



CHAPTER 5

Sockets Direct Protocol

This chapter describes the Sockets Direct Protocol and includes the following sections:

- [Introduction, page 5-1](#)
- [Configuring IPoIB Interfaces, page 5-1.](#)
- [Converting Sockets-Based Application, page 5-2](#)
- [SDP Performance, page 5-4](#)
- [Netperf Server with IPoIB and SDP, page 5-6](#)



Note

See the “[Root and Non-root Conventions in Examples](#)” section on [page ix](#) for details about the significance of prompts used in the examples in this chapter.

Introduction

SDP is an IB-specific upper layer protocol. It defines a standard wire protocol to support stream sockets networking over IB. SDP enables sockets-based applications to take advantage of the enhanced performance features provided by IB and achieves lower latency and higher bandwidth than IPoIB running sockets-based applications. It provides a high-performance, zero-copy data transfer protocol for stream-socket networking over an IB fabric. You can configure the driver to automatically translate TCP to SDP based on source IP, destination, or application name.

Configuring IPoIB Interfaces

SDP uses the same IP addresses and interface names as IPoIB. Configure the IPoIB IP interfaces if you have not already done so. (See [Chapter 3, “IP over IB Protocol.”](#))

Converting Sockets-Based Application

This section describes how to convert sockets-based applications. You can convert your sockets-based applications to use SDP instead of TCP by using one of two conversion types. This section includes the following topics:

- [Explicit/Source Code Conversion Type, page 5-2](#)
- [Automatic Conversion Type, page 5-2](#)

Explicit/Source Code Conversion Type

The explicit or source code conversion type method converts sockets to use SDP based on application source code. This method is useful when you want full control from your application when using SDP.

To use this method, change your source code to use `AF_INET_SDP` instead of `AF_INET` when calling the `socket()` system call.

`AF_INET_SDP` is defined as 26. Add the following line of code to the beginning of your program:

```
#define AF_INET_SDP 26
```

Automatic Conversion Type

This section describes automatic conversion type. Use a text editor to open the `libsdp` configuration file (located in `/usr/local/topspin/etc/libsdp.conf`). This file defines when to automatically use SDP instead of TCP. You may edit this file to specify connection overrides. Use the environment variable `LIBSDP_CONFIG_FILE` to specify an alternate configuration file.

The automatic conversion type method converts socket streams based upon a destination port, listening port, or program name.

Load the installed `libsdp.so` library using either of these two methods:

- Set the `LD_PRELOAD` environment variable to `libsdp.so` before running the executable.
- Add the full path of the library into `/etc/ld.so.preload`. This action causes the library to preload for every executable that is linked with `libc`.

This configuration file supports two main types of statements:

- **log**

The **log** keyword sets logging-related configurations. The log settings take immediate effect, so they are defined at the beginning of the file.

- **match**

The **match** keyword enables the user to specify when `libsdp` replaces `AF_INET/SOCK_STREAM` sockets with `AF_INET_SDP/SOCK_STREAM` sockets.

Log Statement

This section describes the log statement. The log directive allows the user to specify which debug and error messages are sent and where they are sent. The log statement format is as follows:

log [destination stderr | syslog | file *filename*] [min-level 1-9]

Command	Description
destination	Defines the destination of the log messages.
stderr	Forwards messages to the STDERR.
syslog	Sends messages to the syslog service.
file <i>filename</i>	Writes messages to the file/tmp/ <i>filename</i> .
min-level	Defines the verbosity of the log as follows: 9—Errors are printed. 3—Protocol-matching messages. 2—Socket-creation messages. 1—Function calls and return values.

The file destination must be relative to /tmp. This is to prevent non-superuser accounts from having the ability to create arbitrary files on the system. Any path components of the filename are stripped.

The following example shows how to get the full verbosity printed into the /tmp/libsdp.log file:

```
log min-level 1 destination file libsdp.log
```

The following example shows how to get the full verbosity printed into the /STDERR:

```
log min-level 1 destination stderr
```

Match Statement

This section describes the match statement. The match directive enables the user to specify when libsdp replaces AF_INET/SOCK_STREAM sockets with AF_SDP/SOCK_STREAM sockets. Each match directive specifies a group for which all expressions must evaluate as true (logical and).

The four expressions are as follows:

```
destination ip_port
listen ip_port
shared ip_port
program program_name
```

The syntax description for the match statement is as follows:

destination	This expression enables the user to match a client-connect request and convert the TCP socket to an SDP socket. The rule is applied during the connect system call.
listen	This expression enables the user to match a server-bind request and convert the TCP socket to an SDP socket. The rule is applied during the bind system call.

shared	This expression enables the user to match a server-bind request and then listen and accept incoming connections on both TCP and SDP protocols.
program	This expression enables the user to match the program name.

The `ip_port` matches against an IP address, prefix length, and port range. The format is as follows:

```
ip_addr[/prefix_length][:start_port[-end_port]]
```

The prefix length is optional and missing defaults to /32 (length of one host). The ending port in the range is optional and is missing defaults to the port specified by the starting point. The `ip_addr` variable or `start_port` variable can be *, which means any IP or any port, respectively.

The `program_name` variable matches on shell style globs. The `db2*` value matches on any program with a name starting with `db2`, and the `t?cp` matches on `ttcp`. These are examples of program names:

```
match listen *:5001 program ttcp
match shared *:5002
match destination 192.168.1.0/24
match program db2*
```

SDP Performance

This section describes how to verify SDP performance by running the Netperf Bandwidth test and the Latency test. These tests are described in detail at the following URL:

<http://www.netperf.org/netperf/training/Netperf.html>

To verify SDP performance, perform the following steps:

-
- Step 1** Download Netperf from the following URL:
<http://www.netperf.org/netperf/NetperfPage.html>
 - Step 2** Follow the instructions at <http://www.netperf.org/netperf/NetperfPage.html> to compile Netperf.
 - Step 3** Create a `libsdp` configuration file.

```
host1$ cat > $HOME/libsdp.conf << EOF
> match destination *.*
> match listen *.*
> EOF
```

- Step 4** Run the Netperf server, which forces SDP to be used instead of TCP.

The following example shows how to run the Netperf server with SDP:

```
host1$ LD_PRELOAD=libsdp.so LIBSDP_CONFIG_FILE=$HOME/libsdp.conf netserver
Starting netserver at port 12865
Starting netserver at hostname 0.0.0.0 port 12865 and family AF_UNSPEC
host1$
```

- Step 5** Run the Netperf Bandwidth test, which forces SDP to be used instead of TCP.

The following example shows how to run the Netperf Bandwidth test with SDP:

```
host2$ LD_PRELOAD=libsdp.so LIBSDP_CONFIG_FILE=$HOME/libsdp.conf netperf -H 192.168.0.1 -c
-C -- -m 65536
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.1 (192.168.0.1)
port 0 AF_INET
Recv  Send  Send          Utilization      Service Demand
```

Socket Size bytes	Socket Size bytes	Message Size bytes	Elapsed Time secs.	Throughput 10^6bits/s	Send local % S	Recv remote % S	Send local us/KB	Recv remote us/KB
87380	16384	65536	10.00	6601.82	23.79	21.37	1.181	1.061

The following list describes the parameters for the **netperf** command:

-H	Where to find the server
192.168.0.1	IPoIB IP address
-c	Client CPU utilization
-C	Server CPU utilization
--	Separates the global and test-specific parameters
-m	The message size, which is 65536 in the example above

The notable performance values in the example above are as follows:

Throughput is 6.60 gigabits per second.

Client CPU utilization is 23.79 percent of the client CPU.

Server CPU utilization is 21.37 percent of the server CPU.

Step 6 Run the Netperf Latency test, which forces SDP to be used instead of TCP.

After the test runs once, stop the server so that it does not repeat the test.

The following example shows how to run the Netperf Latency test with SDP:

```
host2$ LD_PRELOAD=libsdp.so LIBSDP_CONFIG_FILE=$HOME/libsdp.conf netperf -H 192.168.0.1 -c
-C -t TCP_RR -- -r 1,1
TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.1
(192.168.0.1) port 0 AF_INET
Local /Remote
Socket Size Request Resp. Elapsed Trans. CPU CPU S.dem S.dem
Send Recv Size Size Time Rate local remote local remote
bytes bytes bytes bytes secs. per sec % S % S us/Tr us/Tr

16384 87380 1 1 10.00 27754.33 6.26 7.22 9.029 10.408
16384 87380
Stop netperf server.
```

The following list describes parameters for the **netperf** command:

-H	Where to find the server
192.168.0.1	IPoIB IP address
-c	Client CPU utilization
-C	Server CPU utilization
-t	Test type
TCP_RR	TCP request response test
--	Separates the global and test-specific parameters
-r 1,1	Request size sent and how many bytes requested back

The notable performance values in the example above are as follows:

Client CPU utilization is 6.26 percent of client CPU.

Server CPU utilization is 7.22 percent of server CPU.

Latency is 18.01 microseconds. Latency is calculated as follows:

$(1 / \text{Transaction rate per second}) / 2 * 1,000,000 = \text{one-way average latency in microseconds}$

Step 7 To end test, shutdown the Netperf server.

The following example shows how to shutdown the Netperf server:

```
host1$ pkill netserver
```

Netperf Server with IPoIB and SDP

This section describes how to use the Netperf server with IPoIB and SDP. When using libsdp, it is possible for the Netperf server to work with both IPoIB and SDP. To use Netperf server with IPoIB and SDP, perform the following steps:

Step 1 Create the libsdp configuration file.

The following example shows how to create the libsdp configuration file:

```
host1$ echo "match shared *:*" > $HOME/both.conf
```

Step 2 Ensure that the Netperf server is not running already, and then start the Netperf server.

The following example stops the Netperf server if it is already running and then starts the server:

```
host1$ pkill netserver
host1$ LD_PRELOAD=libsdp.so LIBSDP_CONFIG_FILE=$HOME/both.conf netserver
Starting netserver at port 12865
Starting netserver at hostname 0.0.0.0 port 12865 and family AF_UNSPEC
```

Step 3 Run the Netperf Bandwidth test, which forces SDP to be used instead of TCP.

The following example shows how to run the Netperf Bandwidth test with SDP:

```
host2$ LD_PRELOAD=libsdp.so LIBSDP_CONFIG_FILE=$HOME/libsdp.conf netperf -H 192.168.0.1 -c
-C -- -m 65536
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.1 (192.168.
0.206) port 0 AF_INET
Recv  Send  Send  Utilization  Service Demand
Socket Socket Message Elapsed  Send  Recv  Send  Recv
Size  Size  Size  Time  Throughput  local  remote  local  remote
bytes bytes bytes secs.  10^6bits/s  % S  % S  us/KB  us/KB

 87380 16384 65536 10.00 6601.82 23.79 21.37 1.181 1.061
```

The following list describes parameters for the **netperf** command:

-H	Where to find the server
192.168.0.1	IPoIB IP address
-c	Client CPU utilization
-C	Server CPU utilization
--	Separates the global and test-specific parameters
-m	The message size, which is 65536 in the example above

The notable performance values in the example above are as follows:

Throughput is 6.60 gigabits per second.

Client CPU utilization is 23.79 percent of the client CPU.

Server CPU utilization is 21.37 percent of the server CPU.

Step 4 Run the Netperf client.

The default test is the Bandwidth test.

The following example shows how to run the Netperf client, which starts the Bandwidth test by default:

```
host2$ netperf -H 192.168.0.1 -c -C -- -m 65536
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.1 (192.168.0.1) port 0
AF_INET
```

Recv Socket Size bytes	Send Socket Size bytes	Send Message Size bytes	Elapsed Time secs.	Throughput 10^6bits/s	Utilization		Service Demand	
					local % S	remote % S	local us/KB	remote us/KB
87380	16384	65536	10.00	2701.06	46.93	48.73	5.694	5.912



Note You must specify the IPoIB IP address when running the Netperf client.

The following list describes parameters for the **netperf** command:

-H	Where to find the server
192.168.0.1	IPoIB IP address
-c	Client CPU utilization
-C	Server CPU utilization
--	Separates the global and test-specific parameters
-m	Message size, which is 65536 in the example above

The notable performance values in the example above are as follows:

Throughput is 2.70 gigabits per second.

Client CPU utilization is 46.93 percent of client CPU.

Server CPU utilization is 48.73 percent of server CPU.

