TCOM 370 NOTES 99-11

ERROR CONTROL PROTOCOLS AND FLOW CONTROL

1. Introduction

Consider a network with a set of nodes (e.g. computers) interconnected by individual transmission links. These individual links between a pair of nodes may be radio or microwave (wireless) links, coaxial cables, optical fiber links, or twisted wire-pair connections. Each node will implement in its hardware and software multiple layers of some communication protocol, which we may take to be the seven-layer OSI reference model. In the Data Link Control (DLC) layer of such a multi-layer communication protocol structure, we may use specific schemes to allow ***reliable transmission*** of data packets between DLC entities at the two ends of a link within the network.

- This can be accomplished with *error detection* techniques to obtain reliable packet transmission by employing some form of ***ARQ*** *(automatic repeat request or automatic request for retransmission)* protocol.

Above the DLC layer in the protocol architecture at a node is the Network layer, which hands down data packets to the DLC layer. These data packets include headers containing, for example, addresses of destination and source, etc. These network layer packets handed to a DLC layer constitute the message packets that the DLC layer is expected to transmit reliably across a particular link to the node at the other end of the link. These message packets with *added framing bits, CRC bits, and other overhead bits such as data-length specification bytes, sequence-number bits, and acknowledgment bytes* which are added by the DLC protocol together form the **information frames** (I-frames) exchanged across the **data link.**

- Separate **acknowledgment** (ACK) and **negative acknowledgment** (NAK) frames may also be used on the data link. These ACK or NAK frames are generally much shorter than the I-frames.

The node that is transmitting I-frames to the other node is called the Primary (P) node, and the recipient node is called the Secondary (S) node. In a full duplex scheme, each node may act as both an S and a P node.

- **Flow control** refers to the procedure implemented by the nodes to ensure that the receive end does not become flooded with I-frames that it cannot process fast enough and would therefore be forced to drop.

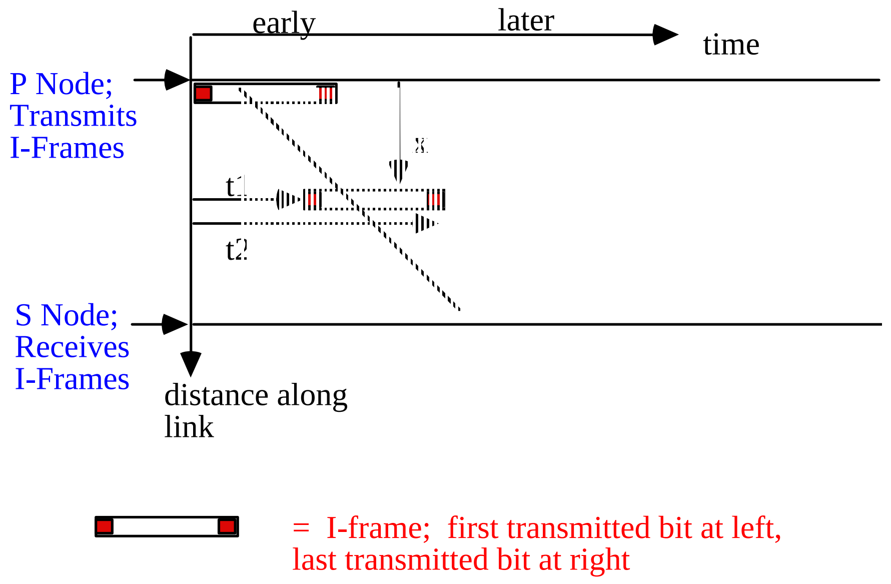
Flow control and ARQ work hand-in-hand in any specific implementation.

(Note: The discussion above pertains to the use of ARQ for exchanging data frames across a single link, implemented at the Data Link Control layer which is above the Physical layer. In the TCP/IP protocol, ARQ is actually implemented at the *Transport Layer* by TCP and is not implemented at the DLC layer. Recall that the transport layer is responsible for ensuring error-free packet transmission, in correct sequence. The general principles are the same; the actual end-to-end transmission delays may be quite different in the two cases.)

2. Frame Sequence Diagrams

It is best to think of these as a two-dimensional depiction of the progress of bits and frames across a physical link. The two dimensions are elapsed time and position (distance) along the physical link.

We show time on the horizontal (left-to-right) axis and position along the link in the vertical (downward) axis. In the figure following, the frame shown in between the P node and the S node is at a distance x between the two. What this means is that at time t_1 the first bit is at distance x from P, and at a later time t_2 the last bit of the frame is at that same position. The dashed line shows the time-distance history of one particular bit in the frame.


early
later
time
P Node; Transmits I-Frames
t1
t2
S Node; Receives I-Frames
distance along link
= I-frame; first transmitted bit at left, last transmitted bit at right

(slanting line in figure shows the progress of a particular bit in the frame)

- The **duration** or transmission time $\mathbf{T_{ix}}$ (here the sub-scripts stand for "I-frame transmission" time or I-frame duration) of the frame **in secs.** is the horizontal length of the frame in the figure. It is the **ratio** $T_{ix} = \frac{N_i}{R}$ where N_i is **the frame length in bits** and R is the **transmit data rate** (in bits per sec.) at which bits are emitted at the transmitter.

Basic Definitions and Idle RQ Efficiency

The transmit data rate R is one of the basic parameters governing operation of the link. It is the raw bit rate that the link can support continuously, but because there are occasional errors we need to implement some form of error control mechanism.

- If the **one-way propagation delay is $\mathbf{T_p}$** (time for any bit to traverse the link), then the ratio $a = \frac{T_p}{T_{ix}}$ is the **propagation delay measured in units of frame durations.**

Obviously T_p depends on the link length and the speed of propagation of electrical signals on the link (usually close to the speed of light).

Thus if frames are sent out one after the other without gap at the transmit end, there can be "*a*" frames "in transit" within the link at any given time. If the receiver starts processing each I-frame without delay, requires negligible processing time, and generates very short ACK frames, the total time elapsed before an ACK frame gets back to the transmit end after it starts sending an I-frame (round-trip delay) is clearly

$$T_t = T_{ix} + 2T_p \qquad \text{or} \qquad 1+2a$$

measured in **I-frame durations.** If the above assumptions are not true (i.e. finite time for processing I frame, and ACK frames at the two ends, finite length of the ACK frame, than these **must** be taken into account in writing the **round-trip delay T_t**.

Under the above simple assumptions, if we use the **Idle RQ** protocol for ARQ in which the P node only sends a new I-frame after it has received an acknowledgment for the current I-frame sent, clearly we can send one frame every $1+2a$ frame durations if there are **no errors**, so the *efficiency of utilization* U in this case is

$$U_{\text{no error}} = \frac{T_{ix}}{T_t} = \frac{1}{1+2a}.$$

More generally, we have to use the **round-trip delay T_t** (in secs.) in the denominator and the frame duration **T_{ix}** (in secs.) in the numerator **and** account for frame errors also, in obtaining U.

3. Utilization Efficiency of Idle RQ with Errors

If an error occurs or an I-frame gets lost, the P node has to retransmit the frame. In Idle RQ the P node will initiate retransmission of the current frame if it receives an ***explicit*** **NAK** for that frame, or (in an ***implicit*** scheme or if the I-frame simply never makes it to S) if it does not receive an **ACK by a certain "time-out" period**. We assume explicit NAKs are sent immediately or that the time-out period is equal to the round-trip delay (minimum time-out period allowable), so that in either case the P node knows at the end of time period T_t if a retransmission is required. Let N_t be the number of transmissions needed for a frame to be received without error; its expected value is $E\{N_t\}$. The utilization efficiency $U=U_{\text{no error}}$ is ***divided by*** $E\{N_t\}$ to get U in the presence of frame errors, since a total average time of $E\{N_t\}T_t$ is now required to send the frame.

We can derive the formula :

$$E\{N_t\} = \frac{1}{1-P_f}$$

where P_f is the probability that any frame will need retransmission (a frame error occurs during transmission). We can find P_f from the bit-error-rate (BER, or P_b) by using $P_f \approx N_i P_b$ if N_i is fairly large and P_b is small enough so that $N_i P_b$ is also small. This comes from the binomial expansion. Now N_t is a random variable taking on values 1,2, 3, ... We have $N_t=1$ with probability $(1–P_f)$, the probability of no frame error; $N_t=2$ with probability $P_f(1–P_f)$, the probability of one frame error followed by successful frame transmission. In general, $P\{N_t=n\}=P_f^{n-1}(1–P_f)$ and we get $E\{N_t\}= \sum_{n=1}^{\infty} n\,P_f^{n-1}\,(1–P_f)$. The sum $\sum_{n=1}^{\infty} n\,P_f^{n-1}$ can be shown to be $\frac{1}{(1-P_f)^2}$ and the result follows.

The utilization is decreased in the presence of errors, from the no-error case, by the factor $1-P_f$, and under the simple assumptions of negligible processing times, short ACK or NAK frame lengths, etc., we have the formula

$$\mathbf{U_{\text{with error}}=\frac{1-P_f}{1+2a}} \; .$$

Clearly, small P_f and small "*a*", or low propagation delay relative to I-frame duration will lead to good utilization efficiency. Thus for short distance and low-rate links, Idle RQ is a reasonable protocol.

4. Frame Sequence Numbers in Idle RQ

Each I-frame that is sent by the P node carries a frame sequence number, and when the S node sends back an ACK frame or a NAK frame it refers to this sequence number.

While consecutive integers may be used for each successive I-frame, the **minimum set** of unique frame sequence numbers needed for the Idle RQ is of **size 2**; that is, with the numbers in the set {0,1} we can use alternating 0,1,0,1,0,... for frame sequence numbering. This requires only a 1-bit field in the frame to carry the sequence number.

Both I and ACK (or NAK) frames carry the sequence number. It is easy to show that (a) frame sequence numbers must be used by **both** I and ACK frames, and that (b) for Idle RQ **1-bit sequence numbers** will work fine. This can be shown by considering all the possibilities (no errors, I-frame errors, ACK frame errors, etc.). The fact that 1-bit sequence numbers can be used (i.e. the numbers 0 and 1 are re-used over and over) follows from the fact that the P node will not transmit a new frame unless it has received a valid ACK for its previous transmission, and the S node will not *accept* an I-frame that does not have the number it is expecting (i.e. 1 following a previous 0 and 0 following a previous 1). Note however that in Idle RQ the *S node will always generate an ACK for a frame it receives correctly* (whether it accepts it or not). The fact that ACK and NAK frames should also use sequence numbers in them for the I-frame they are referring to follows from considerations of special situations such as, say, when an ACK or NAK frame suffers a long delay in getting back to the P node.

5. Continuous RQ: Selective Repeat and Go-Back-N

While the **Idle RQ** error control protocol is generally used for character-oriented applications and uses **half-duplex** operation on the transmission link, **Continuous RQ** protocols for error control operate on **full-duplex** links (allowing simultaneous two-way communication) and are generally used for bit-oriented transmission.

In continuous RQ the P node can send out I-frames one after the other without gaps between them. After a certain propagation delay, these frames are received in sequence at the S node, which sends back ACK frames after each I-frame received correctly. Under no-error conditions, the protocol can utilize the link with 100% efficiency, since the link is always sending I-frame data with no gaps, and I-frames are not repeated (every T_{ix} seconds the S node receives one new I-frame of duration T_{ix}).

- ***The P node should maintain in a "re-transmit list" or buffer, copies of I-frames it has transmitted but which have not been acknowledged,*** because the S node may request a retransmission in the event of frame error.

It is precisely how frame error conditions are handled that distinguish the various continuous RQ schemes from each other.

Selective Repeat (SR) using only ACK frames, no NAKs, (Implicit SR)

- S sends an ACK for *each* un-errored I-frame it receives; if it receives a frame it already has, it ignores it as a duplicate (but sends an ACK for it anyway).
- S node maintains *link receive list or buffer* to hold I-frames that may have been accepted out-of-sequence;
- If P receives ACK with number beyond what it expects in sequence, it enters *re-transmit state,* suspending transmission of new I-frames until all its *un-ACK'ed* frames with numbers less than the out-of-sequence ACK number have been re-sent.

Since P will realize it needs to re-transmit only when it receives an "out-of-sequence" ACK, to prevent an indefinite delay if the last one in a sequence of I-frames is in error, P also implements a time-out interval for receipt of an ACK after each I-frame it sends.

Go-Back-N Protocol

- S *accepts* only in-sequence, un-errored, I-frame; *has no buffer.*
- S sends an ACK for each un-errored, in-sequence, I-frame it receives and accepts; an ACK numbered N actually acknowledges all frames *up to and including* I-frame N.
- S sends an explicit NAK for frame number it is expecting upon getting un-errored I-frame with sequence number beyond what it is expecting; then *ignores* this and further I-frames until it gets one it is waiting for.
- If P receives NAK, it enters *re-transmit state,* re-transmitting I-frames starting from the one that it got NAK for until all I-frames awaiting ACKs in its transmit list (buffer) have been re-sent. Then continues with new I-frame transmission.

S also applies a time-out interval after it sends a NAK to protect against lost NAKs.

Go-Back-N has no buffer requirements for storage of out-of-sequence frames at S. However, it is somewhat less efficient than SR because it re-transmits some already-transmitted frames (ignored by S) beyond the errored one, when a retransmit request is processed.

[In a dual version of Implicit SR, called Explicit SR, explicit NAKs (called **"selective reject"**) are also used. Here ACKs are sent and convey the same meaning as in Go-Back-N. NAks are generated as in Go-Back-N, but instead of ignoring further good I-frames they are simply accepted but no ACKs are issued, until the requested frame has been received.**]**

6. Sliding Window Flow Control

Idle RQ is based on what is known as **stop-and-wait** flow control. The transmitter sends a new frame only if the previous one has been acknowledged. Thus P is not allowed to have more than **1** unacknowledged frame held in its retransmit list for possible retransmission. The **transmit or send window** is said to be of **maximum size 1.** Similarly S will hold **only the one** frame that follows in-sequence from its last correctly received frame, out of sequence frames will not be held, they are discarded. The **receive window** is also of maximum size 1.

In general, we can allow the maximum transmit window size to be some integer K. (Usual terminology for this is "transmit window size is K", rather than "maximum transmit window size is K"). Think of the **transmit (send) window** size K as the length of **longest block of unacknowledged frames** the P node is allowed to have buffered at any given time. Similarly the (maximum) receive window is the size of the longest block of error-free frames that can be held.

- Block of frames in this context has a size equal to that of all consecutive frames between and including the earliest frame in the block and the latest frame in the block, regardless of whether there are or are not intermediate frames present in the block.

For example, if P is already awaiting ACKs for two consecutive frames at a certain point, and its window size or maximum window size is K, then at this point it is allowed to send K-2 more frames following the first 2 in sequence (its window is still open or available if K>2). If in the meantime an acknowledgment comes in that includes one **for the earliest unacknowledged frame,** this frees up at least one more slot or opens the available transmit window by at least 1 frame.

S can *deliver* frames to its user (process or protocol layer) only *in sequence*. It has to be able to hold a block of frames for this reason in SR. In general, if the user attached to S is busy and unable to accept an in-sequence frame from S, and the S window gets full, S will stop sending ACK's until its window opens again. This is one aspect of flow control. Thus even for go-back-N there is an S window of size 1 frame. In our analysis and subsequent discussion we assume that the user attached to S is always ready to get frames from S.

NOTE: If an ACK comes in for a frame which does **not include** the earliest frame awaiting acknowledgment, then the transmit window **does not** open up by 1. (This is related to how we have defined "Block of frames" above).
For **Continuous RQ using go-back-N**, this cannot happen and ACK's will always come in sequence. For **selective repeat** ACK's could return for a higher numbered frame when a lower numbered one is still awaiting an ACK at the transmitter. In such cases P could flood the receive window at S if it opens the send window by 1, assuming the receive window is of the same size K

Thus the parameter K must be thought of in the following way in general: the transmit window is open (available) if the **difference** between the earliest unacknowledged frame number (LWE or lower window edge) and the frame number of the next frame for transmission (UWE or upper window edge) is **UWE-LWE < K,** or else transmission is stopped. For example, if I-frame(5) is the earliest one awaiting an ACK at P and I-frame(9) is the next I-frame awaiting transmission, it can be transmitted as long as the send window size is larger than 4.

Similarly, S will hold frames that can fall in its allowed window at any time. If the S window is of size K, then the receiver will accept **up to** K correctly received frames beyond the last one it receives in sequence and acknowledges (sends on to its attached user). It can actually only accept any new correctly received frame for which the sequence number is no larger than K beyond the last in-sequence frame it has acknowledged. For example, in Selective Repeat if frame 5 and all before it have been received and acknowledged and the receive window size K=3, then at this point the receiver will accept any frame up to frame 8. The receiver will not accept 9 until at least 6 has come in, because it cannot deliver frames out of sequence and has to hold them. After getting 8 it might then receive 7 and then 6 correctly, at which point it sends out an ACK for all three and delivers all three in sequence to its network layer. If Frame 6 came first it would immediately ACK this and be ready to receive any frame up to number 9.

- For **Idle RQ** the **transmit window** size = **1**, **receive window** size **=1**
- For **Go-Back-N** the **transmit window** size = **K**, **receive window** size = **1**
- For **selective repeat**, S accepts out-of-sequence frames and the **receive window** size = **transmit window** size = **K**

Note that no matter what the protocol is, S can only pass on frames **in sequence** to its user.

7. Sequence Numbers for Continuous RQ with Transmit Window Size K

For **go-back-N**, if the window size at P is K, since at S it is always 1, we need at least **K+1 unique sequence numbers**. Suppose we have b bits available to carry the sequence number in each frame. Then we can form 2^b different numbers $\{0,1, ..., 2^b-1\}$, and the largest value for the transmit window size K is **$2^b - 1$.** With b=3 for example, we would use the sequence numbers 0,1,...7,0,1,2,...6,7,0,1,2,....... and be able to support a maximum window size of K=7. To show this is the best possible for K (i.e. we cannot use a larger K with

3-bit sequence numbers or 8 different numbers), consider what would happen if K=8 is used and all 8 I-frames numbered 0 through 7 are received correctly but their ACK's are lost on the return path. After a time-out period, P would send the first frame over again as I-frame(0). But S would interpret it as the next I-frame it is waiting for (beyond number 7), and would not be able to recognize it as a duplicate of an earlier frame.

For **selective repeat** we can show similarly that for transmit window size K (and receive window size therefore K) we need 2K unique sequence numbers. With b bits for sequence numbering we therefore are limited to a maximum window size of $K = \frac{1}{2} 2^b = 2^{b-1}$. This can be verified by considering what happens say for the case b=3 if any K larger than 4 is used. If K=5 is used and the first 5 I-frames are received correctly and ACKed, the receiver is open for 5 more frames numbered 5,6,7,0,1. If the ACKs are not received at the transmitter which then times-out and re-sends these frames, the receiver would begin to accept frames 0 and 1 as if they were new frames and not recognize that they are duplicates. With K=4 there is always a different set of numbers expected in consecutive windows.

8. Utilization Efficiencies for Continuous RQ

In the absence of errors the situation is fairly easy. If the transmit window size is K, and the round-trip delay is (1+2a) in I-frame durations, then

$$\boxed{U_{\text{no error}} = \frac{K}{1+2a} \text{ as long as } K < 1+2a \quad (\text{otherwise } U_{\text{no error}} = 1)}$$

This is because after K frames have been sent the P node has to wait until ACKs are received, the round trip delay being larger than K I-frame durations. Otherwise of course $U_{\text{no error}}=1$, since there is no waiting involved. If the round-trip delay is computed exactly to be T_t seconds (including processing delays and ACK frame durations) then we have $U_{\text{no error}} = \frac{KT_{ix}}{T_t}$ for $KT_{ix} < T_t$, obviously.

In the presence of errors we have to modify the formulas. One thing to note is that any of the formulas given for these cases of error are ***approximations;*** exact analysis is quite difficult in most cases.

Selective Repeat Utilization:

If P_f is the probability of frame error we have to compute the expected number of times a frame has to be transmitted (including retransmissions) for every frame that is correctly received. For Idle RQ this was simple because each time we retransmitted one frame. For **selective repeat** the same situation holds, since we retransmit only the frame that is in error. If the transmit window size **K is large enough** so that the flow of frames is not interrupted because of the need to repeat a frame, then we simply modify the U=1 result for this situation without errors to become **$U=1-P_f$** (using the same argument that we used for Idle RQ in the presence of errors). For K<1+2a, frame errors will also reduce the utilization efficiency by the same factor. In general, if $\mathbf{U_{no\ error}}$ is the *utilization without error for continuous RQ* than with frame error probability P_f we get

$$\mathbf{U_{with\ error} = U_{no\ error}\,(1-P_f)} \qquad \textbf{(Selective Repeat)}$$

Go-Back-N:

Here we have to consider the fact that when a frame is received in error the transmitter backs off to that frame number and starts retransmitting **all** frames starting from that frame, including those that it may already have sent out because its window size K is larger than 1.

If M ***re***-transmissions of a frame are needed (total number of transmissions for the frame is N_t=M+1), and K<1+2a, then a total of MK+1 frames need to be sent (one for the initial erroneous frame and M **re-**transmissions of K frames to get the initial frame correctly). On the other hand if K≥1+2a, we need to send only 1+2a and not K frames for each retransmit because the transmitter is able to know about an error in a round-trip time that is 1+2a frame durations under our simple approximations. (More generally we have to use more exact roundtrip times). We can then find the expected number of frames that have to be transmitted given P_f under both scenarios. We already know (from the Idle RQ analysis) that $E\{N_t\}= E\{M\}+1 = \frac{1}{1-P_f}$ so that $E\{MK+1\}= KE\{M\}+1 = \frac{KP_f}{1-P_f}+1 = \frac{KP_f+1-P_f}{1-P_f}$ This is the factor by which $U_{no\ error}$ decreases if K<1+2a, otherwise we have to replace K by 1+2a. We get therefore

$$U_{\text{with error}} = U_{\text{no error}} \frac{1-P_f}{1-P_f + \min\{(1+2a),K\}P_f} \qquad \text{(Go-Back-N)}$$

Note that the formula for the case $K \geq 1+2a$ for go-back-N in Halsall is not correct. Use the above result for Go-Back-N.

$U_{\text{no error}}$ in the expressions above for $U_{\text{with error}}$ for SR and go-back-N are given as K/(1+2a) or 1 for cont. RQ.

These formulas can be easily modified for cases where the round trip delay T_t is not just 1+2a frame durations. (How does the last result above change?).

9. Effective Rate

Once the average utilization rate U has been obtained, we note that if the transmit data rate is R bits/sec then on the average the **link is able to deliver** RU bits/sec. and the average **frame delivery rate** is $\frac{RU}{N_i}$ frames/sec., where N_i is the length in bits of each I-frame. Furthermore, since each frame carries overhead bits in addition to the actual network layer bits, the effective bit rate seen by the network layer is reduced from RU by a factor which is the ratio of the number of network-layer data bits D_i in a frame to the total frame length N_i. Thus **effective bit rate seen by the network layer** = $\frac{D_i RU}{N_i}$.

10. Further Notes

The protocols we have discussed can be implemented with different details; for example the go-back-N can be implemented without the use of NAK's, with transmitter time-outs and ACK's only. (This is the case for TCP/IP in the transport layer. TCP/IP does not use ARQ at the DLC layer).

In drawing frame sequence diagrams, it is not always necessary to also show the "link retransmission" and "link receive" lists at each new frame arrival at the P or S nodes. The resulting **simple frame sequence diagrams** are much quicker to draw. (Note that when we say "new frame arrival at the P node" we mean that the network layer at the P node has a packet to send that it forwards to its DLC layer, where the I-frame is readied for transmission.)