Topic-3
Selective Repeat Request Protocol

Prepared by K.Lakshmipathi raju, Assistant professor
Dept of IT

Why Selective Repeat Protocol?

The go-back-n protocol works well if errors are less, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the selective repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.

Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors) :

- Receiver must be able to accept packets out of order.
- Since receiver must release packets to higher layer in order, the receiver must be able to buffer some packets.

Retransmission requests :

- **Implicit –** The receiver acknowledges every good packet, packets that are not ACKed before a time-out are assumed lost or in error.Notice that this approach must be used to be sure that every packet is eventually received.
- **Explicit –** An explicit NAK (selective reject) can request retransmission of just one packet. This approach can expedite the retransmission but is not strictly needed.
- One or both approaches are used in practice.

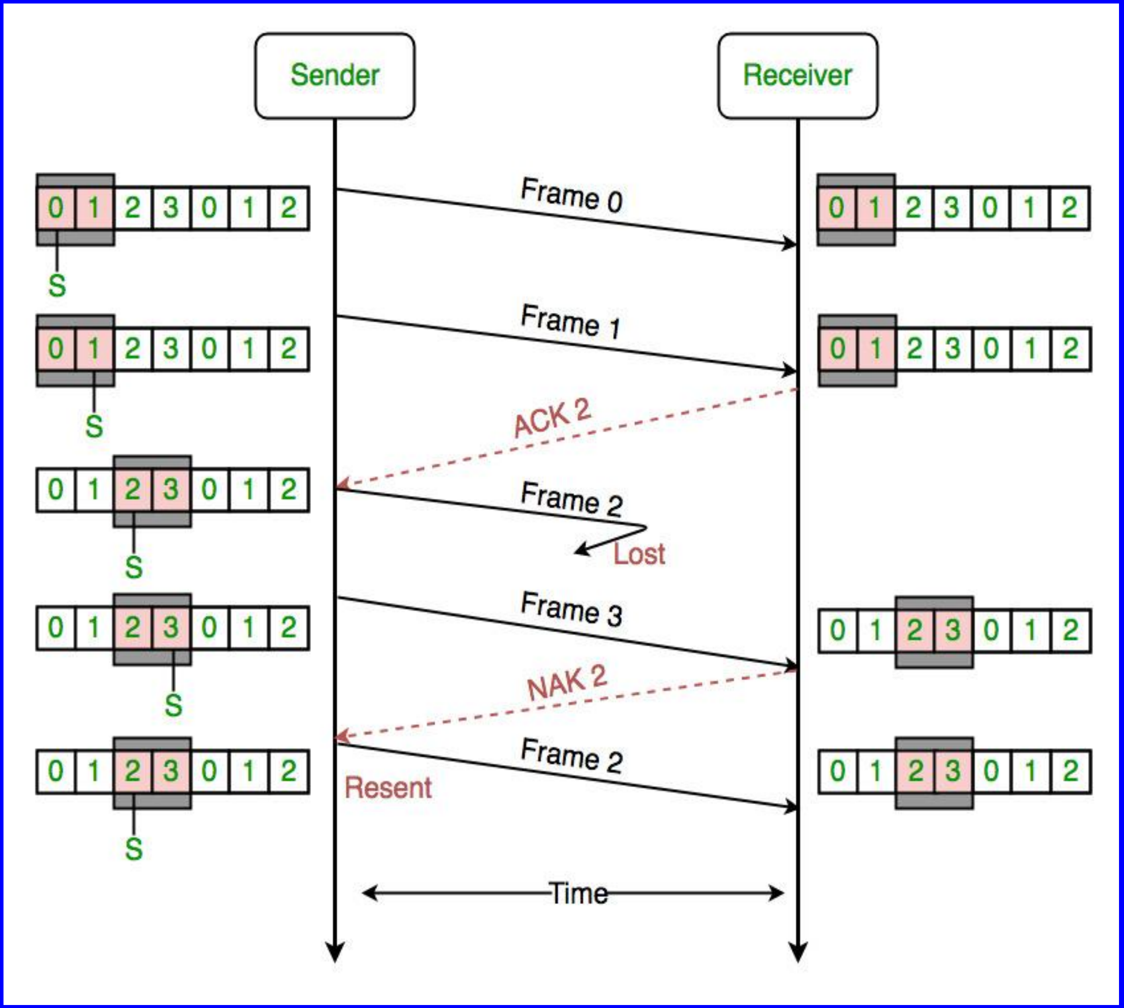
Selective Repeat Protocol (SRP) :

This protocol(SRP) is mostly identical to GBN protocol, except that buffers are used and the receiver, and the sender, each maintain a window of size. SRP works better when the link is very unreliable. Because in this case, retransmission tends to happen more frequently, selectively retransmitting frames is more efficient than retransmitting all of them. SRP also requires full duplex link. backward acknowledgements are also in progress.

Selective Repeat Protocol (SRP) :

- Sender's Windows (Ws) = Receiver's Windows (Wr).
- Window size should be less than or equal to half the sequence number in SR protocol. This is to avoid packets being recognized incorrectly. If the windows size is greater than half the sequence number space, then if an ACK is lost, the sender may send new packets that the receiver believes are retransmissions.
- Sender can transmit new packets as long as their number is with W of all unACKed packets.
- Sender retransmit un-ACKed packets after a timeout – Or upon a NAK if NAK is employed.
- Receiver ACKs all correct packets.
- Receiver stores correct packets until they can be delivered in order to the higher layer.
- In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

Selective Repeat Protocol (SRP) :



Selective Repeat Protocol (SRP) :

Efficiency of Selective Repeat Protocol (SRP) is same as GO-Back-N’s efficiency :

Efficiency = N/(1+2a)

Where a = Propagation delay / Transmission delay

Buffers = N + N

Sequence number = N(sender side) + N (Receiver Side)

The Stop and Wait ARQ offers error and flow control, but may cause big performance issues as sender always waits for acknowledgement even if it has next packet ready to send. Consider a situation where you have a high bandwidth connection and propagation delay is also high (you are connected to some server in some other country though a high speed connection), you can’t use this full speed due to limitations of stop and wait.

Sliding Window protocol handles this efficiency issue by sending more than one packet at a time with a larger sequence numbers. The idea is same as pipelining in architectures.

Terminologies

Transmission Delay (Tt) – Time to transmit the packet from host to the outgoing link. If B is the Bandwidth of the link and D is the Data Size to transmit

$$Tt = D/B$$

Propagation Delay (Tp) – It is the time taken by the first bit transferred by the host onto the outgoing link to reach the destination. It depends on the distance d and the wave propagation speed s (depends on the characteristics of the medium).

$$Tp = d/s$$

Efficiency – It is defined as the ratio of total useful time to the total cycle time of a packet. For stop and wait protocol,

Total cycle time = Tt(data) + Tp(data) + Tt(acknowledgement) + Tp(acknowledgement)
= Tt(data) + Tp(data) + Tp(acknowledgement)
= Tt + 2*Tp

Since acknowledgements are very less in size, their transmission delay can be neglected.

Efficiency = Useful Time / Total Cycle Time
= Tt/(Tt + 2*Tp) (For Stop and Wait)
= 1/(1+2a) [Using a = Tp/Tt]

Effective Bandwidth(EB) or Throughput – Number of bits sent per second.

EB = Data Size(L) / Total Cycle time(Tt + 2*Tp)

Multiplying and dividing by Bandwidth (B),

= (1/(1+2a)) * B [Using a = Tp/Tt]

= Efficiency * Bandwidth

Capacity of link – If a channel is Full Duplex, then bits can be transferred in both the directions and without any collisions. Number of bits a channel/Link can hold at maximum is its capacity.

Capacity = Bandwidth(B) * Propagation(Tp)

For Full Duplex channels,

Capacity = 2*Bandwidth(B) * Propagation(Tp)

Concept Of Pipelining

In Stop and Wait protocol, only 1 packet is transmitted onto the link and then sender waits for acknowledgement from the receiver. The problem in this setup is that efficiency is very less as we are not filling the channel with more packets after 1st packet has been put onto the link. Within the total cycle time of Tt + 2*Tp units, we will now calculate the maximum number of packets that sender can transmit on the link before getting an acknowledgement.

In Tt units ----> 1 packet is Transmitted.

In 1 units ----> 1/Tt packet can be Transmitted.

In Tt + 2*Tp units -----> (Tt + 2*Tp)/Tt packets can be Transmitted

------> 1 + 2a[Using a = Tp/Tt]

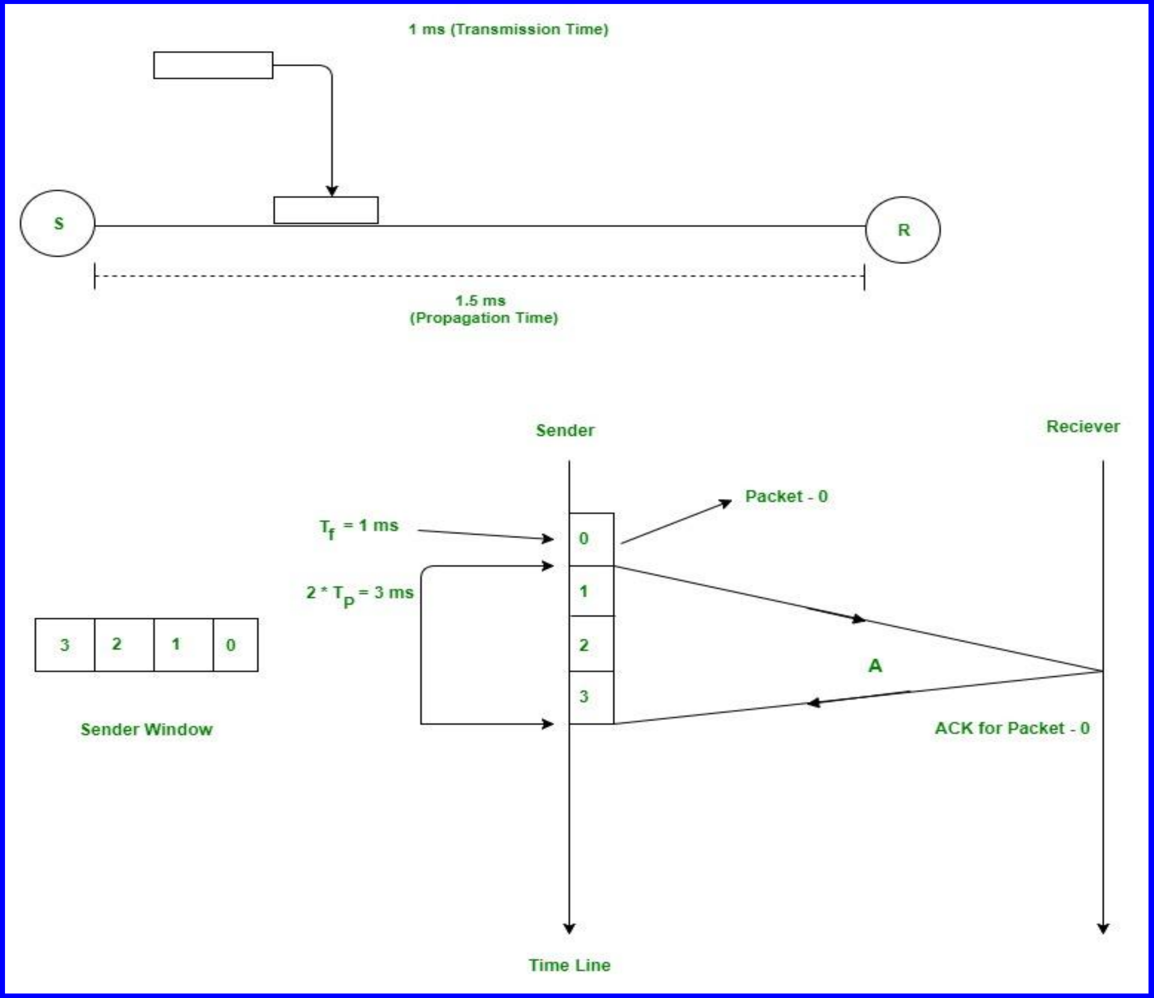
Maximum packets That can be Transmitted in total cycle time = 1+2*a

Example :

Consider Tt = 1ms, Tp = 1.5ms.

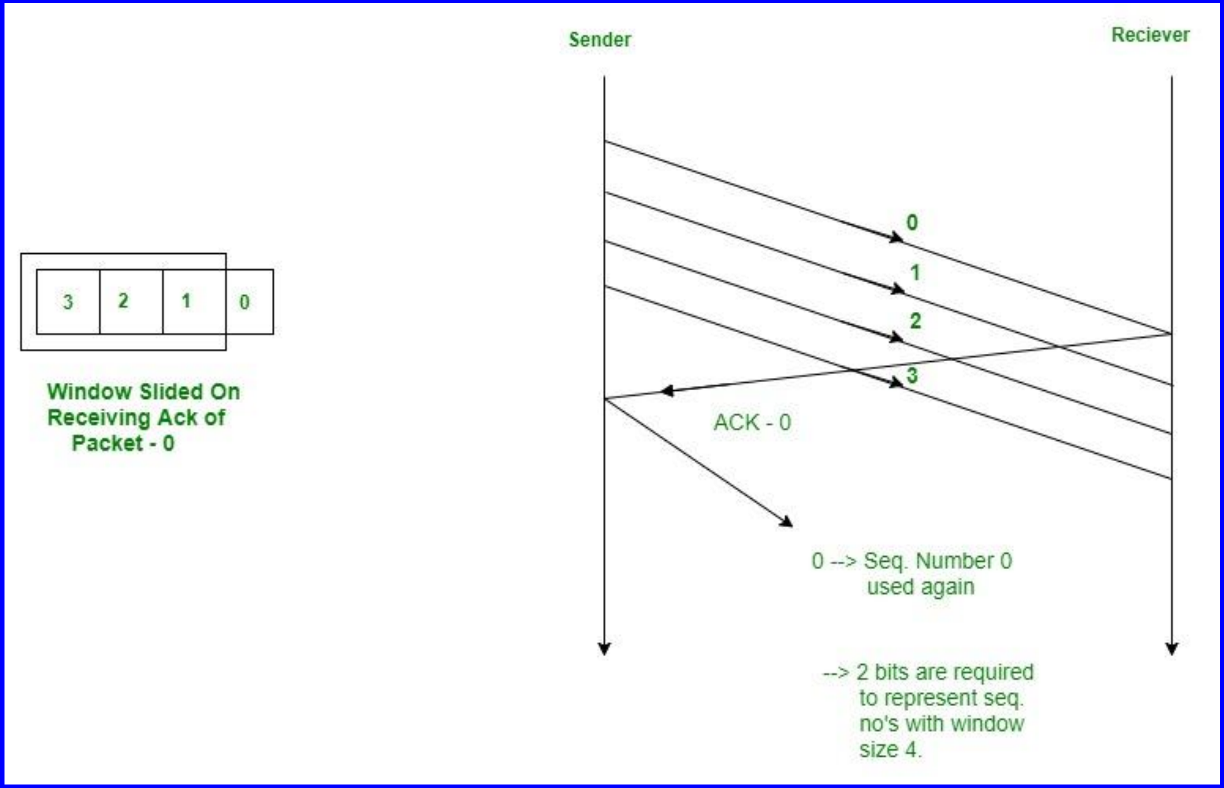
In the picture given below, after sender has transmitted packet 0, it will immediately transmit packets 1, 2, 3. Acknowledgement for 0 will arrive after 2*1.5 = 3ms. In Stop and Wait, in time 1 + 2*1.5 = 4ms, we were transferring one packet only. Here we keep a **window of packets which we have transmitted but not yet acknowledged**.

Concept Of Pipelining



Concept Of Pipelining

After we have received the Ack for packet 0, window slides and the next packet can be assigned sequence number 0. We reuse the sequence numbers which we have acknowledged so that header size can be kept minimum as shown in the diagram given below.



Minimum Number Of Bits For Sender window

As we have seen above,

Maximum window size = 1 + 2*a   where a = Tp/Tt

Minimum sequence numbers required = 1 + 2*a.

All the packets in the current window will be given a sequence number.

Number of bits required to represent the sender window = **ceil(log2(1+2*a)).**

But sometimes number of bits in the protocol headers is pre-defined. Size of sequence number field in header will also determine the maximum number of packets that we can send in total cycle time. If N is the size of sequence number field in the header in bits, then we can have 2^N sequence numbers.

Window Size ws = **min(1+2*a, 2^N)**

If you want to calculate minimum bits required to represent sequence numbers/sender window, it will be **ceil(log2(ws))**.

Thank You