CptS 464/564 Project #1
"Simple Publisher & Subscriber"
Given: Tuesday, February 18, 2020
Due: 11:59pm, Sunday, ~~March 1, 2020~~, March 8, 2020 via BlackBoard
Weight: 10% of Final Grade

February 18, 2020

1 Overview

In this project, you will create your first middleware program using RTI (RealTime Innovations) DDS (Data Distribution Service). It is a basic program that publishes some information over the network *selectively*, by employing a QoS policy. The purpose of this project is to give you an idea of how middleware actually works in a real world situation and get you familiar with RTI Connext® DDS. This first project shouldn't cost you more than five hours but just get you started a little bit, and get over the initial learning curve in using RTI DDS. RTI DDS is network middleware for real-time distributed applications. It provides the communications service that programmers need to distribute time critical data between embedded and/or enterprise devices or nodes. RTI Data Distribution Service uses a publish-subscribe communications model to make data-distribution efficient and robust.

NB: Before starting this project, you must install RTI DDS on your laptops and/or desktops, please follow Section 5 for doing so.

2 Concepts

The TIME_BASED_FILTER QosPolicy allows you to specify that data should not be delivered more than once per specified epoch/period for data-instances of a DataReader/Subscriber, regardless of how fast DataWriters/Publishers are publishing/generating new samples of data.

This QoS policy allows you to optimize resource usage (CPU and possibly network bandwidth) by only delivering the required amount of data to different Subscribers.

DataWriters may send data faster than needed by a DataReader. For example, a DataReader of sensor data that is displayed to a human operator in a GUI application

does not need to receive data updates faster than a user can reasonably perceive changes in data values. This is often measure in tenths (0.1) of a second up to several seconds. However, a DataWriter of sensor information may have DataReaders that are processing the sensor information to control parts of the system and thus need new data updates in measures of hundredths (0.01) or thousandths (0.001) of a second.

With this QoS policy, different DataReaders can set their own time-based filters, so that data published faster than the period set by a DataReader will be dropped by the middleware and not delivered to the DataReader. Note that all filtering takes place on the reader side (as opposed to GridStat, which we will discuss later).

3 Specification

We will have two main kinds of objects in our system: `MsgPublisher`, and `MsgSubscriber`. In this project, the Publisher application (MsgPublisher) publishes every 0.25 seconds.

`MsgSubscriber` sets a time-based filter with a minimum separation of 1 second (you could make this configurable as well, for instance, through a command line parameter). Samples arriving earlier than the time slice of 1 seconds are silently dropped (you could however, record them on a file or so). So, samples will arrive faster than the minimum separation time, as `MsgPublisher` generates a new message every 0.25 seconds.

You must not perform any filtering on the `MsgPublisher` side, only `MsgSubscriber` (the reader) will have the necessary code to filter out messages ariving within the 1 second interval. Rest of the messages will be displayed by `MsgSubscriber`, with the sequence number of the message displayed.

You could set the time-based filter QoS via an XML file, or programmatically. Following is an example of the QoS filter in the USER_QOS_PROFILES.xml file:

```
<datareader_qos>
       <time_based_filter>
              <minimum_separation>
                     <sec>1</sec>
                    <nanosec>0</nanosec>
              </minimum_separation>
       </time_based_filter>
</datareader_qos>
```

If you want to set the time-based filter QoS programmatically, you will find relevant code examples in the RTI DDS example folder.

4 Functionality

You would need to perform the following:

- `MsgPublisher`: publishes data of type `Msg` (mentioned in the IDL file) periodically every 0.25 seconds. Please use "`CS464/564 Project 1 <your_username>`" as topic name. That is, use either “464” or “564”, not “464/564”.

- `MsgSubscriber`: Displays the message received from the publisher if the time difference between the last message’s display time and the current message arrival time is at least 1 second. Again, please use "CS464/564 Project 1 <your_username>" as topic name. Please note that the subscriber will not get any message from publisher if they have different domain ID or topic names. Conversely, if you want to play around

- You should also add a counter (starting from zero) for the messages that the subscriber receives from the publisher. Remember, even if you are throwing away a message, the counter should be incremented.

The program output (for publisher and subscriber) should be similar to:

```
//Publisher:

Writing Apple     #0
Writing Spinach #1
Writing Carrots #2
Writing Lentil #3
Writing Banana #4
Writing Potato #5
Writing Pear #6
Writing Avocado #7
Writing Cherry #8
Writing Orange #9
...

// Subscriber:

Msg Received #0:
sender: JohnDoe (Please use your own name here)
message: Apple!

Msg Received #4:
sender: JohnDoe
message: Banana!

Msg Received #9:
sender: JohnDoe
message: Orange! ...
```

The variables and their types should be defined in an Interface Definition Language (IDL) file, for instance pr1.idl could look like the following.

```
// Msg.idl
const long MSG_LEN=256;
struct Msg {
        string<MSG_LEN> sender;
        string<MSG_LEN> message;
};
```

You can use either Java, C or C++ for coding. One or two of you might want to get ambitious and try another language. This is OK, but do it first in either Java, C, or C++. You'll find example of these in the distribution, but with any other language you are on your own.

5 Installing RTI DDS on your computer and example codes

Here is the official guide which describes how to download and install RTI Data Distribution Service. It also lays out the core value and concepts behind the product and takes you step-by-step through the creation of a simple example application. Developers new to RTI should read this document. In general, installation is quite straightforward. You would need to obtain an evaluation license (generally valid for 30 days), also available from the downloads page. Following are the instructions for installing RTI DDS on Mac/Linux, Windows users please refer to the installation guide.

- Download the correct RTI DDS software according to your machine|O/S**. Be sure to select "University research and education" as your license type, and register with your WSU email address (not a gmail or hotmail one, etc).**
    - Find it at https://www.rti.com/downloads, select a University research and education License. Then download the appropriate installer from https://www.rti.com/downloads/connext-files
- On linux you may need to change the permissions of the downloaded file and add the execute bit with: chmod +x installerName
- On mac, you may have to right click the installer in the mounted dmg and select open
- On Linux, open the .run file and follow the GUI install prompts
- Get the evaluation license from your email:
    - On Linux you may have to set it with: export RTI_LICENSE_FILE=/full/path/to/rti_license.dat
    - On most platforms, the rti launcher will ask you to specify the location of the license file
- You may have to set RTI DDS compiler/linker paths:

- In Linux: source /path/to/RTI-5.3.0/RTI/rti_set_bash_5.3.0
  Or export the path to the bin directory it in .bash_profile/.bashrc
- Mac, add to your ~/.bash_profile: export PATH="/Applications/rti_connext_dds-5.3.0/bin:${PATH}"
- In windows, Path is defined under Control Panel>System>Advanced system settings>Environment Variables under the System Variables section

Check out some RTI DDS example codes from here:

- https://community.rti.com/examples

Also, check out the Hello World example: http://community.rti.com/rti-doc/530/ndds/doc/html/api_cpp/HelloWorld_8cxx-example.html

5.1 Building your project

- Open up two terminals, ssh into 2 different terminals
- You may need to set up the paths in PATH environment variable, based on where RTI is installed.
- Under your home create a file named pr1.idl (this is the Interface Definitation Language file, which is used to specify the data model).
- Generate source code and makefile from IDL – run rtiddsgen [1]
  - For C++:
    rtiddsgen pr1.idl -language C++ -example x64Linux2.6gcc4.4.5
    -replace -ppDisable
  - For Java:
    rtiddsgen pr1.idl -language Java -example x64Linux2.6gcc4.4.5jdk -replace –ppDisable

    You may have to replace the architecture after the –example to match your build environment, use rtiddsgen –help for a complete list
- Modify the generated source code files according to the project requirements
  - For C++: modify publisher.cxx and subscriber.cxx
  - For Java: modify MsgPublisher.java and MsgSubscriber.java
- Compile
  - For C++: make -f makefile_pr1_x64Linux2.6gcc4.4.5
  - For Java: make -f makefile_pr1_x64Linux2.6gcc4.4.5jdk

[1] All you need to start with is the .idl file to start with, and RTI will generate all the other requisite files, such as project source file templates (one for publisher, and other for subscriber), headers, and makefiles.

- Run your programs
  - For C++ –
    In one terminal: obj/x64Linux2.6gcc4.4.5/publisher
    In another terminal: obj/x64Linux2.6gcc4.4.5/subscriber
  - For Java –
    In one terminal: make -f makefile_pr1_x64Linux2.6gcc4.4.5jdk MsgPublisher
    In another terminal: make -f makefile_pr1_x64Linux2.6gcc4.4.5jdk MsgSubscriber
  - Another way for Java –
    In one command window: java -classpath .:$NDDSHOME/class/nddsjava.jar MsgPublisher
    In another command window: java -classpath .:$NDDSHOME/class/nddsjava.jar MsgSubscriber

6 Deliverables and General Instructions

The project is due at 11:59 PM on March 1. This means that all files must be submitted via Blackboard by this time. You *must* put them all in a folder, and compress it. Please follow this naming convention of the compressed file: CS<464/564>Project1<your_username>.zip

- All the files you created or modified by hand. Note, this does not include files that are generated by the IDL compiler and not modified by you. Also turn-in any additional files that you have modified or created.
- We need to see a sample output to assess that your program is correctly running (i.e, the publisher sends messages and the subscriber receives and displays messages). Copy the output from the screen or redirect it to a file. Please also take a clear/legible screen-shot when your programs are running. Put all these files into a folder called output so we know where to look for them. Create a PDF file called "CS<464/564>Project1Output<your_username>.pdf" It should include a sample ouput file, a screen shot of it running, and the source files below (formatted nicely if possible).
- The only files that you will be turning in are listed below.

  For C/C++:

  * pr1.idl
  * publisher.cxx/c
  * subscriber.cxx/c
  * Sample output
  * Screenshot of running programs

For Java:

- pr1.idl
- MsgPublisher.java
- MsgSubscriber.java
- Sample output
- Screenshot of running programs

. Troubleshooting installation

- If you get an error like this:

  RTI Data Distribution Service No source for License information Please contact support@rti.com with any questions or comments.
  DDSDomainParticipant_impl::create_disabledI:!create participant
  DDSDomainParticipant_impl::createI:!create participant
  DomainParticipantFactory_impl::create_participant():
  !create failure creating participant create_participant error
  Then the shell does not know RT_LICENSE_FILE path, or the path has a typo, or the file is not readable by anyone other than *root*, et al.

- You would need to open multiple terminal windows to run publisher and subscriber programs, so all paths that are set on one window/shell, must also be set on the other.

Note: You can use RTI DDS Forum to learn. However, please do not embarrass either WSU or your professor. That is, try to find the answers in the forums, make sure you have looked through the documentation, example programs, etc.