REAL-TIME INNOVATIONS, INC.

RTI® Data Distribution Service
C++ Throughput Test Example
Instructions

© 2006, Real-Time Innovations, Inc.
3975 Freedom Circle
Santa Clara, CA 95054
Phone (781) 306-0470 • Fax (781) 306-0472

This document and the contents herein, are proprietary and confidential to Real-Time Innovations, Inc. Distribution of the contents in any format without prior express permission of Real-Time Innovations, Inc. is prohibited.

Introduction

This document provides instructions on using the RTI Data Distribution Service throughput test example, which is provided in source code and is command line driven.

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. When 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 is indicative 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 and not multicast.

Test operation

The test code contains two applications: one for the publishing node, one for the subscribing node(s). The user starts the applications from the command line. The publication application will wait for the subscribing applications to announce their participation. Once the publisher recognizes that the specified numbers of subscribers are 'on-line,' it starts the test.

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.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Building the Test

The example is found in example/CPP/performance/throughput. For most operating systems, an example makefile is provided. For Windows, a sample project file is provided instead. The data type for the test is in Throughput.idl. The source files are:

- ThroughputArgs.h, ThroughputArgs.cxx: Command line arguments parsing.
- TimeManager.h, TimeManager.cxx: Test duration control.
- PerfMon.h, PerfMon.cxx: On supported platforms, CPU and memory usage measurement.
- ThroughputQos.h, ThroughputQos.cxx: QoS settings for entities that are used in the test.
- ThroughputTransport.h, ThroughputTransport.cxx: Transport configuration.
- ThroughputCommon.h, ThroughputCommon.cxx: Common utilities for both throughput publisher and subscriber.
- Throughput_publisher.cxx: Publisher application.
- Throughput_subscriber.cxx: Subscriber application.

Regardless of the platform, this test *requires* the following preprocessor defines, in addition to the platform-specific defines described in the RTI DDS Platform Notes:

- RTI_MULTICAST: Do not define if your platform does not support multicast.
- RTI_SHARED_MEMORY: Do not define if your platform does not support shared memory.
- RTI_IPV6: Do not define if your platform does not support IPv6 transport.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Building the Test using the Makefile

A sample makefile is provided with the throughput example. To generate a makefile specific to your platform, use rtiddsgen with the example flag, as follows:

```
rtiddsgen –example <your arch> -notypecode Throughput.idl
```

Since all the source files for this example are already present in the directory, rtiddgen may print messages saying that some files already exist and will not be replaced. You can safely ignore these messages.

In the generated makefile, add the preprocessor defines for multicast, shared memory and IPv6 to the DEFINES_ARCH_SPECIFIC variable, if your platform supports those features:

```
DEFINES_ARCH_SPECIFIC = -DRTI_UNIX -DRTI_MULTICAST –DRTI_SHARED_MEMORY –DRTI_IPV6
```

Add the additional source files (ThroughputArgs.cxx, TimeManager.cxx, PerfMon.cxx, ThroughputCommon.cxx, ThroughputQos.cxx, ThroughputTranport.cxx) to the COMMMONSOURCES variable in the generated makefile:

```
COMMONSOURCES = Throughput.cxx ThroughputPlugin.cxx ThroughputSupport.cxx
ThroughputArgs.cxx TimeManager.cxx PerfMon.cxx ThroughputCommon.cxx
ThroughputQos.cxx ThroughputTransport.cxx
```

Refer to the provided makefile as an example.

Building the Test on Windows using a Workspace

On Windows, use rtiddsgen to generate a workspace and a project file for compiling the throughput example:

```
rtiddsgen –example <your arch> -notypecode Throughput.idl
```

Since all the source files for this example are already present in the directory, rtiddgen may print messages saying that some files already exist and will not be replaced. You can safely ignore these messages.

Add the additional source files (ThroughputArgs.cxx, TimeManager.cxx, PerfMon.cxx, ThroughputCommon.cxx, ThroughputQos.cxx, ThroughputTranport.cxx) to the publisher and subscriber projects.

Add the preprocessor definitions RTI_MULTICAST and RTI_SHARED_MEMORY, if appropriate to your platform.

On Windows systems, we use the PDH library to measure CPU usage, so please add pdh.lib as an additional link library.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Special Note for VxWorks Users using Kernel Mode

If you will be running the test on VxWorks *in kernel mode*, please read the following:

Since this test runs in the same process as the kernel, there is no "main" function in the publisher or subscriber. In addition, VxWorks has a limit of 10 command-line options, which precludes you from running the entry point subscriber or publisher function directly. If the preprocessor define RTI_VXWORKS is set, the publisher and subscriber are created with entry point functions that read the command-line parameters from a file.

To use this functionality, create a file readable by the VxWorks OS and put your desired command line in this file. An example configuration file for a publisher would look like this:

```
-domain 88 –nic pub_IP -transport 1 -peer 1@sub1_IP \
-subscribers 1 –duration 10 –size 1024 –demand 1000:1000:9000
```

(Where pub_IP is the publisher's IP address and sub1_IP is the subscriber's IP address.)

Now pass the entry-point function (subscriber_main for subscriber, or publisher_main for publisher) the complete path to the configuration file as the only parameter.

On VxWorks 6.x systems, there is a known bug in the VxWorks kernel that manifests itself as a "memPartFree" error if the sequence of calls fopen-fread-fclose is encountered. This sequence of calls is used when reading in configuration parameters from a configuration file. If you encounter this problem, please hard-code the configuration parameters in the source files for the publisher and subscriber.

If you will be running in RTP mode, the test runs in a separate process from the kernel, and thus has a regular "main" function. None of the above limitations exist for RTP mode processes.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Publishing Application's Command-Line Options

The following command-line options are available for the publishing application. All options are preceded by the minus sign (-) and some take additional information where required. Test output can be captured using redirection on the command line. Some flags are mandatory.

All publishing applications use the participant index of 0.

- **-nic <IP>**. The IP address of any available network interface on the machine. This parameter is required so that RTI DDS is restricted to send data through this interface. By default, RTI DDS 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.
- **-domain #**. The domain ID that will be used for the test. Note that this test can be run at the same time as existing RTI DDS 4.x applications, provided that the domain number is unique.
- **-transport #**. This determines which transport to use for the test. 1 selects UDP over IPv4, 2 selects Shared Memory, and 8 selects UDP over IPv6.
- **-peer <peer>**. This argument is repeated as many times as there are peers. Since the subscriber only needs to discover the publisher, it will only specify one peer, but the publisher may specify multiple peers in a one-to-many test.
- **-subscribers #**. The number of subscribing applications that are participating in the test. Note that there may be more than one subscribing application (participant), on each node.
- **-duration #**. The number of seconds to be sustained for each demand run.
- **-size #**. Payload size.
- **-demand <initial effort>:<incremental effort>:<end effort>**. This specifies an effort range. As a practical matter, we control the effort with this parameter. This number is how many samples the publisher should write consecutively between 10ms sleep periods.
- **-reliable**. This tells the test to use reliable communication. By default, the test uses best-effort communication. This does not apply to raw transport tests.
- **-pull_on_write**. If specified while in reliable mode, this turns off push-on-write behavior. Otherwise, the writer will use push-based reliability.
- **-ttl #**. This argument indicates the number of Time-To-Live (TTL) hops for the multicast packet. This argument needs to be used for multicast test.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Subscribing Application's Command-Line Options

The following command-line options are available for the subscribing application. All options are preceded by the minus sign (-) and some take additional information where required. Test output can be captured using redirection on the command line. Unless otherwise stated, all options are mandatory.

- **-nic <IP>**. The IP address of any available network interface on the machine. This parameter is required so that RTI DDS is restricted to the send output through this interface. By default, RTI DDS 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.
- **-domain #**. The domain ID that will be used for the test. Note that this test can be run at the same time as existing RTI DDS 4.x applications, provided that the domain number is unique.
- **-transport #**. This determines which transport to use for the test. 1 selects UDP over IPv4, 2 selects Shared Memory, and 8 selects UDP over IPv6.
- **-index #**. Participant index number.
- **-peer <initial peers list>**. The descriptors of up to 32 peer nodes which are taking part in the test. To change the limit of the number of descriptors, modify the #definition for THROUGHPUT_MAX_SUBSCRIPTIONS in ThroughputGlobalSettings.hxx. The max_participant_index in each descriptor should match the –index number of the corresponding subscriber IP.
- **-mcast <multicast address to receive data on>**. Multicast address.
- **-reliable**. This tells the test to use reliable communication. By default, the test uses best-effort communication.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Example Command Line Arguments

In all of the following tests, the publisher is started first so that it can wait for subscribers to come on-line. In the table below, let pub_IP, sub1_IP, sub2_IP, sub3_IP, sub4_IP be the IP address of the publisher, subscriber 1, subscriber 2, subscriber 3, subscriber 4 IP address respectively. All IP address must belong to the same network.

Scenario	Publisher Arguments	Subscriber Arguments	
1-1 BE 1024 bytes over shmem	`-domain 88 –nic pub_IP -transport 2 -peer 1@shmem:// -subscribers 1 –duration 10 –size 1024 –demand 1000:1000:9000`	1	`-domain 88 –index 1 –nic sub1_IP -transport 2 -peer 0@shmem://`
1-4 BE 8192 bytes over shmem	`-domain 88 –nic pub_IP -transport 2 -peer 4@shmem:// -subscribers 4 –duration 10 –size 8192 –demand 1000:1000:9000`	1	`-domain 88 –index 1 –nic sub1_IP -transport 2 -peer 0@shmem://`
		2	`-domain 88 –index 2 –nic sub2_IP -transport 2 -peer 0@shmem://`
		3	`-domain 88 –index 3 –nic sub3_IP -transport 2 -peer 0@shmem://`
		4	`-domain 88 –index 4 –nic sub4_IP -transport 2 -peer 0@shmem://`
1-1 pull SR over shmem	`-domain 88 –nic pub_IP -transport 2 -peer 1@shmem:// -subscribers 1 –duration 10 –size 1024 –demand 1000:1000:9000 -reliable`	1	`-domain 88 -index 1 –nic sub1_IP -transport 2 -peer 0@shmem:// -reliable`
1-4 push SR over shmem, effort starting at 100, finishing at 900, with 100 increment	`-domain 88 –nic pub_IP -transport 2 -peer 4@shmem:// -subscribers 4 –duration 10 –size 1024 –demand 100:100:900 –reliable`	1	`-domain 88 –index 1 –nic sub1_IP -transport 2 -peer 0@shmem:// -reliable`
		2	`-domain 88 –index 2 –nic sub2_IP -transport 2 -peer 0@shmem:// -reliable`
		3	`-domain 88 –index 3 –nic sub3_IP -transport 2 -peer 0@shmem:// -reliable`
		4	`-domain 88 -index 4 –nic sub4_IP -transport 2 -peer 0@shmem:// -reliable`

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Scenario	Publisher Arguments	Subscriber Arguments	
1-1 BE over UDP unicast	-domain 88 –nic pub_IP -transport 1 -peer 1@sub1_IP -subscribers 1 –duration 10 –size 1024 –demand 1000:1000:9000	1	-domain 88 -index 1 –nic sub1_IP -transport 1 -peer 0@pub_IP
1-4 BE over UDP unicast	-domain 88 –nic pub_IP -transport 1 -peer 1@sub1_IP -peer 2@sub2_IP -peer 3@sub3_IP -peer 4@sub4_IP -subscribers 4 –duration 10 –size 1024 –demand 1000:1000:9000	1	-domain 88 -index 1 –nic sub1_IP -transport 1 -peer 0@pub_IP
		2	-domain 88 -index 2 –nic sub2_IP -transport 1 -peer 0@pub_IP
		3	-domain 88 -index 3 –nic sub3_IP -transport 1 -peer 0@pub_IP
		4	-domain 88 -index 4 –nic sub4_IP -transport 1 -peer 0@pub_IP
1-4 BE 32 bytes over UDP multicast	-domain 88 –nic pub_IP -transport 1 -peer 1@sub1_IP –peer 2@sub2_IP –peer 3@sub3_IP -peer 4@sub4_IP -subscribers 4 –duration 10 –size 32 –demand 1000:1000:9000 –ttl 1	1	-domain 88 -index 1 –nic sub1_IP -transport 1 -peer 0@pub_IP –mcast 225.3.2.1
		2	-domain 88 -index 2 –nic sub2_IP -transport 1 -peer 0@pub_IP –mcast 225.3.2.1
		3	-domain 88 -index 3 –nic sub3_IP -transport 1 -peer 0@pub_IP –mcast 225.3.2.1
		4	-domain 88 -index 4 –nic sub4_IP -transport 1 -peer 0@pub_IP –mcast 225.3.2.1

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.

Output Analysis

Typical output from Publisher

```
Starting test...
bytes  samples  Mbit/sec duration demand CPU %  memory
----- --------- -------- -------- ------ ------ ------
 1024,   162988,  44.505,   30.00,   100,  4.67,  64.9
 1024,   208667,  56.974,   30.00,   200,  1.00,  64.9
 1024,   233846,  63.850,   30.00,   300, 10.75,  64.9
 1024,   240650,  65.710,   30.00,   400,  8.18,  64.9
 1024,   259260,  70.795,   30.00,   500, 11.08,  64.9
 1024,   263401,  71.901,   30.01,   600, 15.94,  64.9
 1024,   269965,  73.718,   30.00,   700, 13.63,  64.9
 1024,   283201,  77.330,   30.00,   800, 13.72,  64.9
 1024,   286523,  78.238,   30.00,   900, 11.87,  64.9
```

Typical output from Subscriber

```
Starting test...
bytes demand samples  lost(A) Nreject lost(N)  Mbit/s  duration CPU % memory
----- ----- -------- ------- ------- ------- -------- -------- ------ ------
 1024,  100, 162988,       0,      0,       0,  44.502,   30.00,  6.77,  62.5
 1024,  200, 208667,       0,      0,       0,  56.985,   30.00,  8.77,  62.5
 1024,  300, 233846,       0,      0,       0,  63.843,   30.01,  9.90,  62.5
 1024,  400, 240650,       0,      0,       0,  65.714,   30.00, 10.05,  62.5
 1024,  500, 259260,       0,      0,       0,  70.790,   30.00, 11.18,  62.5
 1024,  600, 263401,       0,      0,       0,  71.908,   30.01, 11.40,  62.5
 1024,  700, 269965,       0,      0,       0,  73.726,   30.00, 11.52,  62.5
 1024,  800, 283201,       0,      0,       0,  77.314,   30.01, 11.93,  62.5
 1024,  900, 286523,       0,      0,       0,  78.241,   30.00, 12.22,  62.5
```

Explanation of Results

Note that both the throughput and samples go up with demand. The CPU usage on both the publisher and the subscriber also increase.[1]

[1] CPU and memory usage is measured by the PerfMon class shipped with the test. Currently, only Linux fully supports performance measurement. On Windows XP, only CPU measurement is supported. On Windows 2000, performance measurement is not supported even though the program itself runs fine.

Copyright © 2007 Real-Time Innovations, Inc. All rights reserved.