with a modified dis_irm.conf. If you directly use RPMs to update the existing Dolphin-SCI RPM like using `rpm -U`, the existing and modified dis_irm.conf will be moved to `dis_irm.conf.rpmsave`, and the default dis_irm.conf will replace previously modified version.

If you update your system with the SIA as described in Chapter 5, Update Installation, SIA will take care that the existing dis_irm.conf will be preserved and stay effective.

### 3. Using dis_diag

Dis.diag is a diagnostic utility that can be used to perform diagnostic of the local PCI Express network controller and network. It will report various events and situations as errors or warnings. dis.diag should only be run on a well configured and functional network not report any errors or warnings.

Dis.diag supports various options, please run dis_diag -h for details

```
[root@scox-2 ~]# dis_diag -h
Usage: dis_diag  [options]
Options:
    -help        This Help
    -v           Verbose output (Max)
    -n           Skip diagnostic. Print local configuration information
    -P           Skip probing of remote nodes / topology investigation
    -A           Skip use of dishosts.conf file. Probe all nodes.
    -a (adapter) Perform diagnostic only for selected adapter
    -V (level)   Verbose Level 0 - 9
    -clear       Clear IRM statistics only
    -noVer       Skip the driver version check

[root@scox-2 ~]#
```

Below you will find an example running dis_diag with verbose level 9. We have added some comments to help explaining some of the details. A high level of PCIe chip knowledge is required to fully understand all details. If you have problems and need help, please send the output of dis_diag -V 9 to support.

```
[root@Dakar-A ~]# dis_diag -V 9
Dolphin diagnostic tool -- dis_diag version 5.2.0 ( Fri Jun 17 18:05:30 CEST 2016 )

<<<<<<<<<<<<<<<<<<<<<<<<<<< VARIOUS INFORMATION >>>>>>>>>>>>>>>>>>>>>>>>>>>

Dolphin diagnostic tool -- dis_diag version 5.2.0 ( Fri Jun 17 18:05:30 CEST 2016 )

Dis.diag compiled in 64 bit mode
Driver : Dolphin IRM (GX) 5.2.0.0 Jun 16th 2016 (rev 37746)
Date   : Fri Jul 15 04:11:50 CEST 2016
System : Linux Dakar-A 2.6.32-573.22.1.el6.x86_64 #1 SMP Wed Mar 23 03:35:39 UTC 2016 x86_64 x86_64 x86_64

Version information both on the driver (release version and date, SVN revision), and Node software.
```
Number of configured local adapters found: 1

Hostbridge : , 0x0

Adapter 0 > Type : PXH810
            NodeId : 4
            Serial number : PXH810-CD-000103
            PXH chip family : PLX_DRACO_2
            PXH chip vendorId : 0x10b5
            PXH chip device : 0x8717
            PXH chip revision : 0xC-A
            EEPROM version NTB mode : 0007
            Card revision : CD

>>> Basic information on this Adapter. Multiple adapters can be shown for systems with more than one adapter.
>>> NodeId shows the NodeId of this Adapter.

            Topology type : Direct 2 nodes
            Topology Autodetect : No
            Number of enabled links : 1
            PCIe slot state : x8, Gen3 (8 GT/s)

>>> Shows the connection the adapter has to the system – should match documentation.

            Clock mode slot : Port
            Clock mode link : Global
            Max payload size (MPS) : 128
            Multicast group size : 2 MB
            Prefetchable memory size : 256 MB (BAR2)
            Non-prefetchable size : 64 MB (BAR4)

>>> Firmware-configurable settings for this adapter.

*************************** PXH ADAPTER 0 LINK 0 STATE ****************************

            Link 0 uptime : 2366245 seconds
            Link 0 state : ENABLED
            Link 0 state : x8, Gen3 (8 GT/s)

>>> Shows the connection the adapter has to this fabric (other adapter/SBC or switch/switchboard)
>>> Also shows the state of this link – here, enabled, and up for several days.
>>> For adapters supporting more than one link, the information is displayed per link.

            Link 0 cable inserted : 1
            Link 0 port active : 1
            Link 0 configuration : NTB

*************************** PXH ADAPTER 0 STATUS ****************************

            Adapter state : 1
            Chip temperature : 0 C
            EEPROM init done : 0

*************************** PXH ADAPTER 0, PARTNER INFORMATION FOR LINK 0 ****************************

            Partner adapter type : PXH810
            Partner serial number : PXH810-000112
            Partner link no : 0
            Partner number of ports : 1

>>> Shows what the information collected from the other adapter/switch/switchboard on this link.
******************** TEST OF ADAPTER 0 ********************

OK: PXH chip alive in adapter 0.
OK: Link alive in adapter 0.
==> Local adapter 0 ok.

******************** TOPOLOGY SEEN FROM ADAPTER 0 ********************

Adapters found: 2
----- List of all nodes found:

Nodes detected: 0004 0008

>>> Lists NodeIds of nodes detected on this fabric. The NodeId of this node (configured above) should be present.
>>> Nodes expected to be reachable but missing will cause a warning to be displayed.
Chapter 12. FAQ

This chapter contains some frequently asked questions and answers.

1. Software

1.1.1. The eXpressWare drivers (IRM etc) fails to load after it has been installed for the first time.

Please follow the procedure below to determine the cause of the problem.

1. Verify that the PCIe chip has been recognized by the machine and that the appropriate firmware has been loaded.

2. Check the kernel message log for relevant messages. This can be done as follows:

   Depending on which messages you see, proceed as described below:

   PXH Driver : Preallocation failed
   The driver failed to preallocate memory which will be used to export memory to remote Cluster Nodes. Rebooting the Cluster Node is the simplest solution to defragment the physical memory space. If this is not possible, or if the message appears even after a reboot, you need to adapt the preallocation settings (see Section 3.1, “ dis_irm.conf ”).

   PXH Driver: Out of vmalloc space
   See FAQ ???.

3. If the driver still fails to load, please contact support and provide the driver's kernel log messages.

   # dmesg > /tmp/syslog_messages.txt

1.1.2. Although the Network Manager is running on the Cluster Management Node, and all Cluster Nodes run the Node Manager, configuration changes are not applied to the adapters. I.e., the NodeId is not changed according to what is specified in /etc/dis/dishosts.conf on the Cluster Management Node.

The adapters in a Cluster Node can only be re-configured when they are not in use. This means, no adapter resources must be allocated via the dis_irm kernel module. To achieve this, make sure that upper layer services that use dis_irm (like dis_sisci and dis_supersockets) are stopped.

On most Linux installations, this can be achieved like this (dis_services is a convenience script that come with the Dolphin software stack):

   # dis_services stop
   ...
   # service dis_irm start
   ...
   # service dis_nodemgr start
   ...

1.1.3. The Network Manager on the Cluster Management Node refuses to start.

In most cases, the interconnect configuration /etc/dis/dishosts.conf is corrupted. This can be verified with the command testdishosts. It will report problems in this configuration file, as in the example below:

   # testdishosts
   socket member node-1_0 does not represent a physical adapter in dishosts.conf
   DISHOSTS: signed32 dishostsAdapternameExists() failed

In this case, the adapter name in the socket definition was misspelled. If testdishosts reports a problem, you can either try to fix /etc/dis/dishosts.conf manually, or re-create it with the Dolphin Network Config-
urator GUI, dis_netconfig or the command line version of the Dolphin Network Configurator, dis_mkconf.

If this does not solve the problem, please check /var/log/dis_networkmgr.log for error messages. If you can not fix the problem reported in this log file, please contact Dolphin support providing the content of the log file.

1.1.4. After a Cluster Node has booted, or after I restarted the Dolphin drivers on a Cluster Node, the first connection to a remote Cluster Node using SuperSockets does only deliver Ethernet performance. Retrying the connection then delivers the expected SuperSockets performance. Why does this happen?

Make sure you run the Cluster Node manager on all Cluster Nodes of the cluster, and the Network Manager on the Cluster Management Node being correctly set up to include all Cluster Nodes in the configuration (/etc/dis/dishosts.conf). The option Automatic Create Session must be enabled.

This will ensure that the low-level "sessions" (Dolphin内部) are set up between all Cluster Nodes of the cluster, and a SuperSockets connection will immediately succeed. Otherwise, the set-up of the sessions will not be done until the first connection between two Cluster Nodes is tried, but this is too late for the first connection to be established via SuperSockets.

1.1.5. Socket benchmarks show that SuperSockets are not active as the minimal latency is much more than 10us.

The half round-trip latency (ping-pong latency) with SuperSockets typically starts between 3 and 4us for very small messages. Any value above 7us for the minimal latency indicates a problem with the SuperSockets configuration, benchmark methodology of something else. Please proceed as follows to determine the reason:

1. Is the SuperSockets service running on both Cluster Nodes? /etc/init.d/dis_supersockets status should report the status running. If the status is stopped, try to start the SuperSockets service with /etc/init.d/dis_supersockets start.

2. Is LD_PRELOAD=libksupersockets.so set on both Cluster Nodes? You can check using the ldd command. Assuming the benchmark you want to run is named sockperf, do ldd sockperf. The libksupersockets.so should appear at the very top of the listing.

3. Are the SuperSockets configured for the interface you are using? This is a possible problem if you have multiple Ethernet interfaces in your Cluster Nodes with the Cluster Nodes having different hostnames for each interface. SuperSockets may be configured to accelerate not all of the available interfaces.

   To verify this, check which IP addresses (or subnet mask) are accelerated by SuperSockets by looking at /proc/net/af_ssocks/socket_maps (Linux) and use those IP addresses (or related hostnames) that are listed in this file.

4. If the SuperSockets service refuses to start, or only starts into the mode running, but not configured, you probably have a corrupted configuration file /etc/dis/dishosts.conf: verify that this file is identical to the same file on the Cluster Management Node. If not, make sure that the Network Manager is running on the Cluster Management Node (/etc/init.d/dis_networkmgr start).

5. If the dishosts.conf files are identical on Cluster Management Node and Cluster Node, they could still be corrupted. Please run the dis_netconfigdis_mkconf on the Cluster Management Node to have it load /etc/dis/dishosts.conf; then save it again (dis_mkconf and dis_netconfig will always create syntactically correct files).

6. Please check the system log using the dmesg command. Any output there from either dis_ssocks or af_ssocks should be noted and reported to <pci-support@dolphinics.com>.

1.1.6. I am running a mixed 32/64-bit platform, and while the benchmarks latency_bench and sockperf from the DIS installation show good performance of SuperSockets, other applications do only show Ethernet performance for socket communication.
Please use the `file` command to verify if the applications that fail to use SuperSockets are 32-bit applications. If they are, please verify if the 32-bit SuperSockets library can be found as `/opt/DIS/lib/libksupersockets.so` (this is a link). If this file is not found, then it could not be built due to a missing or incomplete 32-bit compilation environment on your build machine. This problem is indicated by the message `# WARNING: 32-bit applications may not be able to use SuperSockets of the SIA`.

If on a 64-bit platform 32-bit libraries can not be built, the RPM packages will still be built successfully (without 32-bit libraries included) as many users of 64-bit platforms only run native 64-bit applications. To fix this problem, make sure that the 32-bit versions of the glibc and libgcc-devel (packages) are installed on your build machine, and re-build the binary RPM packages using the SIA option `--build-rpm`, making sure that the warning message shown above does not appear. Then, replace the existing RPM package Dolphin-SuperSockets with the one you have just build. Alternatively, you can perform a complete re-installation.

1.1.7. I have added the statement `export LD_PRELOAD=libksupersockets.so` to my shell profile to enable the use of SuperSockets. This works well on some machines, but on other machines, I get the error message `ERROR: ld.so object 'libksupersockets.so' from LD_PRELOAD cannot be preloaded : ignore whenever I log in. How can this be fixed?

This error message is generated on machines that do not have SuperSockets installed. On these machines, the linker can not find the libksupersockets.so library.

This can be fixed to set the `LD_PRELOAD` environment variable only if SuperSockets are running. For a sh-type shell such as bash, use the following statements in the shell profile (`$HOME/.bashrc`):

```
[ -d /proc/net/af_ssocks ] && export LD_PRELOAD=libksupersockets.so
```

1.1.8. How can I permanently enable the use of SuperSockets for a user?

This can be achieved by setting the `LD_PRELOAD` environment variable in the users' shell profile (i.e. `$HOME/.bashrc` for the `bash` shell). This should be done conditionally by checking if SuperSockets are running on this machine:

```
[ -d /proc/net/af_ssocks ] && export LD_PRELOAD=libksupersockets.so
```

Of course, it is also possible to perform this setting globally (in `/etc/profile`).

1.1.9. I can not build SISCI applications that are able to run on my cluster because the Cluster Management Node (where the SISCI-devel package was installed by the SIA) is a 32-bit machine, while my cluster Cluster Nodes are 64-bit machines (or vice versa). I fail to build the SISCI applications on the Cluster Nodes as the SISCI header files are missing. How can this deadlock be solved?

When the SIA installed the cluster, it has stored the binary RPM packages in different directories `node_RPMS`, `frontend_RPMS` and `source_RPMS`. You will find a SISCI-devel RPM that can be installed on the Cluster Nodes in the `node_RPMS` directory. If you can not find these RPM file, you can recreate them on one of the Cluster Nodes using the SIA with the `--build-rpm` option.

Once you have the Dolphin-SISCI-devel binary RPM, you may need to install it on the Cluster Nodes using the `--force` option of `rpm` because the library files conflict between the installed SISCI and the SISCI-devel RPM:

```
# rpm -i --force Dolphin-SISCI-devel.<arch>.<version>.rpm
```
Appendix A. Self-Installing Archive (SIA) Reference

Dolphin provides the complete software stack as a self-installing archive (SIA). This is a single file that contains binaries, source code as well as a setup script that can perform various operations, i.e. compiling, installing and testing the required software on all Cluster Nodes and the Cluster Management Node. A short usage information will be displayed when calling the SIA archive with the --help option. The complete list of options are available with the --help-verbose option.

The SIA will generate a log file during the installation process. This file will be placed in the current directory.

1. SIA Operating Modes

This section explain the different operations that can be performed by the SIA.

1.1. Full Cluster Installation

The full cluster installation mode will install and test the full cluster in a wizard-like guided installation. All required information will be asked for interactively, and it will be tested if the requirements to perform the installation are met. This mode is the default mode, but can also be selected explicitly.

The installer will look for pre-built RPMs in the current directory and offer to skip the build process. The default answer is No.

Option: --install-all

1.2. Heterogenous Cluster Installation

The heterogenous cluster installation mode will install the full cluster on a cluster of homogenous architecture, but with heterogenous versions of Linux in a wizard-like guided installation. All required information will be asked for interactively, and it will be tested if the requirements to perform the installation are met. To start the heterogenous cluster installation

The installer will look for pre-built Management Node (front_end) RPMs in the current directory and offer to skip the build process. The default answer is No.

Option: --install-all-hetero

1.3. Node Installation

Only build and install the kernel modules and Node Manager service needed to run an interconnect Cluster Node. Kernel header files and the kernel configuration are required (package kernel-devel), but no GUI packages (like qt, qt-devel, X packages).

The installer will look for pre-built RPMs in the current directory and offer to skip the build process. The default answer is No.

Option: --install-node

1.4. Cluster Management Node Installation

The Cluster Management Node installation will only build and install those RPM packages on the local host that are needed to have this host run as a Cluster Management Node. However, due to limitations of the current build system, the kernel headers and configuration are still required for Linux systems.
The installer will look for pre-built RPMs in the current directory and offer to skip the build process. The default answer is No.

Option: --install-frontend

1.5. Installation of Configuration File Editor

Build and install the GUI-based cluster configuration tool dis_netconfig, or the command line version dis_mkconf (if the window manager is not installed, or needed). This tool is used to define the topology of the interconnect and the placement of the Cluster Nodes within this topology. With this information, it can create the detailed cabling instructions (useful to cable non-trivial cluster setups) and the cluster configuration files dishosts.conf and networkmanager.conf needed on the Cluster Management Node by the Network Manager.

Option: --install-editor

1.6. Building RPM Packages Only

Build all source and binary RPMs on the local machine. Both, kernel headers and configuration (kernel-devel) as well as GUI development package (qt-devel) are needed on the local machine.

Option: --build-rpm

1.7. Extraction of Source Archive

It is possible to extract the sources from the SIA as a tar archive DIS.tar.gz in the current directory. This is required to build and install on non-RPM platforms, or when you want source-code access in general.

Option: --get-tarball

2. SIA Options

Next to the different operating modes, a number of options are available that influence the operation. Not all options have an impact on all operating modes.

2.1. Node Specification

In case that you want to specify the list of Cluster Nodes not interactively but on the command line, you can use the option --nodes together with a comma-separated list of hostnames and/or IP addresses to do so.

Example:

```
--nodes n01,n02,n03,n04
```

If this option is provided, existing configuration files like /etc/dis/dishosts.conf will not be considered.

2.2. Installation Path Specification

By default, the complete software stack will be installed to /opt/DIS. To change the installation path, use the --prefix option.

Example:

```
--prefix /usr/dolphin
```

This will install into /usr/dolphin. It is recommended to install into a dedicated directory that is located on a local storage device (not mounted via the network). When doing a full cluster install (--install-all, or default operation), the same installation path will be used on all Cluster Nodes, the Cluster Management Node and potentially the installation machine (if different from the Cluster Management Node).
2.3. Installing from Binary RPMs

If you are re-running an installation for which the binary RPM package have already been built, you can save time by not building these packages again, but use the existing ones. The packages have to be placed in two subdirectories `node_RPMS` and `frontend_RPMS`, just as the SIA does. Then, provide the name of the directory containing these two subdirectories to the installer using the `--use-rpms` option.

Example:
```
--use-rpms $HOME/dolphin
```

The installer does not verify if the provided packages match the installation target, but the RPM installation itself will fail in this case.

2.4. Enforce Installation

If the installed packages should be replaced with the packages build from the SIA you are currently using even if the installed packages are more recent (have a higher version number), use the option `--enforce`. This will enforce the installation of the same software version (the one delivered within this SIA) on all Cluster Nodes and the Cluster Management Node no matter what might be installed on any of these machines. Examples:

```
--enforce
```

2.5. Configuration File Specification

When doing a full cluster install, the installation script will automatically look for the cluster configuration files `dishosts.conf` and `networkmanager.conf` in the default path `/etc/dis` on the installation machine. If these files are not stored in the default path (i.e. because you have created them on another machine or received them from Dolphin and stored them someplace else), you can specify this path using the `--config-dir` option.

Example:
```
--config-dir /tmp
```

The script will look for both configuration files in `/tmp`.

If you need to specify the two configuration files being stored in different locations, use the options `--dishosts-conf <filename>` and `--networkmgr-conf <filename>`, respectively, to specify where each of the configuration files can be found.

2.6. PCIe Link width

The Dolphin eXpressWare drivers have an option to try to enforce a minimum PCIe link width on the PCIe backplane. By using the `--link-width <1|2|4|8|16>` you will set the minimum required link-width in the `/etc/dis/dishosts.conf` configuration file. If your cluster consists of equipment with different capabilities, you need to edit the `/etc/dis/dishosts.conf` to specify the link width for each host.

```
--link-width 8
```

2.7. eXpressWare CUDA® integration

The Dolphin eXpressWare drivers have an option to integrate with the NVIDIA CUDA programming environment. By using the `--cuda-support` option, the installer will configure and enable CUDA applications to use SISCI functionality to do GPU RDMA transfers to/from GPUs over the PCI Express network.

To use this option, the CUDA programming environment and drivers must be installed and configured. The eXpressWare drivers will fail to load if the CUDA environment is uninstalled or not available.

Please carefully follow the instructions and questions provided by the installer.

CUDA® is a parallel computing platform and programming model invented by NVIDIA.
2.8. Batch Mode

In case you want to run the installation unattended, you can use the `--batch` option to have the script assume the default answer for every question that is asked. Additionally, you can avoid most of the console output (but still have the full log file) by providing the option `--quiet`. This option can be very useful if you are upgrading an already installed cluster. I.e., to enforce the installation of newly compiled RPM packages and reboot the Cluster Nodes after the installation, you could issue the following command on the Cluster Management Node:

```
# ./Dolphin_eXpressWare-<version> --batch --reboot --enforce >install.log
```

After this command returns, your cluster is guaranteed to be freshly installed unless any error messages can be found in the file `install.log`.

2.9. Non-GUI Build Mode

When building RPMs only (using the `--build-rpm` option), it is possible to specify that no GUI-applications (`dis_admin` and `dis_netconfig`) should be build. This is done by providing the `--disable-gui` option. This removes the dependency on the QT libraries and header files for the build process.

```
--disable-gui
```

2.10. Software Removal

To remove all software that has been installed via SIA, simply use the `--uninstall` option:

```
--uninstall
```

This will remove all packages from the Cluster Node, and stop all drivers (if they are not in use). A more thorough cleanup, including all configuration data and possible remainings of non-SIA installations, can be achieved with the `--wipe` options:

```
--wipe
```

This option is a superset of `--uninstall`.

```
--wipe-all
```

This will attempt to contact all nodes specified in the `dishosts.conf` file and execute `--wipe` on the whole cluster. The result is a cluster-wide uninstallation.
Appendix B. dis_admin Reference

1. Startup

Check out dis_admin -h for startup options. In order to connect to the Network Manager you may either start dis_admin with the -cluster option, or choose the Connect button after the startup is complete. Type the hostname or IP-address of the machine running the PCI Express Network Manager.

**Note**

Only one dis_admin process can connect to the Network Manager at any time. If you should ever need to connect to the Network Manager while another dis_admin process is blocking the connection, you can restart the Network Manager to terminate this connection. Afterwards, you can connect to the Network Manager from your dis_admin process (which needs to be running on a different machine than the other dis_admin process).

2. Interconnect Status View

2.1. Icons

As a visual tool, dis_admin uses icons to let the user trigger actions and to display information by changing the icon shape or color. The icons with all possible states are listed in the tables below.

<table>
<thead>
<tr>
<th>Table B.1. Node or Adapter State</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Dolphin Network Manager has a valid connection to Dolphin Node Manager on the Cluster Node.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Dolphin Network Manager cannot reach the Cluster Node using TCP/IP, but the PCI Express network reports no additional problems.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>The adapter is wrongly configured, broken or the driver is in an invalid state.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Dolphin Network Manager has detected that the adapter needs firmware update. Please contact support for upgrade.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>Adapter has gone into a faulty state where it cannot read system interrupts and has been isolated by the Dolphin IRM driver.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table B.2. Link State</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>20 Gb/s Green pencil strokes indicates that the links are active. Fat active lines indicates x8 connections.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>10 Gb/s Red pencil strokes indicate that a link is inactive. Typically, a cable is unplugged, or not seated well.</td>
</tr>
</tbody>
</table>
Gray pencil strokes indicate that links have been disabled. Links are typically disabled when they are not in use. Thin green lines indicate active x4 connections.

A red dot (cranberry) indicates that a problem has occurred since the last check. The red dot will disappear if the problem was transient. You may run dis_diag (dis_diag -V 0, dis_diag -V 1 or dis_diag -V 9 differ in verbosity) from the Node menu to investigate the problem more closely. A red dot is also used to signal if a link has been trained to another PCI Express width than the user configured it to.

Yellow pencil strokes indicate that the link is in protected mode. It is probably in the process of becoming active, or the link training may have failed.

2.2. Operation

2.2.1. Cluster Status

The area at the top right informs about the current cluster status and shows settings of dis_admin and the connected Network Manager. A number of settings can be changed in the Cluster Settings dialog that is shown when pressing the Settings button.

- Fabric status shows the current status of the fabric, UP, DEGRADED, FAILED or UNSTABLE (see below).
- Check Interval dis_admin shows the number of seconds between each time the Network Manager sends updates to the Dolphin dis_admin GUI.
- Check Interval Network Manager shows the number of seconds between each time the Network Manager receives updates from the Node Managers.
- Topology shows the current topology of the fabric.

Fabric is UP when all Cluster Nodes are operational and all links OK and therefore plotted in green.
Figure B.1. Fabric is UP

Fabric is DEGRADED when some links operate at x4 in a x8 setup.

Figure B.2. Fabric is DEGRADED
Fabric is REDUCED when all Cluster Nodes are not reachable by Ethernet, but we still have full connectivity for all reachable Cluster Nodes. In the snapshot below jelen-03 is unreachable by Ethernet, but all reachable Cluster Nodes operate fully.

**Figure B.3. Fabric is REDUCED**

Fabric is in status FAILED if several links are broken in a way that breaks the full connectivity. In the snapshot below the PCIe link connecting port 1 of Jelen-03 to port 0 of Jelen-04 is unplugged. Node jelen-03 and jelen-04 can not communicate via PCIe in this situation, and SuperSockets-driven sockets will have fallen back to Ethernet.

**Figure B.4. Fabric has FAILED due to loss of connectivity**
The fabric status is also set to FAILED if one or more Cluster Nodes are dead as this Cluster Node can not be reached via PCI Express in a 2 Cluster Node direct topology. The reason for a Cluster Node being dead can be

- Node is not powered up. Solution: power up the Cluster Node.
- Node has crashed. Solution: reboot the Cluster Node.
- The IRM low-level driver is not running. Solution: start the IRM driver like
  
  
  `# service dis_irm start`

- The node manager is not running. Solution: start the node manager like
  
  `# service dis_nodemgr start`

- The adapter is in an invalid state or is missing. Please check the Cluster Node, and also consider the related topic in the Chapter 12, FAQ.

**Figure B.5. Fabric has FAILED due to dead Cluster Nodes**

### 2.2.2. Node Status

The status of a Cluster Node icon tells if a Cluster Node is up, Ethernet is out, or faulty, and a link is broken (red), disabled (grey) or up (green). When selecting a Cluster Node you will see details in the Node Status area:

To get more information on the interconnect status for a Cluster Node, get its diagnostics via Node Diag -V 1.

- Serial number. A unique serial number calculated based on the local MAC address.
- Adapter Type: The Dolphin part number of the adapter
3. Node and Interconnect Control

3.1. Admin Menu

The items in the Admin menu specifies information that are relevant for the Dolphin Admin GUI

Figure B.6. Options in the Admin menu

- Connect to the Network Manager running on the local or a remote machine.
- Disconnect from the Network Manager.
- Refresh Status of the Cluster Node and interconnect (instead of waiting for the update interval to expire).

3.2. Cluster Menu

The commands in the cluster menu are executed on all Cluster Nodes in parallel and the results are displayed by dis_diag. When choosing one of the fabric options the command will be executed on all Cluster Nodes in that fabric.

Figure B.7. Options in the Cluster menu

Each fabric in the cluster has a sub-menu Fabric <X>. Within this sub-menu, the Diag (-V 0), Diag (-V 1), Diag (-V 9) are diagnostics functions that can be used to get more detailed information about a fabric that shows problem symptoms.

- Diag (-V 0) prints only errors that have been found.
• Diag (-V 1) prints more verbose status information (verbosity level 1).

• Diag (-V 9) prints the full diagnostic information including all error counters (verbosity level 9).

• Diag -clear clears all the error counters. This helps to observe if error counters are changing.

• Diag -prod prints production information about the PCI Express interconnect (serial number, card type, firmware revision etc)

• The Test option is described in Chapter 4, Initial Installation, Section 4.2, “Fabric Test”

The other commands in the Cluster menu are:

• Settings displays the Cluster Settings dialog (see below).

• Reboot cluster nodes reboot all Cluster Nodes after a confirmation.

• Power down cluster nodes powers down all Cluster Nodes after a confirmation.

• Toggle Network Manager Verbose Settings to increase/decrease the amount of logging from the Dolphin Network Manager.

• Ethernet to test the quality of your Ethernet connections in the cluster.

• Test PCIe Connections is described in Chapter 4, Initial Installation, Section 4.1, “Cable Test”

### 3.3. Node Menu

The options in the Node menu are identical to the options in the Cluster and Cluster Fabrics <X> menu, only that commands are executed on the selected Cluster Node only. The only additional option is Settings that is described in the Section 3.5, “Adapter Settings”.

**Figure B.8. Options in the Node menu**

![Node Menu Screenshot](image)

### 3.4. Cluster Settings

The Dolphin Interconnect Manager provides you with several options on how to run the cluster.
Figure B.9. Cluster configuration in dis_admin

All options in this dialog are for experts only. A normal user does not need to alter any of these settings.

- Check Interval Admin alters the number of seconds between each time the Network Manager sends updates to the dis_admin GUI.

- Check Interval Network Manager alters the number of seconds between each time the Network Manager receives updates from the Node Managers.

- Topology specifies that topology that you configured the cluster in, while Topology found displays the auto-determined topology. Changes to the topology setting can be performed with dis_netconfig.

- Remove Session to dead nodes lets you decide whether to remove the session to Cluster Nodes that are unavailable.

- Wait before removing session defines the number of seconds to wait until removing sessions with a Cluster Node that has died or became inaccessible by other means.

- Automatic Create Sessions to new nodes lets you decide if the Network Manager shall create sessions to all available Cluster Nodes.

- Alert script lets you choose to enable/disable the use of a script that may alert the cluster status to an administrator.

- IRM Driver lets you choose to enable/disable the IRM driver.

3.5. Adapter Settings

The Advanced Settings button in the Cluster Node menu allows you to retrieve more detailed information about an adapter and to disable/enable links of this adapter.
Figure B.10. Advanced settings for a Cluster Node

![Adapted Settings Dialog]

- Prefetch Memsize shows the maximum amount of remote memory that can be accessed by this Cluster Node.

  A changed value will not become effective until the IRM driver is restarted on the Cluster Node, which has to be done outside of dis_admin. Setting this value too high (> 512MB) can cause problem with some machines, especially for 32bit platforms.

4. Interconnect Testing & Diagnosis

4.1. Cable Test

Test PCIe Connections tests the cluster for faulty cabling by reading the partner information (serial number, adapter number, port number, adapter type) from the other Cluster Node on individual links. Using this test, you can verify that the cables are connecting the right Cluster Nodes, which means it servers to ensure that the physical cabling matches the interconnect description in the dishosts.conf configuration file.

This test is very useful after a fresh installation, but also every time you worked on the cabling. It will only take a few seconds to complete and display its results in an editor. This allows you to copy or print the test result to fix the described problems right at the cluster.

**Warning**

Please note that while this test is running, all traffic over the PCI Express interconnect will be blocked. Although this will not introduce any communication errors except the delay, it therefore is recommended to run the test on an idle cluster.

SuperSockets will fall back to Ethernet while this test is running.
4.2. Fabric Test

The Test option for each fabric of a cluster verifies the connection quality of the links that make up the fabric. It will search for bad connections by imposing the maximum amount of traffic on individual rings and observe the internal error counters of all adapters involved.

Figure B.11. Result of running cable test on a good cluster

```
[RESULT OF TESTING CONNECTIVITY]
Here is a list of all the cable connections and their status:
Fabric 0: Connection from moe link 0: SCI OUT to notch link 0: SCI IN. [OK]
Fabric 0: Connection from notch link 0: SCI OUT to moe link 0: SCI IN. [OK]
Fabric 0: Connection from red link 0: SCI OUT to milhouse link 0: SCI IN. [OK]
Fabric 0: Connection from milhouse link 0: SCI OUT to red link 0: SCI IN. [OK]
Fabric 0: Connection from moe link 1: SCI OUT to red link 1: SCI IN. [OK]
Fabric 0: Connection from notch link 1: SCI OUT to milhouse link 1: SCI IN. [OK]
Fabric 0: Connection from red link 1: SCI OUT to moe link 1: SCI IN. [OK]
Fabric 0: Connection from milhouse link 1: SCI OUT to notch link 1: SCI IN. [OK]
```

Figure B.12. Result of cable test on a problematic cluster

```
[RESULT OF TESTING CONNECTIVITY]
Here is a list of all the cable connections and their status:
moe [TCP PROBLEM] - Not able to get remote node information.
[ BROKEN CONNECTION] Fabric 0: Inspect connection from notch link 0: SCI OUT to moe link 0: SCI IN. [NOT OK]
Fabric 0: Connection from red link 0: SCI OUT to milhouse link 0: SCI IN. [OK]
Fabric 0: Connection from milhouse link 0: SCI OUT to red link 0: SCI IN. [OK]
moe [TCP PROBLEM] - Not able to get remote node information.
Fabric 0: Connection from notch link 1: SCI OUT to milhouse link 1: SCI IN. [OK]
[ BROKEN CONNECTION] Fabric 0: Inspect connection from red link 1: SCI OUT to moe link 1: SCI IN. [NOT OK]
Fabric 0: Connection from milhouse link 1: SCI OUT to notch link 1: SCI IN. [OK]
```
Note

To perform this test, the SISCI RPM has to be installed on all Cluster Nodes. This is the case if the installation was performed via SIA. If SISCI is not installed on a Cluster Node, an error will be logged and displayed as shown below.

Warning

Please note that while this test is running, all traffic over the interconnect will be blocked. Although this will not introduce any communication errors except the delay, it therefore is recommended to run the test on an idle cluster.

SuperSockets will fall back to Ethernet while this test is running.
Figure B.13. Result of fabric test without installing all the necessary rpms

[result of testing cluster]

Fabric 0 Here is a list of all problematic cable connections:

[SISCI FAILED] Fabric 0, reported on moe: SISCI has to be installed on both moe and notch to run this test.

[SISCI FAILED] Fabric 0, reported on notch: SISCI has to be installed on both notch and moe to run this test.

[SISCI FAILED] Fabric 0, reported on red: SISCI has to be installed on both red and milhouse to run this test.

[SISCI FAILED] Fabric 0, reported on milhouse: SISCI has to be installed on both milhouse and red to run this test.

[SISCI FAILED] Fabric 0, reported on moe: SISCI has to be installed on both moe and notch to run this test.

[SISCI FAILED] Fabric 0, reported on notch: SISCI has to be installed on both notch and moe to run this test.

[SISCI FAILED] Fabric 0, reported on red: SISCI has to be installed on both red and moe to run this test.

[SISCI FAILED] Fabric 0, reported on milhouse: SISCI has to be installed on both milhouse and notch to run this test.
Figure B.14. Result of fabric test on a proper fabric

5. Troubleshooting Best Practice

How to locate a Cluster Node or cable that makes trouble.
Appendix C. Configuration Files

1. Cluster Configuration

A PCI Express cluster requires a dishosts.conf configuration file for the interconnect topology and the SuperSockets acceleration of existing Ethernet networks, and a networkmanager.conf file that contains the basic options for the mandatory Network Manager. Both of these files should be created using the GUI tool dis_netconfig. Using this tool vastly reduces the risk of creating an incorrect configuration file.

1.1. dishosts.conf

The file dishosts.conf is used as a specification of the PCI Express interconnect (in a way just like /etc/hosts specifies Cluster Nodes on a plain IP based network). It is a system wide configuration file and should be located with its full path on all Cluster Nodes at /etc/dis/dishosts.conf.

Templates of this file can be found in /opt/DIS/etc/dis/.

A syntactical and semantic validation of dishosts.conf can be done with the tool testdishosts.

The Dolphin Network Manager and diagnostic tools will always assume that the current file dishosts.conf is valid. If dynamic information read from the network contradicts the information read in the dishosts.conf file, Dolphin Network Manager and diagnostic tools will assume that components are mis configured, faulty or removed for repair.

dishosts.conf is by default automatically distributed to all Cluster Nodes when the Dolphin Network Manager software is started. Therefore, do edit and maintain this file on the Cluster Management Node only. You should create and maintain dishosts.conf by using the Dolphin Network Configurator, dis_netconfig GUI: /opt/DIS/sbin/dis_netconfig or the command line version of the Dolphin Network Configurator, dis_mkconf: /opt/DIS/sbin/dis_mkconf

Normally, there is no reason to edit this file manually. To make changes in dishosts.conf effective, the Network Manager needs to be restarted.

# service dis_networkmgr restart

In case that SuperSockets settings have been changed, dis_ssocks_cfg needs to be run on every Cluster Node as SuperSockets are not controlled by the Network Manager.

The following sections describe the keywords used.

1.1.1. Basic settings

#PX-HARDWARE

The first line of this file should contain an entry specifying what hardware you are going to use. If the #PX-HARDWARE is specified, a PLX based SBC is expected and the parsing of this file will fail if the hardware that you intended to use is other PCI Express chipset.

#IX-HARDWARE

The first line of this file should contain an entry specifying what hardware you are going to use. If the #IX-HARDWARE is specified, an IDT based SBC is expected and the parsing of this file will fail if the hardware that you intended to use is other PCI Express chipset.

DISHOSTVERSION [ 0 | 1 | 2 | 3 | 4]

The version number of the dishosts.conf is specified after the keyword DISHOSTVERSION. The DISHOSTVERSION should be put on the first line of the dishosts file that is not a comment. DISHOSTVERSION 4 or higher is required for the DIS 5.0 software.

HOSTNAME: <hostname/IP>

Each cluster Cluster Node is assigned a unique dishostname, which has to be equal to its hostname. The hostname is typically the network name (as specified in /etc/hosts), or the Cluster Nodes IP-address. Examples:
HOSTNAME: host1.dolphinics.no
HOSTNAME: 193.69.165.21

ADAPTER: <physical adaptername> <nodeId> <adapterNo> <link width>

A Dolphin network Cluster Node may hold several physical adapters. Information about a Cluster Node's physical adapters is listed right below the hostname. All Cluster Nodes specified by a HOSTNAME need at least one physical adapter. This physical adapter has to be specified on the next line after the HOSTNAME. The physical adapters are associated with the keyword ADAPTER. The link width is PCI Express link width x4 or x8.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>name</th>
<th>nodeId</th>
<th>adapter</th>
<th>link width</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER:</td>
<td>host1_a0</td>
<td>4</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>ADAPTER:</td>
<td>host1_a1</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

1.1.2. SuperSockets settings

The SuperSockets configuration is responsible for mapping certain IP addresses to the PCI Express network. It defines which network interfaces are enabled for Dolphin SuperSockets.

It is passed down to the SuperSockets kernel module via the dis_ssocks_cfg command. You need to run this command to make changes in dishosts.conf effective.

**Note**

dis_ssocks_cfg will not be successful if there are established SuperSockets connections. You need to close all existing connections to be able to change the setup.

SOCKET: <host/IP> <physical or virtual adaptername>

Enables the given <host/IP> for SuperSockets using the specified adapter.

In the following example we assume host1 and host2 have two network interfaces each designated host1pub, host1prv, host2pub, host2prv, but only the 'private' interfaces hostXprv are enabled for SuperSockets using a striping adapter:

- SOCKET: host1prv host1_s
- SOCKET: host2prv host2_s

Starting with DISHOSTVERSION 2 SuperSockets can handle dynamic IP-to-NodeId mappings. I.e. a certain IP address does not need to be bound to a fixed machine but can roam in a pool of machines. The address resolution is done at runtime. For such a configuration a new type of adapter must be specified:

SOCKETADAPTER: <socket adaptername> [ SINGLE | REDUNDANT ] <adapterNo> [ <adapterNo> ... ]

This keyword basically only defines an adapter number, which is not associated to any NodeId.

Example:

- SOCKETADAPTER: sockad_s SINGLE 0 1

Defines socket adapter "sockad_s" in standard mode using physical adapters 0.

Such socket adapters can now be used in order to define dynamic mappings, and, in extension to DISHOSTVERSION 1 whole networks can be specified for dynamic mappings:

SOCKET: [ <IP> | <hostname> | <network/mask_bits> ] <socket adapter>

Enables the given address/network for SuperSockets and associates it with a socket adapter.

It is possible to mix dynamic and static mappings, but there must be no conflicting entries.

Example:

- SOCKET: host1 sockad_s
- SOCKET: host2 sockad_s
- SOCKET: host3 host3_s
1.2. networkmanager.conf

The networkmanager.conf specifies the startup parameters for the Dolphin Network Manager. It is created by the dis_netconfig utility.

1.3. cluster.conf

This file must not be edited by the user. It is a status file of the Network Manager that consists of the user-specified settings from networkmanager.conf and derived settings of the Cluster Nodes. It is created by the Network Manager.

2. SuperSockets Configuration

The following sections describe the configuration files that specifically control the behaviour of Dolphin SuperSockets. Next to these files, SuperSockets retrieve important configuration information from dishosts.conf as well (see above).

2.1. supersockets_profiles.conf

/etc/dis/supersockets_profiles.conf defines system-wide protocol settings for all SuperSockets applications using LD_PRELOAD. It can be used to fine-tune the behaviour of SuperSockets to specific requirements or applications. However, in most cases, the default setting will work well, and you need a good understanding to use these options usefully.

The options set in /etc/dis/supersockets_profiles.conf are read when a process opens its first socket, and then applied to all sockets that this process will open. This means, that established socket connections, and sockets that are created by processes that had already sockets open before supersockets_profiles.conf was changed, are not effected by changes you make to this file.

Additionally, all settings of supersockets_profiles.conf described below can be overridden by environment variables named SSOCKS_<option>. Example:

export SSOCKS_DISABLE_FALLBACK=1

Empty lines and lines starting with # are ignored.

SYSTEM_POLL [ 0 | 1 ]

Usage of poll/select optimization. Default is 0 which means that the SuperSockets optimization for the poll() and select() system calls is used. This optimization typically reduces the latency without increasing the CPU load. To only use the native system methods for poll() and select(), set this value to 1.

RX_POLL_TIME <int>

Receive poll time [µs]. Default is 30. Increasing this value may reduce the latency as the CPU will spin longer to wait for new data until it blocks sleeping. Reducing this value will send the CPU to sleep earlier, but this may increase message latency.

TX_POLL_TIME <int>

Transmit poll time [µs]. Default is 0, which means that the CPU does not spin at all when a no buffers at the receiving side are available. Instead, it will immediately block until the receiver reads data from these buffers (which makes buffer space available again for sending). The situation of no available receive buffers does rarely occur, and increasing this value is not recommended.

MSQ_BUF_SIZE <int>

Message buffer size [byte]. Default is 128KB. This value determines how much data can be sent without the receiver reading it. It has no significant impact on bandwidth.

MIN_DMA_SIZE <int>

Minimum message size for DMA [byte]. Default is 0 (DMA disabled).
MAX_DMA_GATHER <int>
Maximum number of messages gather into a single DMA transfer. Default is 1.

MIN_SHORT_SIZE <int>
Switch point [byte] from INLINE to SHORT protocol. Default depends on driver.

MIN_LONG_SIZE <int>
Switch point [byte] from SHORT to LONG protocol. Default depends on driver.

FAST_GTOD [ 0 | 1 ]
Usage of accelerated gettimeofday(). Default is 0 which disables this optimization. Set to 1 to enable it.

DISABLE_FALLBACK [ 0 | 1 ]
Control fall-back from SuperSockets to native sockets. Default is 0, which means fall-back (and fall-forward) is enabled. To ensure that only SuperSockets are used (i.e. for benchmarking), set it to 1.

ASYNC_PIO [ 0 | 1 ]
Usage of fully asynchronous transfers. Default is 0, which means that all data transfers are performed by the CPU running the process that called the send function. To enable asynchronous transfers, set this option to 1. If enabled, the SHORT and LONG protocol is processed by a dedicated kernel thread. By this, the sending process is available immediately, and the actual data transfer is performed asynchronously. This generally increases throughput and reduces CPU load with affecting small message latency.

FASTLOOPBACK [ 0 | 1 ]
Control acceleration of loop-back connections. Default is 1, that means that all loop-back connections are accelerated by SuperSockets. Set it to 0 if you don't want loop-back connections to be handled by SuperSockets.

2.2. supersockets_ports.conf
/etc/dis/supersockets_ports.conf is used to configure the port filter for SuperSockets. If no such file exists all ports will be enabled by default. It is, however, recommended to exclude all system ports. A suitable port configuration file is part of the SuperSockets software package. You can adjust it to your specific needs.

To make changes in supersockets_ports.conf effective, you need to run dis_ssocks_cfg on this Cluster Node. Changes do not affect established socket connections.

# Default port configuration for Dolphin SuperSockets
# Ports specifically enabled or disabled to run over SuperSockets.
# Any port not specifically covered, is handled by the default:
EnablePortsByDefault yes

# Recommended settings:

# Disable the privileged ports used by system services.
DisablePortRange tcp 1 1023
DisablePortRange udp 1 1023

# Disable Dolphin Interconnect Manager service ports.
DisablePortRange tcp 3443 3444

The following keywords are valid:

EnablePortsByDefault [ yes | no ]
Determines the policy for unspecified ports.

DisablePortRange [ tcp | udp ] <from> <to>
Explicitly disables the given port range for the given socket type.

EnablePortRange [ tcp | udp ] <from> <to>
Explicitly enables the given port range for the given socket type.
3. Driver Configuration

The Dolphin drivers are designed to adapt to the environment they are operating in; therefore, manual configuration is rarely required. The upper limit for memory allocation of the low-level driver is the only setting that may need to be adapted for a cluster, but this is also done automatically during the installation.

**Warning**

Changing parameters in these files can affect reliability and performance of the PCI Express interconnect.

3.1. **dis irm.conf**

*dis irm.conf* is located in the *lib/modules* directory of the DIS installation (default */opt/DIS*) and contains options for the hardware driver (*dis irm* kernel module). Only a few options are to be modified by the user. These options deal with the memory pre-allocation in the driver:

**Warning**

Changing other values in *dis irm.conf* than those described below may cause the interconnect to malfunction. Only do so if instructed by Dolphin support.

Whenever a setting in this file is changed, the driver needs to be reloaded to make the new settings effective. Please note that some of the possible settings are commented out in the *dis irm.conf* file. Please remove the leading # to change these settings.

### 3.1.1. Resource Limitations

These parameters control memory allocations that are only performed on driver initialization.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dis_max_segment_size_megabytes</td>
<td>Sets the maximum size of a memory segment that can be allocated for remote access. Some systems may lock up if too much memory is requested.</td>
<td>MB</td>
<td>integers &gt; 0</td>
<td>4</td>
</tr>
<tr>
<td>max-vc-number</td>
<td>Maximum number of virtual channels (one virtual channel is needed per remote memory connection; i.e. 2 per SuperSocket connection)</td>
<td>n/a</td>
<td>integers &gt; 0</td>
<td>1024</td>
</tr>
</tbody>
</table>

The upper limit is the consumed memory; values > 16384 are typically not necessary.

### 3.1.2. Real time behavior

These parameters control some driver real-time settings. Changes here are normally only needed if you run a real time application or simulation using the SISCI API.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>linkWatchdogEnabled</td>
<td>Controls the link watchdog behaviour. The link watchdog is a high availability feature to ensure detection of non</td>
<td>Seconds</td>
<td>0 : Disabled. Integers &gt; 0 : Watchdog</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>period in seconds.</td>
<td></td>
</tr>
</tbody>
</table>
### Configuration Files

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>operational links</td>
<td>This feature is normally not needed but should be left on for additional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>high availability. The feature introduces a microsecond level jitter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Should be turned off for real-time applications.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sessionHeartbeatsEnabled</td>
<td>Controls the session heartbeats. The session heartbeat mechanism is used</td>
<td>n/a</td>
<td>0 : Disabled</td>
<td>1 - enabled</td>
</tr>
<tr>
<td></td>
<td>for end to end internal heart beating. This feature introduces a microsecond</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>level jitter. Should be turned off for real-time applications.</td>
<td></td>
<td>1 : Enabled.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.3. Memory Preallocation

Preallocation of memory is recommended on systems without IOMMU (like x86 and x86_64). The problem is the memory fragmentation over time which can cause problems to allocate large segments of contiguous physical memory after the system has been running for some time. To overcome this situation, options has been added to let the IRM driver allocate blocks of memory upon initialization and to provide memory from this pool under certain conditions for allocation of remotely accessible memory segments.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>number-of-megabytes-preallocated</td>
<td>Defines the number of MiB memory the IRM shall try to allocate upon</td>
<td>MB</td>
<td>0: disable preallocation</td>
<td>16 (may be increased by the installer</td>
</tr>
<tr>
<td></td>
<td>initialization.</td>
<td></td>
<td>&gt;0: MB to preallocate in as few blocks</td>
<td>script)</td>
</tr>
<tr>
<td></td>
<td>as possible.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use-sub-pools-for-preallocation</td>
<td>If the IRM fails to allocate the amount memory specified by</td>
<td>n/a</td>
<td>0: disable sub-pools</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>number-of-megabytes-preallocated it will by default repetitively decrease</td>
<td></td>
<td>1: enable sub-pools</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the amount and retry until success. By enabling use-sub-pools-for-preallocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the IRM will continue allocate memory (possibly in small chunks),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>until the amount specified by number-of-megabytes-preallocated is reached.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option Name</td>
<td>Description</td>
<td>Unit</td>
<td>Valid Values</td>
<td>Default Value</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>block-size-of-preallocated-blocks</td>
<td>To allocate not a single large block, but multiple blocks of the same size, this parameter has to be set to a value &gt; 0. Pre-allocating memory this way is useful if the application to be run on the cluster uses many memory segments of the same (relatively small) size.</td>
<td>bytes</td>
<td>0: don't preallocate memory in this manner</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: size in bytes (will be aligned upwards to page size boundary) of each memory block.</td>
<td></td>
</tr>
<tr>
<td>number-of-preallocated-blocks</td>
<td>The number of block to be pre-allocated (see previous parameter)</td>
<td>n/a</td>
<td>0: don't preallocate memory in this manner</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: number of blocks</td>
<td></td>
</tr>
<tr>
<td>minimum-size-to-allocate-from-preallocated-pool</td>
<td>This sets a lower limit on the size of memory segments the IRM may try to allocate from the preallocated pool. The IRM will always request the system for resolving memory request less than this size. Due to the aspect of the need of the preallocation mechanism, there is a &quot;hard&quot; lower limit of one SCI_PAGE (currently 8K). The minimum size is defined in 1K blocks.</td>
<td>KB</td>
<td>0: always allocate from pre-allocated memory</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: try to allocate memory that is smaller than this value from non-preallocated memory</td>
<td></td>
</tr>
<tr>
<td>try-first-to-allocate-from-preallocated-pool</td>
<td>Directs the IRM when to try to use memory from the preallocated pool.</td>
<td>n/a</td>
<td>0: The preallocated memory pool becomes a backup solution, only to be used when the system can't honor a request for additional memory. 1:IRM to prefers to allocate memory from the preallocated pool when possible.</td>
<td>1</td>
</tr>
</tbody>
</table>

### 3.1.4. Logging and Messages

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>link-messages-enabled</td>
<td>Control logging of non critical link messages during operation.</td>
<td>n/a</td>
<td>0: no link messages</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: show link messages</td>
<td></td>
</tr>
</tbody>
</table>
### Configuration Files

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Unit</th>
<th>Valid Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>notes-disabled</td>
<td>Control logging of non-critical notices during operation.</td>
<td>n/a</td>
<td>0: show notice messages</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: no notice messages</td>
<td></td>
</tr>
<tr>
<td>warn-disabled</td>
<td>Control logging of general warnings during operation.</td>
<td>n/a</td>
<td>0: show warning messages</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: no warning messages</td>
<td></td>
</tr>
<tr>
<td>dis_report_resource_outtages</td>
<td>Control logging of out-of-resource messages during operation.</td>
<td>n/a</td>
<td>0: no messages</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: show messages</td>
<td></td>
</tr>
<tr>
<td>notes-on-log-file-only</td>
<td>Control printing of driver messages to the system console</td>
<td>n/a</td>
<td>0: also print to console</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: only print to kernel message log</td>
<td></td>
</tr>
</tbody>
</table>

### 3.2. dis_ssocks.conf

dis_ssocks.conf is a configuration file for the SuperSockets (dis_ssocks) kernel module. The values defined within this file are passed to the dis_ssocks kernel module when it is loaded (part of SuperSockets startup).

If a value different from the default is required, edit and uncomment (remove the `#`) the appropriate line.

```plaintext
#address_family=27;
#rds_compat=0;
```

The following keywords are valid:

**address_family**

*AF_SSSOCKS address family index.* Default value is 27. If not set, the driver will automatically choose another index between 27 and 32 until it finds an unused index. The index currently used can be retrieved via the `/proc` file system like `cat /proc/net/af_ssocks/family`.

If this value is set explicitly in dis_ssocks.conf, this value will be chosen, and no search for unused values is performed if this value should already be taken (SuperSockets startup will fail).

Generally, this value is only required if SuperSockets should be used explicitly without the preload library, like when using SuperSockets within the kernel.

**rds_compat**

*RDS compatibility level.* Default is 0.
Appendix D. Platform Issues and Software Limitations

This chapter lists known issues of PCI Express with certain hardware platforms and limitations of the software stack. Some of these limitations can be overcome by changing default settings of runtime parameters to match your requirements.

1. Platforms with Known Problems vs Dolphin PCI Express software

Low PIO performance
PowerPC systems do not support CPU write combining. The effect of this is that every CPU store will be transmitted to the remote memory as a small PCIe transfer. This will normally have a huge impact on the write performance compared to e.g. an Intel CPU running at the same speed. PowerPC users should use DMA operations to transfer larger amounts of data.

2. IRM

Resource Limitations
The IRM (Interconnect Resource Manager) manages the hardware and related software resources of the PCI Express interconnect. Some resources are allocated once when the IRM is loaded. The default setting are sufficient for typical cluster sizes and usage scenarios. Please contact Dolphin support if you hit a resource limitation.

3. SISCI

Please consult the software release note for details.

Heterogeneous Cluster Operation (Endianess)
SISCI supports clusters where Cluster Nodes use different endian representation (mixing little endian or big endian). Managing the data in a bi-endian configuration is the responsibility of the application.

4. SuperSockets

Please consult the software release note for details.

Heterogeneous Cluster Operation (Endianess)
SuperSockets supports clusters where Cluster Nodes use different endian representation (mixing little endian or big endian). Managing the data in a bi-endian configuration is the responsibility of the application.

Sending and Receiving Vectors
The vector length for the `writev()` and `sendmsg()` functions is limited to 16. For `readv()` and `recvmsg()`, the vector length is not limited.

Socket Options
The following socket options are supported by SuperSockets for communication over PCI Express:

- `SO_DONTROUTE` (implicit, as SuperSockets don't use IP packets for data transport and thus are never routable)
- `TCP_NODELAY`
- `SO_REUSEADDR`
- `SO_TYPE`
The following socket options are passed to the native (fall-back) socket:

- SO_SENDBUF and SO_RECVBUF (the buffer size for the SuperSockets is fixed).

All other socket options are not supported (ignored).

Fall-forward for Stream Sockets

While SuperSockets offer fully transparent fall-back and fall-forward between PCI Express-based communication and native (Ethernet) communication for any socket (TCP or UDP) while it is open and used, there is currently a limitation on sockets when they connect: a socket that has been created via SuperSockets and connected to a remote socket while the PCI Express interconnect was not operational will not fall forward to PCI Express communication when the interconnect comes up again. Instead, it will continue to communicate via the native network (Ethernet).

This is a rare condition that typically will not affect operation. If you suspect that one Cluster Node performs not up to the expectations, you can either contact Dolphin support to help you diagnose the problem, or restart the application making sure that the PCI Express interconnect is up.

Removal of this limitation as well as a simple way to diagnose the precise state of a SuperSockets-driven socket is scheduled for updated versions of SuperSockets.

Resource Limitations

SuperSockets allocate resources for the communication via PCI Express by means of the IRM. Therefore, the resource limitations listed for the IRM indirectly apply to SuperSockets as well. To resolve such limitations if they occur (i.e. when using very large number of sockets per Cluster Node), please refer to the relevant IRM section above. SuperSockets logs messages to the kernel message log for two typical out-of-resources situations:

- **No more VCs available.** The maximum number of virtual channels needs to be increased.

- **No more segment memory available.** The amount of pre-allocated memory needs to be increased