Sender process(rt_srv):
Before anything starts, sender should build connection will receiver, once it gets request from receiver, it will reply permission with taking a Time stamp Receiving TimeStamp.
In the request packet, receiver will includes Window size and Latency window to help sender initialize the window.

1.Get data from local app and take the send time stamp(send_TS), then store it into the window and send it to the Receiver.

2.Check whether we should shift the window.
If (sendTS+base_delta+Latency_Window<=Sender_now +clock_diff)
then we shift otherwise, do nothing.
The condition could convert to sendTS + ½ RTT +LatencyWindow<=Sender_now

3.Receive the ACK packet from receiver and update the ½ RTT
Send the ACKACK packet
check whether we have NACK and resend packet based on the condition :
sendTS+latencyWindow >Sender_now
(if is NACK type, which is type 6, keep original sendTS, get time of N_sendTS to receiver)

4. If Sender receive the request, which is message type 3, it will send decline, sending message with type 5.

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 BaseDelta.
BaseDelta = Now - Receive Time1
½ RTT = (Now – sendTS)/2

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 clock_diff 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.
Depends on the condition : sendTS+ base_Delta + lantencywindow and now.
If less than or equal, deliver the packet
Otherwise keep into the buffer.

Data Structure of Sender
2d array Buffer Window
Time ½ RTT
Time window
Time timer_array /* array to store sendTS in order to shift window and resend pkts */

Data Structure of Receiver
2d array Latency Window
Time Base_Delta
Time ½ RTT
array to store recent 50 ½ RTT /*aim for updating ½ 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
T (ns)*2.5M bytes /s / 1400 bytes = T *2.5*10^3/ 1400 = T*25/14

Data Structure of Message
/* 0-> Sender sends data to Receiver
1->Sender sends ACKACK
2->Receiver sends ACK
3->Receiver sends request
4->Sender sends permission
5-> Sender sends decline
6 -> Sender sends data to Receiver, but it is request from receiver (NACK)
*/
Type
Seq /* Sequence number of Data Packet*/

Char data []

ACK
NACK[]

Send_TS /* Send time of packet */
N_Send_TS /* Resend time of packet*/
Receive1_TS /*First time receiver receives the packet */
ACKACK_TS/*The time when sender gets the ACK */
LatencyWindow

Window Size /*Calculate by receiver*/
HalfRTT /*Measure of ½ RTT */