ARQ and Flow Control

Automatic Repeat reQuest (ARQ) is a technique to ensure reliable delivery of data by retransmitting frames that are received with errors or not at all. Flow control is the term for various techniques to stop or slow down transmissions to match the rate at which a receiver can accept data.
After this lecture you should be able to: explain how ACK frames ensure error-free transmissions; select an appropriate type of ARQ (from stop-and-wait, go-back-N and selective repeat) based on channel error rate and delay; and select appropriate flow-control method(s) to avoid over-flows.

Retransmission Protocols

Many frame-oriented data communication protocols require that the receiver acknowledge correct reception of each frame to ensure that no data is lost.

When a message is received without errors the receiver sends an "ACK" (acknowledgment) packet back to the sender to confirm correct reception.

A transmitter using ARQ sets a timer when a frame is transmitted. If the timer "goes off" (expires) before the receiver receives the corresponding ACK frame it knows the frame (or the ACK) was not correctly received and retransmits the frame. This technique is called ARQ (for Automatic Repeat reQuest) because the receiver does not have to ask for the frame to be retransmitted.

There are various types of ARQ. The version just described is called **Stop and Wait ARQ**. The transmitter waits after sending each frame until it receives the ACK or the timer expires. This is the simplesty type of ARQ since the transmitter only has to store one frame in case a retransmission is required. However, it is inefficient if there is a long delay between transmitter and receiver.

Exercise 1: Considering only the propagation delays, what is the minimum delay between transmitted frames if no ACKs are lost?

Efficiency can be improved by allowing the transmitter to send more than one frame without waiting for each one to be acknowledged. The transmitter adds a serial number (or a count of the total packets or bytes transmitted thus far) to each frame. The serial number of count for the most recently received packet is sent back with each ACK. This version of ARQ is more complex since the transmitter must store all un-acknowledged frames. This increases memory requirements at the transmitter.

There are also two ways that retransmissions can be handled:

The first, **Go Back N ARQ**, requires the transmitter to transmit all frames starting at the first unacknowledged one.

The second, **Selective Repeat ARQ**, allows the receiver to acknowledge individual frames (or ranges of byte counts) received. This is more efficient because the transmitter then has to retransmit the lost frames. However, it makes the implementation more complex because the transmitter must store all unacknowledged frames in case they are not ACK'ed and the receiver must store any frames not received in order so that the correct sequence of packets can be reassembled.

Exercise 2: Create a table summarizing the three different types of ARQ. Include: throughput, transmitter memory, receiver memory and relative complexity.

Exercise 3: A data communication system operates at 1 Mb/s and uses 10000-bit data frames and 100-bit ACK frames. What are the frame durations? What is the throughput if there is no channel delay and no errors? If the round-trip channel delay is a 0.5s (typical for satellite links)? If go-back-N ARQ is used, assuming the transmitter can store all unacknowledged frames?

Exercise 4: Assume a transmitter has an ARQ timeout that is 5 packet durations and fails to get an ACK for every 10th frame (e.g. due to periodic noise bursts). Ignoring ACK delay and overhead, what is the throughput using go-back-N ARQ? Using Selective ARQ?

In many protocols ACKs can be appended as "piggyback" information on data frames being sent in the reverse direction. This reduces the overhead of sending ACKs.

Some protocols allow the use of a negative acknowledgment (NACK) that allows a receiver to ask for a retransmission if it knows that a frame was lost (e.g. a packet is received out of order). This can be faster than waiting for a timeout.

Flow Control

If the data sink cannot accept data as quickly as it is being received, it is possible for the received data to overflow the "buffer" memory available in a receiver.

To avoid such overflows we can use "flow control" signals. For example, on serial interfaces a "clear-to-send" signal output by the receiver can tell the transmitter when the receiver is ready to accept data. This is often called "hardware" flow control.

To avoid the need for a dedicated flow-control signal line the receiver can send special characters to the transmitter to tell it when to stop and start sending data. This is called "software" flow control. The XON (start sending, also control-Q) and XOFF (stop sending, also control-S) ASCII control characters are often used for this. However, this means that these characters cannot be used as data making it more difficult to sent binary data.

Another flow control method is to use ARQ with a limited "window" size (N). The sink can wait to output ACKs for received frames until there is room for additional frames.

Exercise 5: Which of the above flow control methods can be used with frame-oriented protocols? Which can be used on unidirectional links?