dpinit: Creates DPSession structure and reserves memory. Sets attributes of the structure. Returns the created structure.

dpclose: Closes the DPsession and free the memory allocated for it.

dpmaxdgram: Returns the maximum buffer size for the DP

dpServerInit: Initializes the DP server and returns NULL if it failed. Creates the socket fd. Sets server information. Sets socket options. Returns the DP Connection structure.

dpClientInit: Initializes the DP client. Creates the socket FD. Fills in the server information and sets the inbound address to the same thing as the outbound address and returns the DP client structure.

dprecv: Retrieves receive length using the dp struct. Returns if the length DP_CONNECTION_CLOSED if the connection was closed. If the received length is greater than the size of the pdu, it copies the dp buffer into the buff variable. Returns the dgram size.

dprecvdgram: Returns if the buffer is greater than the max size. Receives the raw bytes. Sets the error code if the number of bytes in are less than the size of the pdu. Copies the buffer into a pdu struct. Checks if there is no error code and then increments the sequence number. Creates the outbound pdu. Checks if there is an error code and sends the outpdu and returns if the actual size sent does not equal the pdu size. Checks the inbound pdu mtype and sets the outpdu mtype accordingly and sends the outpdu. Returns an error if the actual send size is less thant he pdu size. If the mtype is DP_MT_CLOSE it also closes the DP. Function returns the bytes in.

dprecvraw: Throws an error if the address isnt initialized. Receives bytes from the dp udp socket. Throws an error if the bytes received are less than 0. Sets the outSockAddr.isAddrInit to true. Checks if the bytes are greater than the size of the pdu and does nothing. Creates an inPdu from the buffer and prints it. Returns the bytes received.

dpsend: Returns DP_BUFF_UNDERSIZED if the send buffer is greater than the max. Sends the dgram and returns the sent size.

dpsenddgram: Throws an error if the outSockAddr isnt initialized. Returns DP_ERROR_GENERAL if the send buffer size is grater than the MAX buff size. Build the outPdu . Sets builds the DP Buffer using the send buffer. Calculates the total send size and sends the data. Prints a warning if the sent byte size is not equal to the total send size. Increments the outpdu sequence number. Return the amount of bytes sent minus the size of the dp pdu.

dpsendraw: Throws an erro fi the outSockAddr is not initialized. Creates the outPdu using the send buffer. Sends the bytes using the udp socket and and prints the outPdu. Returns the amount of bytes sent.

dplisten: Throws an error if the inSockAddr is not initialized. Creates a pdu struct. Receives raw data from the dp. Throws an error if the size of the data received is not equal to the size of the pdu. SEts the pdu mtype. Increments the sequence number. Calls dpsendraw. Throws an error if the size of the data sent does not equal the pdu. Sets the dp isConnected to true. returns True.

dpconnect: Throws an error if the outSockAddr is not initialized. Builds a dp pdu struct. Sends the pdu and throws an error if the size sent does not equal the size of the pdu. Receives raw and throws an error if the size

sent does not equal the size of the dp pdu. Throws an error if the pdu size does not equal DP_MT_CNTACK. Increments the sequence number and sets dp isConnected to true. Returns true.

dpdisconnect: Builds a pdu. Calls dpsendraw and throws an error if the send size does not equal the size of the pdu. Calls dprecvraw and throws an error if the received size does not equalk the dp pdu size. Throws an error if the pdu mtype does not equal DP_MT_CLOSEACK. Closes the dp. Returns DP_CONNECTION_CLOSED.

dp_prepare_send: Throws an error if the buffer size is less than the size of a dp pdu. Creates an empty buffer and copies the pdu to it. Returns the address after the pdu.

print_out_pdu: Prints the pdu details.

print_in_pdu: Prints the pdu details.

print_pdu_details: Prints the pdu details.

pdu_msg_to_string: Converts the PDU mtype to string.

2. The advantage of the 3 sub-layeres is that it allows for modularity. High level layer is used to check the buffer size and call the intermediate level layer function. The intermediate level layer function is used to build the pdu and dp buffer and the lower level function is used to actually send and receive the raw data. An improvement could be adding more error handling between the layers.

3. Sequence numbers are used to track the order of messages exchanged.

4. A limitation of this is that there is a potential for increased latency and less throughput compared to TCP. It allowed implementation to be simpler because it avoided the complexities of retransmission timers.

5. TCP sockets require a connection to be established before data can be exchanged. UDP does not have this requirement. TCP maintains a connection state that both ends are aware of. UDP is stateless and does not maintain a connection state.