Sender process(rt_srv):
Before anything starts, sender should build connection will receiver, once it gets request from receiver, it will reply permission, and Take Receive Time1, We have the sendTs, Basically we could initial Base Delta.

1.Get data from local app, calculate the deliveryTime then store it into the window and send it to the Receiver. (deliveryTime = sendTS+ baseDelta + lantency Window)

2.Check whether we should shift the window.
If (packet delivery time < present time+ base_Delta) then we shift otherwise, do nothing.

3.Receive the ACK packet from receiver and update the Base_Delta, RTT.
Send the ACKACK packet And check whether we have NACK and resend packet if its(delivery time > present time + base_Delta)
When resending the Nack, we replace sendTS with present time,but keep the same delivery time.

4. If Sender receive the request, which is message type 3,it will send decline.

Receiver process(rt_rcv):
Before anything starts, receiver should keep sending a request to sender until it gets reply.
If it gets the permission, then we go to main body part. Else, it will exit.
In this process, we could initial our RTT and BaseDelta.
RTT = now – sendTs
BaseDelta = Receive Time1 – sendTs.

The receiver has three main responsibilities.
The first responsibility is Receiving the packet from Sender.
There could be two types of packet.

1. Data Packet: Check whether we already had the packet. If not, check whether the delivery time is not expired. If not write it into our buffer.
2. ACKACK Packet: the receiver gets packets and adjust the Base_delta and Drift.
   RTT =(recvTime2– recvTime1) = (ACKACK_TS- Send_TS)
   Base_delta = ½ RTT + clock_diff= recvTime2 – ACKACK_TS = recvTime1- Send_TS

The second responsibility is sending the ACK and NACKs to sender.
The third responsibility is Delivering the packet on Delivery Time.

Data Structure of Sender
2d array Buffer Window
Time Base_Delta
Time RTT
Time clock_diff /*Optional*/
Time lantencywindow

Data Structure of Receiver
2d array Latency Window
Time   Base_Delta
Time   RTT
Time clock_diff   /*Optional*/
array to store recent 50 RTT   /*aim for updating RTT*/
array to store recent 50 Base_Delta   /* aim for updating Base_Delta*/

Latency Window Size= 1s * 20Mbps = 2.5 Mbtyes = 2.5*10^6 bytes <=1786 *1400 byte packets
We will set the size of latency window to 1786

Data Structure of Message
/* 0-> Sender sends data to Receiver
    1->Sender sends ACKACK
     2->Sender Receive ACK from Receiver
     3->Receiver sends request
     4->Sender sends permission
    5-> Sender sends decline
*/
Type
Seq   /* Sequence number of Data Packet*/
Delivery Time   /* The delivery time of data*/
DATA

ACK
NACK
ACKACK   /* ACK of ACK*/

Send_TS   /*Send time of packet
                        It could change when we resend the Packet*/
Receive1_TS   /*First time receiver receives the packet   */
ACKACK_TS   /*The time when sender gets the ACK */
Receive2_TS   /* No need, it is the time when receiver gets ACKACK */

Base_Delta   /*Measure of ½ RTT+ clock _diff*/
RTT   /*Round trip time between sender and receiver*/

Question: When updating the Base_Delta, how to deal with the delivery time of the packet already in the latency window.