RTI Connext DDS

Core Libraries

Throughput Performance Test Example Using Java

Instructions

Version 5.3.0

Your systems. Working as one.

Technical Support
Real-Time Innovations, Inc.
232 E. Java Drive
Sunnyvale, CA 94089

Phone:	(408) 990-7444
Email:	support@rti.com
Website:	https://support.rti.com/

Contents

Throughput Test Example Using Java

This document provides instructions on using the *RTI® Connext® DDS* throughput test example.

1 Introduction

In this test, the publisher publishes to one or more subscriber applications. The test goes through the following phases:

1. The publisher signals the subscriber applications that it will commence, and then starts its own clock. You may specify the duration of the test.
2. The subscriber starts measuring the number of samples received.
3. After the desired duration is over, the publisher signals the subscribers that one experiment is over. The subscriber will then divide the number of samples received by the elapsed time to report the throughput observed at the receiver. If there are multiple subscribers, the throughput reported may be different. The publisher also reports the throughput on its end.

Maximum throughput is achieved when the publisher sends as fast as the subscribers can handle messages without dropping a packet. That is, the maximum throughput is obtained somewhere between the publisher sending too slowly (not maximizing the available pipe) and the publisher swamping the subscriber (overflowing the pipe).

For this reason, the test makes the publisher try a range of sending rates; the range is provided as an input to the test. For the absolute maximum throughput to be observed, the optimal sending rate must be in the range. An ever increasing reported throughput throughout the test range indicates that the tested range is too small. To observe the maximum throughput, the range must be extended to include faster sending rates.

By default, the message size ranges from 16 bytes to 8K where the maximum size is indicated in the IDL file. To ensure a controlled environment, the performance test uses unicast for discovery instead of multicast.

2 Test Operation

The test code contains two applications: one for the publishing node, one for the subscribing node(s).

1. You will start the applications from the command line (publisher, then subscribers).
2. The publication application waits for the subscribing applications to announce their participation.
3. Once the publisher recognizes that the specified numbers of subscribers are online, it starts the test.
4. The test sends a burst of data, sleeps 10ms, and repeats the cycle for a specified duration. You control the amount of data and the test duration with command-line options.

3 Paths Mentioned in Documentation

The documentation refers to:

❏ **<NDDSHOME>**

This refers to the main installation directory for *Connext DDS*. The default installation paths are:

For UNIX-based systems: **/home/your user name/rti_connext_dds-*version***

For Windows systems: **C:\Program Files\rti_connext_dds-*version***

When you see **<NDDSHOME>**, replace it with your installation path.

❏ **<path to examples>**

Examples are copied into your home directory the first time you run *RTI Launcher* or any script in **<NDDSHOME>/bin**. This document refers to the location of the copied examples as **<path to examples>.** The paths to the examples are:

For UNIX-based systems:
/home/your user name/rti_workspace/*version*/examples/

For Windows systems:
C:\Users\your user name\Documents\rti_workspace\*version*\examples\

When you see **<path to examples>**, replace it with the appropriate path.

4 Building the Test Applications

The example is in **<path to examples>/connext_dds/java/performance/throughput**.

Important: Make sure the **NDDSHOME** environment variable is set. See the *RTI Connext DDS Core Libraries Getting Started Guide* for more information.

4.1 Generating a Makefile with rtiddsgen

The source code and a sample makefile are provided in the directory. To generate a makefile specific to your architecture, execute the following:

```
<NDDSHOME>/bin/rtiddsgen -language Java -example <your Java arch> \
  -notypecode Throughput.idl
```

On sparcSol2.10gcc3.4.2, for example, the Java arch is sparcSol2.10jdk. Since all the source files for this example are already present in the directory, *rtiddsgen* may print messages saying that some files already exist and will not be replaced. You can safely ignore these messages.

Note: Remove the newly generated **USER_QOS_PROFILES.xml** file (its default profiles are not consistent with the QoS used by the test).

4.2 Building the Test Application with the Generated Makefile

To build and run the test, you need JDK 1.7 or later.

The *rtiddsgen* utility generates **makefile_Throughput_<*your Java architecture*>**, which you can use to build the example. Please execute:

```
gmake -f makefile_Throughput_<your Java architecture>
```

4.3 Running the Test Using the Generated Makefile

Once the Java files are compiled, you can run the main methods of the ThroughputPublisher and ThroughputSubscriber, specifying the right classpath (please see the generated makefile for the list). The generated makefile also helps you run the example, providing the two targets: ThroughputPublisher and ThroughputSubscriber.

Use the following two commands to run the publisher and subscriber applications:

```
gmake ARGS="<command-line options>" \
  -f makefile_Throughput_<your Java architecture> ThroughputPublisher

gmake ARGS="<command-line options>" \
  -f makefile_Throughput_<your Java architecture> ThroughputSubscriber
```

If you have trouble running the test, please first ensure that you can run the HelloWorld example. See the *RTI Connext DDS Core Libraries Getting Started Guide* and *Platform Notes* for detailed instructions.

Note for Testing on an AIX System: Because the QoS in the test uses changed thread priorities, on AIX platforms, you must run the test with *root* privileges (see the *Platform Notes*, Section 2.1). Alternatively, you can change the test so that it does not change thread priorities; to do so, in ThroughputPublisher.java, comment out the changes to:

- ❑ **participant_qos.event.thread.priority**
- ❑ **participant_qos.receiver_pool.thread.priority**
- ❑ **publisher_qos.asynchronous_publisher.thread.priority**

5 Command-Line Options

All options are preceded by a minus sign (**-**). Some options take additional information where required. Test output can be captured using redirection on the command line.

All parameters are optional; the defaults can be found in the main function (vx_publisher_main and vx_subscriber_main for VxWorks Kernel Mode).

For examples, please see Example Command-Line Options (Section 6).

Table 5.1 **Command-Line Options**

Option	Publishing Application	Subscribing Application
-domainId #	Sets the domain ID. This test can be run at the same time as other *Connext* applications, provided that the domain ID is unique.	
-participantId #	N/A (Publishing application always uses 0.)	Sets the participant ID number. Use a unique value for each application running on the same host that uses the same domain ID. Default: 1
-nic <IP>	Restricts *Connext* to sending output through this interface. This can be the IP address of any available network interface on the machine. By default, *Connext* will attempt to contact all possible subscribing nodes on all available network interfaces. While this may be interesting for some users, we are focusing on a simple case. Even on a multi-NIC machine, the performance over one NIC vs. another may be different (e.g., Gbit vs. 100 Mbit), so choosing the correct NIC is critical for a proper test.	
-transport #	Determines which transport to use for the test: 1 = UDP over IPv4 2 = Shared Memory 8 = UDP over IPv6 (Please see the Known Issues section of the *RTI Core Libraries and Utilities Release Notes* for information on using IPv6.)	
-peer *<IP address or hostname>* or -peer shmem://	Specifies each peer taking part in the test. For the publishing application, specify the IP address or hostname of each subscribing application. If all subscribing applications are on the same remote host, you will only need to specify that one hostname. If each subscribing application is on a separate host, specify each one individually. If the publishing and subscribing applications are on the same host, specify that you are using shared memory: **shmem://** . For the subscribing application, just specify the IP address or hostname of the publishing application. You can specify up to 16 peers; to change this limit, modify the #definition of MAX_TEST_SUBSCRIBERS in ThroughputPublisher.java and ThroughputSubscriber.java. When performing the test with two or more subscribing applications, the **-subscribers** flag (on the publishing side) and the **-participantId** flag (on both the publishing and subscribing sides) are both mandatory.	
-reliable	Tells the test to use reliable communication. By default, the test uses best-effort communication.	
-mcast_recv_addr *<IP>*	N/A	Specifies the multicast address to use for receiving data.

Table 5.1 **Command-Line Options**

Option	Publishing Application	Subscribing Application
-subscribers #	Sets the number of subscribing applications participating in the test. Note that there may be more than one subscribing application (participant) on each node.	N/A
-duration #	Sets the number of seconds to be sustained for each demand run.	
-size #	Sets the payload size.	
-demand *<initial effort>*: *<incremental effort>*: *<end effort>*	Controls how many samples the publisher should write consecutively between 10ms sleep periods.	
-recoveryTime <time in ms>	Specifies the sleep period to be used after writing a specific effort (specified by **-demand**).	N/A
-strength <value>	Sets the DataWriter's Ownership-Strength QoS.	
-maxBlockingTime *<time in ms>*	Sets the max_blocking_time field in the DataWriter's Reliability QoS.	
-no_push_on_write	If specified while in reliable mode, this turns off push-on-write behavior. Otherwise, the writer will use push-based reliability.	
-multicast_ttl #	Indicates the number of Time-To-Live (TTL) hops for the multicast packet. This argument must be used for a multicast test.	
-bw_limit *<maxMbps>*	Limits the flow to the specified Mbit/s. (Only applies to large data.)	
-asynchronous	Sends data using a separate thread and enables flow control. This option is implied if the data size exceeds the transport MTU.	

6 Example Command-Line Options

On Solaris and Linux systems, set **RTI_CLASSPATH** to **$(NDDSHOME)/lib/java/nddsjava.jar** and **LD_LIBRARY_PATH** to **$(NDDSHOME)/lib/java**.

On Windows systems, set **RTI_CLASSPATH** to **%NDDSHOME%/lib/java/nddsjava.jar** and **Path** to **%NDDSHOME%/lib/java**.

Assuming you are executing the test in the **<path to examples>/connext_dds/java/performance/throughput/** directory, you can start the publisher and subscribers with the following the commands:

```
gmake ARGS="<command-line options>" \
-f makefile_Throughput_<your Java architecture> ThroughputPublisher

gmake ARGS="<command-line options>" \
-f makefile_Throughput_<your Java architecture> ThroughputSubscriber
```

In the following tests, start the publisher first, so it can wait for subscribers to come online.

In Table 6.1, pub_IP, sub1_IP, sub2_IP, sub3_IP, and sub4_IP represent the IP addresses of the publisher, subscriber 1, subscriber 2, subscriber 3, and subscriber 4, respectively. In the Scenario column, BE means Best Effort reliability, SR means Strict Reliability; 1-4 means 1 publisher to 4 subscribers. All IP addresses must belong to the same network.

Table 6.1 **Example Command-Line Options**

Scenario	Command-Lines Options
1-1 BE 1024 bytes over shmem	**Publisher:** `-domainId 88 -nic pub_IP -transport 2 -peer shmem:// -subscribers 1 \ -duration 10 -size 1024 -demand 1000:1000:9000` **Subscriber:** `-domainId 88 -participantId 1 -nic sub1_IP -transport 2 -peer shmem://`
1-4 BE 8192 bytes over shmem	**Publisher:** `-domainId 88 -nic pub_IP -transport 2 -peer shmem:// -subscribers 4 \ -duration 10 -size 8192 -demand 1000:1000:9000` **1st Subscriber:** `-domainId 88 -participantId 1 -nic sub1_IP -transport 2 -peer shmem://` **2nd Subscriber:** `-domainId 88 -participantId 2 -nic sub2_IP -transport 2 -peer shmem://` **3rd Subscriber:** `-domainId 88 -participantId 3 -nic sub3_IP -transport 2 -peer shmem://` **4th Subscriber:** `-domainId 88 -participantId 4 -nic sub4_IP -transport 2 -peer shmem://`

Table 6.1 **Example Command-Line Options**

Scenario	Command-Lines Options

1-1
pull SR over shmem
	Publisher:
	`-domainId 88 -nic pub_IP -transport 2 -peer shmem:// -subscribers 1 \`
	`-duration 10 -size 1024 -demand 1000:1000:9000 -reliable \`
	`-no_push_on_write`
	Subscriber:
	`-domainId 88 -participantId 1 -nic sub1_IP -transport 2 \`
	`-peer shmem:// -reliable`

1-4
push SR over shmem, effort starting at 100, finishing at 900, increments of 100
	Publisher:
	`-domainId 88 -nic pub_IP -transport 2 -peer shmem:// -subscribers 4 \`
	`-duration 10 -size 1024 -demand 100:100:900 -reliable`
	1st Subscriber:
	`-domainId 88 -participantId 1 -nic sub1_IP -transport 2 \`
	`-peer shmem:// -reliable`
	2nd Subscriber:
	`-domainId 88 -participantId 2 -nic sub2_IP -transport 2 \`
	`-peer shmem:// -reliable`
	3rd Subscriber:
	`-domainId 88 -participantId 3 -nic sub3_IP -transport 2 \`
	`-peer shmem:// -reliable`
	4th Subscriber:
	`-domainId 88 -participantId 4 -nic sub4_IP -transport 2 \`
	`-peer shmem:// -reliable`

Table 6.1 **Example Command-Line Options**

Scenario	Command-Lines Options
1-4 BE over UDP unicast	**Publisher:** `-domainId 88 -nic pub_IP -transport 1 -peer sub1_IP -subscribers 1 \` `-duration 10 -size 1024 -demand 1000:1000:9000` **Subscriber:** `-domainId 88 -participantId 1 -nic sub1_IP -transport 1 -peer pub_IP`
1-4 BE 32 bytes over UDP multicast	**Publisher:** `-domainId 88 -nic pub_IP -transport 1 -peer sub1_IP -peer sub2_IP \` `-peer sub3_IP -peer sub4_IP -subscribers 4 -duration 10 -size 32  \` `-demand 1000:1000:9000 -multicast_ttl 1` **1st Subscriber:** `-domainId 88 -participantId 1 -nic sub1_IP -transport 1 -peer pub_IP \` `-mcast_recv_addr 225.3.2.1` **2nd Subscriber:** `-domainId 88 -participantId 2 -nic sub2_IP -transport 1 -peer pub_IP \` `-mcast_recv_addr 225.3.2.1` **3rd Subscriber:** `-domainId 88 -participantId 3 -nic sub3_IP -transport 1 -peer pub_IP \` `-mcast_recv_addr 225.3.2.1` **4th Subscriber:** `-domainId 88 -participantId 4 -nic sub4_IP -transport 1 -peer pub_IP \` `-mcast_recv_addr 225.3.2.1`

7 Output Analysis

Typical output from Publisher:

```
Starting test...
bytes  samples  Mbit/sec duration effort
----- --------- -------- -------- ------
 1024,   162988,  44.505,   30.00,   100
 1024,   208667,  56.974,   30.00,   200
 1024,   233846,  63.850,   30.00,   300
 1024,   240650,  65.710,   30.00,   400
 1024,   259260,  70.795,   30.00,   500
 1024,   263401,  71.901,   30.01,   600
 1024,   269965,  73.718,   30.00,   700
 1024,   283201,  77.330,   30.00,   800
 1024,   286523,  78.238,   30.00,   900
```

Typical output from Subscriber:

```
Starting test...
bytes effort samples  lost(A) Nreject lost(N)  Mbit/s duration
----- ----- -------- ------- ------- ------- -------- --------
 1024,  100, 162988,      0,      0,       0,  44.502,   30.00
 1024,  200, 208667,      0,      0,       0,  56.985,   30.00
 1024,  300, 233846,      0,      0,       0,  63.843,   30.01
 1024,  400, 240650,      0,      0,       0,  65.714,   30.00
 1024,  500, 259260,      0,      0,       0,  70.790,   30.00
 1024,  600, 263401,      0,      0,       0,  71.908,   30.01
 1024,  700, 269965,      0,      0,       0,  73.726,   30.00
 1024,  800, 283201,      0,      0,       0,  77.314,   30.01
 1024,  900, 286523,      0,      0,       0,  78.241,   30.00
```

Explanation of Results:

Note that both the throughput and samples go up with effort.