جامعة الملك فهد للبترول والمعادن
King Fahd University of Petroleum & Minerals

COE 523, Term 232

Distributed Computing

Assignment# 2
Due date: Saturday, April 27, 2024
Instructor: Dr. Ayaz ul Hassan Khan

Objective:

In this programming assignment, you are going to implement a scenario-based system that uses the publish-subscribe communication model with RTI Connext DDS to transmit patient vital sign data in a distributed computing environment. The system demonstrates various aspects of DDS, including QoS, dynamic joining and leaving.

Problem Statement:

A hospital needs to monitor the vital signs of multiple patients in real-time using a network of sensors. The sensors should be able to publish patient data to a central server, which will store the data and alert healthcare providers in case of any abnormalities. The healthcare providers should be able to subscribe to patient data for specific patients, and the system should be able to handle the dynamic joining and leaving of sensors and healthcare providers.

Requirements:

Design and implement a distributed system that uses the publish-subscribe communication model with RTI Connext DDS to monitor patient vital signs in real time. Your system should include the following components:

1. **Sensors:** Simulate multiple sensors that generate patient vital sign data, including heart rate, blood pressure, and oxygen saturation. The sensors should publish data to a central server using RTI Connext DDS.
2. **Central Server:** Receive and store patient vital sign data from multiple sensors using RTI Connext DDS. The central server should also be responsible for distributing the data to healthcare providers based on their subscriptions.
3. **Healthcare Providers:** Simulate multiple healthcare providers who are interested in monitoring the vital signs of specific patients. The healthcare providers should subscribe to patient data for specific patients using RTI Connext DDS.
4. **QoS:** Your system should demonstrate the use of different QoS settings, such as reliability and resource usage. You should also experiment with different QoS settings to observe their effects on system performance and reliability.

5. **Dynamic Joining and Leaving:** Your system should handle dynamic joining and leaving of sensors and healthcare providers. You should experiment with adding and removing sensors and healthcare providers at runtime to observe how the system handles these changes.
6. **Documentation:** Document your code and provide a report that includes a description of your system design, observations about system performance and reliability, and recommendations for future improvements.

Note: You can use any programming language (python recommended) that supports RTI Connext DDS. You can also modify the scenario to suit your specific requirements.

Good luck with your assignment!

Required Components:

- **Sensors:** Python script that simulates sensors and publishes patient vital sign data using RTI Connext DDS.
- **Central Server:** Python script that receives and stores patient vital sign data from sensors using RTI Connext DDS. The central server also distributes the data to healthcare providers based on their subscriptions.
- **Healthcare Providers:** Python scripts that simulate healthcare providers and subscribe to patient data for specific patients using RTI Connext DDS.
- **QoS:** The system demonstrates the use of different QoS settings, such as reliability and resource usage, by configuring RTI Connext DDS accordingly.
- **Dynamic Joining and Leaving:** The system handles dynamic joining and leaving of sensors and healthcare providers by using the DDS discovery service to discover and register new participants at runtime.

Possible Implementation Strategy:

1. Sensors
   a. Create a Python script that generates patient vital sign data and publishes it using RTI Connext DDS.
   b. The script should use the DDS API to create a DDS domain participant, topic, and publisher.
   c. The script should generate patient data using random values for heart rate, blood pressure, and oxygen saturation.
   d. The script should publish patient data to the central server using the DDS publisher.
2. Central Server
   a. Create a Python script that receives and stores patient vital sign data from sensors using RTI Connext DDS.
   b. The script should use the DDS API to create a DDS domain participant, topic, subscriber, and data reader.

   c. The script should receive patient data using the DDS data reader and store it in a database or file.
   d. The script should distribute patient data to healthcare providers based on their subscriptions using the DDS publisher.

3. Healthcare Providers
   a. Create a Python script that subscribes to patient data for specific patients using RTI Connext DDS.
   b. The script should use the DDS API to create a DDS domain participant, topic, subscriber, and data reader.
   c. The script should subscribe to patient data using the DDS data reader and display it in a user interface or log file.
   d. The script should allow healthcare providers to specify their subscription criteria, such as patient ID or vital sign readings, using the DDS API.
4. QoS
   a. Configure RTI Connext DDS to use different QoS settings, such as reliability and resource usage, by modifying the XML configuration files.
   b. Run experiments to observe the effects of different QoS settings on system performance and reliability.
5. Dynamic Joining and Leaving
   a. Use the DDS discovery service to discover and register new participants at runtime.
   b. Implement logic in the central server and healthcare provider scripts to handle dynamic joining and leaving of sensors and healthcare providers.
   c. Run experiments to observe how the system handles adding and removing sensors and healthcare providers at runtime.

Note: This is just one possible implementation strategy for the assignment. There are many ways to approach this problem, and you may choose to implement the system differently.

Grading Scheme:

1. Implementation (50 points)
   a. Sensors: 10 points
      i. Generates and publishes patient data
   b. Central Server: 20 points
      i. Receives and stores patient data
      ii. Distributes patient data to healthcare providers
   c. Healthcare Providers: 20 points
      i. Subscribes to patient data
      ii. Displays patient data that meet subscription criteria
2. Use of RTI Connext DDS (40 points)
   a. QoS: 20 points
      i. Uses appropriate QoS settings for topics, publishers, and subscribers
   b. Dynamic Joining and Leaving: 20 points
      i. Handles dynamic joining and leaving of sensors and healthcare providers

3. Code quality (10 points)
    - a. Code is well-documented and easy to read: 5 points
    - b. Code follows best practices for Python programming: 5 points

Total: 100 points