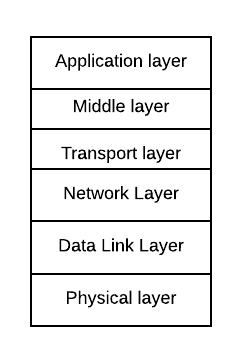
**RDT PROTOCOL IN APPLICATION LAYER**

**SPECIFICATION**

In this protocol, whatever data is to be sent is sent through a Middle layer which is also at the application level which ensures reliability of the data to be sent but using unreliable UDP below its layer. All the steps needed to make sure the receiver receives the data will be taken care of by this additional layer which is present between the application which needs to send data and the transport layer UDP.

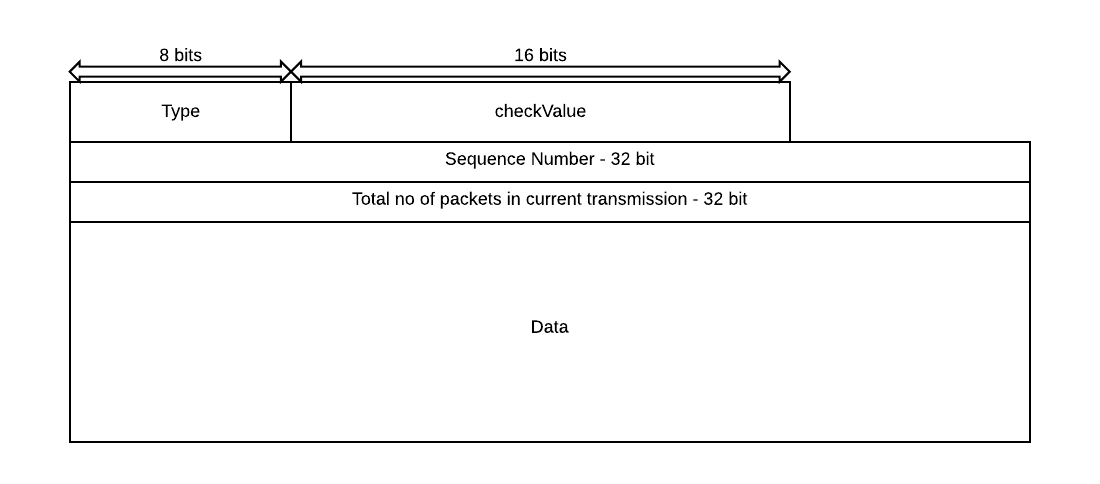
As soon as the Middle layer receives data from any application, it will be dividing the data to be sent into packets containing a specific format with custom headers included in it. The headers will contain the fields like the type of packet(sending data or ack), sequence number, check-valve and the total number of packets in the current transmission. We are not having any header length here because we are not having any variable parts in the header. The total length of the header is 88 bits as shown below in the diagram.



**PACKET STRUCTURE:**

The value in the ‘Type’ field will be a character ‘0’ for data and ‘1’ for ACK. So, its size is 8 bits or 1 byte. The check-value is the value obtained by hashing the data and other headers of the packet. Sequence numbers are integers from 0 to **232**. The ‘Total number of packets’ field is useful for the receiver to make up the original data which was fragmented.

The size of each packet is at most 65,535 bytes because that is the maximum size a UDP packet can carry. Anything more than that will cause the transport layer protocol UDP to fragment that packet. If any of the fragmented packets is lost, then other fragmented packets of the same original packet would be useless even if they reached the destination.



**SEQ AND ACK NUMBERS:**

Before sending the data, all the sequence numbers used are stored at the sender side. The timer is started for each packet sent. The time set is RTT + (some buffer time). All the packets are sent(in a sequence from first to the last packet) using UDP. The receiver will send the ACK for the received packets. It also contains the same custom headers as mentioned above. However, the data part will be the sequence number of the packet receiver wanted to ACK.

The sender then marks the sequence numbers of the received ACKs. After the timeout, if there are any remaining packets to be ACKed, then the sender will send all those remaining packets once again assuming that they are lost previously or corrupted. Again the timer is set. This is repeated until all the packets get ACKed. But if all ACKs are received before the timeout, then the sender will stop the timer and wait for the next transmission.

Furthermore, until the complete data of the current transmission is transferred to the receiver, the sender would not send other data.

**RETRANSMISSION:**

If ack is not received after some time (RTT + buffer time), the sender will retransmit the packet. After a certain retransmission limit, the sender is informed that the data could not be sent.

**RTT ESTIMATION:**

RTT value is initially calculated by sending a packet to the receiver and measuring time to receive ack. After this RTT is estimated using the following formula.

**EstimatedRTT = (1-𝝰)\*EstimatedRTT + 𝝰\*SampleRTT**

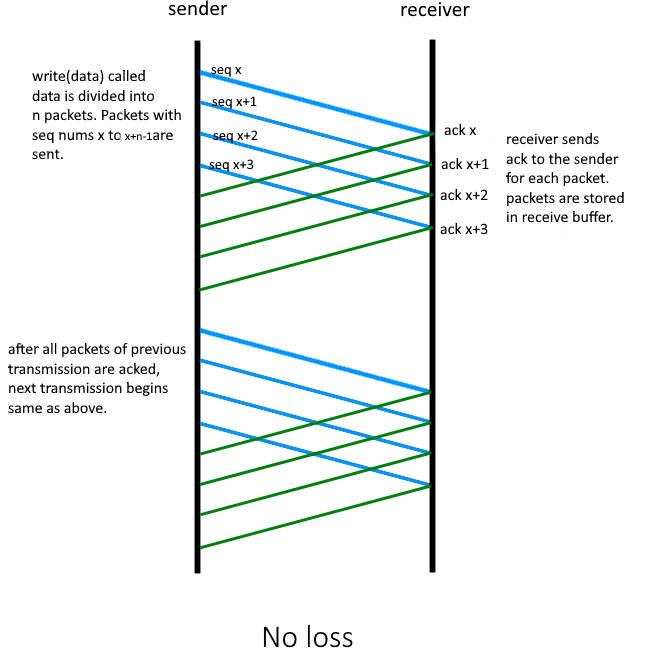
SampleRTT is measured for between every packet sent and ack received.

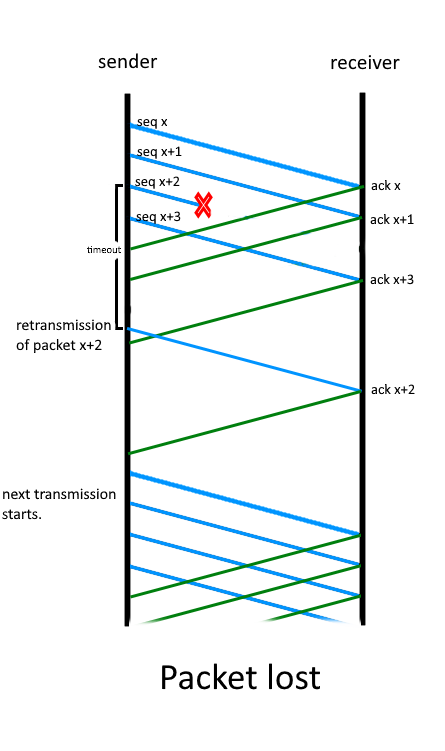
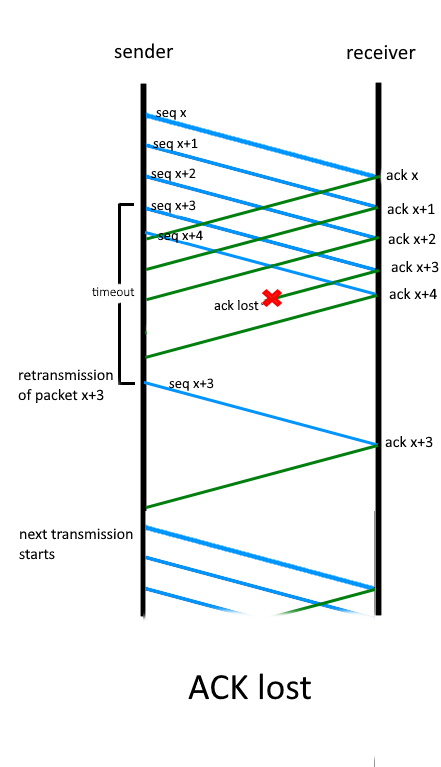
**RECEIVER:**

In the case of the receiver, it will store all the received packets in a buffer until the number of packets received is equal to the value in the ‘Total no of packets in current transmission’ field in any of the received packets. After receiving a packet, the receiver will verify the check-value field. If it is not matching with the value in the ‘check-value’ field, then it simply discards the packet. For all the correct and in-sequence packets’ headers are removed and the data part will be joined. For all the correct and out of order packets, they are stored in the buffer in its place until its turn comes to be unpacked. For all correctly received packets, an ACK will be sent to the sender. The value of ‘Type’ field will be ‘1’, the sequence number will be the next unused sequence number from the receiver side, the ‘Total no of packets’ field will be some arbitrary value(because it is not needed), the ‘check-value’ field contains the hashed value of the data field with other headers and the sequence number of the received packet will be present in the data field.

For the subsequent transmissions, the sequence numbers which are unused previously are used. After all the **232** sequence numbers are used, then we again start from 0 and continue till **232** and repeat the process. We use this procedure because there may be some packets which may take a very long time to reach the destination which will then cause confusion for the receiver if a packet of subsequent transmissions has the same sequence number.

**SOME EXAMPLE SITUATIONS:**

****

****