COMP260 – Distributed programming Worksheet 3
Making robust client server applications that supports multiple clients

Introduction

The goal of this worksheet is to develop robust networking approaches that can be used for your assignment based off the network-based SUD that you developed in last weeks' worksheet.

This session will also be used for assessing part A of the first assignment, the UML design assessment.

Explore the code samples from the lecture

I have placed code samples for exception handling sockets (client & server), threaded clients and servers and a PyQt5 example. Load up these projects and have a look to see how they work. However, do note that the threaded examples do not include much exception handling – that is an exercise for the student, as is building a bi-directional send/receive framework for the client/server architecture.

Network SUD to MUD part 1: robust network components

In last weeks' workshop, you developed version of your SUD that would work as a client server model. This task is to take that client server code and build robustness into it using socket exception handling for the critical client and server issues that were highlighted in this week's lecture. On successful completion of this task, you should be able demonstrate several of the use cases described in assignment 1.

Network SUD to MUD part 2: multiple users

Building further on your existing network SUD functionality, you can refactor your client and server applications to support multiple users with appropriate threading and thread safe containers. Again, the architecture and code approaches to implement this was presented in this weeks' lecture and your challenge is to use the appropriate approaches and containers.

Experimenting with PyQt

PyQt will provide a non-blocking client interface for your MUD. A good starting point with this is to experiment with PyQt functionality to get an insight into its operations and limitations. https://github.com/mfitzp/15-minute-apps contains a collection of simple applications that use Qt as a UI.

Taking it further

1. Last week's workshop outlined making a client/server noughts and crosses game. If you were unable to do this last week, now would be an ideal time to implement it. Start by thinking about the underlying model, view and controller for noughts and crosses and how these can be described in UML using class hierarchies, state diagrams, flowcharts and message flows. Then build a solution.

   If you managed to develop the client/server noughts and crosses game from the previous tutorial, this would be an excellent application to Qt-ify as well as making it a robust network application.

2. If you managed to get anywhere with the chat service, you will have most likely discovered that Python's standard output blocking made it very difficult to use chat. PyQt gives you an ideal interface to address these problems.