Simulation and Comparison of ARQ protocols

Sheetal Shalini
Computer Science and Engineering
National Institute of Technology
Karnataka-575025
Email: sheetalsh456@gmail.com

Navya R S
Computer Science and Engineering
National Institute of Technology
Karnataka-575025
Email: navy8296@gmail.com

***Abstract*—The purpose of the data link layer is to transfer blocks of data without error between two adjacent devices. Adjacent devices are physically connected by a communication channel such as telephone lines, coaxial cables, optical fibres, or satellites.Transmission of data over the link would be very simple indeed if no error ever occurred. Unfortunately, this is not so in a real physical link. A data link protocol thus has to be designed to ensure an error-free transmission and also to achieve an efficiency of the data transfer as high as possible. This is when error control protocols come into the picture. Since real-world channels are noisy ones, we discuss,compare and henceforth present a case study on error control protocols for a noisy channel. The case study will include a detailed explanation of each protocol, along with their diagrammatic representations, correctness, comparison, as well as drawbacks and suggestions for any improvements that can be incorporated for the betterment of these protocols.**

***Keywords*—IEEEtran, journal, LaTeX, paper, template, stop and wait, go back n, selective repeat, error, control, protocols, comparison.**

I. Introduction

In information theory and coding theory with applications in computer science and telecommunication, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases. Whenever packets of data need to be transmitted from point A (the transmitter) to point B (the receiver),there is always a chance that something bad happens to them while they move through the medium between A and B. Some packets may be corrupted or even lost entirely. When data-frame is transmitted, there is a probability that data-frame may be lost in the transit or it is received corrupted. In both cases, the receiver does not receive the correct data-frame and sender does not know anything about any loss.In such case, both sender and receiver are equipped with some protocols which helps them to detect transit errors such as loss of data-frame. Hence, either the sender retransmits the data-frame or the receiver may request to resend the previous data-frame.To cope with this, ARQ (Automatic Repeat request) protocols have been used to provide a more reliable way of communication between the transmitter and the receiver. The fundamental idea is to use acknowledgments (ACKs) sent from the receiver to the sender for data frames sent from the sender to the receiver. The concepts used in ARQ are:

1. Timers: Sender keeps a timer. If an ACK is not received before the timer expires, it resends the frame.

2. Need for sequence numbers for the frames: If a frame is properly received, but the ACK is lost, then the sender will time-out and retransmit. This will result in the receiver receiving duplicate frames. Hence frames need sequence numbers to allow the receiver to detect duplicates.

3. Need for sequence numbers in the ACKs: If a frame 0 is sent, but the timer times-out before the ACK is received, the sender will resend frame 0. If the ACK for the first frame 0 arrives soon after the sender retransmitted frame 0, the sender will assume that the ACK was for the retransmitted frame 0 and hence send frame 1. Now if frame 1 gets lost, but the receiver sends another ACK for the retransmitted frame 0, the sender may mistakenly think this ACK is for frame 1. If ACKs said what frames they were acknowledging (using their sequence numbers), this problem would not arise.

4. Checkpointing: periodically sending ENQ (enquiries) to see what frames were received.

The requirements for error control mechanism are:

1. Error detection - The sender and receiver, either both or any, must ascertain that there is some error in the transit.

2. Positive ACK - When the receiver receives a correct frame, it should acknowledge it.

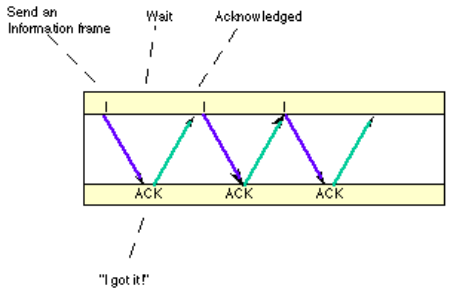
3. Negative ACK - When the receiver receives a damaged frame or a duplicate frame, it sends a NACK back to the sender and the sender must retransmit the correct frame.

4. Retransmission: The sender maintains a clock and sets a timeout period. If an acknowledgement of a data-frame previously transmitted does not arrive before the timeout the sender retransmits the frame, thinking that the frame or its acknowledgement is lost in transit.

II. Stop-And-Wait ARQ

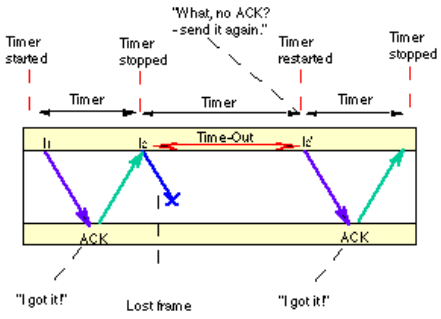
Stop and Wait transmission is the simplest reliability technique and is adequate for a very simple communications protocol. A stop and wait protocol transmits a Protocol Data Unit (PDU) of information and then waits for a response. The receiver receives each PDU and sends an Acknowledgement (ACK) PDU if a data PDU is received correctly, and a Negative

Acknowledgement (NACK) PDU if the data was not received. In practice, the receiver may not be able to reliably identify whether a PDU has been received, and the transmitter will usually also need to implement a timer to recover from the condition where the receiver does not respond. Under normal transmission the sender will receive an ACK for the data and then commence transmission of the next data block. For a long delay link, the sender may have to wait an appreciable time for this response. While it is waiting the sender is said to be in the ”idle” state and is unable to send further data.



Stop and Wait ARQ - Waiting for Acknowledgment (ACK) from the remote node.

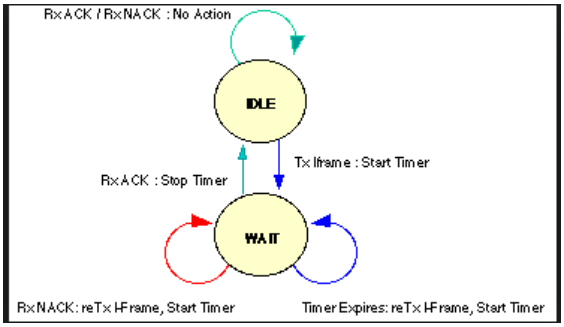
The blue arrows show the sequence of data PDUs being sent across the link from the sender (top to the receiver (bottom). A Stop and Wait protocol relies on two way transmission (full duplex or half duplex) to allow the receiver at the remote node to return PDUs acknowledging the successful transmission. The acknowledgements are shown in green in the diagram, and flow back to the original sender. A small processing delay may be introduced between reception of the last byte of a Data PDU and generation of the corresponding ACK. When PDUs are lost, the receiver will not normally be able to identify the loss (most receivers will not receive anything, not even an indication that something has been corrupted). The transmitter must then rely upon a timer to detect the lack of a response.



Stop and Wait ARQ - Retransmission due to timer expiry

In the diagram, the second PDU of Data is corrupted during transmission. The receiver discards the corrupted data (by noting that it is followed by an invalid data checksum). The sender is unaware of this loss, but starts a timer after sending each PDU. Normally an ACK PDU is received before this the timer expires. In this case no ACK is received, and the timer counts down to zero and triggers retransmission of the same PDU by the sender. The sender always starts a timer following transmission, but in the second transmission receives an ACK PDU before the timer expires, finally indicating that the data has now been received by the remote node.

The state diagram (also showing the operation of NACK) is shown below:



Stop and Wait ARQ - State Diagram (Green for ACK, Blue for Data, Red for NACK)

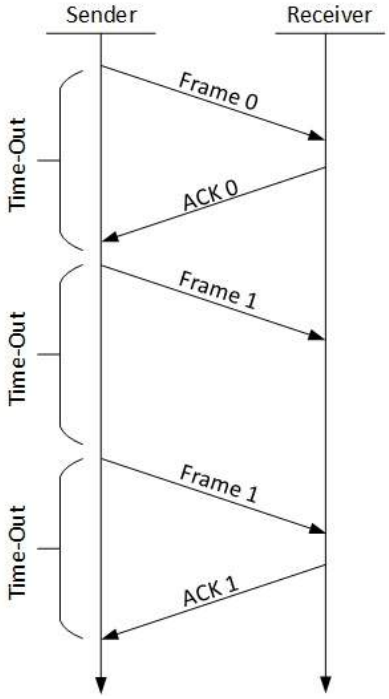
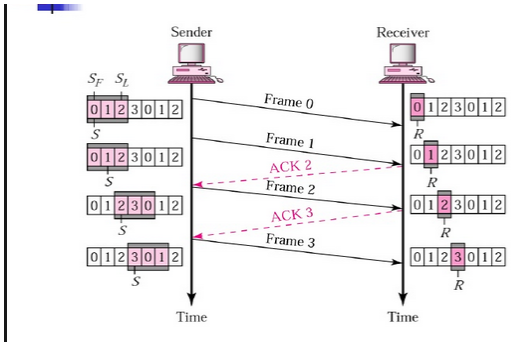


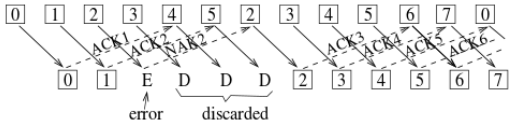
Illustration of the working of Stop and Wait ARQ

III. Go-Back-N ARQ

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1. It can transmit N frames to the peer before requiring an ACK. The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends.



The receiver will discard any frame that does not have the exact sequence number it expects (either a duplicate frame it already acknowledged, or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame.[1] Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.



To support Go-Back-N ARQ, a protocol must number each PDU which is sent. (PDUs are normally numbered using modulo arithmetic, which allows the same number to be re-used after a suitably long period of time. The time period is selected to ensure the same PDU number is never used again for a different PDU, until the first PDU has ”left the network” (e.g. it may have been acknowledged)). The local node must also keep a buffer of all PDUs which have been sent, but have not yet been acknowledged. The receiver at the remote node keeps a record of the highest numbered PDU which has been correctly received. This number corresponds to the last acknowledgement PDU which it may have sent. Go-Back-N ARQ is a more efficient use of a connection than Stop-and-wait ARQ, since unlike waiting for an acknowledgement for each packet, the connection is still being utilized as packets are being sent. In other words, during the time that would otherwise be spent waiting, more packets are being sent. However, this method also results in sending frames multiple times if any frame was lost or damaged, or the ACK acknowledging them was lost or damaged, then that frame and all following frames in the window (even if they were received without error) will be re-sent.

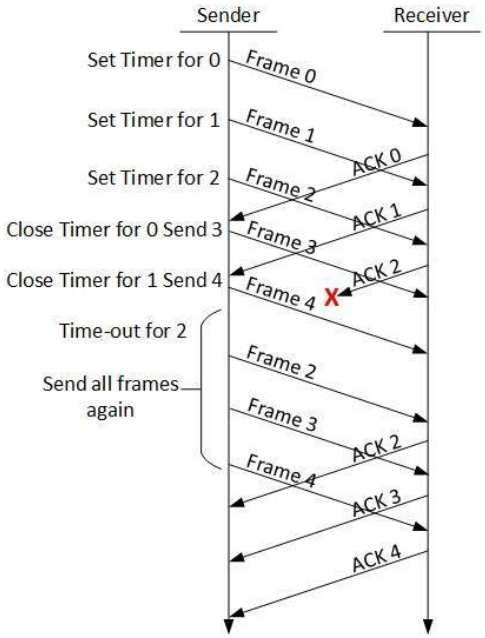
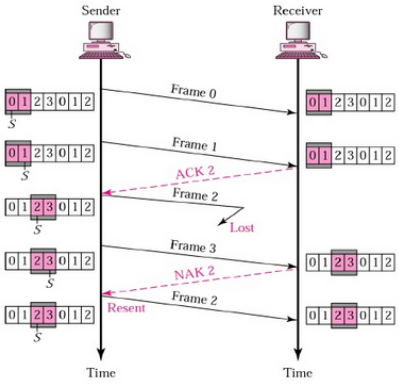


Illustration of the working of go back n ARQ

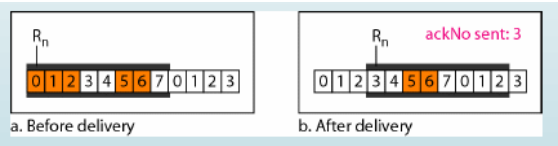
IV. Selective Repeat ARQ

Selective Repeat ARQ / Selective Reject ARQ is a specific instance of the Automatic Repeat-Request (ARQ) protocol used to solve sequence number dilemma in communications. It is part of the automatic repeat-request (ARQ). With selective repeat, the sender sends a number of frames specified by a window size even without the need to wait for individual ACK from the receiver as in Go-Back-N ARQ. The receiver may selectively reject a single frame, which may be retransmitted alone; this contrasts with other forms of ARQ, which must send every frame from that point again. The receiver accepts out-of-order frames and buffers them. The sender individually retransmits frames that have timed out. It may be used as a protocol for the delivery and acknowledgement of message units, or it may be used as a protocol for the delivery of subdivided message sub-units.

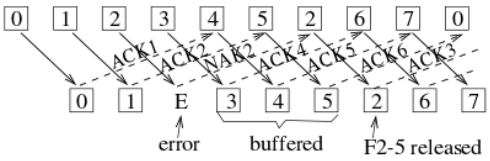


The go-back-n protocol works well if errors are rare, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the selective repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one. When used as the protocol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

delivery of messages, the sending process continues to send a number of frames specified by a window size even after a frame loss. Unlike Go-Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error; this is the general case of the sliding window protocol with both transmit and receive window sizes greater than 1. The receiver process keeps track of the sequence number of the earliest frame it has not received, and sends that number with every acknowledgement (ACK) it sends. If a frame from the sender does not reach the receiver, the sender continues to send subsequent frames until it has emptied its window. The receiver continues to fill its receiving window with the subsequent frames, replying each time with an ACK containing the sequence number of the earliest missing frame. Once the sender has sent all the frames in its window, it re-sends the frame number given by the ACKs, and then continues where it left off. In this protocol, both sender and receiver maintain a window of outstanding and acceptable sequence numbers, respectively. The senders window size starts out at 0 and grows to some predefined maximum. The receivers window, in contrast, is always fixed in size and equal to the predetermined maximum. The receiver has a buffer reserved for each sequence number within its fixed window. Associated with each buffer is a bit (arrived) telling whether the buffer is full or empty. Whenever a frame arrives, its sequence number is checked by the function between to see if it falls within the window.



The size of the sending and receiving windows must be equal, and half the maximum sequence number (assuming that sequence numbers are numbered from 0 to n1) to avoid miscommunication in all cases of packets being dropped. To understand this, consider the case when all ACKs are destroyed. If the receiving window is larger than half the maximum sequence number, some, possibly even all, of the packets that are resent after timeouts are duplicates that are not recognized as such. The sender moves its window for every packet that is acknowledged.



To support Go-Back-N ARQ, a protocol must number each PDU which is sent. (PDUs are normally numbered using modulo arithmetic, which allows the same number to be re-used after a suitably long period of time. The time period is selected to ensure the same PDU number is never used again for a different PDU, until the first PDU has ”left the network” (e.g. it may have been acknowledged)). he local node must also keep a buffer of all PDUs which have been sent, but have not yet been acknowledged. The receiver at the remote node keeps a record of the highest numbered PDU which has been correctly received. This number corresponds to the last acknowledgement PDU which it may have sent.

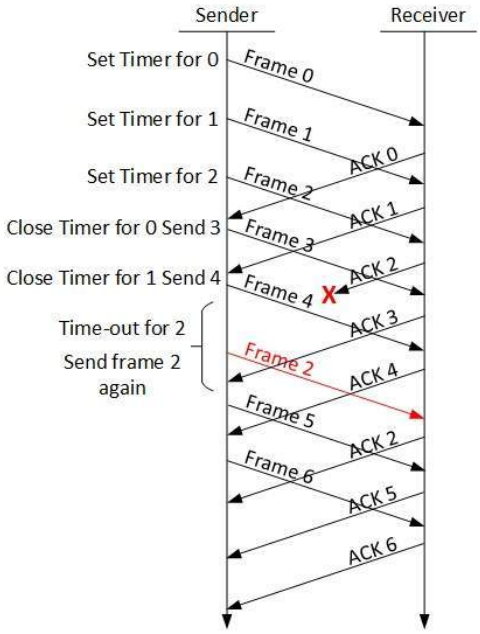


Illustration of the working of selective repeat ARQ

V. Comparison

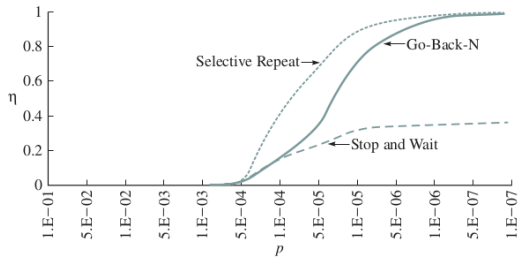


FIGURE 5.24 Transmission efficiency of ARQ protocols

1. The efficiency of the selective-repeat protocol is higher than that of the go-back-N protocol if the communication channel capacity is considered as the only communication resource. On the other hand, the selective-repeat protocol requires more buffer memory due to the larger receive window width (almost twice as much as the go-back-N protocol), and, due to the more complex memory management at the receiving side, also more receivers processing power.

2. Both go-back-N and selective-repeat protocols are much more efficient than the stop-and-wait protocol if the transmit window width is large enough. However, the efficiency of the selective-repeat protocol is higher that the efficiency of the go-back-N protocol in most cases.

3. The stop-and wait protocol is a special case of both go-back-N and selective-repeat protocols and also that the behaviour of go-back-N and selective-repeat protocols is identical in the absence of loss.

The case study will include detailed comparisons and ways of improvement.

REFERENCES

[1] Research Paper on IMPORTANCE OF SLIDING WINDOW PROTOCOL by Balwinder Kaur, Mandeep Kaur, Pooja Mudgil and Harjeet Singh.

[2] Research Paper on The Importance of Sliding Window Protocol Generalisation in a Communication Protocols Course by Drago Hercog, University of Ljubljana.

[3] Research Paper on An Improved Selective-Repeat ARQ Strategy by E. J. WELDON, JR., MEMBER, IEEE.

[4] Research Paper on A GENERAL ANALYSIS TECHNIQUE FOR ARQ PROTOCOL PERFORMANCE IN HIGH SPEED NETWORKS by Ian F. Akyildiz and Wei Liu.

[5] 'The Data Link Layer: ARQ Protocols' lecture by Eytan Modiano, MIT.

[6] B.A.Forouzan,Data Communications and Networking 3rd edition, Mc Graw Hill Publications.

[7] David Haccoun, Samuel Pierre, The Communications Handbook, Chap. 14, 2002 CRC Press Release.

[8] M.Veeraraghaban, Analysis of error control and flow control schemes, March 30, 2004.

[9] Research Report on 'Throughput Analysis of a class of selective repeat protocols in high-speed environments' by Phuoc Tran-Gia, Hamid Ahmadi and Parviz Kermani.

[10] International Journal of Advanced Research in Computer Science and Software Engineering Research Paper on 'Analysis for Go-Back-N ARQ Protocols on Multi Channels by using AIMD Congestion Control Algorithm' by Radha Krishna Reddy, Ashim Roy, G.Sireesha and K.Pushpa Rani.

[11] Research Paper on 'Performance Analysis of ARQ Protocols using a Theorem Prover' by Osman Hasan and Sofi'ene Tahar.