

CptS 464/564 Project #1

Simple Publisher & Subscriber

Given: Wednesday, September 7, 2011

Due: 6pm Friday, September 23, 2011, via ANGEL

Weight: 5% of Final Grade

Overview

In this project, you will create your first middleware program using RTI (Real-Time Innovations) DDS (Data Distribution Service). It is a basic program that publishes information over the network. The purpose of this project is to give you an idea of how middleware actually works and get you familiar with RTI DDS. This first project shouldn't cost you more than three 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.

Problem setting

Design and implement a simple Publisher/Subscriber system, using RTI DDS. We will have two main kinds of objects in our system: `LittleMsgPublisher`, and `LittleMsgSubscriber`.

`LittleMsgPublisher`: publishes data of type `LittleMsg` periodically every 4 seconds. Please use "CS464/564 Project 1 your_username" as topic name.

`LittleMsgSubscriber`: Checks for and displays the message received from the publisher. It prints out a message of its own every 6 seconds. 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.

The data is defined in `little.idl`:

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

Expected output:

```
//Publisher:

Writing LittleMsg, #0
Writing LittleMsg, #1
```

```
Writing LittleMsg, #2
...

//Subscriber:

LittleMsg subscriber sleeping for 6 sec...
LittleMsg Received:
    sender: Dario (Please use your own name here)
    message: Hello world!

LittleMsg subscriber sleeping for 6 sec...
LittleMsg Received:
    sender: Dario
    message: Hello world!

...
```

Additional Work for 564

Students enrolled in CptS 564 should add a counter (starting from zero) for the messages that the subscriber receives from the publisher. So the expected output for subscriber would be:

```
//Subscriber:

LittleMsg subscriber sleeping for 6 sec...
LittleMsg Received #0:
    sender: Dario (Please use your own name here)
    message: Hello world!

LittleMsg subscriber sleeping for 6 sec...
LittleMsg Received #1:
    sender: Dario
    message: Hello world!

...
```

The message counter for the subscriber of the assignment is only 10% of your grade, so we strongly suggest you get the base program going first, and then add the message counter for the subscriber. Once you get the rest going, it should at take no more than 30 minutes.

Implementation language

You can use either Java or C++.

Implementation steps

We have 6 virtual machines whose names are from rti-dds-01.eecs.wsu.edu to rti-dds-06.eecs.wsu.edu. Please ssh to one or more of them to do your programming projects. You will work under your EECS account home directory (/net/stu/ugrads/your_username or /net/stu/grads/your_username).

1. Set your environment variables:

Please add the following two lines to the `.bashrc` file under your EECS account home directory:

```
export NDDSHOME=/opt/RTI/ndds.4.5c
export PATH=$NDDSHOME/scripts:$PATH
```

If you use Java, please add two more lines to your `.bashrc` file:

```
export PATH=/usr/local/jdk1.6.0_21/bin:$PATH (or your own Java path)
export LD_LIBRARY_PATH=$NDDSHOME/lib/i86Linux2.6gcc4.1.1jdk:$LD_LIBRARY_PATH
```

2. Under some subdirectory of your EECS account home directory, create a file named `little.idl` as specified above.

3. Run `rtiddsgen`:

For C++:

```
rtiddsgen little.idl -language C++ -example i86Linux2.6gcc4.1.1
-replace -ppDisable
```

For Java:

```
rtiddsgen little.idl -language Java -example i86Linux2.6gcc4.1.1jdk
-replace -ppDisable
```

4. Modify the generated source code files according to the problem setting specified above:

For C++: modify `little_publisher.cxx` and `little_subscriber.cxx`

For Java: modify `LittleMsgPublisher.java` and `LittleMsgSubscriber.java`

5. Build your programs:

For C++: `gmake -f makefile_little_i86Linux2.6gcc4.1.1`

For Java: `gmake -f makefile_little_i86Linux2.6gcc4.1.1jdk`

6. Run your programs:

For C++:

In one command window: `./obj/i86Linux2.6gcc4.1.1/little_publisher`

In another window: `./obj/i86Linux2.6gcc4.1.1/little_subscriber`

For Java:

In one command window:

```
gmake -f makefile_little_i86Linux2.6gcc4.1.1jdk LittleMsgPublisher
```

In another command window:

```
gmake -f makefile_little_i86Linux2.6gcc4.1.1jdk LittleMsgSubscriber
```

An alternate way for Java:

In one command window:

```
java -classpath .:$NDDSHOME/class/nddsjava.jar LittleMsgPublisher
```

In another command window:

```
java -classpath .:$NDDSHOME/class/nddsjava.jar LittleMsgSubscriber
```

Note: You can run any number of publishers and subscribers programs, and can add and remove them dynamically from the domain.

Turn-ins

The project is due at 6PM on Friday September 23. This means that all files must be stored under your EECS account home directory by this time and they must remain untouched after this time.

Via ANGEL you are to turn in:

1. 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.
2. Sample output when your program is correctly running. (That is, the publisher sends messages and the subscriber receives and displays messages). You need to copy the output from the screen (or send it to a file) and print it out. Please also provide a printout of the screen copy when your programs are running.

The files that you will be turning in are listed in the Appendix II. Do not modify the last accessed dates on your created or modified files after the submission date, in case we need to check the files.

Appendix-I

The files generated when you compile the IDL file are listed below.

For C++:

```
little.h  
USER_QOS_PROFILES.xml
```

```
littleSupport.h
littleSupport.cxx
little_publisher.cxx
little_subscriber.cxx
littlePlugin.h
littlePlugin.cxx
little.cxx
makefile_little_i86Linux2.6gcc4.1.1
```

For Java:

```
MSG_LEN.java
USER_QOS_PROFILES.xml
Makefile_little_i86Linux2.6gcc4.1.1jdk
LittleMsgTypeSupport.java
LittleMsgTypeCode.java
LittleMsgSeq.java
LittleMsgDataWriter.java
LittleMsgDataReader.java
LittleMsg.java
LittleMsgPublisher.java
LittleMsgSubscriber.java
```

Appendix-II

The files to be turned in are listed below.

For C++:

1. little_publisher.cxx
2. little_subscriber.cxx
3. sample output
4. screenshot of running programs (on Windows you can hit the "PrtScn" key to get your entire desktop, use ALT-PrtScn to get only the active window).

For Java:

1. LittleMsgPublisher.java
2. LittleMsgSubscriber.java
3. sample output
4. screenshot of running programs (on Windows you can hit the "PrtScn" key to get your entire desktop, use ALT-PrtScn to get only the active window).

Also turn-in any additional files that you have modified or created.

Appendix-III

[\\$NDDSHOME/doc/pdf/RTI_DDS_GettingStarted.pdf](#) is a 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.