Red Hat Enterprise Linux 8

Configuring InfiniBand and RDMA networks

A guide to configuring InfiniBand and RDMA networks on Red Hat Enterprise Linux 8

Last Updated: 2022-01-14

Red Hat Enterprise Linux 8 Configuring InfiniBand and RDMA networks

A guide to configuring InfiniBand and RDMA networks on Red Hat Enterprise Linux 8

Legal Notice

Abstract

This document describes what InfiniBand and remote direct memory access (RDMA) are and how to configure InfiniBand hardware. Additionally, this documentation explains how to configure InfiniBand-related services.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. UNDERSTANDING INFINIBAND AND RDMA

InfiniBand refers to two distinct things:

- The physical link-layer protocol for InfiniBand networks
- The InfiniBand Verbs API, which is an implementation of the remote direct memory access (RDMA) technology

RDMA provides access to the memory from one computer to the memory of another computer without involving either computer's operating system. This technology enables high-throughput and low-latency networking with low CPU utilization.

In a typical IP data transfer, when an application on one machine sends data to an application on another machine, the following happens on the receiving side:

1. The kernel must receive the data.
2. The kernel must determine that the data belongs to the application.
3. The kernel wakes up the application.
4. The kernel waits for the application to perform a system call into the kernel.
5. The application copies the data from the kernel's own internal memory space into the buffer provided by the application.

This process means that most network traffic is copied across the main memory of the system if the host adapter uses direct memory access (DMA), or otherwise at least twice. Additionally, the computer executes a number of context switches to switch between the kernel and application context. Both context switches can cause a high CPU load at high traffic rates and slow down other tasks.

RDMA communication bypasses the kernel intervention in the communication process, unlike the normal IP communication. This reduces the CPU overhead. The RDMA protocol enables the host adapter to know when a packet comes in from the network, which application should receive it, and where in the application's memory space, the packet should be stored. Instead of sending the packet to the kernel to be processed and then copied into the user application's memory, with InfiniBand, the host adapter places the contents of the packet directly in the application's buffer. This process requires a separate API, the InfiniBand Verbs API, and applications must support this API before they can use RDMA.

Red Hat Enterprise Linux 8 supports both the InfiniBand hardware and the InfiniBand Verbs API. Additionally, Red Hat Enterprise Linux supports the following technologies that allow to use the InfiniBand Verbs API on non-InfiniBand hardware:

- Internet Wide Area RDMA Protocol (iWARP): A network protocol that implements RDMA over IP networks.
- RDMA over Converged Ethernet (RoCE), which is also known as InfiniBand over Ethernet (IBoE): A network protocol that implements RDMA over Ethernet networks.

Additional resources

- Configuring RoCE

CHAPTER 2. CONFIGURING ROCE

This section explains background information about RDMA over Converged Ethernet (RoCE), as well as how to change the default RoCE version, and how to configure a software RoCE adapter.

Note that there are different vendors, such as Mellanox, Broadcom, and QLogic, who provide RoCE hardware.

2.1. OVERVIEW OF ROCE PROTOCOL VERSIONS

RoCE is a network protocol that enables remote direct memory access (RDMA) over Ethernet.

The following are the different RoCE versions:

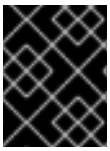
RoCE v1

The RoCE version 1 protocol is an Ethernet link layer protocol with ethertype **0x8915** that enables communication between any two hosts in the same Ethernet broadcast domain.
By default, when using a Mellanox ConnectX-3 network adapter, Red Hat Enterprise Linux uses RoCE v1 for the RDMA Connection Manager (RDMA_CM).

RoCE v2

The RoCE version 2 protocol exists on top of either the UDP over IPv4 or the UDP over IPv6 protocol. The UDP destination port number 4791 is reserved for RoCE v2.
By default, when using a Mellanox ConnectX-3 Pro, ConnectX-4 Lx, or ConnectX-5 network adapter, Red Hat Enterprise Linux uses RoCE v2 for the RDMA_CM, but the hardware supports both RoCE v1 and RoCE v2.

The RDMA_CM sets up a reliable connection between a client and a server for transferring data. RDMA_CM provides an RDMA transport-neutral interface for establishing connections. The communication uses a specific RDMA device, and data transfers are message-based.



IMPORTANT

Using RoCE v2 on the client and RoCE v1 on the server is not supported. In this case, configure both the server and client to communicate over RoCE v1.

Additional resources

- Temporarily changing the default RoCE version

2.2. TEMPORARILY CHANGING THE DEFAULT ROCE VERSION

Using the RoCE v2 protocol on the client and RoCE v1 on the server is not supported. If the hardware in your server only supports RoCE v1, configure your clients to communicate with the server using RoCE v1. This section describes how to enforce RoCE v1 on the client that uses the **mlx5_0** driver for the Mellanox ConnectX-5 Infiniband device. Note that the changes described in this section are only temporary until you reboot the host.

Prerequisites

- The client uses an InfiniBand device that uses, by default, the RoCE v2 protocol.

- The InfiniBand device in the server only supports RoCE v1.

Procedure

1. Create the **/sys/kernel/config/rdma_cm/*mlx5_0*/** directory:

```
# mkdir /sys/kernel/config/rdma_cm/mlx5_0/
```

2. Display the default RoCE mode. For example, to display the mode for port 1:

```
# cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode

RoCE v2
```

3. Change the default RoCE mode to version 1:

```
# echo "IB/RoCE v1" > /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
```

2.3. CONFIGURING SOFT-ROCE

Soft-RoCE is a software implementation of remote direct memory access (RDMA) over Ethernet, which is also called RXE. Use Soft-RoCE on hosts without RoCE host channel adapters (HCA).

IMPORTANT

The Soft-RoCE feature is provided as a Technology Preview only. Technology Preview features are not supported with Red Hat production Service Level Agreements (SLAs), might not be functionally complete, and Red Hat does not recommend using them for production. These previews provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

This section describes how to configure Soft-RoCE.

Prerequisites

- An Ethernet adapter is installed in the system.

Procedure

1. Install the **iproute libibverbs**, **libibverbs-utils**, and **infiniband-diags** packages:

```
# yum install iproute libibverbs libibverbs-utils infiniband-diags
```

2. Display the **rdma** links on the system:

```
# rdma link show
```

3. Load the **rdma_rxe** kernel module and add new RXE device:

```
# rdma link add rxe0 type rxe netdev eth0
```

where,

- **rxe0** specifies the name of the **rdma** link to add.
- **rxe** specifies the **rdma** type to use.
- **netdev** specifies the network device to which the link is bound.

4. View the state of all **rdma** links on the system:

```
# rdma link show
link rxe0/1 state ACTIVE physical_state LINK_UP netdev eth0
```

5. Optional: list the available RDMA devices in the system:

```
# ibv_devices

    device                 node GUID
    ------              ----------------
    rxe0                505400fffed5e0fb
```

Alternatively, use the **ibstat** utility to display a detailed status:

```
# ibstat rxe0

CA 'rxe0'
 CA type:
 Number of ports: 1
 Firmware version:
 Hardware version:
 Node GUID: 0x505400fffed5e0fb
 System image GUID: 0x0000000000000000
 Port 1:
  State: Active
  Physical state: LinkUp
  Rate: 100
  Base lid: 0
  LMC: 0
  SM lid: 0
  Capability mask: 0x00890000
  Port GUID: 0x505400fffed5e0fb
  Link layer: Ethernet
```

CHAPTER 3. CONFIGURING THE CORE RDMA SUBSYSTEM

This section describes how to configure the **rdma** service and increase the amount of memory that users are allowed to pin in the system.

3.1. RENAMING IPOIB DEVICES

By default, the kernel names IP over InfiniBand (IPoIB) devices, for example, **ib0**, **ib1**, and so on. To avoid conflicts, Red Hat recommends creating a rule in the **udev** device manager to create persistent and meaningful names, such as **mlx4_ib0**.

Prerequisites

- An InfiniBand device is installed in the host.

Procedure

1. Display the hardware address of the device. For example, to display the address for the device named **ib0**, enter:

```
# ip link show ib0
8: ib0: >BROADCAST,MULTICAST,UP,LOWER_UP< mtu 65520 qdisc pfifo_fast state UP
mode DEFAULT qlen 256
    link/infiniband 80:00:02:00:fe:80:00:00:00:00:00:00:00:02:c9:03:00:31:78:f2 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
```

   The last eight bytes of the address, marked in bold in the example, are required to create a **udev** rule in the next step.

2. Edit the **/etc/udev/rules.d/70-persistent-ipoib.rules** file, and append an **ACTION** rule. For example, to configure a rule that renames the device with the **00:02:c9:03:00:31:78:f2** hardware address to **mlx4_ib0**, append the following line:

```
ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32",
ATTR{address}=="?*00:02:c9:03:00:31:78:f2", NAME="mlx4_ib0"
```

3. Reboot the host:

```
# reboot
```

Additional resources

- **udev(7)** man page
- Understanding IPoIB hardware addresses

3.2. INCREASING THE AMOUNT OF MEMORY THAT USERS ARE ALLOWED TO PIN IN THE SYSTEM

RDMA operations require pinning of physical memory. This means that the kernel is not allowed to write memory into the swap space. If a user pins too much memory, the system can run out of memory, and the kernel terminates processes in order to free up more memory. For this reason, memory pinning is a privileged operation.

If non-root users run large RDMA applications, it can be necessary to increase the amount of memory these users can pin in the system. This section describes how to configure an unlimited amount of memory for the **rdma** group.

Procedure

- As the **root** user, create the **/etc/security/limits.d/rdma.conf** file with the following content:

```
@rdma soft memlock unlimited @rdma hard memlock unlimited
```

Verification steps

1. Log in as a member of the **rdma** group after you edited the **/etc/security/limits.d/rdma.conf** file.
   Note that Red Hat Enterprise Linux applies updated **ulimit** settings when the user logs in.

2. Use the **ulimit -l** command to display the limit:

```
$ ulimit -l
unlimited
```

   If the command returns **unlimited**, the user can pin an unlimited amount of memory.

Additional resources

- **limits.conf(5)** man page

3.3. CONFIGURING THE RDMA SERVICE

The **rdma** service manages the RDMA stack in the kernel. If Red Hat Enterprise Linux detects InfiniBand, iWARP, or RoCE devices, the **udev** device manager instructs **systemd** to start the **rdma** service.

Procedure

1. Edit the **/etc/rdma/rdma.conf** file, and set the variables of the modules you want to enable to **yes**. The following is the default **/etc/rdma/rdma.conf** on Red Hat Enterprise Linux 8:

```
# Load IPoIB
IPOIB_LOAD=yes
# Load SRP (SCSI Remote Protocol initiator support) module
SRP_LOAD=yes
# Load SRPT (SCSI Remote Protocol target support) module
SRPT_LOAD=yes
# Load iSER (iSCSI over RDMA initiator support) module
ISER_LOAD=yes
# Load iSERT (iSCSI over RDMA target support) module
ISERT_LOAD=yes
# Load RDS (Reliable Datagram Service) network protocol
RDS_LOAD=no
# Load NFSoRDMA client transport module
XPRTRDMA_LOAD=yes
# Load NFSoRDMA server transport module
```

```
SVCRDMA_LOAD=no
# Load Tech Preview device driver modules
TECH_PREVIEW_LOAD=no
```

2. Restart the **rdma** service:

```
# systemctl restart rdma
```

CHAPTER 4. CONFIGURING AN INFINIBAND SUBNET MANAGER

All InfiniBand networks must have a subnet manager running for the network to function. This is true even if two machines are connected directly with no switch involved.

It is possible to have more than one subnet manager. In that case one acts as a master and another subnet manager acts as a slave that will take over in case the master subnet manager fails.

Most InfiniBand switches contain an embedded subnet manager. However, if you need a more up-to-date subnet manager or if you require more control, use the **OpenSM** subnet manager provided by Red Hat Enterprise Linux.

4.1. INSTALLING THE OPENSM SUBNET MANAGER

This section describes how to install the OpenSM subnet manager.

Procedure

1. Install the **opensm** package:

```
# yum install opensm
```

2. Configure OpenSM if the default installation does not match your environment.
   If only one InfiniBand port is installed, the host should act as the master subnet manager, and no custom changes are needed. The default configuration works without any modification.

3. Enable and start the **opensm** service:

```
# systemctl enable --now opensm
```

Additional resources

- **opensm(8)** man page

4.2. CONFIGURING OPENSM USING THE SIMPLE METHOD

This section describes how to configure OpenSM if you do not need any customized settings.

Prerequisites

- One or more InfiniBand ports are installed on the server.

Procedure

1. Obtain the GUIDs for the ports using the **ibstat** utility:

```
# ibstat -d device_name

CA 'mlx4_0'
  CA type: MT4099
  Number of ports: 2
```

```
Firmware version: 2.42.5000
Hardware version: 1
Node GUID: 0xf4521403007be130
System image GUID: 0xf4521403007be133
Port 1:
   State: Active
   Physical state: LinkUp
   Rate: 56
   Base lid: 3
   LMC: 0
   SM lid: 1
   Capability mask: 0x02594868
   Port GUID: 0xf4521403007be131
   Link layer: InfiniBand
Port 2:
   State: Down
   Physical state: Disabled
   Rate: 10
   Base lid: 0
   LMC: 0
   SM lid: 0
   Capability mask: 0x04010000
   Port GUID: 0xf65214fffe7be132
   Link layer: Ethernet
```

NOTE

Some InfiniBand adapters use the same GUID for the node, system, and port.

2. Edit the **/etc/sysconfig/opensm** file and set the GUIDs in the **GUIDS** parameter:

```
GUIDS="GUID_1 GUID_2"
```

3. Optionally, set the **PRIORITY** parameter if multiple subnet managers are available in your subnet. For example:

```
PRIORITY=15
```

Additional resources

- **/etc/sysconfig/opensm**

4.3. CONFIGURING OPENSM BY EDITING THE OPENSM.CONF FILE

This section describes how to configure OpenSM by editing the **/etc/rdma/opensm.conf** file. Use this method to customize the OpenSM configuration if only one InfiniBand port is available.

Prerequisites

- Only one InfiniBand port is installed on the server.

Procedure

1. Edit the **/etc/rdma/opensm.conf** file and customize the settings to match your environment. After updating an **opensm** package, the **yum** utility overrides the **/etc/rdma/opensm.conf** and creates a copy which is the new OpenSM configuration file **/etc/rdma/opensm.conf.rpmnew**. So, you can compare the previous and new file to identify changes and incorporate them manually in file **opensm.conf**.

2. Restart the **opensm** service:

```
# systemctl restart opensm
```

4.4. CONFIGURING MULTIPLE OPENSM INSTANCES

This section describes how to set up multiple instances of OpenSM.

Prerequisites

- One or more InfiniBand ports are installed on the server.

Procedure

1. Optionally, copy the **/etc/rdma/opensm.conf** file to **/etc/rdma/opensm.conf.orig** file:

```
# cp /etc/rdma/opensm.conf /etc/rdma/opensm.conf.orig
```

   When you install an updated **opensm** package, the **yum** utility overrides the **/etc/rdma/opensm.conf**. With the copy created in this step, you can compare the previous and new file to identify changes and incorporate them manually in the instance-specific **opensm.conf** files.

2. Create a copy of the **/etc/rdma/opensm.conf** file:

```
# cp /etc/rdma/opensm.conf /etc/rdma/opensm.conf.1
```

   For each instance you create, append a unique and continuous number to the copy of the configuration file.

   After updating the **opensm** package, the **yum** utility stores the new OpenSM configuration file as **/etc/rdma/opensm.conf.rpmnew**. Compare this file with your customized **/etc/rdma/opensm.conf.*** files, and manually incorporate the changes.

3. Edit the copy you created in the previous step, and customize the settings for the instance to match your environment. For example, set the **guid**, **subnet_prefix**, and **logdir** parameters.

4. Optionally, create a **partitions.conf** file with a unique name specifically for this subnet and reference that file in the **partition_config_file** parameter in the corresponding copy of the **opensm.conf** file.

5. Repeat the previous steps for each instance you want to create.

6. Start the **opensm** service:

```
# systemctl start opensm
```

The **opensm** service automatically starts a unique instance for each **opensm.conf.*** file in the **/etc/rdma/** directory. If multiple **opensm.conf.*** files exist, the service ignores settings in the **/etc/sysconfig/opensm** file as well as in the base **/etc/rdma/opensm.conf** file.

Additional resources

- When you install an updated **opensm** package, the **yum** utility stores the new OpenSM configuration file as **/etc/rdma/opensm.conf.rpmnew**. Compare this file with your customized **/etc/rdma/opensm.conf.*** files and manually incorporate the changes.

4.5. CREATING A PARTITION CONFIGURATION

This section describes how to create InfiniBand partition configurations for OpenSM. Partitions enable administrators to create subnets on InfiniBand similar to Ethernet VLANs.

IMPORTANT

If you define a partition with a specific speed, such as 40 Gbps, all hosts within this partition must support at least this speed. If a host does not meet the speed requirements, it cannot join the partition. Therefore, set the speed of a partition to the lowest speed supported by any host with permission to join the partition.

Prerequisites

- One or more InfiniBand ports are installed on the server.

Procedure

1. Edit the **/etc/rdma/partitions.conf** file and configure the partitions.

NOTE

All fabrics must contain the **0x7fff** partition, and all switches and all hosts must belong to that fabric.

For example, add the following content to the file to create the **0x7fff** default partition at a reduced speed of 10 Gbps, and a partition **0x0002** with a speed of 40 Gbps:

```
# For reference:
# IPv4 IANA reserved multicast addresses:
#   http://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt
# IPv6 IANA reserved multicast addresses:
#   http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml
#
# mtu =
#   1 = 256
#   2 = 512
#   3 = 1024
#   4 = 2048
#   5 = 4096
#
# rate =
#   2  =   2.5 GBit/s
#   3  =  10   GBit/s
```

```
#   4 = 30   GBit/s
#   5 =  5   GBit/s
#   6 = 20   GBit/s
#   7 = 40   GBit/s
#   8 = 60   GBit/s
#   9 = 80   GBit/s
#   10 = 120   GBit/s

Default=0x7fff, rate=3, mtu=4, scope=2, defmember=full:
   ALL, ALL_SWITCHES=full;
Default=0x7fff, ipoib, rate=3, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff   # IPv4 Broadcast address
   mgid=ff12:401b::1           # IPv4 All Hosts group
   mgid=ff12:401b::2           # IPv4 All Routers group
   mgid=ff12:401b::16          # IPv4 IGMP group
   mgid=ff12:401b::fb          # IPv4 mDNS group
   mgid=ff12:401b::fc          # IPv4 Multicast Link Local Name Resolution group
   mgid=ff12:401b::101         # IPv4 NTP group
   mgid=ff12:401b::202         # IPv4 Sun RPC
   mgid=ff12:601b::1           # IPv6 All Hosts group
   mgid=ff12:601b::2           # IPv6 All Routers group
   mgid=ff12:601b::16          # IPv6 MLDv2-capable Routers group
   mgid=ff12:601b::fb          # IPv6 mDNS group
   mgid=ff12:601b::101         # IPv6 NTP group
   mgid=ff12:601b::202         # IPv6 Sun RPC group
   mgid=ff12:601b::1:3         # IPv6 Multicast Link Local Name Resolution group
   ALL=full, ALL_SWITCHES=full;

ib0_2=0x0002, rate=7, mtu=4, scope=2, defmember=full:
      ALL, ALL_SWITCHES=full;
ib0_2=0x0002, ipoib, rate=7, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff   # IPv4 Broadcast address
   mgid=ff12:401b::1           # IPv4 All Hosts group
   mgid=ff12:401b::2           # IPv4 All Routers group
   mgid=ff12:401b::16          # IPv4 IGMP group
   mgid=ff12:401b::fb          # IPv4 mDNS group
   mgid=ff12:401b::fc          # IPv4 Multicast Link Local Name Resolution group
   mgid=ff12:401b::101         # IPv4 NTP group
   mgid=ff12:401b::202         # IPv4 Sun RPC
   mgid=ff12:601b::1           # IPv6 All Hosts group
   mgid=ff12:601b::2           # IPv6 All Routers group
   mgid=ff12:601b::16          # IPv6 MLDv2-capable Routers group
   mgid=ff12:601b::fb          # IPv6 mDNS group
   mgid=ff12:601b::101         # IPv6 NTP group
   mgid=ff12:601b::202         # IPv6 Sun RPC group
   mgid=ff12:601b::1:3         # IPv6 Multicast Link Local Name Resolution group
   ALL=full, ALL_SWITCHES=full;
```

CHAPTER 5. CONFIGURING IPOIB

By default, InfiniBand does not use the internet protocol (IP) for communication. However, IP over InfiniBand (IPoIB) provides an IP network emulation layer on top of InfiniBand remote direct memory access (RDMA) networks. This allows existing unmodified applications to transmit data over InfiniBand networks, but the performance is lower than if the application would use RDMA natively.

NOTE

Internet Wide Area RDMA Protocol (iWARP) and RoCE networks are already IP-based. Therefore, you cannot create an IPoIB device on top of IWARP or RoCE devices.

The Mellanox devices starting from ConnectX-4 and above on RHEL 8 use Enhanced IPoIB mode by default (datagram only). Connected mode is not supported on these devices.

5.1. THE IPOIB COMMUNICATION MODES

You can configure an IPoIB device either in **Datagram** or **Connected** mode. The difference is what type of queue pair the IPoIB layer attempts to open with the machine at the other end of the communication:

- In the **Datagram** mode, the system opens an unreliable, disconnected queue pair.
  This mode does not support packages larger than the InfiniBand link-layer's Maximum Transmission Unit (MTU). The IPoIB layer adds a 4 byte IPoIB header on top of the IP packet being transmitted. As a result, the IPoIB MTU must be 4 bytes less than the InfiniBand link-layer MTU. As 2048 is a common InfiniBand link-layer MTU, the common IPoIB device MTU in **Datagram** mode is 2044.

- In the **Connected** mode, the system opens a reliable, connected queue pair.
  This mode allows messages larger than the InfiniBand link-layer MTU, and the host adapter handles packet segmentation and reassembly. As a result, there is no size limit imposed on the size of IPoIB messages that can be sent by InfiniBand adapters in **Connected** mode. However, IP packets are limited because of the **size** field and TCP/IP headers. For this reason, the IPoIB MTU in **Connected** mode is **65520** bytes maximum.

  The **Connected** mode has a higher performance, but consumes more kernel memory.

If a system is configured to use the **Connected** mode, it still sends multicast traffic in the **Datagram** mode, because InfiniBand switches and fabric cannot pass multicast traffic in **Connected** mode. Additionally, the system falls back to **Datagram** mode, when communicating with any host that is not configured in the **Connected** mode.

While running application that sends multicast data up to the maximum MTU on the interface, you must configure the interface in **Datagram** mode or configure the application to cap the packet send size at a size that will fit in datagram-sized packets.

5.2. UNDERSTANDING IPOIB HARDWARE ADDRESSES

IPoIB devices have a 20 byte hardware address that consists of the following parts:

- The first 4 bytes are flags and queue pair numbers.

- The next 8 bytes are the subnet prefix.

The default subnet prefix is **0xfe:80:00:00:00:00:00:00**. After the device connects to the subnet manager, the device changes this prefix to match the one configured in the subnet manager.

- The last 8 bytes are the Globally Unique Identifier (GUID) of the InfiniBand port that the IPoIB device is attached to.

NOTE

Because the first 12 bytes can change, don't use them in **udev** device manager rules.

Additional resources

- Renaming IPoIB devices

5.3. CONFIGURING AN IPOIB CONNECTION USING NMCLI COMMANDS

This procedure describes how to configure an IPoIB connection using **nmcli** commands.

Prerequisites

- An InfiniBand device is installed on the server, and the corresponding kernel module is loaded.

Procedure

1. Create the InfiniBand connection. For example, to create a connection that uses the **mlx4_ib0** interface in the **Connected** transport mode and the maximum MTU of **65520** bytes, enter:

```
# nmcli connection add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode Connected mtu 65520
```

2. Optional: set a **P_Key** interface. For example, to set **0x8002** as **P_Key** interface of the **mlx4_ib0** connection, enter:

```
# nmcli connection modify mlx4_ib0 infiniband.p-key 0x8002
```

3. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server of the **mlx4_ib0** connection, enter:

```
# nmcli connection modify mlx4_ib0 ipv4.addresses '192.0.2.1/24'
# nmcli connection modify mlx4_ib0 ipv4.gateway '192.0.2.254'
# nmcli connection modify mlx4_ib0 ipv4.dns '192.0.2.253'
# nmcli connection modify mlx4_ib0 ipv4.method manual
```

4. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server of the **mlx4_ib0** connection, enter:

```
# nmcli connection modify mlx4_ib0 ipv6.addresses '2001:db8:1::1/32'
# nmcli connection modify mlx4_ib0 ipv6.gateway '2001:db8:1::fffe'
# nmcli connection modify mlx4_ib0 ipv6.dns '2001:db8:1::fffd'
# nmcli connection modify mlx4_ib0 ipv6.method manual
```

5. Activate the connection. For example, to activate the **mlx4_ib0** connection:

```
# nmcli connection up mlx4_ib0
```

5.4. CONFIGURING AN IPOIB CONNECTION USING NM-CONNECTION-EDITOR

This procedure describes how to configure an IPoIB connection using the **nm-connection-editor** application.

Prerequisites

- An InfiniBand device is installed in the server, and the corresponding kernel module is loaded.
- The **nm-connection-editor** package is installed.

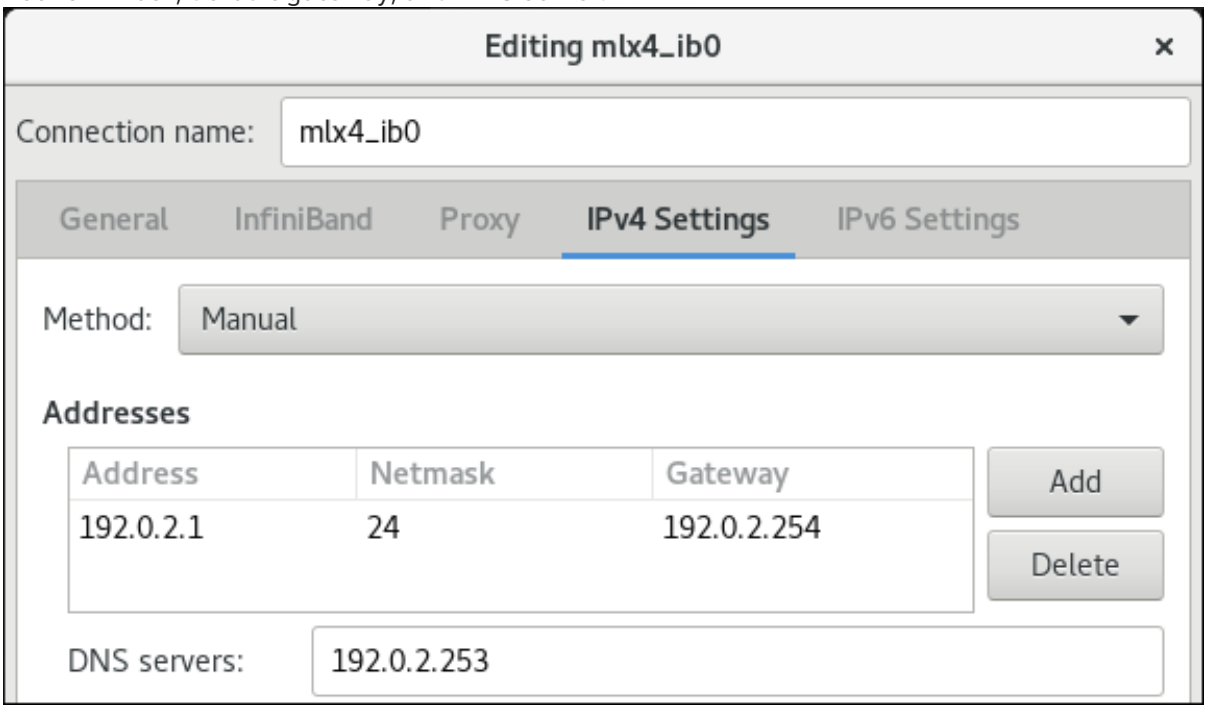
Procedure

1. Open a terminal, and enter:

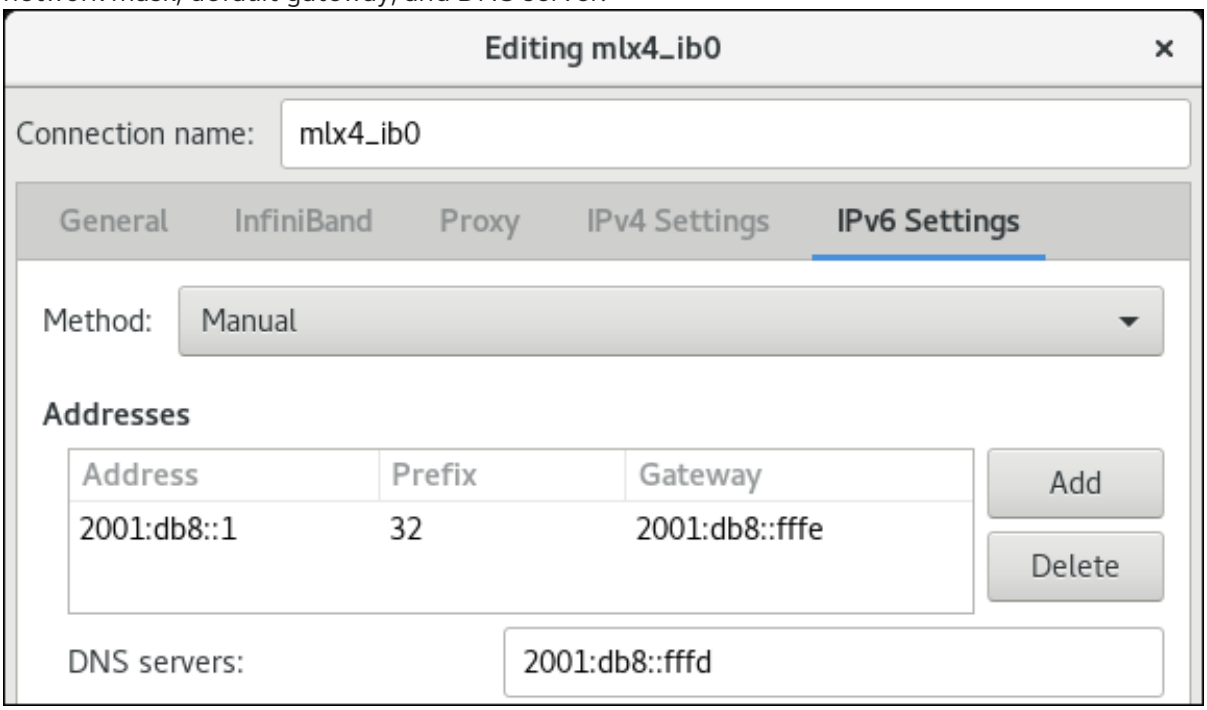
```
$ nm-connection-editor
```

2. Click the **+** button to add a new connection.
3. Select the **InfiniBand** connection type, and click **Create**.
4. On the **InfiniBand** tab:
   a. Optionally, change the connection name.
   b. Select the transport mode.
   c. Select the device.
   d. Optional: set an MTU.

5. On the **IPv4 Settings** tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:



6. On the **IPv6 Settings** tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:



7. Click **Save** to save the team connection.

8. Close **nm-connection-editor**.

9. Optional: set a **P_Key** interface. Note that you must set this parameter on the command line, because the setting is not available in **nm-connection-editor**.
For example, to set **0x8002** as **P_Key** interface of the **mlx4_ib0** connection, enter:

```
# nmcli connection modify mlx4_ib0 infiniband.p-key 0x8002
```

CHAPTER 6. TESTING INFINIBAND NETWORKS

This section provides procedures how to test InfiniBand networks.

6.1. TESTING EARLY INFINIBAND RDMA OPERATIONS

This section describes how to test InfiniBand remote direct memory access (RDMA) operations.

NOTE

This section applies only to InfiniBand devices. If you use iWARP or RoCE/IBoE devices, which are IP-based, see:

- Testing an IPoIB using the ping utility
- Testing an RDMA network using qperf after IPoIB is configured

Prerequisites

- RDMA is configured.
- The **libibverbs-utils** and **infiniband-diags** packages are installed.

Procedure

1. List the available InfiniBand devices:

```
# ibv_devices

    device                 node GUID
    ------              ----------------
    mlx4_0              0002c903003178f0
    mlx4_1              f4521403007bcba0
```

2. Display the information for a specific InfiniBand device. For example, to display the information of the **mlx4_1** device, enter:

```
# ibv_devinfo -d mlx4_1

hca_id: mlx4_1
    transport:                  InfiniBand (0)
    fw_ver:                     2.30.8000
    node_guid:                  f452:1403:007b:cba0
    sys_image_guid:             f452:1403:007b:cba3
    vendor_id:                  0x02c9
    vendor_part_id:             4099
    hw_ver:                     0x0
    board_id:                   MT_1090120019
    phys_port_cnt:              2
        port:   1
            state:              PORT_ACTIVE (4)
            max_mtu:            4096 (5)
            active_mtu:         2048 (4)
            sm_lid:             2
```

```
            port_lid:          2
            port_lmc:          0x01
            link_layer:        InfiniBand

        port:   2
            state:             PORT_ACTIVE (4)
            max_mtu:           4096 (5)
            active_mtu:        4096 (5)
            sm_lid:            0
            port_lid:          0
            port_lmc:          0x00
            link_layer:        Ethernet
```

3. Display the basic status of an InfiniBand device. For example, to display the status of the **mlx4_1** device, enter:

```
# ibstat mlx4_1

CA 'mlx4_1'
    CA type: MT4099
    Number of ports: 2
    Firmware version: 2.30.8000
    Hardware version: 0
    Node GUID: 0xf4521403007bcba0
    System image GUID: 0xf4521403007bcba3
    Port 1:
        State: Active
        Physical state: LinkUp
        Rate: 56
        Base lid: 2
        LMC: 1
        SM lid: 2
        Capability mask: 0x0251486a
        Port GUID: 0xf4521403007bcba1
        Link layer: InfiniBand
    Port 2:
        State: Active
        Physical state: LinkUp
        Rate: 40
        Base lid: 0
        LMC: 0
        SM lid: 0
        Capability mask: 0x04010000
        Port GUID: 0xf65214fffe7bcba2
        Link layer: Ethernet
```

4. Use the **ibping** utility to ping from a client to a server using InfiniBand:

   a. On the host that acts as a server, start **ibping** in server mode:

   ```
   # ibping -S -C mlx4_1 -P 1
   ```

   This command uses the following parameters:

   - **-S**: Enables the server mode.

- **-C *InfiniBand_CA_name***: Set's the CA name to use.
- **-P *port_number***: Sets the port number to use, if the InfiniBand provides multiple ports.

b. On the host that acts as client, use **ibping** as follows:

```
# ibping -c 50 -C mlx4_0 -P 1 -L 2
```

- **-c *number***: Sends these number of packets to the server.
- **-C *InfiniBand_CA_name***: Set's the CA name to use.
- **-P *port_number***: Sets the port number to use, if the InfiniBand provides multiple ports.
- **-L *port_LID***: Sets the Local Identifier (LID) to use.

Additional resources

- **ibping(8)** man page

6.2. TESTING AN IPOIB USING THE PING UTILITY

After you configured IPoIB, use the **ping** utility to send ICMP packets to test the IPoIB connection.

Prerequisites

- The two RDMA hosts are connected in the same InfiniBand fabric with RDMA ports.
- The IPoIB interfaces in both hosts are configured with IP addresses within the same subnet.

Procedure

- Use the **ping** utility to send ICMP packets to the remote host's InfiniBand adapter:

```
# ping -c5 192.0.2.1
```

This command sends five ICMP packets to the IP address **192.0.2.1**.

6.3. TESTING AN RDMA NETWORK USING QPERF AFTER IPOIB IS CONFIGURED

This procedure describes examples how to display the InfiniBand adapter configuration and measure the bandwidth and latency between two hosts using the **qperf** utility.

Prerequisites

- The **qperf** package is installed on both hosts.
- IPoIB is configured on both hosts.

Procedure

1. Start **qperf** on one of the hosts without any options to act as a server:

```
# qperf
```

2. Use the following commands on the client. The commands use port **1** of the **mlx4_0** host channel adapter in the client to connect to IP address **192.0.2.1** assigned to the InfiniBand adapter in the server.

   a. To display the configuration, enter:

```
# qperf -v -i mlx4_0:1 192.0.2.1 conf

conf:
    loc_node   =  rdma-dev-01.lab.bos.redhat.com
    loc_cpu    =  12 Cores: Mixed CPUs
    loc_os     =  Linux 4.18.0-187.el8.x86_64
    loc_qperf  =  0.4.11
    rem_node   =  rdma-dev-00.lab.bos.redhat.com
    rem_cpu    =  12 Cores: Mixed CPUs
    rem_os     =  Linux 4.18.0-187.el8.x86_64
    rem_qperf  =  0.4.11
```

   b. To display the Reliable Connection (RC) streaming two-way bandwidth, enter:

```
# qperf -v -i mlx4_0:1 192.0.2.1 rc_bi_bw

rc_bi_bw:
    bw              =  10.7 GB/sec
    msg_rate        =   163 K/sec
    loc_id          =  mlx4_0
    rem_id          =  mlx4_0:1
    loc_cpus_used   =    65 % cpus
    rem_cpus_used   =    62 % cpus
```

   c. To display the RC streaming one-way bandwidth, enter:

```
# qperf -v -i mlx4_0:1 192.0.2.1 rc_bw

rc_bw:
    bw              =  6.19 GB/sec
    msg_rate        =  94.4 K/sec
    loc_id          =  mlx4_0
    rem_id          =  mlx4_0:1
    send_cost       =  63.5 ms/GB
    recv_cost       =    63 ms/GB
    send_cpus_used  =  39.5 % cpus
    recv_cpus_used  =    39 % cpus
```

Additional resources

- **qperf(1)** man page