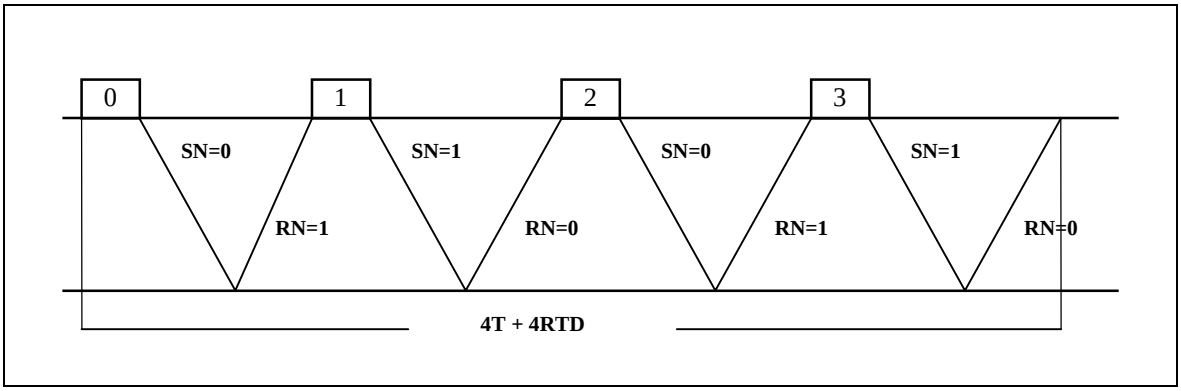
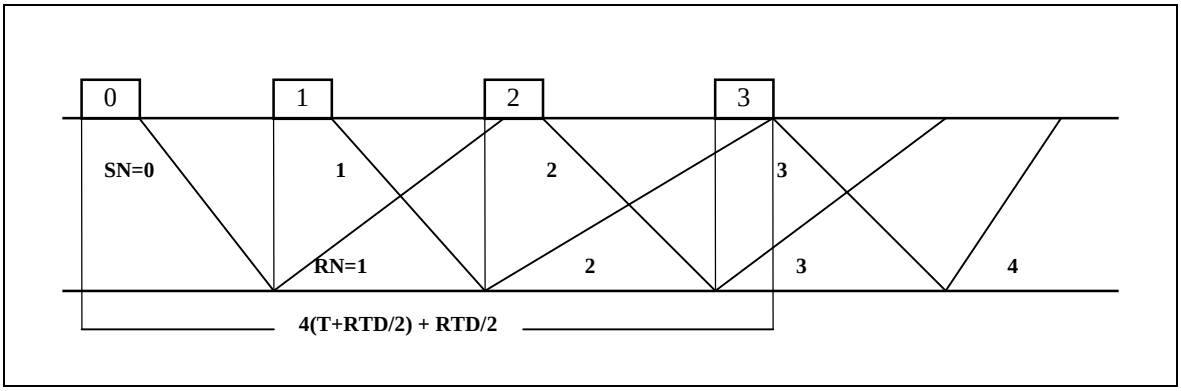
2.4.2 Go-BACK-N ARQ

- Also known as Sliding Window
- Most popular ARQ today
- Motivation: consider transmission of 4 packets

(1) With Stop and Wait



(2) Alternative : Let sender send 4 packets before waiting for any ACK



Saves time over stop and wait: ($time_{wind}$ / $time_{stop\&wait}$) ≈ RTD/2 for large n

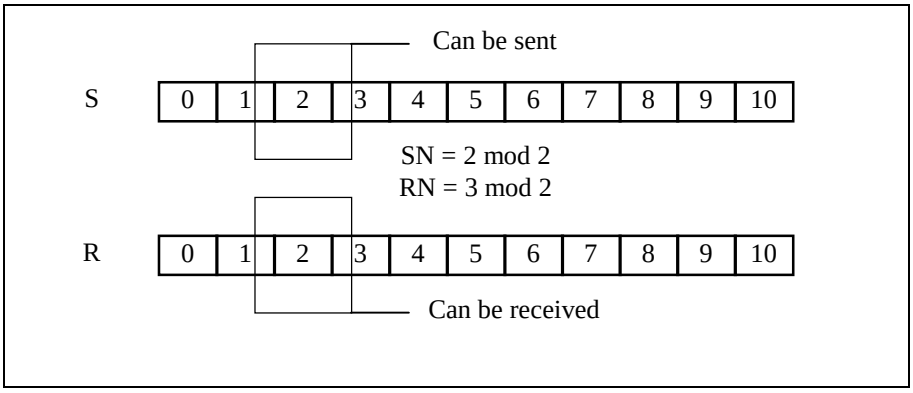
---

Note: Go-Back-N will use :

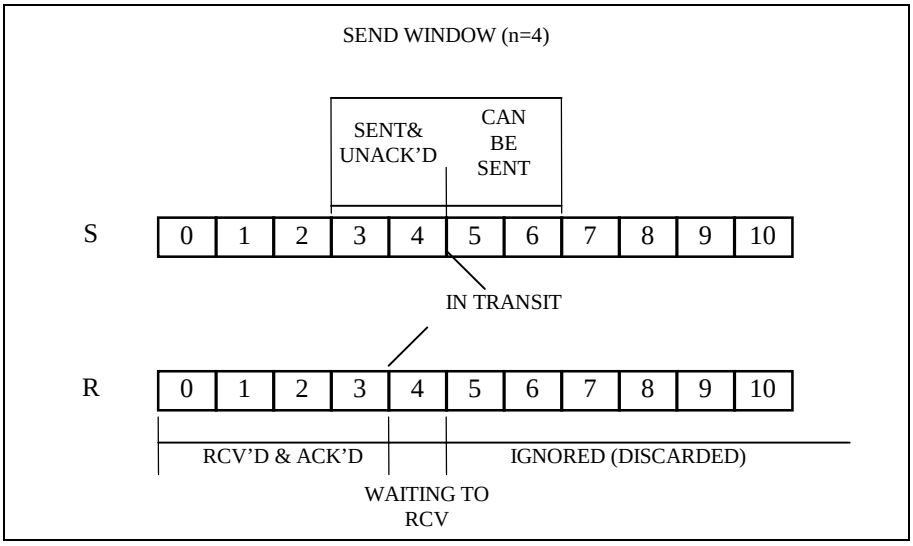
- SN
- RN

just as stop and wait did. But domain of SN, RN will be n+1 rather than 2.

Stop and Wait



Sliding Window



View: The sender window slowly moves left to right. Point & quickly moves left to right within sender window.

The lower edge of 5's window is advanced whenever 5 receives a frame with RQ inside window larger than lower window edge (RQ > 3 in this example).

Q. Why doesn't R accept <u>any</u> packet with SN ≥ RN?

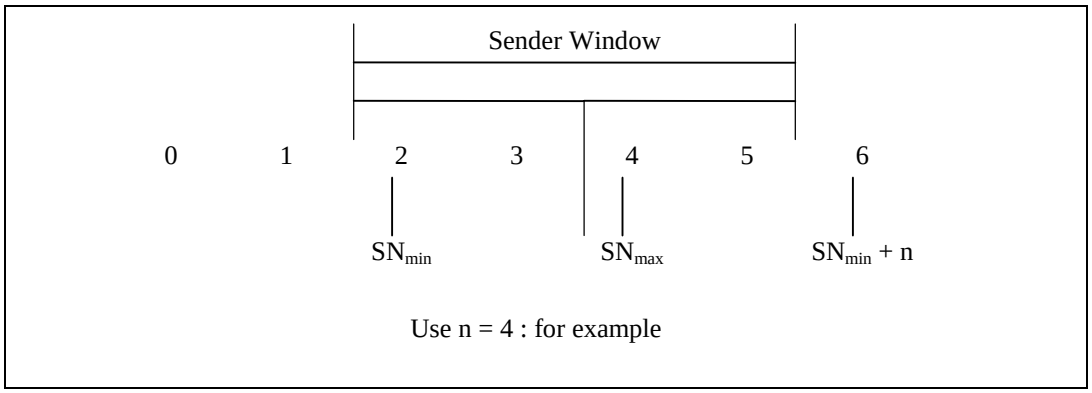
A. Receiver would need to buffer m packets, and could accept SN to SN + m - 1. This is2.4.3-Selective Repeat.

Algorithm

Variables

Sender: 1) SN_{min} (smallest unack pck #)

2) Sn_{max} (smallest unsent pck #)



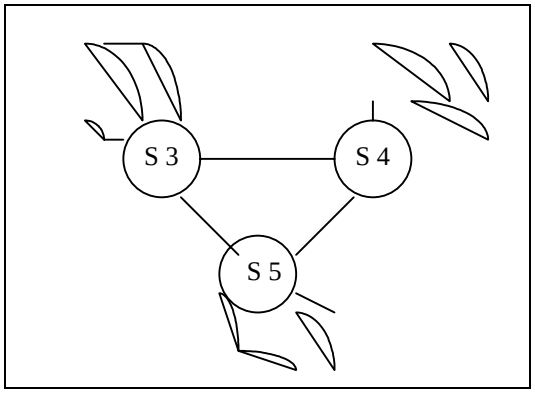
Receiver : RN (same as meaning as in stop and wait - next SN expected)

Sender Algorithm

S1 : Q. Initial condition is what?

A. $SN_{min} = SN_{max} = 0$

State Machine



S2 : Arbitrary, bounded delay between when condition for a state is enabled, and when state's actions are executed.

[Assume n times. Could be layer 2 or 4] [3's denote the "timer" view.]

S3 : If $SN_{max} < SN_{min} + n$ and packet available from higher layer:

- accept packet
- assign packet SN_{max}
- set time for packet SN_{max}
- transmit packet
- $SN_{max} := SN_{max} + 1$

S4 : If error-free frame received from B with $RN > SN_{min}$:

- $SN_{min} := RN$; [Clear timers for all packets with SN < RN]

S5 : If $SN_{min} < SN_{max}$ and no frame currently in transmission : [SN's timer pops:]

- choose a packet with $SN_{min} \leq SN < SN_{max}$
- transmit SN^{th} packet
- [reset timer]

Receive Algorithm

S1 : Initially : RN = 0

S2 : If error-free frame received from A with SN = RN :

- release packet to higher layer
- RN++

S3 : At arbitrary times, but within bounded delay after receiving error free data frame from A :

- transmit frame to A with RN in request # fold

How is "arbitrary delay" handled?

- Timer :
  - set time for packet SN; resend SN on pop; clear timer when RN = SN.
  - or set time for entire window
- You can order operations in above algorithm many ways (e.g., S5 does not require retransmission in ascending SN order.

Show fig:

- 2.24 Normal
- 2.25 Packet Loss
- 2.26 ACK Loss
- 2.27 Long Reverse Packets

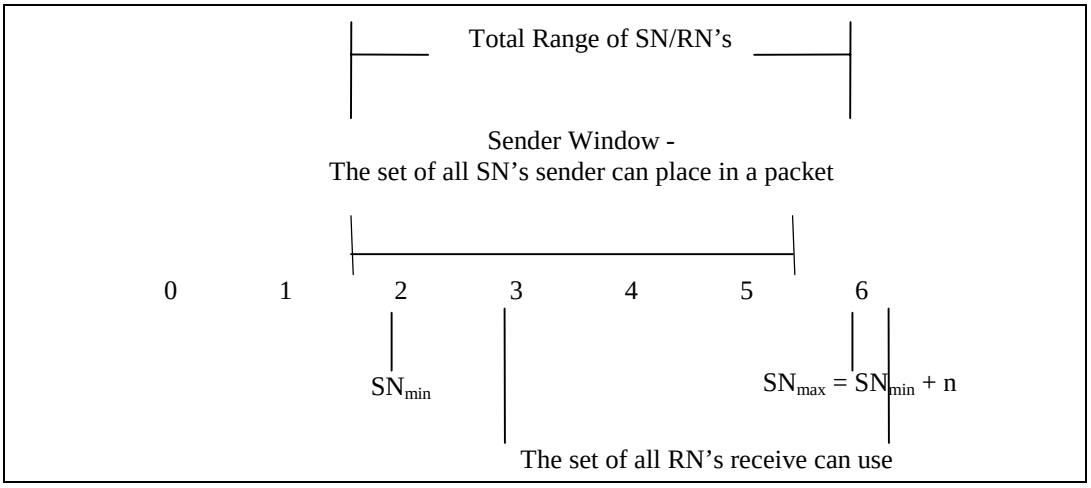
Go-Back-N

Some way to handle retransmission in S5 of sender:

1) Set timer. On timer pop, retransmit $SN_{min} \ldots SN_{max-1}$
2) After A sends entire window, cycle back and resend window
3) A waits for request from B for retransmission.

Q. When send window closed, what value does SN_{max} have ?

A. SN_{min} + n: Total range of SN/RN's used is $SN_{min}..SN_{max}$ + n [= 2..6 for n =4]



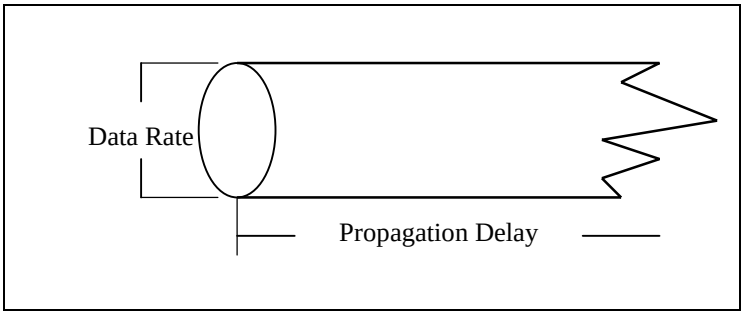
Q. What is max # SN's needed?

A. n+1 (only one old packet can be in transit-see proofs in book)

Q. How do you select n?

A. Factor of :

- Sender buffer space and # timers feasible
- Desired # bits for SN field in header
- Loop delay : large pipe — larger n



TCP uses separate sequence # for each byte, not each packet hence TCP must be replaces/modified in gigabit (UTP not).

Efficiency of Go-Back-N

Retransmission for 3 reasons

- case 1 : Error in forward directions
- case 2 : Error in reverse directions
- case 3 : Longer frames in reverse direction
- case 4 : Timer < RTD (Dr.Abrams's reason)
- case 5 : High variance in RTD with timer set near mean RTD (Dr.Abrams's reason)

Consider case 3:

First consider mean packet size equal in both directions, but exponentially distribution. Then,

Prob[frame not ACK'd by time window is exhausted] = 1/[2(n+1)]

Q. Stop and Wait: What % of time does case 3 occur?
A. n = 1; 1/[2(1+1)] = 25 %
Note : n => 7 => Prob. = 1/16

Consider case 2: Set n large and not a problem

Q. Why?
A. ACK for SN = n lost, but ACK for SN = n+1 slides to ACK n and n+1

Consider case 1: Large n (good for (3) or (2)) is **BAD**.

If packet close to left window edge is error, then much BW is wasted sending rest of window, which receiver ignores.

Some solutions:

- RCVR sends ACK immediately upon receiving frame in error
- RCVR puts RN in trailer of frame, reducing feedback bus frame

2.4.3 Selective Repeat ARQ

Go-Back-N retransmits ≥ 1 RTD of frames upon single error

Bad if :

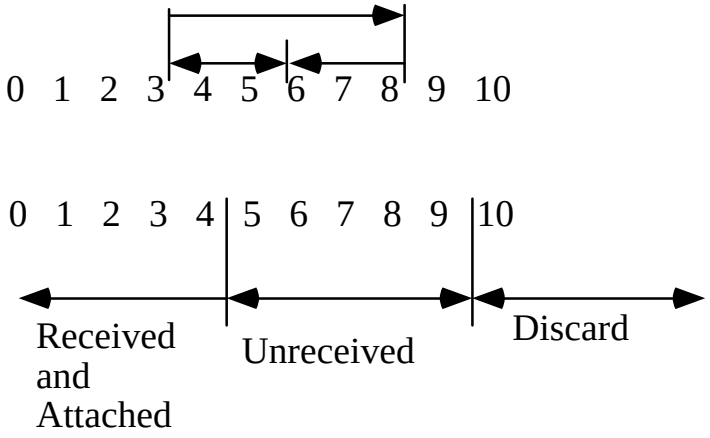
- error probability is high (cellular phone)
- n is very large (satellite or Gbps-fiber)

So in Selective Repeat:

(BG) 2.4.3 Selective Repeat

- Modification of Go back N:

  -- Receiver buffers packets received out of order
  -- Receive window:



  -- Typically receiver window size = sender size

- BG: "Receiver requests retransmission of any packets missing from received sequence."

- TCP: Same meaning of RN ACK's - send upper half of window (wastefully)

Upper bound on fraction packets successfully derived to B: $n \leq 1 - p$
where p = probability of frame error.
So if p = 0.1, (1- 0.1)* 100 = 90% of frames delivered successfully.
In 10 Mbps medium, you get ≤ 9 Mbps. But, you cannot get this bound
Q. Why?
A. Finite time to propagate

1) information from B to A that error occurred
2) retransmission frame

(1) + (2) = RTD

Let β = exp[#frames sent in RTD]

$$\eta \leq \frac{1-p}{1+p\beta}$$

- $\beta \Rightarrow \infty$; η bound approaches 0
- $\beta \Rightarrow 10$; $\eta < (1-p)/2$ or 50 % drop (for stop and wait)

More facts on Selective Repeat;

1) SN range is 2n (vs n + 1 for Go-Back-N)
2)
   - Selective repeat wins due to receiver buffering out of order packets
   - To do even better, put more ACK information than RN into B⎯⎯ A traffic, to handle multitude-frame errors:
     A) B sends lowest j packet # s not yet received. [But this wastes BW due to large frame headers.]
     B) B sends k bits, one bit per packet after RN. [BIT MAP].