CS3205-Assignment#4: Go-Back-N and Selective Repeat Protocols

VijayaSimhaReddy - CS18D018

Department of Computer Science & Engineering
Indian Institute of Technology Madras, India

Outline of Discussion

- Data Link Layer
- Sliding Window Protocols
- Comparing GO-BACK-N vs S-R ARQs
- Assignment#4 Deliverables

Data Link Layer in OSI reference model

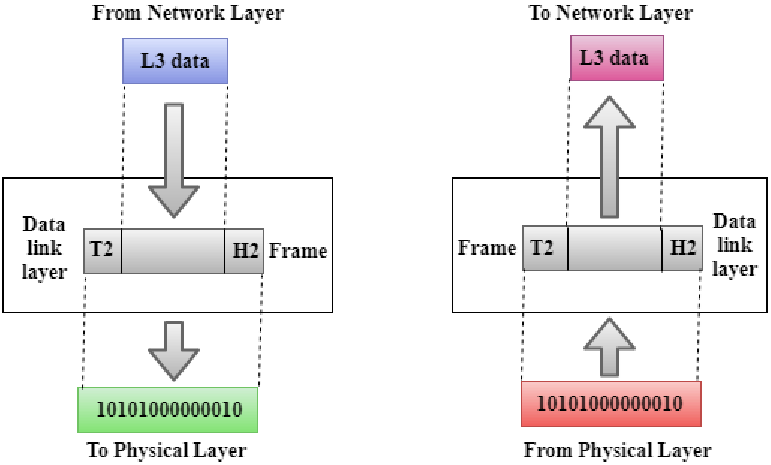


Figure: DLL in OSI (Flow of Frames from sender to receiver)

- Main functionalities of DLL :
  1. Framing
  2. Physical/MAC Addressing
  3. Flow and Error Control (Stop-and-Wait,GBN,SR ARQs)
  4. Access Control (ALOHA,CSMA/CD,CA,Token,Poll,TDM etc.)

https://www.javatpoint.com/osi-model

GO-BACK-N Automatic Repeat Request (1/4)

Need for GBN

- Stop-and-wait ARQ is **less efficient**
- Need to use **frame pipelining** so as to send multiple frames without waiting for ACks
- Need to achieve better link utilization, throughput so as to use **bandwidth effectively**

G-B-N ARQ

- Sender Window Size (SWS) is **N** and Receiver Window Size (RWS) is always **1**
- **Cumulative** acknowledgements are used in GBN
- Bits to be allotted for sequence number field is $\lceil log_2(N+1) \rceil$ in GBN
- Receiver receives **in-order** frame only and cannot accept (discards) frames out-of-sequence
- Sender must **re-send entire window** in the event of an erroneous/lost frame

GO-BACK-N Sender and Receiver (2/4)

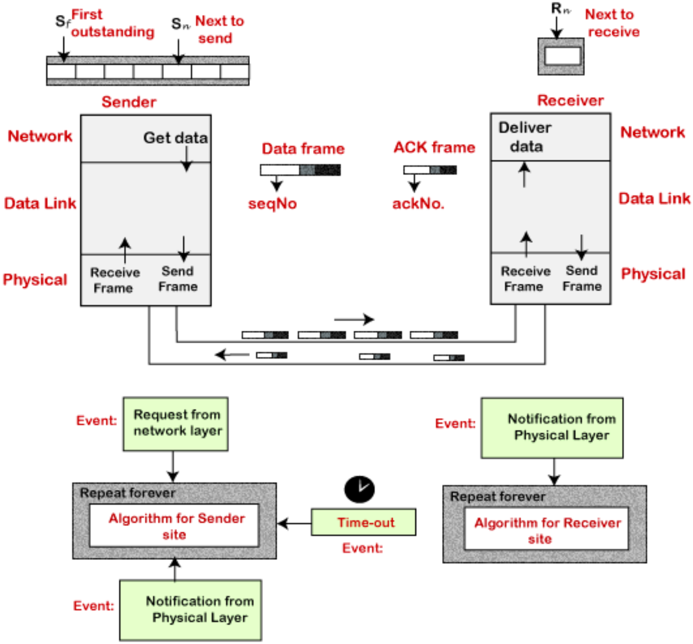
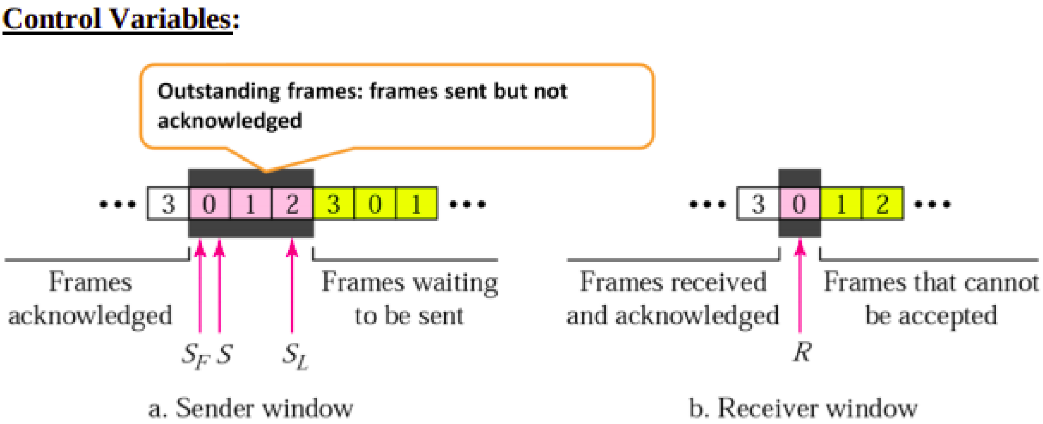


Figure: Design of Go-Back-N ARQ

https://www.javatpoint.com/sliding-window-protocol

Sliding Windows in GBN-ARQ (3/4)


Control Variables:
Outstanding frames: frames sent but not acknowledged
3
0
1
2
3
0
1
Frames acknowledged
Frames waiting to be sent
S_F S
S_L
a. Sender window
3
0
1
2
Frames received and acknowledged
Frames that cannot be accepted
R
b. Receiver window

- Variables that keeps track of **sliding** window :
  1. S: holds the sequence number of **recently sent** frame
  2. S_F: holds sequence number of **first frame** in the window
  3. S_L: holds the sequence number of **last frame**
  4. R: sequence number of the **frame expected** to be received

https://www.nielit.gov.in/gorakhpur/sites/default/files/Gorakhpur/alevel_2_dcn_07apr_PT.pdf

Understanding G-B-N ARQ Iteratively (4/4)

Lost Data Frame Case

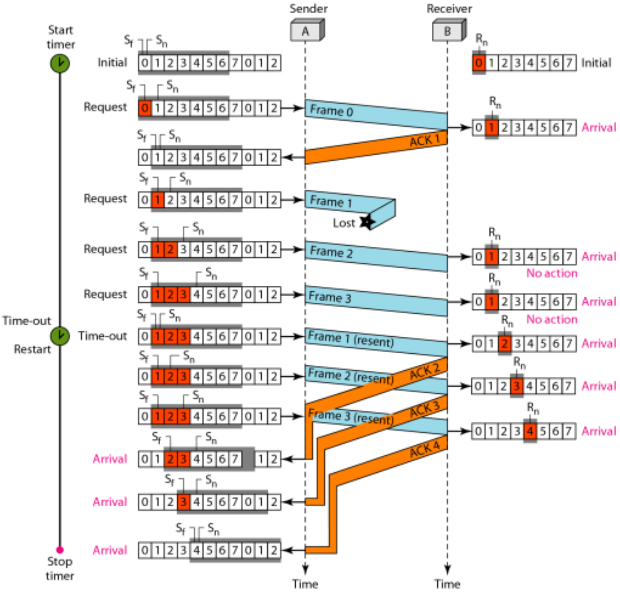


Figure: Illustrating GO-BACK-N ARQ

https://www.nielit.gov.in/gorakhpur/sites/default/files/Gorakhpur/alevel_2_dcn_07apr_PT.pdf

Selective Repeat ARQ (1/4)

Need for SR

- Need to address **issues** in GBN
- Major problem with Go-Back-N is **resending the entire window** when an error occurs, because receiver can only accept frame in-order
- Need for receiver to be able to accept packets **out-of-order** using buffer space
- Need for a **superior** protocol to combine advantages of both Stop-Wait and GBN

SR ARQ

- Selective Repeat attempts to **retransmit only those packets** that are actually lost (due to errors)
- **Independent** acknowledgements are used in SR ARQ
- Bits to be allotted for sequence number field is $\lceil log_2(N+N)\rceil$ in SR
- **Sorting mechanism** at receiver's side adds to more complex implementation with **SWS = RWS**

SELECTIVE REPEAT Sender and Receiver (2/4)

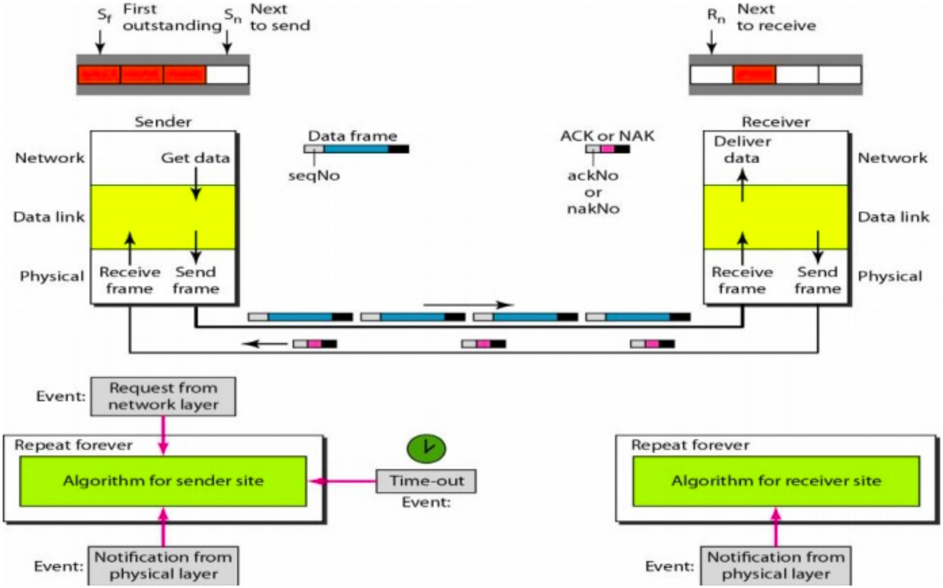


Figure: Design of SR ARQ

https://www.brainkart.com/article/Noisy-Channels-Protocol_13447/

Sliding Windows in SR-ARQ (3/4)

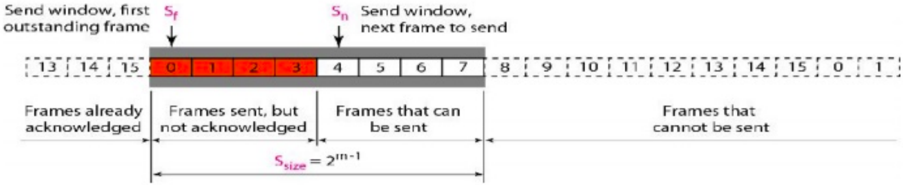

Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

Figure: SR Sender Window

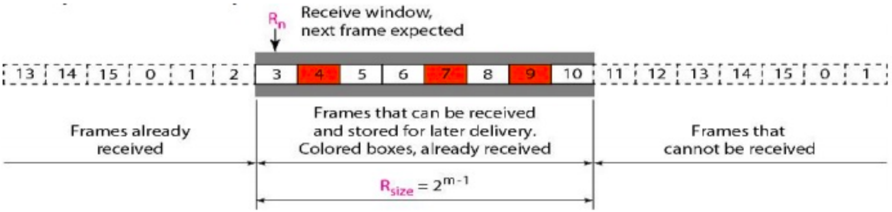

R_n
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

Figure: SR Receiver Window

https://www.brainkart.com/article/Noisy-Channels-Protocol_13447/

Understanding S-R ARQ Iteratively (4/4)

Lost Data Frame Case

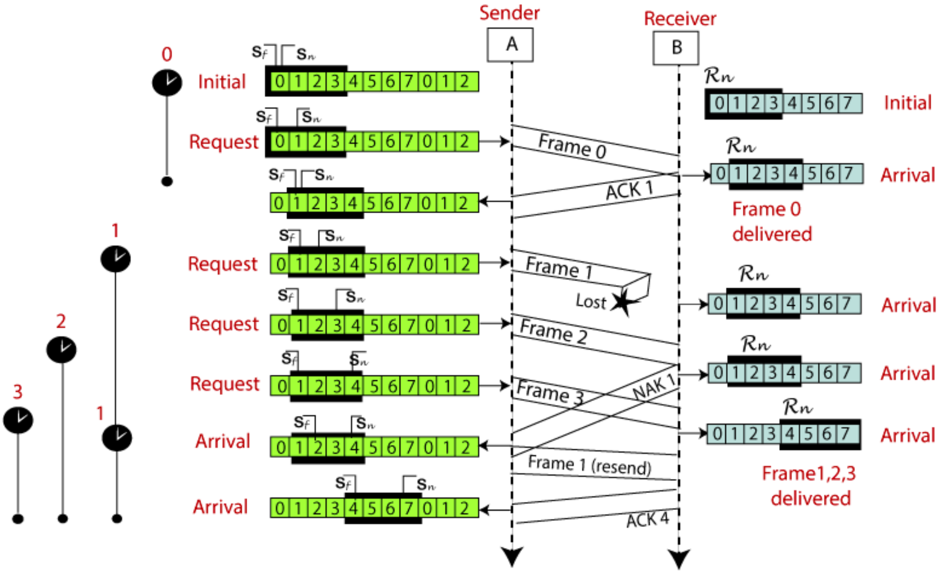
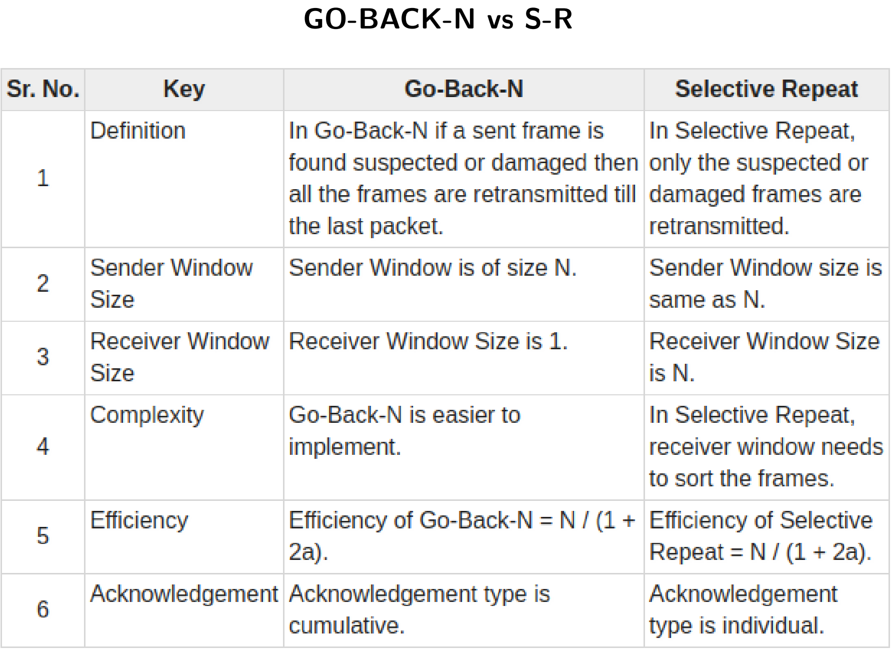


Figure: Illustrating S-R ARQ

https://www.javatpoint.com/sliding-window-protocol

Comparing GBN and SR (1/3)



GO-BACK-N vs S-R

Sr. No.	Key	Go-Back-N	Selective Repeat
1	Definition	In Go-Back-N if a sent frame is found suspected or damaged then all the frames are retransmitted till the last packet.	In Selective Repeat, only the suspected or damaged frames are retransmitted.
2	Sender Window Size	Sender Window is of size N.	Sender Window size is same as N.
3	Receiver Window Size	Receiver Window Size is 1.	Receiver Window Size is N.
4	Complexity	Go-Back-N is easier to implement.	In Selective Repeat, receiver window needs to sort the frames.
5	Efficiency	Efficiency of Go-Back-N = N / (1 + 2a).	Efficiency of Selective Repeat = N / (1 + 2a).
6	Acknowledgement	Acknowledgement type is cumulative.	Acknowledgement type is individual.

https://www.tutorialspoint.com/difference-between-go-back-n-and-selective-repeat-protocol

Which Protocol is effective ? (2/3)

CasesA : GO-BACK-N

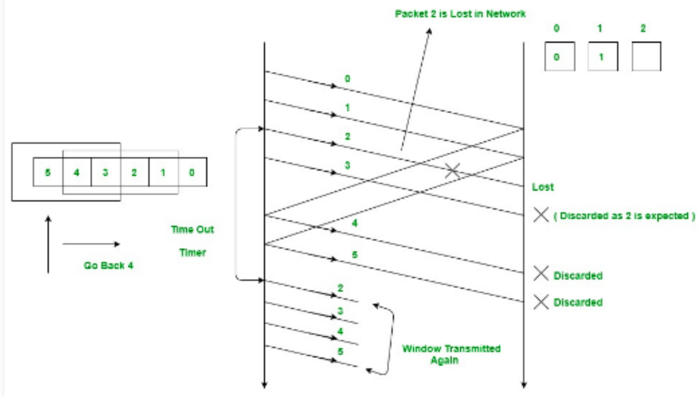
- When Buffer Space is of more concern than **bandwidth**
- **Less complexity**,less CPU cycles
- Useful in **less re-transmissions** scenarios and less sequence numbers available
- If **error rate is low**, use Go-back-N.

CasesB : Selective-Repeat

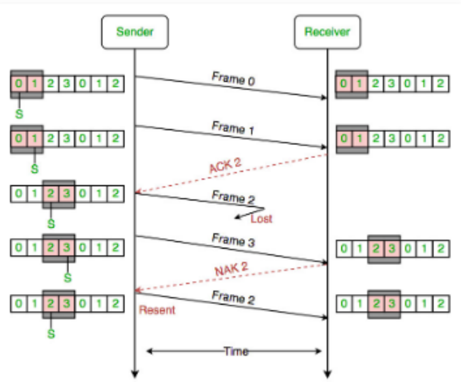
- When Bandwidth is of more concern than **buffer space**
- **More processing power**,cpu cycles at receiver
- In **erroneous/noisy links** having more re-transmits and more available sequence numbers
- If **error rate is high**, selective repeat is better in terms of number of retransmits

GBN vs SR Protocols (3/3)

GO-BACK-N vs Selective Repeat



GO-BACK-4



SR ARQ

https://www.geeksforgeeks.org/sliding-window-protocol-set-3-selective-repeat/
https://www.geeksforgeeks.org/sliding-window-protocol-set-2-receiver-side/

Implementing S-R ARQ (Sender)

- Packet Generation with packet length follows **Uniform distribution**, rate of generating packets in **periodic** time intervals (packets/sec)
- Transmit buffer to store number of packets depends on **buffer size**
- Need to keep track of **Time-out timers** for each packet
- Given an n-bit sequence number, the maximum window size will be 2^{n-1}
- Next packet can be sent when **outstanding**(unacknowledged) packets not exceed window size
- **ACK(Independent/Selective)** receive event needs to update local state variables, cancel T.Os, removing packet from transmit buffer
- For each ACK, **RTT** calculation, **Avg.RTT update** for acknowledged packets so far
- Time Expiry/Time Out event results in **retransmitting selective** packet

Assumptions for SR-Sender

- Each packet length follows **Uniform(40, MAX_PACKET_LENGTH)** bytes
- First byte(s) of the packet contains its **unique** sequence number
- Time out timers are assumed to be **300 millisec** for initial **10** packets and all subsequent packets T.O timers set to $2 \times \text{RTT}_{\text{Avg}}$
- Termination Criteria is either no.of **successful ACKs = MAX_PACKETS** or no.of retransmissions for any packet > **10**

Output format

Upon termination, SR sender needs to print below metrics:

- PACKET_GEN_RATE
- PACKET_LENGTH
- ReTransmission Ratio
- Average RTT Value for ALL Acked Packets
- In case of DEBUG mode :

```
Seq #:  Time Generated: xx:yy RTT: zz Number of Attempts: aa
```

Implementing S-R ARQ (Receiver)

- Listens on **UDP socket** port for reading packets from sender
- **Random decision** on corrupt or Error-free packet so that deciding whether to drop or not
- Takes PACKET_ERROR_RATE as CLP infering to **packet drop probability**
- Reads the packet and extracts the sequence number to be used in **ACKs**
- Buffering **out-of-order** packets
- **Discards** packets in case of FULL Receive Buffer
- **Sorting** w.r.t Sequence Numbers

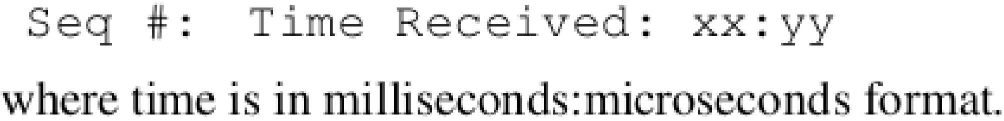
Assumptions for SR-Receiver

- ACK packets are **NOT dropped** and are always assumed to be **delivered** to the sender
- Terminates only after acknowledging MAX_PACKETS
- SWS $=$ RWS
- Takes the role of **UDP server** and keeps listening until termination

Output format

Upon termination, SR receiver needs to print :

- DEBUG mode :(-d flag Set in Command Line)



```
Seq #:  Time Received: xx:yy
```

where time is in milliseconds:microseconds format.

UDP Sockets API

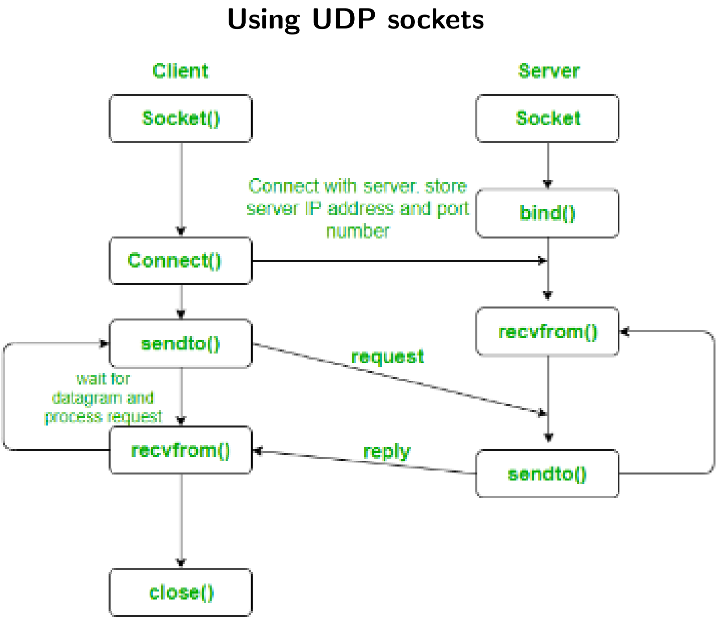


Figure: sendto() and recvfrom() in UDP sockets

https://www.geeksforgeeks.org/udp-client-server-using-connect-c-implementation/

Implementing GO-BACK-N ARQ (Sender)

- Packet Generation with packet length from CLP, rate of generating packets in periodic time intervals **(packets/sec)**
- Maximum size of sender transmit buffer MAX_BUFFER_SIZE in terms of number of packets
- Given an **n-bit** sequence number, the maximum sender window size will be $2^n - 1$
- Next packet can be sent when **outstanding**(unacknowledged) packets not exceed window size
- **ACKs(Cumulative)** receive event needs to update local state variables, cancel T.Os, removing packet from transmit buffer
- For each packet, **RTT** calculation, **Avg.RTT update** for acknowledged packets so far
- Time Expiry/Time Out event results in retransmitting **all packets** from the first unacknowledged packet

Assumptions for GO-BACK-N-Sender

- Each packet generated is of **same length**
- First byte(s) of the packet contains its **unique** sequence number
- Time out timers are assumed to be **100 millisec** for initial **10** packets and all subsequent packets T.O timers set to $2 \times \text{RTT}_{\text{Avg}}$
- Termination Criteria is either no.of **successful ACKs = MAX_PACKETS** or no.of retransmissions for any packet $>$ **5**

Output format

Upon termination, GBN sender needs to print below metrics:

- PACKET_GEN_RATE
- PACKET_LENGTH
- ReTransmission Ratio
- Average RTT Value for ALL Acked Packets
- In case of DEBUG mode :

```
Seq #:  Time Generated: xx:yy RTT: zz Number of Attempts: aa
```

Implementing Go-Back-N ARQ (Receiver)

- Listens on **UDP** socket port for reading packets from sender
- **Random decision** on Corrupt or Error-free packet so that deciding whether to drop it or not
- Takes PACKET_ERROR_RATE as CLP infering to **packet drop probability**
- Reads the packet and extracts the sequence number to be used in **ACKs**
- If sequence number matches the **NEXT EXPECTED** sequence number, it transmits an ACK to sender and updates local state variables
- **No Buffering** thereby out-of-order packets will be discarded

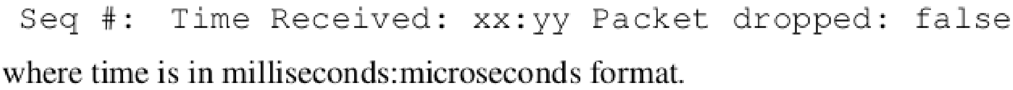
Assumptions for GBN-Receiver

- ACK packets are **NOT dropped** and are always assumed to be delivered to the sender
- Terminates only after acknowledging **MAX_PACKETS**
- RWS is always **1**
- **No buffer** space and no sorting

Output format

Upon termination, GBN receiver needs to print :

- DEBUG mode :(-d flag Set in Command Line)



```
Seq #: Time Received: xx:yy Packet dropped: false
```

where time is in milliseconds:microseconds format.

Hints (Optional)

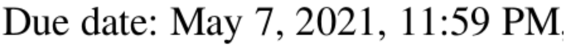
1. Use of Built-in Timer functions
2. Use of Locks/Mutex, Condition Variables on Shared Resource Updates
3. Use of Built-in Thread Libraries to accomplish Multi-Threading
4. Use of Built-in Random Generator functions
5. Maintaining appropriate control variables/pointers for managing Sliding Windows
6. Using proper Data Structures

Technical Report

- **Variable Parameters**
  1. PACKET_GEN_RATE
  2. RANDOM_DROP_PROB
  3. PACKET_LENGTH
- **Metrics of Importance to compare**
  1. Average RTT
  2. Re-transmission Ratio

Grading Scheme

- Platform : Linux, C/C++/Java
- GO-BACK-N Implementation : [**30 Pts**]
- SELECTIVE REPEAT Implementation : [**30 Pts**]
- REPORT : [**20 Pts**]
- VIVA : [**20 Pts**]
- No readme/make file : [**-10 Pts**]



Due date: May 7, 2021, 11:59 PM,

Thank You

References

[1] https://www.programminglogic.com/sockets-programming-in-c-using-udp-datagrams/

[2] https://www.javatpoint.com/sliding-window-protocol

[3] https://www.brainkart.com/article/Noisy-Channels-Protocol_13447/

[4] https://www.geeksforgeeks.org/udp-client-server-using-connect-c-implementation/

[5] https://www.nielit.gov.in/gorakhpur/sites/default/files/Gorakhpur/alevel_2_dcn_07apr_PT.pdf

[6] https://www.cs.dartmouth.edu/$\sim$ *campbell*/*cs*60/*socketprogramming*.*html*