CPSC441 Assignment 4 Design Document (Brett Gattinger, 30009390)

I implemented retransmissions using the Timer object and TimerTask thread object. I setup the Timer object (called timeout_timer) and the TimerTask thread it ran (called TimeoutRestransmitPacketTask) within the main while loop inside the transferFile() function. The while loop would read from the input stream of the file and create an Datagram to be sent via UDP to the server, each time this happened a Timer object and TimerTask thread object would also be instantiated. The TimerTask thread object would be instantiated with all the data necessary to create a complete copy of the Datagram that was currently being transmitted by the main thread, along with a thread-safe reference to the UDP socket to the server. Once the main thread transmitted the current packet out of the UDP socket, the Timer object would schedule the TimerTask thread to continually retransmit it's copy of the packet each time the timeout period had elapsed until the Timer object was cancelled. This cancellation would only occur once an appropriate Ack was received back from the server on the main thread. If the sequence number of the Ack was not appropriate, the main thread would resume listening on the UDP socket until a proper Ack was received.

If a timeout occurred at the same time an ack was received from the server then a copy of the last packet to be sent to the server, before the Ack, would be retransmitted by the TimerTask thread to the server. Because this retransmitted packet would be its own copy created by the TimerTask thread at its creation BEFORE the main thread would go to listen for Acks, the TimerTask thread wouldn't be accessing any shared data at the same time as main thread. And since UDP is thread safe in that it guarantees the integrity of Datagram packets, one can reasonably conclude then that no race conditions occur in this scenario.