Prioritized Retransmission: A TCP Flow Control Mechanism

Avanish Kumar, Dharamdas Kumhar

Abstract: ***Now-a-days, efforts have been made in the research of transmission control protocols to improve the performance of the flow control mechanism. Internet communication and services daily increase the variety and quantity of their capacity and needs. Therefore the flow control mechanism will have to consider valuable for traffic control, especially on high-speed networks. Initially there are some challenges, for instance, loss of packet, processing capacity, performance, buffer overflows and deadlocks with which daily traffic is confronted. This paper analyzes and reviewed the strengths and weaknesses of the different flow control mechanisms used in TCP. To overcome the weaknesses of these flow controls, we suggest a priority retransmission mechanism. Here we have priority on the Negative Acknowledgment (NACK), we resend the package on the basis of the minimum sequence number of the NACK. In buffer of priority retransmission Automatic Repeat request (ARQ) mechanism, the packet is released to the communication link in a First in First out (FIFO) manner. That is why the priority retransmission ARQ gives the optimum performance.***

Keywords: automatic repeat request, communication network, flow control, priority retransmission, sliding window.

I. INTRODUCTION

When we interact with anyone, we have split our knowledge. It is exchanges of information among two machines through some form of communication path such as a wireless or wire cables. In modern communication networks, flow and congestion control are very important for efficient and fair utilization of resources as well as avoiding congestion collapse. At present, Internet or email basically, the Transmission Control Protocol gives reliable transmission between end-to-end nodes. Most of the present distributed systems including communication networks extensively employ the Transmission Control Protocol (TCP) for reliable transmission among different nodes. Since TCP is renovated for actual delivery rather than be times delivery, TCP on wireless devices may frequently incur significant end-to-end delays as long as waiting for retransmission of missing or damaged packets and rearrangements of out-of-order packets on wireless communication links [1]. Transport protocols like the transmission control protocol is reliable are tuned to carry out well in classic wire line networks wherever packet losses occur principally as a result of congestion. However, computer networks with wireless links as well bear f3530rom significant packets impairment because of bit errors and handoffs. TCP make a response to all the packet losses by citing congestion control algorithms as well as avoidance algorithms and this outcome in decay end-to-end performance in a lossy and wireless node. To realize a reliable end-to-end interconnection oriented transmission, TCP usage positive or negative acknowledgments and retransmission the information. A flow control mechanisms generally use implicit and explicit knowledge of the network conditions transmitted by the receiver to regulate the transmission rate or the size of the transmitter window. The TCP transfer rate is controlled by the window. In case of a loss of packets, the window is halved and the window is incremented after confirmation of the receipt of all the packets belonging to the current window [2].

The flow control mechanism can be easily implemented and proven to be effective. However, the sender receives the feedback information, identifies congestion and receives at least one round-trip time to make the necessary changes. To determine when a packet is lost and must be forwarded, the sender uses a forwarding timer. It is clear that an accurate RTT estimate is required to allow retransmission of lost packets on time and redelivery of packets with unexpected delays in the network. This process can take place several times in the current zone on the Internet. At the same time, congestion can be caused or exacerbated, resulting in failure of the packets and improper use of network resources [3]. The most commonly used flow control protocol was the DECbit scheme [4] that uses explicit network signals and slow start scheme [5] using packet harm as the implicit congestion signal. Since the introduction of these systems and, many other systems have been introduced to further improve performance. These flow control methods should not be developed properly because the network reaches the gigabit limit and does not work properly. So that we will, however, proposing a new flow control policy.

Manuscript published on November 30, 2019.
* Correspondence Author
Dr. Avanish Kumar*, Chairman (CSTT) & Director (CHD), Ministry of Human Resource Development, Govt. Of India, New Delhi, India. Email: dravanishkumar@yahoo.com
Dharamdas Kumhar, Department of Mathematical Sciences & Computer Applications, Bundelkhand University, Jhansi (U.P.), India. Email: prajapatibu@yahoo.co.in

II. FLOW CONTROL

There are different types of protocols of flow control in transport level. Both source–to-destination and entry-to-exit control level are termed end-to-end (flow) control despite that they are applied at different layers. End-to-end flow control cannot guarantee that resources are available at intermediate nodes, only at the destination. Node level flow control is very important because the transmission device can transmit information much faster than the receiving device can receive and process. This can happen if the traffic load of the receiver device is higher than the traffic load of the transmission devices or if the processing capacity of the target device is lower than the processing capacity of the transmission devices. Therefore, in the case of data transmission between the transmitter and the receiver, the slow receiver does not support the speed at which the information is transmitted by a very fast transmitter. In such a condition, there is a demand for flow control so that a very high speed sender does not overwhelm a low speed receiver [6]. Flow control is a mechanism that allows transceivers with different speed functions to share with each other. Flow control ensures that the sending station (for example, the server with the maximum power of the processor) does not overload the receiving station (especially the desktop system) with low processor power. Therefore flow control methods refer to the set of rules used to regulate the quantities of data the sender can send before waiting for acknowledgment. Flow control, however, plays an important role in the performance of communication networks. The main features and objectives of flow control in packet networks are Minimize packet loss and latency avoidance, Avoid Deadlock, Prevent flow loss and reduce efficiency due to overload, Distribution of resources between the competing customer and rapid comparison between the network and its users [7].

In a communication network for effective transformations, we make confident the highly reliable and secure data communication. ARQ is being used as a control protocol for flow control. ARQ use timeout retransmission, antithetical and retransmission mechanism, positive acknowledgment and inaccuracy recognition to resolve the errors. ARQ is functional mutually in wired and wireless networks and it provides this error detection and revival founded on feedback messages and retransmission [8].

III. TYPES OF FLOW CONTROL APPROACHES

The network layer is responsible for carrying the packet from the transport layer and sends to the data link layer and packet encapsulate the frame. Reliability can be improved by adding error control to the transport layer, such that discarding the packet and resending the packet, its process will continue awaiting the lost packet accomplish your target [9]. There are normally key functions for error control protocols are identify the error for incoming packets, if an error is not detected, then send a positive acknowledgment (ACK), if an error is detected, then send a negative acknowledgment by the recipient (NACK), and set the timer for lost packet.

Hence there are two popular approaches built-up for flow control to be precise Stop-and-Wait (SAW) ARQ and Sliding Window ARQ as given below fig. 1.1. With the ARQ stop and wait for flow control, the remote host must identify all packets before the next packet is sent. With the sliding window ARQ used by TCP can send several data packets simultaneously so that the network bandwidth can be used more effectively [10].

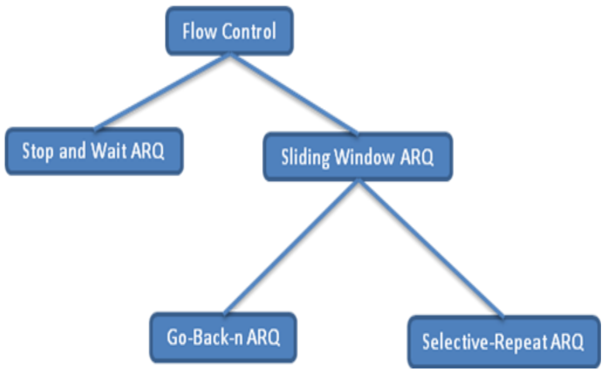


Fig. 1.1. Approaches of Flow Control

A. Stop-and Wait ARQ

This is the simplest variety of flow control. Here, the dispatcher sends one data packet at a time and stay for an acknowledgment earlier than sending the next packet to the recipient. After receiving the data packet, the receiver indicates its willingness to agree to one more packets by sending back a positive acknowledgment (ACK), packet acknowledging the packet just received. Once the data packet is accepted the receiver window slide increase by one bit and the receiver send the ACK. If sending received packet as a negative acknowledgment (NACK), then the sender will resend the packet. In a position whereby a receiver expected a packet correctly and the receiver sends an acknowledgment for that to the dispatcher but the acknowledgment is missing or reaches after a timeout, then the dispatcher sends again the same packet and the recipient received a duplicate as a previous packet. The Timer is also utilized in case packet or acknowledgment is missing for the duration of the transmission. Here, behind transmitting a packet, the dispatcher waits for the ACK or NACK until for a few precise times. The channel utilization and working of a stop-and-wait ARQ is given in fig.1.2.

We know that propagation delay (a) is the time it required for a packet to travel from sender to receiver. Propagation time is the time for a sender to release all the bits of the packet to the destination. In stop-and-wait ARQ the propagation delay is in each direction; result in two of propagation time. So the total for a packet is transmission time and two's of propagation time. The flow control reduces utilization. Utilization defines as the ratio of throughput and network capacity. The throughput is the quantity of useful data that turn up at the other end of the link while the network capacity is a maximum possible amount of data that could be sent across the link.

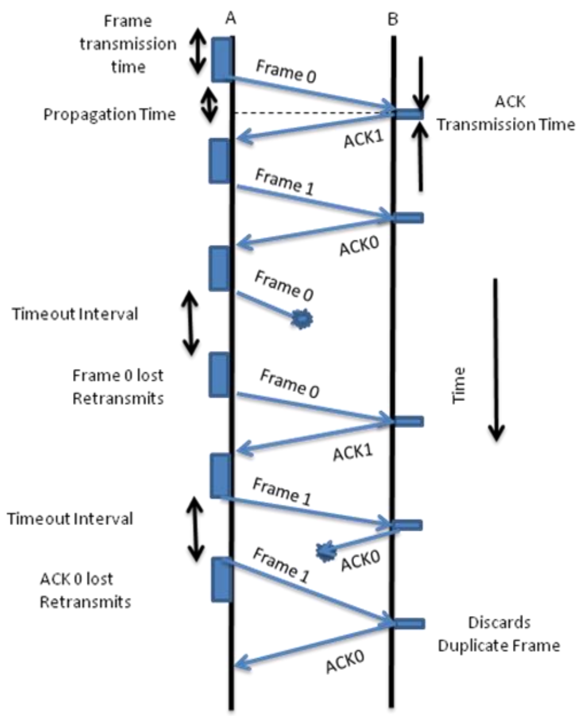


Fig. 1.2. Stop and Wait ARQ Mechanism

Thus channel utilization without errors is can be expressed as,

$$U = \frac{1}{(1+2a)} \qquad (1)$$

and channel utilization with errors is can be expressed as,

$$U = \frac{1-p}{(1+2a)} \qquad (2)$$

Where p is the probability of a packet being received incorrectly, hence (1-p) is the probability of successfully received packet. Where a is defined as the ratio of propagation time and transmission time. Hence from “(1)”, we can say that when a is below one then the channel is inefficiently utilized and when a is greater than one, a channel is always underutilized. Therefore to enhance the channel utilization, we can make use of the sliding window ARQ [11]. The major disadvantage in this is that just one packet can be transmitted at a time, this leads to ineffectiveness if propagation delay is a large amount than the transmission delay. The dispatcher wants to stay for the ACK after each packet is transmitted. The recipient has one bit received window size. There is no pipelining in stop-and-wait ARQ.

B. Sliding Window ARQ

The problems of stop and wait ARQ can be overcome by using the sliding Window ARQ. In the sliding window, the multiple packets, up to a fixed number of packets, are sending before receiving an acknowledgment from the receiver. Multiple packets send by the sender are acknowledged by a receiver using a single ACK packet. In this, the dispatcher can send multiple data packet and receiver accept it without having to stay for an acknowledgment. The sequence number is assigned to packet sequentially to help in maintaining the path of those packets that did received the acknowledgment. The receivers acknowledge a packet by sending acknowledgments that include the sequence number of the next expected. This acknowledgment announces that the receiver is ready to receive a packet, beginning with the number specified. The sliding window can be classified into two parts, namely Go-back-N (GBN) ARQ and Selective-Repeat (SR) ARQ. The Go-back-N ARQ has a width of sender widow is not fixed but the receiver window has one, and the Selective-repeat ARQ have the width of both dispatcher and receiver is greater than one.

1) Go-back-N ARQ

In the go-back-n, the dispatcher can send continues several packets before receiving an acknowledgment, but the receiver can only buffer one packet. We maintain a duplicate of the sent packet until the acknowledgments reach our destination. The receiver side keeps track of sequence number (SeqNo) of the next packet is expected to accept and sends the number with each ACK it sends. When the recipient receives the packets, it keeps on sending ACKs or a NACK, in case a packet is imperfectly received. During transmission, if several packets encountered an error the recipient ignores that packet and any other packet that comes following its [12]. When the dispatcher receives a NACK, it retransmits the packet in error plus all the succeeding packets. If a packet is lost, the recipient sends NACK after receiving the next packet. For fear that there is a long delay previous to sending the NACK; the dispatcher will resend the missing packet after its timer timeout. If the ACK packet sent by the receiver is lost, the sender resends the packet after its timer timeout, as given in fig. 1.3. Once all the packets in the sender’s output buffer window are transmitted, the sender will notice that the entire sent packet in that window is outstanding, beginning with the first packet that was lost [13]. For Go-back-n mechanism, each error generates a requirement to retransmit k packets, rather than just one packet.

Thus the number of a transmitted packet to successfully transmit one packet

$$p = \frac{1-p}{(1-p+pk)} \qquad (3)$$

Where the approximate value of

$$k = 2a + 1 \;\; for \quad N > 2a + 1$$

$$k = N \qquad for \quad N < 2a + 1$$

Therefore, channel utilization for go-back-n mechanism is given below

$$p = \begin{cases} \dfrac{(1-p)}{(1+2ap)} & for \quad N > 2a+1 \\ \dfrac{N(1-p)}{(2a+1)(1-p+pN)} & for \quad N < 2a+1 \end{cases} \qquad (4)$$

Where N is the size of window

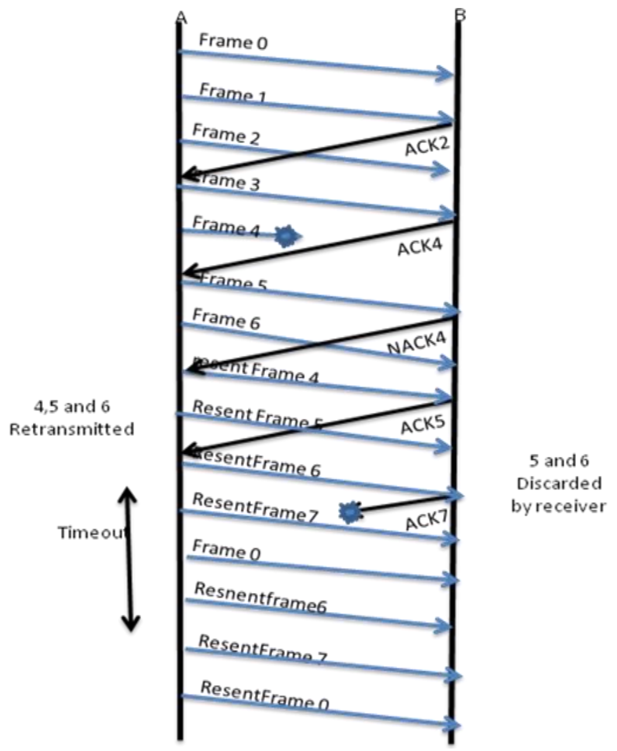


Fig. 1.3. Go-Back-N ARQ Mechanism

Drawback of Go-Back-N ARQ: It transmits all the packets if one packet is damaged or missing. It also transmits packet continuously as long as it does not receive the NACK. The NACK takes some time to reach the sender. Till that time, the sender has already sent some packet. All those will be retransmitted after receiving the NACK.

2) *Selective- Repeat ARQ:*

The selective-repeat mechanism is the improved version of Go-back-N mechanism as it has buffers on joint sides of the recipient as well as a sender. This mechanism allows the dispatcher to have more than one outstanding packet at a particular instant and the recipient to accept out of order packets and accumulate them in its own window [14]. In the selective repeat, the upholding of buffers and logical timers is accurately the same as in the Go-back-n. The only discrepancy is that if a negative ACK is received the dispatcher retransmits the associated packet recognized by the NACK, as given in the fig. 1.4. All others thing similar to retransmission, loop iterations and timeouts are the entire equal as Go-back-n. This is dissimilar from Go-back-n in the logic that it just retransmits the packet for which a NACK is received and not all the successive packets, as the receiver keeps a window of a packet, not the entire series can be retransmitted only the timeout packets or discard packets needs to be transmitted. Here, the sender transmits packets continuously until a NACK reach your destination [15]. While this happens, the dispatcher retransmits the NACK packet without resending the transmitted packets following it the sliding window. According to [16], the GBN is not efficient in communication channels with high error rates since it has to retransmit the erroneous packet and the other entire packet after it. The SR addresses this setback by retransmitting only the NACK packet and is the most effective of the ARQ mechanism. For selective-repeat mechanism, the channel utilization is given below

$$U = \begin{cases} (1-p) & for \quad N > 2a+1 \\ \dfrac{N(1-p)}{(2a+1)} & for \quad N < 2a+1 \end{cases} \qquad (5)$$

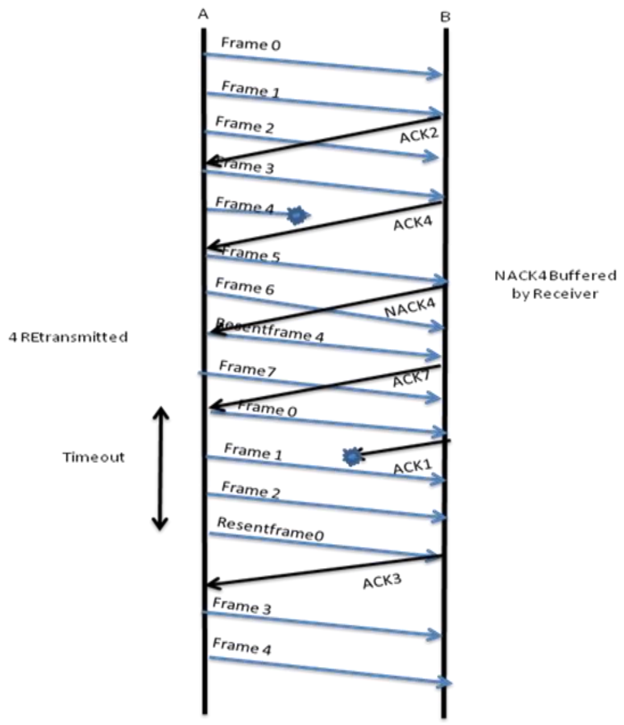


Fig. 1.4. Selective-Repeat ARQ Mechanism

Drawback of Selective - Repeat ARQ: The selective repeater mechanism has some problems to implement. The sequence number is often to be bigger than the window size, ensuring there’s no overlap will occur within the window. This permits recipient and dispatcher to be continuous in synchronization even once packets and ACK are missing at an extremely high rate. The buffering and ACK permit this mechanism to simply handle unhealthy packets, congestion and lost packets. It has been found that a far higher timeout price is required than in return N so as to cut back the quantity of packet sent. A lower timeout value leads to too numerous packets sequential collection out and being retransmitted unnecessarily.

IV. PRIORITIES RETRANSMISSION ARQ

In terms of throughput and delivery delay, SR is one of the most efficient flow control mechanisms for packet switched communications, but this does not guarantee that packets will be sent sequentially. This means that the recipient cannot keep the order of the packages. Therefore, all high-order protocols that require sequential packet delivery can process previously received packets. Therefore, all successful received packets must be stored in a buffer called a re-sequencing buffer until the original packet stream is sent to the top layer according to the outgoing command. Therefore we proposed a new mechanism which overcomes the disadvantages of various types of flow control approaches.

The proposed mechanism is Priorities Retransmission ARQ. It is the combination of Go-back-n and selective-repeat ARQ. Our mechanism works like the Go-back-n and selective-repeat ARQ but it has some modification. It has both sides a buffer i.e. it has buffer dispatcher side and buffers in receiver side as in selective repeat. It allows the dispatcher to have more than one outstanding packet at a specific time. It also allows the receiver to accept out of range packets and store it in its window. In this mechanism, packet retransmissions are put priorities in such a way that no fresh packets are sent until all the NACK packets are exhausted from the retransmission buffer. In the dispatcher side of Priorities Retransmission ARQ, it collects all NACK which are received by recipient side and store it in an array.

If there is only one NACK, which are received by receiver side, then Priorities Retransmission ARQ mechanism simply retransmit that packet for which a NACK is received. However, if there is more than one NACK is received by receiver side then the Priorities Retransmission ARQ mechanism find the minimum sequences number of a NACK which are stored in the array. After that, it retransmitted the minimum sequences number of a NACK. This process is continuing until all the NACK packets are not retransmitted. The parameters used in Prioritized Retransmission are given the table 1.1.

Table I. Description of parameters used in Priorities Retransmission Approach

Parameter	Definition
s_w	Size of a window
s_n	Sender, next packet to send
s_f	First outstanding packet
R_n	Receiver, next packet expected
m	Number of packet
t	Time period
ack	Acknowledgment
$nack$	Negative acknowledgment
ack_no	Acknowledgment sequence number
$nack\_no$	Negative Acknowledgment sequence number
seq_no	Sequence number

The following fig. 1.5 shows the sending side pseudo code for priority retransmission ARQ mechanism. The algorithm has four steps: in the first steps we initialize the variable, second steps define the process of sending the packet at the sender side, the third and fourth steps define the process of accepting the ACK and the process of the timer.

The given below fig. 1.6 shows the Receiving side pseudo code for priority retransmission ARQ mechanism. The

```
I. Initializes the variables
          $s_w = 2^{m-1}$
          $s_n = 0$
          $s_f = 0$
II. The Sender sends the packet
            if (event (request_to_send))
          if ($s_n - s_f \geq s_w$) then sleep()
             else
              storepacket($s_n$)
             sendpacke($s_n$)
             $s_n = s_n + 1$
             start_timer($s_n$)
            end
 III. The sender receives the acknowledgments
          if (event (arrival_ack))
             begin
               receive (packet)          \\ ack or nack
              if (corrupted(packet)) then sleep()
              if (packettype == nack)
                if ($s_f$ < nack_no > $s_n$)
                  begin
                    resend(nack_no)
                    start_timer(nack_no)
                 end
                  if (packettype == ack)
                  if ($s_f$ < ack_no > $s_n$)
                       begin
                     while($s_f$ < ack_no)
                    begin
                       purge($s_f$)
                    stop_time($s_f$)
                  $s_f = s_f + 1$
                   end
                 end
               end
IV. sender set the timer
            if (event (time_out (t)))
              begin
               start_timer(t)
              send_packet(t)     \\only the packet
                                \\which times
                                \\is out is resent
             end
```

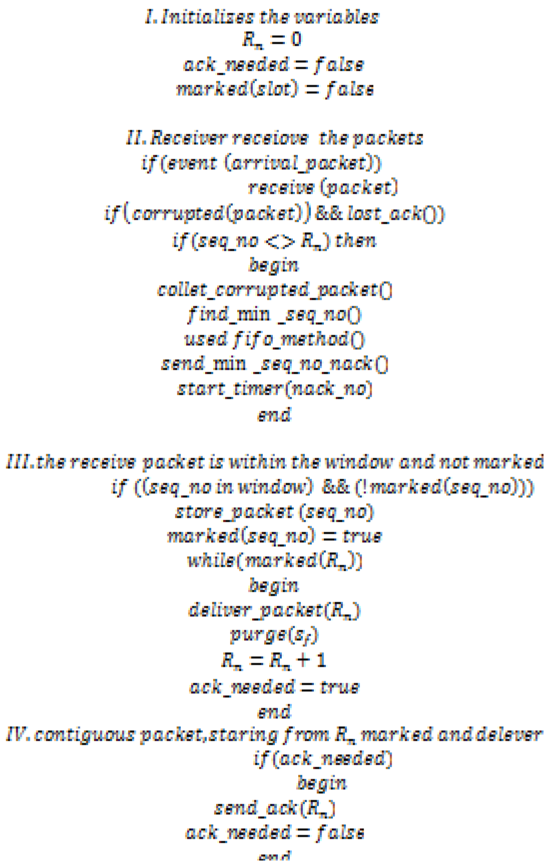
Fig. 1.5. Sending side pseudo code for priority retransmission ARQ mechanism

algorithm has four steps: in the first steps we initialize the variable, the second step defines how the packets are received and if there is an error in the packet received or any packet is lost then how they are collected. After the error-packet is collected then find the minimum sequence number of the packet which has a minimum number of NACK sequence number, then send that NACK sequence number to the sender and all the other NACK sequence number after the minimum sequence number of NACK. The third and fourth step define remain process which completes the cycles.

In priority retransmission ARQ mechanism buffer, packets are released to the communication link on a First Come First Out basis. This effectively prevents the sender from inserting extra fresh packets into an already lossy communication channel. Although this mechanism reduces the overall network throughput, it increases the likelihood of a packet being revived correctly.

This mechanism reduces the delivery delay since the adaptive increment of the number of retransmissions raises the possibility of correcting the erroneous packets.

Moreover, it condenses the number of pending packets as a result of prioritization of retransmission over new transmissions. Although this mechanism reduces the overall network throughput; it increases the likelihood of a frame being received correctly.



```
I. Initializes the variables
    $R_n = 0$
    ack_needed = false
    marked(slot) = false

II. Receiver receiove  the packets
    if (event (arrival_packet))
        receive (packet)
    if (corrupted(packet)) && lost_ack())
    if (seq_no <> $R_n$) then
    begin
        collet_corrupted_packet()
        find_min _seq_no()
        used fifo_method()
        send_min _seq_no_nack()
        start_timer(nack_no)
    end

III.the receive packet is within the window and not marked
    if ((seq_no in window) && (!marked(seq_no)))
        store_packet (seq_no)
        marked(seq_no) = true
        while(marked($R_n$))
        begin
            deliver_packet($R_n$)
            purge($s_f$)
            $R_n = R_n + 1$
            ack_needed = true
        end
IV. contiguous packet,staring from $R_n$ marked and delever
    if (ack_needed)
    begin
        send_ack($R_n$)
        ack_needed = false
    end
```

Fig. 1.6. Receiving side pseudo code for priority retransmission ARQ mechanism

V. SIMULATION AND RESULTS

The performance of the priorities retransmission ARQ is defined with the help of sending the packets from the dispatcher side to the recipient side. For the performance, we assume that; calculate the number of packets sent across the channel among the dispatcher side and receiver side. For this, first, we give the compression between timeout value and the number of packets sent. Therefore in the second, we give the comparison between events and the number of packets sent. Table 1.2 shows the comparison between timeout values and the number of packets sent to the channel.

Table II. Comparison between values of timeout and number of packet sent

S. No.	Value of timeout	Number of packet sent
1	10	3352
2	30	1286
3	50	790
4	70	569
5	90	445

In table 1.2, we have assumed 10% as error rate and kept the event is 1000. Here we change the timeout value and calculate the number of packets sent as given in the fig. 1.7.

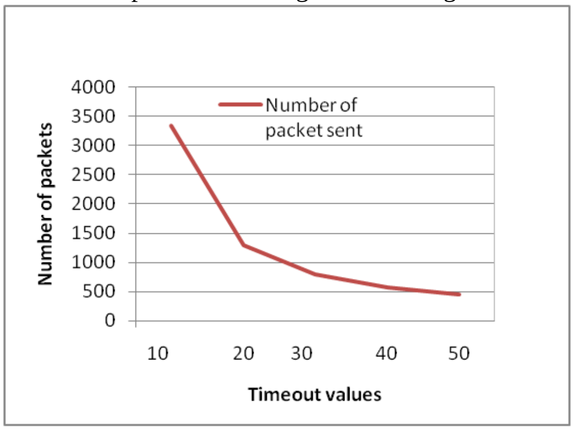


Fig. 1.7. Value of Timeout v/s Number of Packet sent

Table 1.3 shows the comparison between events and the number of packets sent to the channel. In the table-1.3 we have assumed 10% as error rate and timeout values are also 10.

Table III. Comparison between events and number of packet sent

S. No.	Events	Number of packet sent
1	50	17
2	100	39
3	150	57
4	200	63
5	250	90
6	300	102
7	350	126
8	400	140
9	450	153
10	500	167

Here we change the number of events value and calculate the number of packets sent as given in the fig.1.8.

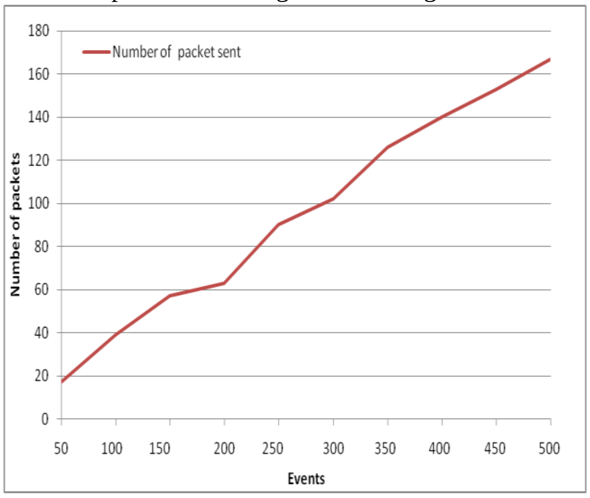


Fig. 1.8. Value of Timeout v/s Number of Packet sent

VI. CONCLUSION

The Use of Internet Services and Communication Network in day to day life with varying speed and memory capacities requirement, here the strength and weakness of such networks are cheeked and to improve the weaknesses, a priority retransmission mechanism has been developed. From table 1.2, table 1.3, fig. 1.7 and fig. 1.8, we find out that network bandwidth affected the packet transmission process whereby high bandwidth led to higher rates of successful packet delivery and hence high throughput while lower bandwidths led to frequent timeouts and hence lower throughputs. In the real time application such as Video conferencing and live telecast, it creates the difficulty of synchronization. So to overcome these problems priorities retransmission ARQ introduce. In priorities retransmission ARQ, we have priorities on the NACK in which we are retransmitting the packet according to the minimum sequence number of NACK. Hence the priorities retransmission ARQ gives the optimum performance. When we use priorities retransmission ARQ the utilization is increased. It is found that overall performance of the Communication Network improved.

REFERENCES

1. Afanasyev, Alexander, Neil Tilley, Peter Reiher, and Leonard Kleinrock. "Host-to-host congestion control for TCP." IEEE Communications surveys & tutorials 12, no. 3, 2010, pp. 304-342.
2. Pahlevanzadeh, Bahareh, SA Hosseini Seno, Tat-Chee Wan, and Rahmat Budiarto. "Comparative study on the development of Ethernet flow control." In 2nd international conference on Sciences and technology (ICSTIE2008), Universiti Teknologi Malaysia, Penang, Malaysia. 2008.
3. Malhotra, Richa, Ronald van Haalen, Michel Mandjes, and Rudesindo Núñez-Queija. "Modeling the interaction of IEEE 802.3 x hop-by-hop flow control and TCP end-to-end flow control." In Next Generation Internet Networks, 2005, pp. 260-267. IEEE, 2005.
4. Ramakrishnan, K. K., and Raj Jain. "A binary feedback scheme for congestion avoidance in computer networks." ACM Transactions on Computer Systems (TOCS) 8, no. 2, 1990, pp. 158-181.
5. Jacobson, Van. "Congestion avoidance and control." In ACM SIGCOMM computer communication review, vol. 18, no. 4, pp. 314-329. ACM, 1988.
6. Pal, Ajit. "Broadcast Communication Networks: Medium Access Control (MAC) Techniques, Version 2 CSE IIT, Kharagpur." 2009.
7. Gerla, Mario, and Leonard Kleinrock. "Flow control: A comparative survey." IEEE Transactions on Communications 28, no. 4, 1980, pp. 553-574.
8. León-García, Alberto, Indra Widjaja, and Jesús Esteban Díaz Verdejo. “Communication networks, fundamental concepts and key architectures”. McGraw-Hill, 2002, pp.845-857.
9. Sklar, Bernard, and Fredric J. Harris. Digital communications: fundamentals and applications. Vol. 2001. Englewood Cliffs, NJ: Prentice-hall, 1988.
10. Mandiwal, Sunil, and T. Subbulakshmi. "Flow Control Policy Protocols Generalization and Dynamic ARQ Error Control Scheme." Research Gate 2014.
11. Behrouz A. Forouzan . “Data communication and Networking”, Fifth edition, New York, McGraw Hill, 2017.
12. Zhang, Mingrui. "Major automatic repeat request protocols generalization and future develop direction." In 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, vol. 2, 2013, pp. 5-8. IEEE.
13. Nyangaresi, Vincent Omollo, Silvance O. Abeka, and Rebecca N. Arika. "Low latency automatic repeat request protocol for time sensitive GSM-enabled smart phone video streaming services." 2018.
14. Anagnostou, Miltiades, and Emmanuel Protonotarios. "Performance analysis of the selective repeat ARQ protocol." IEEE Transactions on Communications 34, no. 2, 1986, pp. 127-135.
15. Gao, M., J. Li, W. Li, and N. Xu. "Delivery Delay Analysis of Selective Repeat ARQ in Underwater Acoustic Communications." Sensor Network Data Communication. 5, no. 134, 2016, pp. 2.
16. Li, Suoping, Yongqiang Zhou, Xijuan Yang, and Zufang Dou. "Performance evaluation of multiple relays cooperative Go-Back-n ARQ with limited retransmission." Journal of Systems Engineering and Electronics 26, no. 6, 2015, pp.1210-1215.

AUTHORS PROFILE

Professor Avanish Kumar Presently working as Chairman, Commission for Scientific and Technical Terminology, and Director, Central Hindi Directorate, Department of Higher Education, Ministry of Human Resource Development, Government of India Professor Avanish Kumar Born in 1969, at Manglour, District Hardwar, Uttrakhand, completed his M. Sc., from Meerut University (presently CCS University), Meerut, M. Phil., from University of Roorkee (presently IIT), Roorkee and Ph. D. in Mathematics, from Gurukula Kangri, University, Hardwar. University of Roorkee awarded him ‘University Medal’ for standing First in First Class. He was then associated with Seventh-Day Adventist College, Roorkee during January 1989 to November 1995. During 1995 to 1999, he has served as Assistant Scientific Officer in the Government of India. In November 1999 he joined Bundelkhand University as Assistant Professor in the Department of Mathematical Sciences and Computer Applications. There he served in various capacity and became full Professor on November 2014. He was awarded ‘Professor R. C. Mehrotra Best Teacher Award and Gold Medal’ for outstanding contribution to the university services. He has presented research papers, delivered invited talk, Chaired technical session and Guest of Honor in more than 60 conferences and seminars. He had supervised 15 Ph. D. students, 27 M. Phil. Students; He has published about 80 research papers in National and International Journal of reputes. His research interests are Modeling, Optimization Techniques and Distributed Computing Systems. Solution provided for Unbalanced Assignment and Transportation problems with application in Distributed Systems.

Dharamdas Kumhar is currently working as Asst Professor in Dept of Mathematical Sciences and Computer Applications at Bundelkhand University, Jhansi. He is pursuing his Ph. D in Computer Science at Bundelkhand University, Jhansi. He received his M.C.A. degree from Vikram University, Ujjain in 2000 and Bachelor’s degree from Dr. Harisingh Gour University, Sagar in 1986. He has 18 years of teaching experience and Academic experience. His research interest focuses mainly Computer Communication Network and Network Security. He has published papers in international and national level Journals & IEEE conferences. Most of the papers are in web of science and Scopus indexed publications.