III. Link Layer ARQ Protocols

Instructor: Patrick Tague **Date**: 14 January 2008

EE 565: Computer-Communication Networks I Winter Quarter 2008

1 Automatic Repeat reQuest (ARQ)

The goal of **Automatic Repeat reQuest (ARQ)** at the link layer is achieve *reliable transmission* between the sending and receiving link layers. Transmission at the link layer is said to be reliable only if the receivin link layer releases packets fo the network layer without errors, in order, and without duplicates or omissions. In order to achieve reliability, the ARQ algorithm must include error detection, acknowledgment of frame reception, recovery from frame loss (e.g. timeout), and retransmission. In formulating an ARQ algorithm, the first property of interest is correctness, i.e. ensuring reliable transmission. The second property of an ARQ algorithm is efficiency, related to the waste of bit-transmission resources due to unnecessary retransmission and waiting. To measure the degree of unnecessary retransmission, we define the **packet rate** η as the average number of packets delivered to the receiver B per frame sent from A to B. If each frame is lost with probability p, the packet rate is necessarily bounded as $\eta \leq 1 - p$. As a measure of time efficiency in the absence of frame loss, we define the **utilization** U of the channel (or of the sender) as the fraction of the total frame exchange time that is spent sending bits into the channel, or

$$U = \frac{\text{time spent sending bits into the channel}}{\text{total time using channel}}. \tag{1}$$

We'll address three types of ARQ algorithms, stop and wait ARQ, go back n ARQ, and selective repeat ARQ.

2 Stop and Wait ARQ

The basic idea of **stop and wait ARQ** is that the sender A transmits a frame to the receiver B and waits to receive either (1) a (positive) acknowledgment (ACK) of correct reception, at which point A will transmit a new frame corresponding to the next packet, or (2) a negative acknowledgement (NAK) of incorrect reception, at which point A will re-transmit a frame with the same packet data as the incorrectly received frame. Since it is also possible that frames are lost between A and B (e.g. due to framing errors), it may also be necessary for A and B to periodically re-transmit or request re-transmission, respectively, referred to as a *timeout*.

Since it is possible that frames are lost between A and B (e.g. due to framing errors), it may be necessary for A and B to periodically re-transmit or re-request frames, respectively, referred to as a *timeout* and requiring each sender and receiver to incorporate a timer into the ARQ algorithm.

Next, suppose the sender A transmits a frame to the receiver B, the frame is correctly received by B, but the ACK from B to A is either lost or receceived by A after a timeout has occurred. The next frame received by B will then contain the same packet data as the correctly received frame. If there is no additional mechanism to distinguish between the packets, the receiver B is unable to determine whether the second frame is (1) a duplicate of the previous frame corresponding to the same network layer packet or (2) a frame corresponding to a second network layer packet that happens to have the same data. To solve this problem, a sequence number SN is used to uniquely identify network layer packets. Similarly, we must include a request number RN in each ACK frame, indicating the SN value of the next packet expected by the receiver.

An additional benefit of using RN is that different ACK and NAK packet types are not required. The receiver can simply increment RN to indicate an ACK but leave RN the same to indicate a NAK. Furthermore, if there is a second data stream being transmitted from B to A, the packet requests for the $B \rightarrow A$ stream can be "piggybacked" onto the frame transmissions for the $A \rightarrow B$ stream.

The stop and wait algorithm for a single $A \rightarrow B$ data stream is given in Fig. 1.

For node A:

1. Set SN to 0.
2. Accept packet from network layer (or wait for it), assign number SN to new packet.
3. Transmit packet SN in a frame with SN in the sequence number field and start a timer.
4. If error-free frame from B is received with $RN > SN$, increment SN and go to step 2. If the timer expires and no such frame is received, go to step 3.

For node B:

1. Set RN to 0.
2. Start a timer. If error-free frame from A is received with $SN = RN$ before the timer expires, release packet to network layer and increment RN. Otherwise, wait for the timer to expire.
3. Transmit frame with RN and go to step 2.

Figure 1: The Stop and Wait ARQ algorithm is given for a single $A \rightarrow B$ data stream.

2.1 Efficiency of Stop and Wait ARQ

Suppose the sender A using a stop and wait ARQ protocol begins a transmission of an L bit frame over a link of rate R at time $t = 0$. A is actually pushing bits onto the link for a total of L/R seconds, stopping transmission at time $t = L/R$. At time $t = L/R + RTT/2$, the last bit in the frame has arrived at B. Assuming the acknowledgment frame is negligibly small and ignoring the processing and queueing delays, A will receive the acknowledgment at time $t = L/R + RTT$. The utilization U of the exchange is thus given by (1) as

$$U = \frac{L/R}{L/R + RTT}.$$

As indicated, the utilization U is low whenever the transmission delay L/R is small relative to the round-trip time RTT.

As an example, suppose a frame of 1000 bits is transmitted over a 100 Mbps link between two nodes separated by a distance of 10 km and assume that bits propagate over the link at a speed of 2×10^8 m/s. The value of RTT is approximately equal to twice the separation distance divided by the propagation speed, or

$$RTT = \frac{2 \times 10^4 \text{ m}}{2 \times 10^8 \text{ m/s}} = 10^{-4} \text{ s},$$

or 100 μs. The transmission delay L/R is computed as

$$L/R = \frac{10^3 \text{ bits}}{10^8 \text{ bits/s}} = 10^{-5} \text{ s},$$

or 10 μs. The utilization U is thus equal to $U = 1/11$, indicating that the channel is being used with an efficiency of approximately 9%.

3 Go Back n ARQ

To remedy the poor utilization of the channel under stop and wait ARQ protocols, we investigate ARQ protocols which allow the sender A to send multiple frames in sequence without waiting for corresponding acknowledgments. The first type of generalized ARQ protocols is **go back n ARQ** in which the sender is allowed to transmit n frames before stopping to wait for acknowledgments. The receiver's algorithm under go back n is the same as in stop and wait ARQ.

In the sender's algorithm for go back n ARQ, the sender A keeps a *window* of n packets, numbered i through $i + n - 1$ (denoted $[i, i + n - 1]$), passed down from the network layer

such that packets 0 through $i-1$ have already been acknowledged by B. As each successive packet is acknowledged, the window is moved, i.e. $i \leftarrow i+1$ and $i+n-1 \leftarrow i+n$, maintaining a length of n. This type of algorithm is thus often referred to as a *sliding-window* algorithm.

Suppose the current sender window is $[i, i+n-1]$. Since B is waiting for only one packet numbered RN, we may run into trouble if a frame is lost. Suppose $RN = i$ and the frame with $SN = i$ is lost. B may still receive frames with $SN = i+1, \ldots, i+n-1$, but these frames are ignored. If desired, the receiver can send a retransmit request with $RN = i$, potentially one such request for each of the $n-1$ ignored packets. Once A finishes processing the n packets in the window (or stops doing so due to the repeated $RN = i$ requests from B), frames will be re-transmitted starting with $SN = i$. This step requiring the SN to reduce to the beginning of the window is where this type of ARQ gets its name. We also notice that if the acknowledgment/request labeled $RN = i$ is lost but one labeled $RN = i+1$ is received by the sender, both packets numbered $SN = i-1$ and $SN = i$ are acknowledged. This concept is referred to as the *cumulative acknowledgment* property of sliding-window algorithms, as reception of a request $RN = i$ implicitly acknowledges all packets with $SN < i$.

We note that it is not necessary for SN and RN to grow unbounded, as the window size is finite. The algorithm can be modified to use sequence numbers computed modulo m, where $m > n$. This is sometimes referred to as **go back n mod m ARQ**. As an exercise, consider what happens if $m = n$.

3.1 Efficiency of Go Back n ARQ

In the absence of errors, the utilization U given in (1) can be pushed quite near 1. The time spent transmitting frames is equal to nL/R, while the total channel utilization time is equal to $L/R + RTT$, here assuming $(n-1)L/R < RTT$. If the round-trip time RTT can be estimated, the value of n can be chosen as the largest n satisfying $(n-1)L/R < RTT$. In general, however, estimation of RTT is non-trivial.

There are three sources of error when using go back n ARQ. First, a data frame can be lost in the forward direction (sender $\rightarrow$ receiver), eventually leading to a timeout and retransmission of the window of n frames. Second, an ACK frame can be lost in the reverse direction (sender $\leftarrow$ receiver), which may lead to a timeout, but may be caught by a cumulative acknowledgement. The third source of error occurs only when the reverse direction ACK frames are piggybacked onto reverse-direction traffic and the reverse-direction frames are much longer than the forward-direction frames. For example, see Figure 2.31 in the text. From the above, we see that the packet rate η can be significantly less than the optimal value $\eta = 1-p$ due to loss of the entire window of frames when a single frame is lost.

4 Selective Repeat ARQ

Go back n ARQ is effective in improving the poor utilization of stop and wait ARQ. However, go back n ARQ still suffers from poor efficiency as a large number (as high as n) of frames are retransmitted each time there is an error in the forward direction, resulting in a low packet rate. This can be remedied, however, by allowing the receiver to buffer out-of-order packets and request retransmission of specific packets, instead of the entire window of n packets. This technique, appropriately named **selective repeat ARQ**, introduces the inclusion of a second window at the receiver side. There are many ways that the receiver can request retransmissions of specific packets. If the probability of frame loss is very small, it is likely that a single frame was lost, so the receiver can specify a single RN value, and the sender can send only the packet with $SN = RN$, hoping the receiver will then respond with a significantly higher RN value, indicating the cumulative acknowledgment of several subsequent frames. Ideally, selective repeat can achieve a packet rate η very near $1 - p$. For details, consult the text.