**C++ Server / Client program exercise**

Please write a C++ Server / Client program according to the following:

**Program logic**

* The program should use the following sub-directories:
  + Incoming (this directory is used to store the message that was received)
  + Outgoing (this directory is used to store outgoing messages before sending them out)
* The program should be able to process several incoming and outgoing messages simultaneously
* The program should “listen” to port 80 for incoming messages (should be a global parameter)
* Outgoing messages should have an arbitrary source port
* Accepting incoming message
  + The program should fill the *TimeArrival* parameter for each received message
  + The Message details should be printed to console
  + The received message should be written as an XML file to Incoming sub-directory
* Sending outgoing messages
  + The program should scan the Outgoing directory for outgoing messages every 5 seconds
  + Outgoing messages should be read from the Outgoing directory in XML format
  + The message should be sent according to the message *ServerIp* parameter
  + The program should fill the message *ClientIp* parameter before it sends the message
  + Once the message was sent it should be deleted from the outgoing directory
* Message
  + Each message contains the following basic fields:
    - *ID* - the message ID
    - *TimeArrival* - the time that the message arrived to its destination
    - *ClientIp* - the client IP
    - *ServerIp* - the server IP
    - *Type* – the message type (can be either "request" or "response")
  + Request message
    - Each request message containing the following fields:
      * *RequestedDataX* - the data requested from the server (X is a number of request parameter)
    - The request message can contain more than one "*RequestedData*" parameter
    - The *RequestedData* parameter can contain either "GetServerIP" or "GetServerName"
    - Each request message should be followed with response message
  + Response message
    - Each response message containing the following fields:
      * *RequestMessageId* - the ID of the original request message
      * *RequestedDataX* - the data info from the server(X is a number of request parameter)
    - The response message can contain more than one "*RequestedData*" parameter
    - The *RequestedData* parameter can contain either the server IP or server hostname

**XML Message examples:**

<message>  
 <Id>1</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.1</ClientIp>  
 <ServerIp>1.1.1.2</ServerIp>  
 <Type>Request</Type>  
 <RequestedData1>GetServerIP</RequestedData1>  
</message>

<message>  
 <Id>2</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.1</ClientIp>  
 <ServerIp>1.1.1.2</ServerIp>  
 <Type>Request</Type>  
 <RequestedData1>GetServerName</RequestedData1>  
</message>

<message>  
 <Id>3</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.1</ClientIp>  
 <ServerIp>1.1.1.2</ServerIp>  
 <Type>Request</Type>  
 <RequestedData1>GetServerIP</RequestedData1>  
 <RequestedData2>GetServerName</RequestedData2>  
</message>

<message>  
 <Id>4</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.2</ClientIp>  
 <ServerIp>1.1.1.1</ServerIp>  
 <Type>Response</Type>  
 <RequestMessageId>1</RequestMessageId>  
 <RequestedData1>1.1.1.2</RequestedData1>  
</message>

<message>  
 <Id>5</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.2</ClientIp>  
 <ServerIp>1.1.1.1</ServerIp>  
 <Type>Response</Type>  
 <RequestMessageId>2</RequestMessageId>  
 <RequestedData2>Test\_machine\_2</RequestedData2>  
</message>

<message>  
 <Id>6</Id>  
 <TimeArrival></TimeArrival>  
 <ClientIp>1.1.1.2</ClientIp>  
 <ServerIp>1.1.1.1</ServerIp>  
 <Type>Response</Type>  
 <RequestMessageId>3</RequestMessageId>  
 <RequestedData1>1.1.1.2</RequestedData1>  
 <RequestedData2>Test\_machine\_2</RequestedData2>  
</message>

**Design and code requirements:**

* Object Oriented Design
  + The code should be written in an Object Oriented way with structure of simple objects containing their logic.
* Multi-threaded system
  + The program should use different threads for incoming and outgoing messages.
  + A thread pool of 2 threads should be created for incoming messages, if more than 2 messages will arrive at the same time the program should dynamically allocate more threads.
  + A thread pool of 2 threads should be created for outgoing messages, if more than 2 messages should be sent at the same time the program should dynamically allocate more threads.
  + All threads should be controlled from one main class
  + The threads for incoming messages should have higher priority than the threads of the outgoing messages
  + Anti-Starvation mechanism should be implemented so a bulk of incoming messages will not prevent outgoing messages from been sent.
* Queue base network implementation
  + A FIFO queue should be created to manage the incoming messages
  + A FIFO queue should be created to manage the outgoing messages
* Documentation
  + Class documentation
    - At the beginning of each class should be a short description of this class and its purpose
  + Function & procedures documentation
    - At the beginning of each function should be
      * Description of this function and its major steps
      * Description of each parameter it gets and its possible values
      * Description of each parameter it returns and its possible values
      * Description of the exception it throws and in what cases
    - Inside the function
      * A one line description before each major steps described at the beginning of the function
      * A one line description before complex code / calculation
  + Global parameters documentation
    - Before each global parameter should be a one line description describe the parameter and its use.
* Exception handling
  + The program should not crash when exception is thrown, every exception should be caught and be written to a log file.