**Producer/Consumer Application**

Author: Matt Ferrante

Date: September 28 2018

The modules were compiled and tested on CentOS 6.9, using the RedHat developer toolset to make use of gcc 4.8 and C++11 features. They make use of the standard library and also depend on the Qt5 framework.

**Build Instructions:**

In top-level 'bsci' directory:

> qmake && make

**Run Instructions**

Module A:

> cd ModuleA

> ./ModuleA

Module B:

> cd ModuleB

> ./ModuleB

Multiple instances of the ModuleB executable can be run concurrently, in separate terminals.

In the unlikely event that the modules fail to compile, there are backup compiled binaries in the bin/ subdirectory.

**A. Data Types**

Both data types are shown here for convenience. Each consists of a fixed-size std::array and a double-precision timestamp.

**A.I PacketT**

typedef std::array<char, 36> PacketDataT;

struct PacketT {

PacketDataT data; // header + 8 channels

TimeStamp timestamp; // in seconds

}

**A.II ChecksumT**

typedef std::array<unsigned int, 8> ChecksumDataT;

struct ChecksumT {

ChecksumDataT sum;

TimeStamp timestamp;

}

**B. Common Classes**

**B.I Config**

The optimal value for many parameters may vary depending on system resources, for example, maximum allowed connections to the Server, maximum output file size for the FileWriter, and TCP packet size for the TcpWriter. The user may also like to change certain defaults such as the FileWriter's output directory, or whether to write to stdout. The Config class is a singleton class made available to all classes in both modules, allowing the user to configure various options in the application at runtime and have the various components lookup the parameters they care about.

**B.II ThreadSafeQueue**

The ThreadSafeQueue is a template class which manages a queue resource known to be shared by multiple threads. It is used by almost every class in this application.

**C. “Producer” – Module A**

**C.I FileReader**

The FileReader class is designed to be run in its own thread, performing the sole function of reading in packets from a binary file one at a time at a constant rate, and pushing the packets to a ThreadSafeQueue, shared by the FrameParser.

**C.II FrameParser**

The FrameParser class pulls PacketT's off the queue being pushed to by FileReader, process them into ChecksumT's, and emits them to be read by anyone accepting the signal. Like the FileReader, it is designed to be run in its own thread. Its job is to wait for data to become available on the queue of PacketTs, and then process the packets. It verifies that the packets' header numbers are sequential, gathers the packets into frames, computes the checksum for each channel, and packs it all into an outgoing ChecksumT.

**C.III TcpWriter**

The TcpWriter class manages the server-side sockets, and writes to them. It waits for ChecksumT's to be ready and pushes them to a queue when they become available. It then waits for enough Checksums to send across the TCP connection, according to a desired TCP packet size, which can be specified by a configurable parameter or set to a default value of 1500 bytes.

**C.IV Server**

The Server class inherits QTcpServer, the Qt framework's built-in TCP Server class. It is implemented as a singleton class, designed to be run in the main application thread. It handles incoming TCP connections initiated on the client side, signaling the Producer application to spawn a new TcpWriter and Qthread to host the writer.

**C.V Producer**

The Producer class doubles as the primary resource manager and the main application window of Module A. In its initialzation routine, it commands the Server to start listening for new connections, sets up two ThreadSafeQueues (one of PacketTs and one of ChecksumTs), and spawns separate threads for FileReader and FrameParser to do their work. It is triggered by a signal from the Server each time a new connection arrives, to spawn a new thread to host a TcpWriter which will communicate with the new client. Since it manages the list of TcpWriters, it also forwards outgoing ChecksumT's to each TcpWriter when they arrive from the FrameParser.

**D. “Consumer” - Module B**

**D.I FileWriter**

The FileWriter waits for QByteArrays to become available on a shared ThreadSafeQueue, and writes them out to an output stream, which can be set to either a standard output file stream or stdout. If stdout is used, the output will be ASCII, but if a filestream is used, the output file will be binary, with each ChecksumT written as a double-precision timestamp followed by eight 4-byte checksums, one per channel.

**D.II Consumer**

Like the Producer, the Consumer class is both the resource manager and the main window of the application. It attempts to open a TCP connection to the server and manages the TcpSocket resource. If successful in opening a connection, it waits for data to become available on its socket, then pushes it to a shared ThreadSafeQueue. It also contains a FileWriter as a member, which runs on a separate thread.