**Requirements**

* We need to build a client server system
* The client and server will run in separate machines
* The client and server can use any pub/sub models implementing DDS to communicate
* The server has a database of Person Info containing Person ID and corresponding Age
* The client can send request to the Server asking for the age of a particular person ID
* The server should then reply the client with the age and ID of the person
* **Good to have features**
  + There may be multiple clients connecting with the same server
  + The server should be scalable and highly responsive

**Implementation Details**

**Dependencies:**

I have used boost Asio library for the async communication between the client and the server:

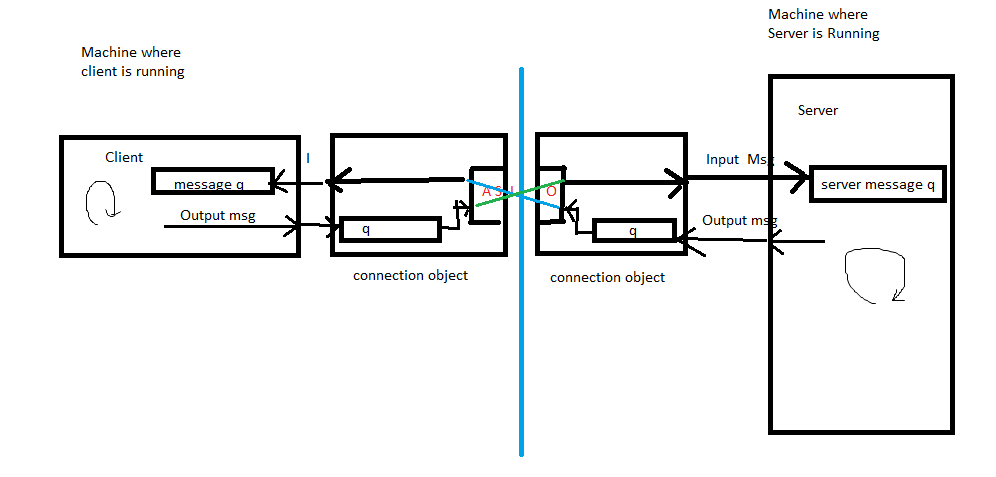
https://think-async.com/Asio/

**Project Structure:**

|  |  |
| --- | --- |
| **Dir** | **Comments** |
|  | The directory CommLayer hides the implementation of the communication layer. The communication layer is a light weight framework built on top of boost Asio C++ library : https://think-async.com/Asio/ |
|  | This folder contains the implementation of a Simple Client that sends age related requests to Server on top of the communication layer. |
|  | This folder contains the implementation of a Simple Server that accepts client connections and processes the query of the clients. The query here is related to age and if the person ID is present in Server’s DB, it sends a message back to the client containing age information |

**Project Architecture:**

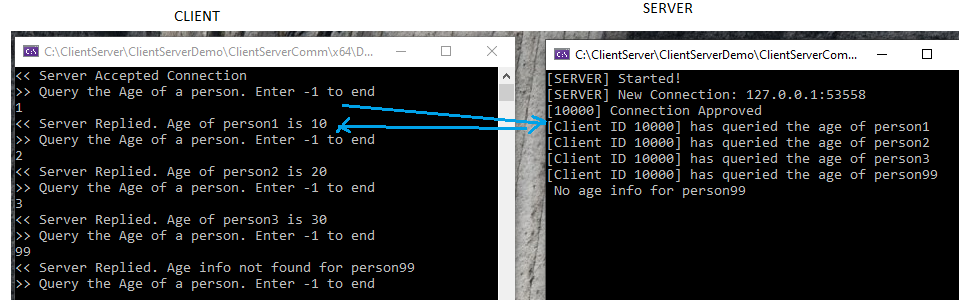
**High Level Architecture of the Communication flow**



The asio library handles the low-level details of networking. A light communication framework based on asio is built which gives the client and server APIs to connect and send messages to each other. The framework is similar to a lightweight pub sub model where messages are pushed to common thread safe queue. Clients and servers send and consume messages from this common thread safe message queue.

**Demo**

To demonstrate the communication between client and server, I have run server and client on local host and port :60000. The Server starts first and waits for any client connection. When a client is run, it attempts to establish connection with the server on the local host running at port 60000. Once the server approves the communication, the system is ready for the exchange of messages. Here is a screenshot of how client queries for the age and the server replies with the age:



Class Hierarchy

Important classes in Connection Layer:

|  |  |  |
| --- | --- | --- |
| Class | File | Responsibility |
| connection | CommLayer/comm\_conn.h | Both client and server rely on connection object to communicate. connection class exposes API to connect to server, connect to client and send messages. A connection contains its own queue to store messages as well as reference to the client or server’s queue as shown in the architecture diagram. The reference to the queue is provided when a client or server creates a connection object. The connection class hides the low level sockets and asio details |
| client\_interface | CommLayer/comm\_client.h | Its an interface for talking to the server. It contains its own queue to store incoming messages from the server. client\_interface uses connection object to connect to the server.It provides an API called connect which accepts a url and port num to connect to the server.  Other API is disconnect. It encapsulates low level asio related details for connection. Ideally a custom client should extend this client\_interface to write only the business logic |
| server\_interface | CommLayer/comm\_server.h | Server has the responsibility to listen to client connections and serve the client requests. Each time a client is connected to a server, a connection is created which is responsible for message exchange. The server\_interface encapsulates all the networking related details and asio and has virtual methods which should be overridden by server classes that extends the server\_interface. . The server\_interface provides various virtual methods like OnMessage/OnClientConnect which can be implemented by the child classes to perform application logic |
|  |  |  |

Application Layer:

|  |  |  |
| --- | --- | --- |
| Class | File | Responsibilty |
| CustomClient | SimpleClient/SimpleClient.cpp | The SimpleClient class is a simple client that extends the client\_interface in the connection layer. All the networking related details are already handled in the base client\_interface class. Hence the simple client can focus on the business logic. This client connects to the server and queries the server for the age of persons |
| AgeInfoServer | SimpleServer/SimpleServer.cpp | The AgeInfoServer is a thin server extending the server\_interface. The AgeInfoServer does not have to deal with the networking details as its handles by the base server\_interface. The AgeInfoServer has a database of person ID and Age information. It waits for client connection and sends the age information to the client |

Further Enhancements

* Adding Unit Tests
* Making the connection layer more open closed so that it can use any DDS libraries
* Encapsulate the Database in Server side and delegate it to DataAccess Layer