Networks Lab
Assignment 2

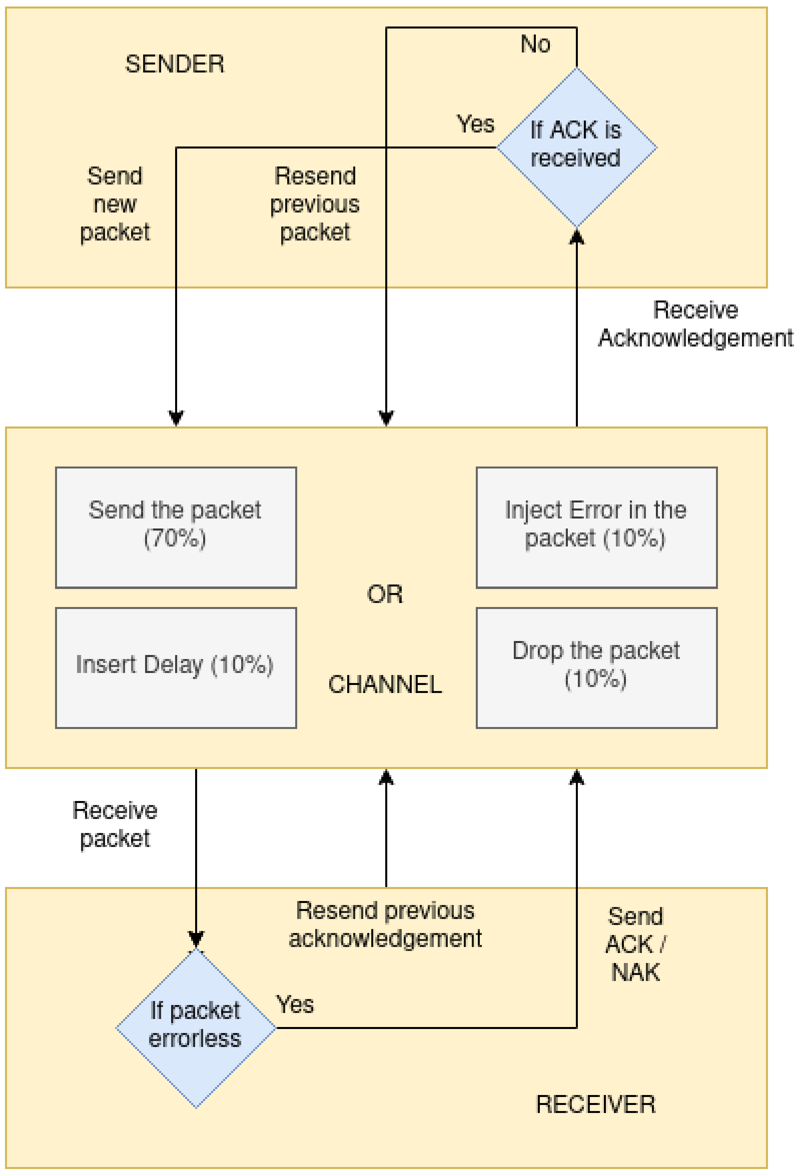
Name: **Koustav Dhar** **Class:** **BCSE UG-III** **Group:** **A1** **Roll:** **001910501022**

Problem Statement:

Implement three data link layer protocols, Stop and Wait, Go Back N Sliding Window, and Selective Repeat Sliding Window for flow control.

Design:

- Sender:
    - Read data from a file.
    - Convert it into a packet following IEEE 802.3 frame format.
    - Start three simultaneous threads.
    - First, to send the packet to the channel(server).
    - Second, to receive the acknowledgments if any.
    - Third, to resend the previous data packets, if positive acknowledgment isn't received.
- Channel:
    - Receives data packets from sender(s).
    - Processes the packets, to simulate a network with noise in it.
        - Sends the packet with a 70% chance.
        - Introduce noise in the data and send erroneous data with a 10% chance.
        - Introduce a delay in sending the packet with a 10% chance.
        - Drop the packet with a 10% chance.
    - Simultaneously receives acknowledgments from the receiver(s), passes them to the sender.
- Receiver:
    - Receives data packets from the channel.
    - Checks if the data is erroneous or not.
    - If data is error-free, then send positive acknowledgment, else drop it.
    - In sliding window protocols, if the packet ranges aren't valid at any moment, send a negative acknowledgment.
    - If the sequence number is still stuck at the last packet, resend the acknowledgment for the previous packet simultaneously.

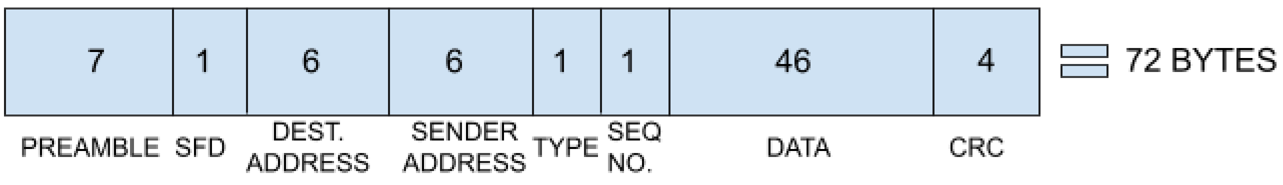


Flow Diagram of the Generalised Network

There can be more than one sender or receiver since it is sort of a multithreaded chat system, but for simplicity, it is demonstrated using one sender and receiver.

Implementation:

We have used the IEEE 802.3 Ethernet Frame Format for our packets here. Size of a frame is 72 bytes.



7	1	6	6	1	1	46	4	72 BYTES
PREAMBLE	SFD	DEST. ADDRESS	SENDER ADDRESS	TYPE	SEQ NO.	DATA	CRC	

IEEE 802.3 ETHERNET FRAME FORMAT

- **PREAMBLE –** Ethernet frame starts with 7-Bytes Preamble. This is a pattern of alternative 0's and 1's which indicates starting of the frame and allows sender and receiver to establish bit synchronization. Initially, PRE (Preamble) was introduced to allow for the loss of a few bits due to signal delays. But today's high-speed Ethernet doesn't need Preamble to protect the frame bits. PRE (Preamble) indicates to the receiver that the frame is coming and allows the receiver to lock onto the data stream before the actual frame begins.

- **Start of frame delimiter (SFD) –** This is a 1-Byte field that is always set to 10101011. SFD indicates that upcoming bits are starting of the frame, which is the destination address. Sometimes SFD is considered the part of PRE, this is the reason Preamble is described as 8 Bytes in many places. The SFD warns station or stations that this is the last chance for synchronization.
- **Destination Address –** This is a 6-Byte field that contains the port address of the destination socket.
- **Source Address –** This is a 6-Byte field that contains the port address of the sender socket.
- **Type -** This is a 1-Byte field that contains the type of the packet, whether it is a data packet or an acknowledgment.
- **Sequence No. -** This is a 1-Byte field that contains the sequence no. of the current frame.
- **Data –** This is the place where actual data is inserted, also known as **Payload**. Both IP header and data will be inserted here if Internet Protocol is used over Ethernet. The maximum data present may be as long as 1500 Bytes. In case data length is less than minimum length i.e. 46 bytes, then padding 0's is added to meet the minimum possible length.
- **Cyclic Redundancy Check (CRC) –** CRC is a 4 Byte field. This field contains a 32-bits hash code of data, which is generated over the Destination Address, Source Address, Length, and Data field. If the checksum computed by destination is not the same as the sent checksum value, the data received is corrupted.

Noise in the channel is introduced as follows:

```
# Function to select if original packet or tainted packet or no packet will
be sent
def process_packet(packet:str):
    # Randomly generate what would be done
    flag=random.randint(0,100)

    # If flag is 0-70, original packet will be sent
    # If flag is 71-80, taint the packet and send
    # If flag is 81-99, drop the packets
    if(flag >= 0 and flag < 100):
        return packet
    elif(flag >= 70 and flag < 80):
        return ErrorInsertion.Error.injectError(packet)
    elif(flag >= 80 and flag < 90):
        time.sleep(0.5)
        return packet
    else:
        return ''
```

Test Cases:

We have generated random ASCII strings without whitespaces, of about 4×10^5 lengths, and performed all the tests on them.

testgenerator.py

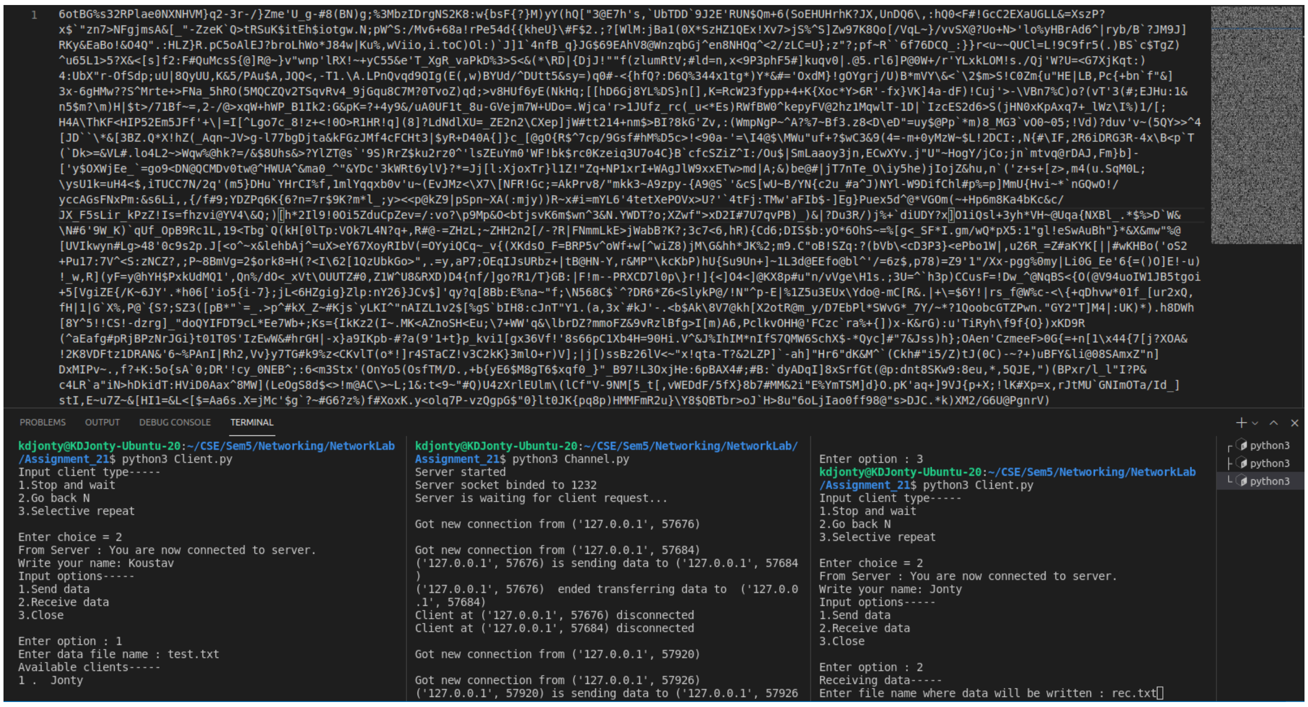
```
import string
import random
import re
```

```
n = 13 * (32 ** 3)

res = ''.join(random.choices(re.sub(r'\s+', '',string.printable), k = n))

file = open("test.txt", "w")
file.write(res)
file.close()
```

Some of the test cases along with terminal output screens are shown below.



```
6otBG%s32RPlae0NXNHVM}q2-3r-/}Zme'U_g-#8(BN)g;%3MbzIDrgNS2K8:w{bsF{?}M)yY(hQ["3@E7h's,`UbTDD`9J2E'RUN$Qm+6(SoEHUHrhK?JX,UnDQ6\,:hQ0<F#!GcC2EXaUGLL&=XszP?
x$`"zn7>NFgjmsA&[_"-ZzeK`Q>tRSuK$itEh$iotgw.N;pW^S:/Mv6+68a!rPe54d{{kheU}\#F$2.;?[WlM:jBa1(0X*SzHZ1QEx!Xv7>jS%^S]Zw97K8Qo[/VqL~}/vvSX@?Uo+N>'lo%yHBrAd6^|ryb/B`?JM9J]
RKy&EaBo!&O4Q".:HLZ}R.pC5oAlEJ?broLhWo*J84w|Ku%,wViio,i.toC)Ol:)`J]1`4nfB_q}JG$69EAhV8@WnzqbGj^en8NHQq^<2/zLC=U};z"?;pf~R``6f76DCQ_:}}r<u~~QUCl=L!9C9fr5(.)BS`c$TgZ)
^u65L1>5?X&<[s]f2:F#QuMcsS{@]R@~}v"wnp'lRX!~+yC55&e'T_XgR_vaPkD%3>S<&(*\RD|{DjJ!""f(zlumRtV;#ld=n,x<9P3phF5#]kuqv0|.@5.rl6]P@0W+/r'YLxkLOM!s./Qj'W?U=<G7XjKqt:)
4:UbX"r-OfSdp;uU|8QyUU,K&5/PAu$A,JQQ<,-T1.\A.LPnQvqd9QIg(E(,w)BYUd/^DUtt5&sy=)q0#-<{hfQ?:D6Q%344x1tg*)Y*&#='OxdM}!gOYgrj/U)B*mVY\&<`\2$m>S!C0Zm{u"HE|LB,Pc{+bn`f"&]
3x-6gHMw??S^Mrte+>FNa_5hRO(5MQCZQv2TSqvRv4_9jGqu8C7M?0TvoZ)qd;>v8HUf6yE(NkHq;[[hD6Gj8YL%DS}n[],K=RcW23fypp+4+K{Xoc*Y>6R'-fx}VK]4a-dF)!Cuj'>-\VBn7%C)o?(vT'3(#;EJHu:1&
n5$m?\m)H|$t>/71Bf~=,2-/@>xqW+hWP_B1Ik2:G&pK=?+4y9&/uA0UF1t_8u-GVejm7W+UDo=.Wjca'r>1JUfz_rc(_u<*Es)RWfBW0^kepyFV@2hz1MqwlT-1D|`IzcES2d6>S(jHN0xKpAxq7+_lWz\I%)1/[;
H4A\ThKF<HIP52Em5JFf'+\|=I[^Lgo7c_8!z+<!0O>R1HR!q](8]?LdNdlXU=_ZE2n2\CXep]jW#tt214+nm$>BI?8kG'Zv,:(WmpNgP~^A?%7~Bf3.z8<D\eD"=uy$@Pp`*m)8_MG3`vO0~05;!Vd)?duv'v~(5QY>>^4
[JD``\*&[3BZ.Q*X!hZ(_Aqn~JV>g-l77bgDjta&kFGzJMf4cFCHt3|$yR+D40A{]}c_[@gO{R$^7cp/9Gsf#hM%D5c>!<90a-'=\I4@$\MWu"uf+?$wC3&9(4=-m+0yMzW~$L!2DCI:,N{#\IF,2R6iDRG3R-4x\B<p`T
(`Dk>=&VL#.lo4L2~>Wqw%@hk?=/&$8Uhs&>?YlZT@s`'9S)RrZ$ku2rz0^'lsZEuYm0'WF!bk$rc0Kzeiq3U7o4C}B`cfcSZiZ^I:/Ou$|SmLaaoy3jn,ECwXYv.j"U"~HogY/jCo;jn`mtvq@rDAJ,Fm}b]-
['y$OXWjEe_`=go9<DN@QCMDv0tw@^HWUA^&ma0_^"&YDc'3kWRt6ylV}?*=Jj[l:XjoxTr}l1Z!"Zq+NP1xrI+WAgJlW9xxETw>md|A;&)be@#|jT7nTe_O\iy5he)jIojZ&hu,n`('z+s+[z>,m4(u.SqM0L;
\ysU1k=uH4<$,iTUCC7N/2q'(m5}DHu`YHrCI%f,1mlYqqxb0v'u~(EvJMz<\X7\[NFR!Gc;=AkPrv8/"mkk3~A9zpy-{A9@S`'&cS[wU~B/YN{c2u_#a^J)NYl-W9DifChl#p%=p]MmU{Hvi~*`nGQwO!/
yccAGsFNxPm:&s6Li,,{/f#9;YDZPq6K{6?n=7r$9K?m*l_;y><<p@kZ9|pSpn~XA(:mjy))R~x#i=mYL6'4tetXePOVx>U?'`4tFj:TMw'aFIb$-]Eg}Puex5d^@*VGOm(~+Hp6m8Ka4bKc&c/
JX_F5sLir_kPzZ!Is=fhzvi@YV4\&Q;)[h*2Il9!0Oi5ZduCpZev=/:vo?\p9Mp&O<btjsvK6m$wn^3&N.YWDT?o;XZwf">xD2I#7U7qvPB)_)&|?Du3R/)j%+`diUDY?x]O1iQsl+3yh*VH~@Uqa{NXBl_.*$%>D`W&
\N#6'9W_K)`qUf_OpB9Rc1L,19<Tbg`Q(kH[0lTp:VOk7L4N?q+,R#@-=ZHzL;~ZHH2n2[/-?R|FNmmLkE>jWabB?K?;3c7<6,hR){Cd6;DIS$b:yO*6OhS~=%[g<_SF*I.gm/wQ*pX5:1"gl!eSwAuBh"}*&X&mw"%@
[UVIkwyn#Lg>48'0c9s2p.J[<o^~x&lehbAj^=uX>eY67XoyRIbV(=OYyiQCq~_v{(XKdsO_F=BRP5v^oWf+w[^wiZ8)jM\G&hh*JK%2;m9.C"oB!SZq:?(bVb\<cD3P3}<ePbo1W|,u26R_=Z#aKYK[||#wKHBo('oS2
+Pu17:7V^<S:zNCZ?,;P~8BmVg=2$ork8=H(?<I\62[1QzUbkGo>",.=y,aP7;OEqIJsURbz+|tB@HN-Y,r&MP"\kcKbP)hU{Su9Un+]~1L3d@EEfo@bl^'/=6z$,p78)=Z9'1"/Xx-pgg%0my|Li0G_Ee'6{=()O]E!-u)
!_w,R](yF=y@hYH$PxkUdMQ1',Qn%/dO<_xVt\OUUTZ#0,Z1W^U8&RXD)D4{nf/]go?R1/T}GB:|F!m--PRXCD7l0p\}r!]{<]O4<]@KX8p#u"n/vVge\H1s.;3U=^`h3p)CCusF=!Dw_^@NqBS<{O(@V94uoIW1JB5tgoi
+5[VgiZE{/K~6JY'.*h06['io5{i-7};jL<6HZgig}Zlp:nY26}JCv$]'qy?q[8Bb:E%na~"f;\N568C$`^?DR6*Z6<SlykP@/!N"^p-E|%1Z5u3EUx\Ydo@-mC[R&.|+\=$6Y!|rs_f@W%c-<\{+qDhvw*01f_[ur2xQ,
fH|1|G`X%,P@`{S?;SZ3([pB*"`=_.>p^#kX_Z~#Kjs`yLKI^"nAIZL1v2$[%gS`bIH8:cJnT"Y1.(a,3x`#kJ'-.<b$Ak\8V7@kh[X2otR@m_y/D7EbPl*SWvG*_7Y/~*?1QoobcGTZPwn."GY2"T]M4|:UK)*).h8DWh
[8Y^5!!CS!-dzrg]_"doQYIFDT9cL*Ee7Wb+;Ks={IkKz2(I~.MK<AZnoSH<Eu;\7+WW'q&\lbrDZ?mmoFZ&9vRzlBfg>I[m)A6,PclkvOHH@'FCzc`ra%+{])x-K&rG):u'TiRyh\f9f{O})xKD9R
(^aEafg#pRjBPzNrJGi}t01T0S'IzEwW&#hrGH|-x}a9IKpb-#?a(9'1+t}p_kvi1[gx36Vf!'8s66pC1Xb4H=90Hi.V^&J%IhIM*nIfS7QMW6SchX$-*Qyc]#"7&Jss)h};OAen'CzmeeF>0G{=+n[1\x44{7[j?XOA&
!2K8VDFtz1DRAN&'6~%PAnI|Rh2,Vv}y7TG#k9%z<CKvlT(o*!]r4STaCZ!v3C2kK}3ml0+r)V];|j[)ssBz26lV<~"x!qta-T?&2LZP]`-ah]"Hr6"dK&M^`(Ckh#"i5/Z)tJ(0C)-~?+)uBFY&li@08SAmxZ"n]
DxMIPv~.,f?+K:5o{sA`0;DR'!cy_0NEB^;:6<m3Stx'(OnYo5(OsfTM/D.,+b{yE6$M8gT6$xqf0_}"_B97!L3OxjHe:6pBAX4#;#B:`dyADqI]8xSrfGt(@p:dnt8SKw9:8eu,*,5QJE,")(BPxr/l_l"I?P&
c4LR`a"iN>hDkidT:HViD0Aax^8MW](LeOgS8d$<>!m@AC\>~L;1&:t<9~"#Q)U4zXrlEUlm\(lCf"V-9NM[5_t[,vWEDdF/5fX}8b7#MM&2i"E%YmTSM]d}O.pK'aq+]9VJ{p+X;!lK#Xp=x,rJtMU`GNImOTa/Id_]
stI,E~u7Z~&[HI1=&L<[$=Aa6s.X=jMc'$g`?~#G6?z%)f#XoxK.y<olq7P-vzQgpG$"0}lt0JK{pq8p)HMMFmR2u}\Y8$QBTbr>oJ`H>8u"6oLjIao0ff98@"s>DJC.*k)XM2/G6U@PgnrV)
```

```
kdjonty@KDJonty-Ubuntu-20:~/CSE/Sem5/Networking/NetworkLab
/Assignment_21$ python3 Client.py
Input client type-----
1.Stop and wait
2.Go back N
3.Selective repeat

Enter choice = 2
From Server : You are now connected to server.
Write your name: Koustav
Input options-----
1.Send data
2.Receive data
3.Close

Enter option : 1
Enter data file name : test.txt
Available clients-----
1 .  Jonty
```

```
kdjonty@KDJonty-Ubuntu-20:~/CSE/Sem5/Networking/NetworkLab/
Assignment_21$ python3 Channel.py
Server started
Server socket binded to 1232
Server is waiting for client request...

Got new connection from ('127.0.0.1', 57676)

Got new connection from ('127.0.0.1', 57684)
('127.0.0.1', 57676) is sending data to ('127.0.0.1', 57684
)
('127.0.0.1', 57676)  ended transferring data to  ('127.0.0
.1', 57684)
Client at ('127.0.0.1', 57676) disconnected
Client at ('127.0.0.1', 57684) disconnected

Got new connection from ('127.0.0.1', 57920)

Got new connection from ('127.0.0.1', 57926)
('127.0.0.1', 57920) is sending data to ('127.0.0.1', 57926
```

```
Enter option : 3
kdjonty@KDJonty-Ubuntu-20:~/CSE/Sem5/Networking/NetworkLab
/Assignment_21$ python3 Client.py
Input client type-----
1.Stop and wait
2.Go back N
3.Selective repeat

Enter choice = 2
From Server : You are now connected to server.
Write your name: Jonty
Input options-----
1.Send data
2.Receive data
3.Close

Enter option : 2
Receiving data-----
Enter file name where data will be written : rec.txt
```

<u>Results:</u>

We have used throughput, efficiency, and the average successful transmission time of a packet as the parameters to compare the 3 protocols, i.e. Stop and Wait ARQ, Go Back N ARQ, and Selective Repeat ARQ.

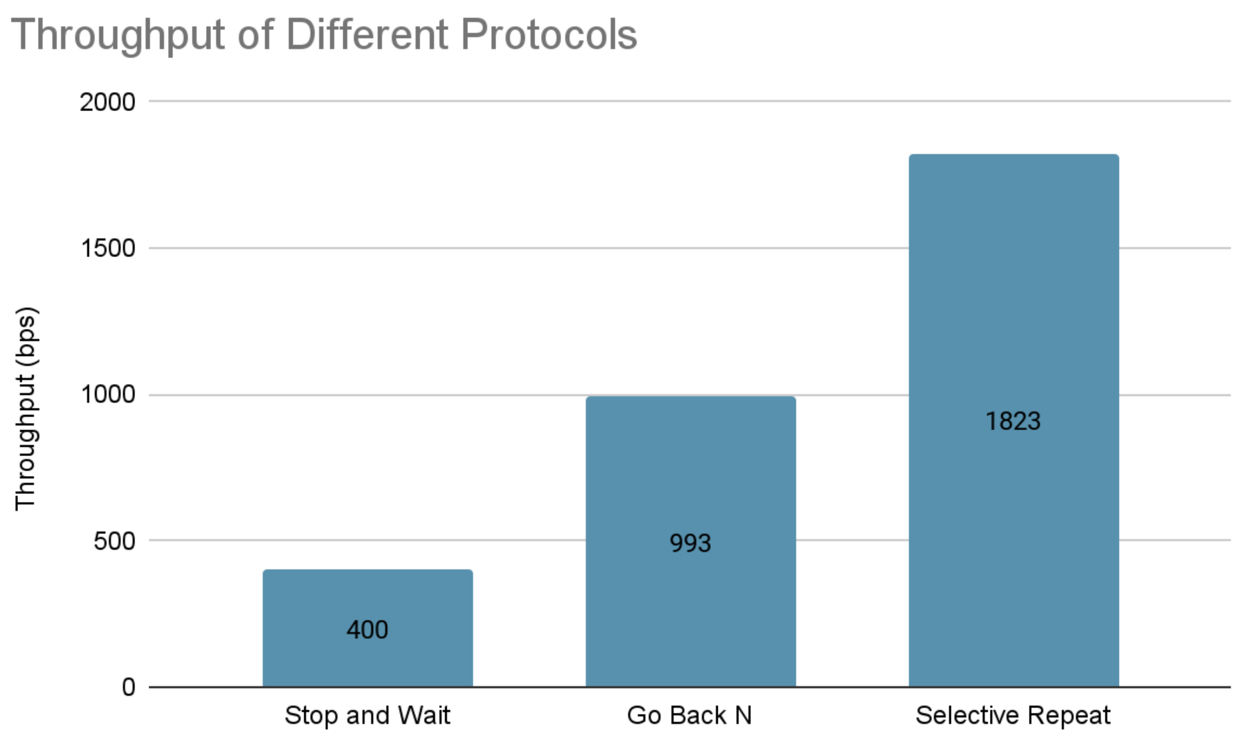
$$Throughput \;=\; \frac{Effective\ no.\ of\ bits\ sent\ (no.\ of\ packets \times packet\ size\ in\ bits)}{Total\ Time\ taken}$$

$$Efficiency \;=\; \frac{Throughput}{Bandwidth}$$

$$Average\ Successful\ Transmission\ Time\ of\ a\ Packet \;=\; \frac{Total\ Time\ taken}{Effective\ no.\ of\ packets}$$

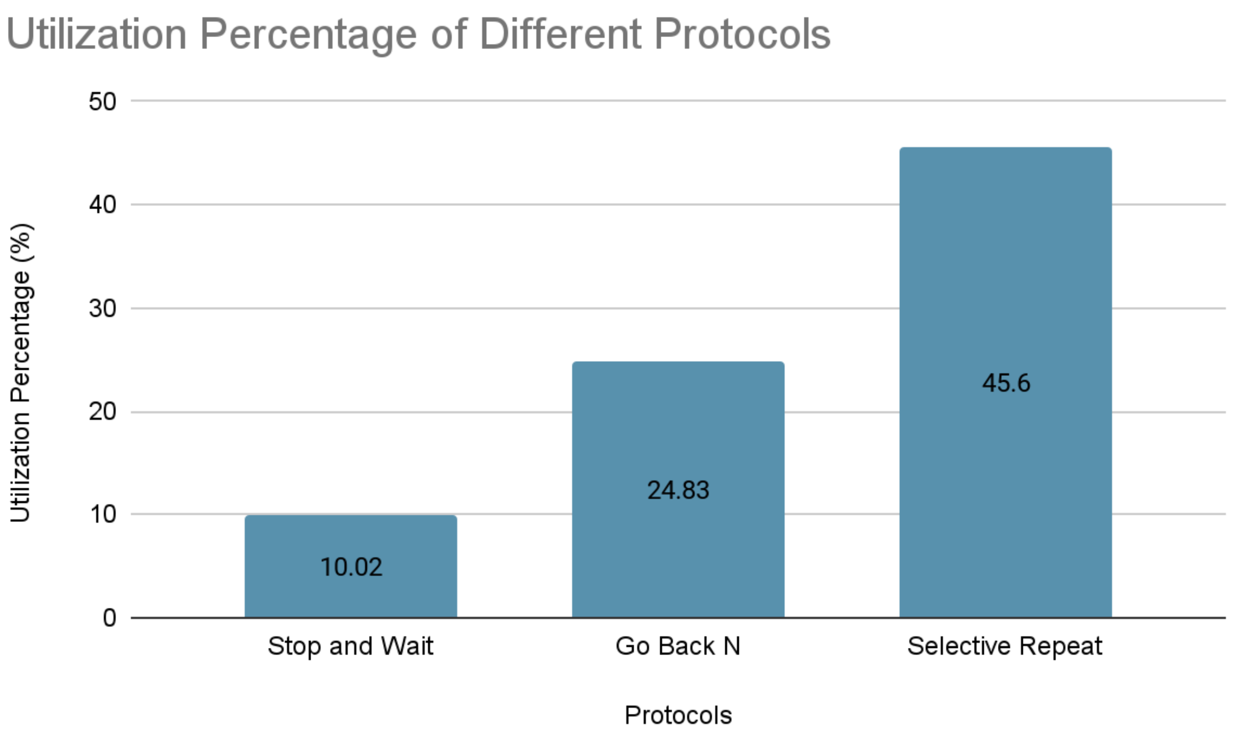
Receiver Throughput

Packets / Protocols	Stop and Wait ARQ	Go Back N ARQ	Selective Repeat ARQ
23552 (512)	383 bps	1059 bps	1924 bps
5888 (128)	375 bps	799 bps	2231 bps
2752 (64)	443 bps	1121 bps	1315 bps
Mean throughput	400 bps	993 bps	1823 bps



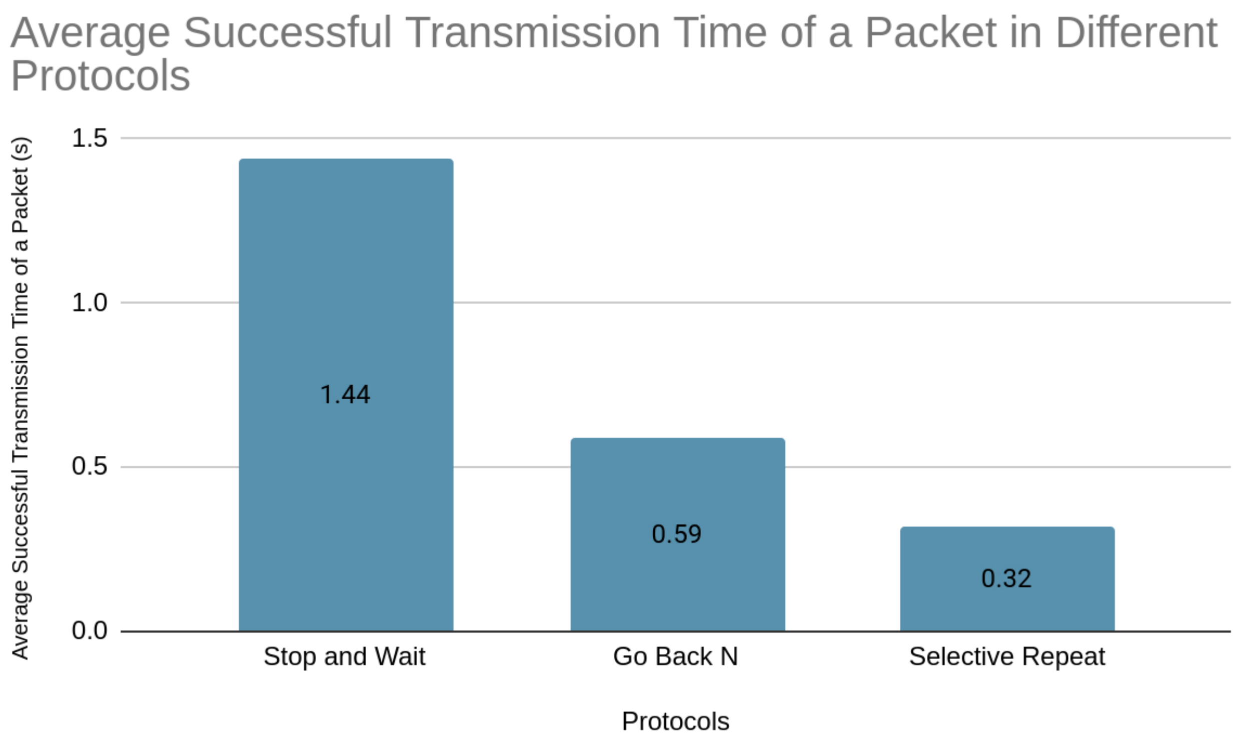
Efficiency

Packets / Protocols	Stop and Wait ARQ	Go Back N ARQ	Selective Repeat ARQ
23552 (512)	9.60%	26.49%	48.12%
5888 (128)	9.39%	19.98%	55.78%
2752 (64)	11.08%	28.04%	32.90%
Mean %	10.02%	24.83%	45.60%



Average Successful Transmission Time of a Packet

Packets / Protocols	Stop and Wait ARQ	Go Back N ARQ	Selective Repeat ARQ
23552 (512)	1.50 s	0.54 s	0.29 s
5888 (128)	1.53 s	0.72 s	0.25 s
2752 (64)	1.30 s	0.51 s	0.43 s
Mean transmission time	1.44 s	0.59 s	0.32 s



<u>Analysis:</u>

From the tables and charts presented in the Results section, we can observe how the data is being transmitted using different protocols. In terms of all the parameters considered, throughput, utilization percentage, and average successful transmission time of a packet, the Selective Repeat protocol has done the job most efficiently, followed by Go Back N protocol and Stop and Wait protocol.
We've considered three types of data sizes for our transmission to observe how the efficiency changes with the size of data and the number of packets actually being sent.
We've not considered **Round Trip Time (RTT)** as a comparison parameter because round trip time should be the same for all the three protocols as it's a property of the network and not the protocol. But for observation purposes, we have tried to find the round trip time too. We can fairly say that in a noiseless channel the successful transmission time will be the round trip time, hence we've simulated one run of the noiseless channel, and we've found the round trip time for this network to be 0.06 s/packet. This run was performed on data of 512 packets.
We've not considered **Bandwidth-Delay Product** as a comparison parameter either because it is too a property of the channel, and doesn't depend on the protocol.

Improvements:

- ➢ This would have been more efficient if it was implemented in a language closer to the system such as C/C++.