Middleware Architectures
[IS3108 / SCS3203]
Assignment 01

1. Objective

This assignment focuses on designing and implementing a simple Publish and Subscribe middleware using Client-Server Sockets Programming concepts and techniques.

2. Introduction

The assignment is related to the **Publish and Subscribe(Pub/Sub) Middleware Architecture** software implementations. In a Pub/Sub architecture, there are message Publishers and Subscribers. A special type of middleware amalgamates the Publishers and Subscribers in an Asynchronous mode of communication. The messages among the participants could be grouped based on unique topics, where a publisher may publish on one or many topics, and a subscriber may also listen to one or many topics.

3. Duration and Team Composition

The duration for the assignment is **2 weeks** and should be done in **teams of 6-8** students.

4. Submission Guidelines

The assignment submission should comply with the following criteria.

1. A short demonstration video/screen-cast of the implementation should be recorded and submitted. There should be a video for tasks 1 - 3, given in the assignment outline.
2. The source code of the implementation should be provided with the submission. Teams are allowed to select a programming language out of the following options - C/C++, Java, Python.
3. Documented feedback to task 4.
4. All content of submission (screencasts, screen capture images, source code and documentation) should be organised into a meaningful directory structure and uploaded to the VLE in compressed/zipped format.

5. Assignment Outline

Task 1: The Client-Server Application

1. Implement a client-server socket application using the selected programming language. The server will listen for connections on a pre-defined PORT where the client will use the server IP and the PORT to connect to the server.

2. The implementation only needs a Command Line Interface (CLI) to show the activities between the Server and the Client applications. Using a Graphical User Interface (GUI) based implementation will not influence the assignment to be graded higher than a CLI-based implementation.

3. Start the Server by passing the PORT as a command line argument.

   E.g.

   my_server_app 5000

4. Start a Client to connect with the server by passing Server IP and Server PORT as command line arguments.

   E.g.

   my_client_app 192.168.10.2 5000

5. After starting the Client, whatever text is typed on the client CLI should be displayed on the server side CLI as standard system output text.

6. The primary objective is to demonstrate the client and server communicating with each other.

7. The client should perpetually run until the "terminate" keyword is typed by a user. After typing the keyword, the client should disconnect from the server and terminate.

Screencast Video: *This should depict the CLI screens of the Server and the Client exchanging messages.*

[Marks 20/100]

Task 2: Publishers and Subscribers

1. Improve the client-server implementation for the Server to handle multiple concurrent Client connections.

2. Multiple client applications should be able to connect to the server and the typed text in a given client CLI should be displayed on the Server terminal CLI.

3. The client application should take the third command line argument to indicate whether the client will act as either "Publisher" or "Subscriber".

   E.g.

   my_client_app 192.168.10.2 5000 PUBLISHER

   my_client_app 192.168.10.2 5000 SUBSCRIBER

4. Further, the server should echo the messages sent by any "Publisher" clients to all the "Subscriber" clients' terminals. For example, when a single server is connected with 10 client applications when a client terminal that is in Publisher mode types a text, it should be displayed on all remaining client terminals that are connected as Subscribers.

5. The Publisher messages should be only shown on Subscriber terminals, not on any Publisher terminals.

***Screencast Video**: This should depict the CLI screens of the Server and several Client terminals exchanging messages. Few of the Client terminals should be publishers and few should be Subscribers.*

[Marks 30/100]

Task 3: Publishers and Subscribers Filtered on Topics/Subjects

1. Improve the implementation of Task 2 to include **"topic/subject"** based filtering of messages among Publishers and Subscribers.

2. The client implementation should be improved to include the fourth command line argument of the topic/subject of either the publisher or subscriber.

   E.g.

   my_client_app 192.168.10.2 5000 PUBLISHER TOPIC_A

   my_client_app 192.168.10.2 5000 SUBSCRIBER TOPIC_A

3. The client application that is a publisher should be sending the messages to the server on a specific topic/subject to route to interested subscriber clients on the same topic/ subject.

4. There can be publishers and subscribers interested in different messages connected concurrently to the server.

***Screencast Video**: This should depict the CLI screens of the Server and several Client terminals exchanging messages on different topics. Few of the Client terminals should be publishers on different topics and few should be Subscribers on topics declared by publishers.*

[Marks 40/100]

Task 4: Enhance the Architecture to Gain Improvement in Availability and Reliability

A single server-based architecture for the distribution of messages among Publishers and Subscribers developed in this assignment has a single point of failure - the server. Propose a new distributed architecture for the Pub/Sub implementation to gain improved availability and reliability over the single server node failure deficiency. Only required to draw and describe the new architecture and improvements to the implementation are not expected.

[Marks 10/100]

6. Mode of Evaluation

The evaluation of the assignment would be done based on the following criteria.

1. Evaluation of the assignment tasks based on the video recording – separate screencast videos for each of the tasks(1-3) listed above should be submitted. Each video should demonstrate the CLI interfaces of server and client applications to justify the implementation guidelines of each task.

2. Evaluation of the source code – The source code for both client and server applications, after the completion of task 3, should be submitted.

3. Documentation for task 4.

4. If the submitted evidence of the assignment is inaccurate, insufficient, plagiarised or not convincing enough, a selective viva would be held for the respective groups.

[Tempus fugit carpe diem]