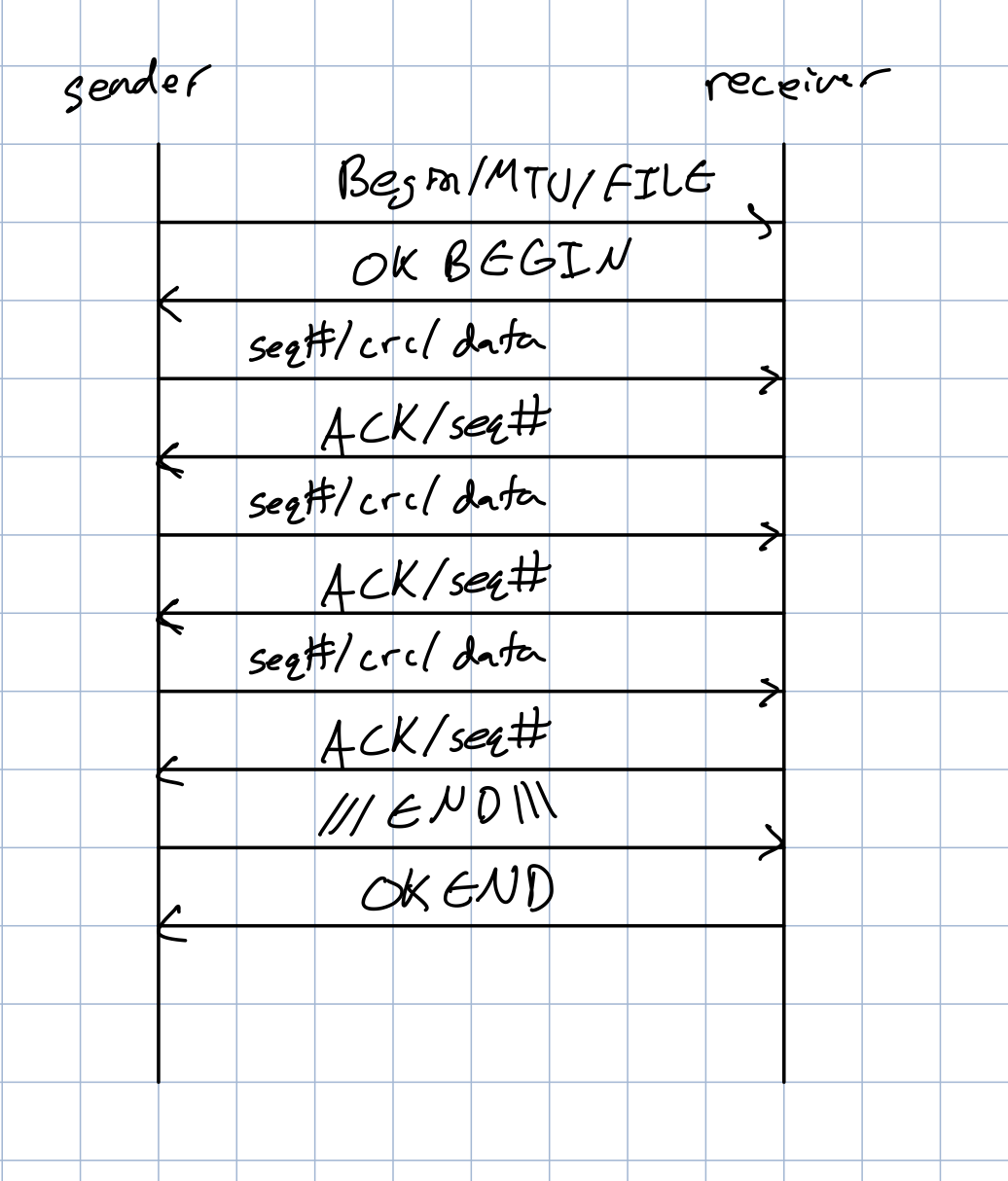
Malcolm Sturgis

Reliable Data Transfer

**Protocol:**

My protocol is built similarly to stop and wait but with some improvements. A big speed improvement is dynamic timeouts. A moving average over the last 5 rtt’s is used and multiplied by 4 for the timeout. This allows for timeouts to happen very rarely while massively decreasing the stopping time. For error prevention, checksums were implemented to prevent bit errors from corrupting the data transfer. Additionally, sequence numbers were used to ensure all data comes sequentially.

This being said, the protocol is not fully robust. If enough packet loss happens to exceed the maximum retries (8) then the protocol will die. Additionally, I have seen some bit errors which cause python’s built-in .decode(‘utf-8’) to error, which halts the transfer. The protocol is able to send consistently on all network qualities excluding terrible. This is due to bit errors messing with the packet interpretations or excessive drops/dupes/ooo/ber causing the retry limit to be hit.



**Figure 1.** Diagram of the transfer protocol.

**Table I.** Transfer time at a variety of connection qualities.

| **Connection Quality** | **Transfer Time** |
| --- | --- |
| Ideal | 19.3 seconds |
| Good | 20.7 seconds |
| Medium | 22.4 seconds |
| Terrible | 32.5 seconds |

**Running:**

No modifications were made to the send, receive, and udp\_box files except for adding in the ability to select my protocol (tagged ‘custom’). It will default to my protocol so no tagging required.