National College of Ireland

National College of Ireland

BSC (Honours) in Computing - Full-time - Year 4

Repeat Terminal Based Assessment Assignment – Jan 2023

Completed by 06 January 2023

Distributed Systems

Dr. Athanasios Staikopoulos

This is an online repeat/deferral Terminal Based Assessment worth 100% of the final grade.

The online assessment must be electronically submitted using the automatic features provided in the module page in Moodle.

Should any student miss the assessment deadline with a valid reason, the student can apply for an extension/rerun using the Extension/Re-run Form online, via NCI360.

The assessment consists of 2 parts (project and questions) as follows:

Section A – Project Overview (76%)

This project is to be done individually.

Aims

The purpose of this project is to gain experience in designing and building distributed systems. You will be expected to demonstrate that your solution meets the objectives set out in this document.

The Problem

For the purpose of this assignment, you assume that you lead a Software Development Consultancy company which has been commissioned to create an application on the specific industry based on your **last digit of your student Id**. In particular, from the following list and based on the last digit of your student id please select the corresponding industry.

For example: If your student ID were x17068381 then your industry would be Smart Buildings (last digit is "one").

Last student digit	Domain (Industry)
0 or 5	Smart City
1 or 6	Smart Living
2 or 7	Smart Health/medical environment
3 or 8	Smart/Automated Business Operations
4 or 9	Smart Buildings

As part of your task you have to develop a set of protocols/messages and build a reference implementation that simulates the operations of a **smart automated environment** (for example hospital, building, city). As a result, your environment could consist of smart services and devices that inter-communicate with each other.

Your devices/services must **publish** themselves and **discover** each other. Your devices/services should communicate via **gRPC**. Two programming languages must be used to develop your solution. One of the languages should be Java.

You should begin by devising your own scenario. There must be a minimum of 3 separate services that would simulate the operations of smart-automated environment. It is key to specify what operations are supported on each "service/device" in a corresponding proto file.

Finally, to demonstrate your implementation a simple client GUI should be developed, operating as a main controller that discovers and uses your **devices/services.**

Deliverables

As part of the project submission, you are **required** to submit the following **three parts**

Report

A short report which details the scenario and services you have chosen. Additionally, this should specify the message formats for data exchange and service actuation.
The report must have all the headings of the marking scheme (see below).

Program Code

A project, or more than one, with all code, well commented. Code must also be available in a private **GitHub** repository, the repo must have a commit history, not a last-minute code dump.

Video Presentation

A recorded video presentation is REQUIRED, that will be used to demonstrate and explain the different parts of your application. The presentations should not exceed 10 minutes in duration.

Marking Scheme

- Report (**5%**) – problem description, solution, writing

- Service Definitions - Use of gRPC (**12%**)
  - For each of the 3 different services a corresponding proto file is defined and used
  - All 3 different types of RPC invocation styles have been used (simple RPC, server-side streaming RPC, client-side streaming RPC, bidirectional streaming RPC)
  - Appropriate and different message structures and field types are used

- Service Implementation - 3 sufficiently complex services are implemented (**26%**)
  - One service must be written in a language other than Java. [8%]
  - The other two services should be written in Java. [2 * 6 = 12%]
  - Service complexity and implementation [3 * 2 = 6%]

- Naming Services - Use of jmDNS (**12**%)
  - Marks for service registration and discovery [3 * 4 = 12%]

- Remote Error Handling (**5%**)
  - appropriate error handling for remote invocations and error messaging
  - user input validation

- Client - Graphical User Interface (GUI) (**6%**)
  - That allows to view (e.g., present, discover), control (parameters) and invoke the services/devices
  - The GUI can be developed in any language, technology of choice (Java application, web based, etc)

- GitHub (**4%**)
  - Maintain a repo with a regular commit history.

- Video Presentation (**6%**) – **Required**
  - demonstrate and explain different/core parts of the application. Provide video or link of your presentation

Section B – Related Questions (24%)

From this section, please implement or answer **any 2 of the following questions**

Q1: Explain the MQTT broker architecture with an example.

[12 marks]

Q2: Explain and illustrate the sequence of events during a remote procedure call?

[12 marks]

Q3: How to choose between CP (consistency) and AP (availability)? Discuss the criteria/ requirements for different types of applications.

[12 marks]