FAST RTPS THROUGHPUT TEST

eProsima
Proyectos y Sistemas de Mantenimiento SL
Ronda del poniente 16 – Bajo K
28760 Tres Cantos Madrid
Tel: + 34 91 804 34 48
info@eProsima.com – www.eProsima.com

Technical Support

- Phone: +34 91 804 34 48
- Email: Support@eProsima.com

Contents

Figure List

Table List

1 Introduction

1.1 Throughput

In communication networks the throughput is usually defined as the rate of successful message delivery over a communication channel. The throughput is usually expressed in bytes per second. There are different methods to measure the throughput of a communication network. The most common ones are to send a large file (or multiple smaller ones) and measure the time that takes to transmit it to another point of the network. Then the amount of data is divided by the time it took to send it.

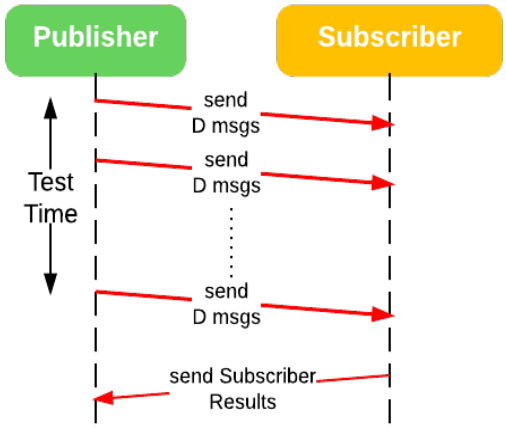
In the case of a RTPS communication, the throughput can be measured sending groups of messages in a certain amount of time and the obtaining the combined size of the transmitted data. However, to obtain the maximum throughput value different message demands (the number of continuous messages send) must be tried in order to find the best value; i.e., one that maximize the available send pipe in the publisher without overflowing the receive queue in the subscriber (producing packet losses).

The test developed to obtain the throughput values for the eProsima RTPS library follows these steps:

1. for msg_size = {16,32,...,8192}
    1. for demand = {500,750,850,1000,1250,1400,1500,1600,1750,2000}
        1. iterations = 0;
        2. Time t1=now()
        3. while(t2-t1<Test_Time)
            1. write demand messages
            2. Time t2 = now()
            3. iterations++;
            4. wait 50 ms
        4. total_time = t2-t1-iterations*50ms
        5. throughput=demand*msg_size*iterations/total_time

For each message size and each demand the test tries to send the maximum number of messages in the test time (which is provided as an argument). In order to avoid blocking the read or write pipes a small wait of 50ms is introduced after the publisher sends a block of messages (the number depends on the demand).

The results obtained are the throughput both at the publisher and at the receiver, as shown in following sections.



2 Throughput Test

The throughput test distributed with this release of eProsima RTPS is develop to be able to determine the maximum throughput value for different messages data size, both in the subscriber and in the publisher side.

2.1 Source Files

The Latency test distributed with this release of eProsima RTPS consist of the following source files:

- ThroughputTypes.cpp/.h: Serialization and deserialization method of the two data types used in the test: LatencyType (a sample number and an array of bytes) and ThroughputCommandType (a specific data used by the endpoints to communication the beginning of the test).
- ThroughputPublisher.cpp/.h: This class contains the publisher side of the application. Two DDS publishers and subscribers are created (a pair for the latency data communication and a pair for the commands). This class is also in charge of saving the times and calculating the time statistics.
- ThroughputSubscriber.cpp/.h: This class contains the subscriber side of the application. As in the publisher side two pairs of publisher/subscriber are created to perform the test.
- main_ThroughputTest.cpp: This is the main file of the application that creates a ThroughputPublisher or a ThroughputSubscriber depending on the command line options.
- payload_demands.csv: A list of the predefined list of payloads and demands used in the test.

2.2 Building the test

A current version of eProsima RTPS library needs to be installed and compiled to be able to build and execute the test.

2.2.1 Unix

The test is provided with a Makefile that should allow the user the compilation in any Unix-based machine, just executing *make* in the home directory of the test. Two executable files are generated: ThroughputTest and ThroughputTestd, one for the release and one for the debug version of the test. To obtain reliable results the release version should be used. **Boost 1.53** needs to be installed on the system.

2.2.2 Windows

The test is provided with a Visual Studio 2010 project, shared with other use tests provided with the release. The test compiles without problem generating two executable files: ThroughputTest.exe and ThroughputTestd.exe, again the release and debug version. In case the user would want to generate its own project it must be noted that, along with **boost 1.53** libraries, the following dependencies are also needed: **Shlwapi.lib** and **Iphlpapi.lib**.

2.3 Test operation

The same application file is used in either side of the test (publisher or subscriber). The behavior of the test will depend on the command line arguments provided to the application.

2.3.1 Command line options

The arguments that can be passed to the application are:

1. “publisher or subscriber”: The first argument indicates the application whether is going to be acting as the publisher side or the subscriber side in this test.
2. The second argument is different depending on the type of endpoint the application is meant to be:
    1. Publisher: number of seconds the publisher should send messages for each demand. For optimal results this time needs to be large enough to allow the test to send sufficient messages. In out tests good results are obtained with times larger that 5 seconds.
    2. The subscriber doesn't need any additional arguments.
3. Thrid and four arguments are optional and can be provided to the publisher. If we chose to provide this arguments only an instance of the test will be carried out with the demand a number of bytes provided by the user instead of using the predefined lists of demands and data sizes included in the payload_demands.csv file provided with the test.
    1. Demand: Number of continuous messages sent to the subscriber.
    2. Bytes: Number of bytes to be used, with a maximum of 24000.

2.3.2 Command line examples

The following table present two example of command line options for two different test.

One to one test with predefined demand and message sizes	***Command***
Publisher:	*ThroughputTest publisher 5*
Subscriber:	*ThroughputTest subscriber*
Test with given demand (2000) and bytes (10000)	
Publisher	*ThroughputTest publisher 5 2000 10000*
Subscriber	*ThroughputTest subscriber*

2.4 Test Results

The throughput results are measured both in the publisher and in the subscriber side. However, for simplicity they are only provided in the publisher side. The subscriber transmits its results to the publisher after each test is finished. An example of the provided results can be observed below:

```
[      TEST      ][              PUBLISHER              ][                    SUBSCRIBER                    ]
[ Bytes, Demand][Sent Samples,Send Time(us), MBits/sec][Rec Samples,Lost Samples,Rec Time(us), MBits/sec]
[------,-------][------------,-------------,----------][-----------,------------,------------,----------]
     16,    500,      152500,      5011845,     3.895,      152500,           0,      5019968,     3.888
     16,    750,      194250,      5014519,     4.958,      194079,         171,      5028145,     4.941
     16,    850,      206550,      5003913,     5.284,      206550,           0,      5018143,     5.269
     16,   1000,      223000,      5005011,     5.703,      223000,           0,      5019052,     5.687
     16,   1250,      256250,      5015956,     6.539,      256250,           0,      5028885,     6.522
     16,   1400,      260400,      5007107,     6.657,      260400,           0,      5020571,     6.639
     16,   1500,      258000,      5012665,     6.588,      258000,           0,      5025132,     6.572
     16,   1600,      270400,      5018617,     6.897,      270293,         107,      5031175,     6.877
     16,   1750,      278250,      5013824,     7.104,      278250,           0,      5026076,     7.086
     16,   2000,      284000,      5031777,     7.224,      283886,         114,      5043818,     7.204
     32,    500,      158000,      5006978,     8.078,      158000,           0,      5021297,     8.055
     32,    750,      198750,      5013067,    10.149,      196947,        1803,      5027527,    10.028
     32,    850,      199750,      5014290,    10.198,      198869,         881,      5027873,    10.126
```