Sender process(rt_srv):
Before anything starts, sender should build connection will receiver, once it gets request from receiver, including Latency window and its size, it will reply permission with taking Time stamp of Receiving Time in order to initial RTT and BaseDelta in receiver side.

In the main part, the first thing we do is checking 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

Sender has two ports, rcv and app in respond to messages from receiver and local app.

If the message is from local app, sender will get data and take the send time stamp and seq then store it into the window and send it to the Receiver.

If the message is from receiver, sender makes different choice based on the type in package.

If the type is 2, sender will receive the ACK packet from receiver and update the ½ RTT
and send the ACKACK packet.
Besides, sender will check whether we have NACK and resend packet based on the condition :
sendTS+latencyWindow >Sender_now
(When resending Nack, sender keeps original sendTS and gets time of N_sendTS to receiver)

If the type is 3, Sender receive the request, sender will compare the address, if the address is not the same as receiver, it send decline with type 5.Otherwise, it just ignore the message.

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 the left hand side is less than or equal than the right hand side, the receiver deliver the packet .Otherwise keep into the buffer.

Data Structure of Sender

Package Array	/* Store an array of package in order to recovery*/
Time Array	

/* array to store sendTS+latency window in order to shift window and resend pkts */

Time ½ RTT
Time latency window

Data Structure of Receiver

Package Array	/* Store an array of package in order to deliver packets on time*/
Time 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 calculation :
T (ns)*2.5M bytes /s / 1400 bytes = T *2.5*10^3/ 1400 = T*25/14

Data Structure of Message
Int Type
```
/* 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)
*/
```

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	/* The 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 */
```