This protocol is implemented for transferring data between 2 Nodes(a client and a server) over UDP. In this case the client is BaseStation class and the server is Lander class. Initially the client has to make a request to server using some commands to which the server responds with appropriate data packets. For the explanation of protocol we will focus on the get command sent from client to server.

Protocol explanation:

The client will send a get command packet to the server. The get command packet will have the following attributes.

**Command**: True: if the packet is generated using some command like ls or get.

False: if the packet is generated only for acknowledgement of the data received.

**Packet Number Expected**: This will have the sequence number of the packet which the client expects.

**Expected File**: This will have the name of the file which is expected from server.

**Receive Window**: This will have the number of packets the client can receive without any trouble.

To this get command the server will send a data response. The data response is sent using a data packet. The Data Packet has following attributes. ***(Note: The server can only send data packets to client and client can only send the command packets to server.)***

**Data**: The byte array containing actual data.

**Packet Number**: The number of the packet being sent.

**Fin Flag**: The flag to indicate the last packet in the data transfer so that the client will not expect any more packets from the server.

The packet drops due to any reason are handles by the client. If the client does not receive an expected chunk of data packets from the server it will send the acknowledgement again to the server with 1/2 of the previous window size. The client will wait again to get the desired data packets from the server. The waiting time at the client side to retransmit the ack packet is 3 seconds.

The protocol uses the checksum provided by UDP to check whether the data received was correct or not.

The client will keep on excepting the data packets from server until a packet with fin **flag = true** does not arrive. After the client receives the last packet it simply terminates i.e. it does not acknowledges the last packet as it is no longer expecting anything from the server. In this scenario the client will send a collective acknowledgement to the server.

Reasons for this protocol being faster than TCP.

* No three-way handshake required. In this case the data is coming from lander to the Base Station, thus two way handshake can be sufficient. In addition to that in two way handshake the Server can directly start sending the packets in the response of the 1st packet from client.
* Use of collective acknowledgements. The client does not need to send ack for each data packet.
* After each successful data transfer the client doubles the receiving window and if there is a packet loss then it reduces the window to half. Note: Here window is specified in terms of number of packets and not number of bytes like in TCP as the MTU is assumed to be fixed.