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.

------------------------

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 (type 0).

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 (type 2) from receiver and update the ½ RTT

Send the ACKACK packet (type 1)

check whether we have NACK and resend packet (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 (type 3) to sender until it gets reply.

If it gets the permission(type 4), then we go to main body part. Else, it will exit.(type 5)

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: (type 0 or 6) 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: (type1)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. (type 2)

// Check available Nack, sendTS+latency window+ base\_delta >now +RTT

The third responsibility is Delivering the packet on Delivery Time.

Depends on 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 lantencywindow

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

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\*/

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 \*/

WindowTime (Time) /\*Latency Window for calculate the size of window \*/

HalfRTT /\*Measure of ½ RTT \*/

Text

Description automatically generated with medium confidence

1.Sender

Initial { type:0 | seq | read data | Delivery Time |

sendTS| }

{ACK, NACK,ACKACK = -1, Base\_Delta = -1, Drift = -1

Receive1\_TS,ACKACK\_TS = NULL}

2.Receiver

Take Receive1\_TS

Type set to 1

Find Nack

Send ACK= seq

Put Updated Base\_Delta, Drift

3.Sender

Take ACKACK\_TS

Type set to 2

Adjust Base\_Delta and Drift

4.Receiver

Get ACKACK\_TS,

calculate new RTT

new Base\_Delta