**Project #3 – Socket-based Communication** due Thursday, April 9 Version #1.1

Purpose:

This project requires you to build a socket-based communication system for sending files and messages between network machines. You will demonstrate this capability with a peer-to-peer system that can send, possibly binary, files and text messages from either end of the communication channel.

Requirements:

Your PeerToPeer communication system:

1. **shall** use standard C++, the standard library, compile and link from the command line, using Visual Studio 2012, as provided in the ECS clusters and operate in the environment provided there[[1]](#footnote-1).
2. **shall** use services of the C++ std::iostream library and Win32 sockets for all input and output in the communication channel. Peers **shall** use the C++ operator new and delete for all dynamic memory management.
3. **shall** support posting text messages to a send queue and getting message from the corresponding receive queue.
4. shall support sending, possibly binary, files across the communication channel in chunks. It is suggested that you send a start text message indicating the beginning of a file transfer, then send blocks with text headers[[2]](#footnote-2) and binary bodies. After the last file block is sent you send an end text message indicating end of file transfer.
5. **shall** demonstrate the communication system by sending both text and file messages from multiple concurrent clients to a single receiver. The receiver **shall** send back an acknowledgement message[[3]](#footnote-3).
6. **shall** support asynchronous communication. That means that a client sends a message and breaks the connection.

If the receiver wishes to send a reply it later connects to the client to send the reply[[4]](#footnote-4).

1. **shall** demonstrate asynchronous processing by starting at least two clients that each send to all of at least two running receivers. The clients each request receivers to process a computational task and break the connection. Receivers must reply to the requestor with the computational results.
2. **shall** provide a reusable channel package. Reusable means that no changes are required to support communication in Project #4., e.g., it provides drop-in support for interprocess and remote messaging.
3. **shall** be submitted containing all packages from your Repository client and server, along with corresponding metadata.
4. Your project submission **shall** be uploaded in a zip file archive, including batch files named compile.bat and run.bat that compile and run your project, to demonstrate you meet all requirements, using appropriate command line arguments. Please also include a Visual Studio solution that demonstrates you meet these requirements.

Set arguments as a storing folder for receiver. Yes

Check arguments. Port limited, filepath, IP address.

How to build the communication lib first, and then rebuild the following projects.

1. VC++ version 11.0 is provided by Visual Studio .Net 2012, and is available in all the ECS clusters. [↑](#footnote-ref-1)
2. The text header should contain a line with a byte count for the body, as the last block may not be full. [↑](#footnote-ref-2)
3. This means that receivers must spawn a thread for each client and maintain a connection on which to reply. [↑](#footnote-ref-3)
4. This means that senders must encode their end point information, e.g., ip address and port number, in the sent message header. [↑](#footnote-ref-4)