FLOW AND ERROR CONTROL

Data link control functions include framing, flow and error control, and software implemented protocols that provide smooth and reliable transmission of frames between nodes.

FRAMING

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination. The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing. The data link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another.

Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Frame Size

Frames can be of fixed or variable size:

1) Fixed-Size Framing

In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells.

2) Variable-Size Framing

In variable-size framing, we need to define the end of the frame and the beginning of the next. Two approaches are used for this purpose: a character-oriented approach and a bit-oriented approach.

- ***Character-Oriented Protocols***

In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII . The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame.

Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to character-oriented framing.

Byte Stuffing: In *byte stuffing* (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag. Thus, byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. If the text contains one or more escape characters followed by a flag. The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if

the escape character is part of the text, an extra one is added to show that the second one is part of the text.

- ***Bit-Oriented Protocols***

In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame.

This flag can create a problem. If the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of I byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing.

Bit Stuffing: In *bit stuffing*, if a 0 and five consecutive I bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. Thus, bit stuffing is the process of adding one extra 0 whenever five consecutive 1's follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

This means that if the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

FLOW AND ERROR CONTROL

The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.

- **Flow Control**

Flow control coordinates the amount of data that can be sent before receiving an acknowledgment. In most protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. The flow of data must not be allowed to overwhelm the receiver. Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily.

Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called a *buffer,* reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

- **Error Control**

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term *error control* refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called *Automatic Repeat Request (ARQ)*.

FLOW CONTROL PROTOCOLS

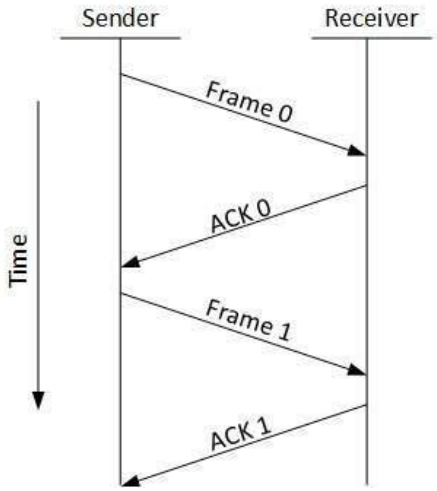
The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer.

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

Two types of mechanisms can be deployed to control the flow:

- **Stop-and-Wait Protocol**

In this method of flow control, the sender sends a single frame to receiver & waits for an acknowledgment. The next frame is sent by sender only when acknowledgment of previous frame is received. This process of sending a frame & waiting for an acknowledgment continues as long as the sender has data to send. To end up the transmission sender transmits end of transmission (EOT) frame.



The main advantage of stop & wait protocol is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So there is no chance of frame being lost.

The main advantage of stop & wait protocol is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So there is no chance of frame being lost.

- **Sliding Window Protocol**

In sliding window method, multiple frames are sent by sender at a time before needing an acknowledgment. Multiple frames sent by source are acknowledged by receiver using a single ACK frame. Sliding window refers to imaginary boxes (buffer) that hold the frames on both sender and receiver side. It provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment. Frames may be acknowledged by receiver at any point even when window is not full on receiver side.

The windows have a specific size in which the frames are numbered modulo-n, which means they are numbered from 0 to n-l. For e.g. if n = 8, the frames are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ...

When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver

sends an ACK containing the number 5. When sender sees an ACK with number 5, it comes to know that all the frames up to number 4 have been received.

ERROR CONTROL PROTOCOLS

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.

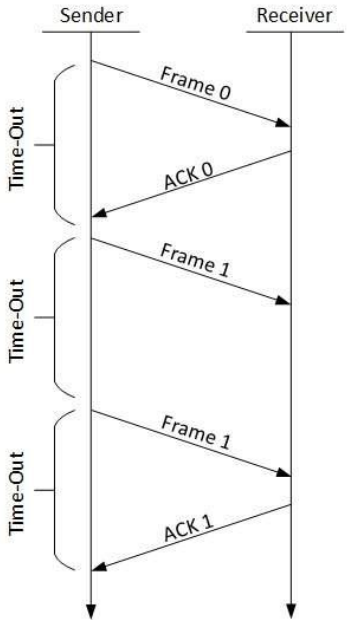
Requirements for error control mechanism:

- **Error detection** - The sender and receiver, either both or any, must ascertain that there is some error in the transit.
- **Positive ACK** - When the receiver receives a correct frame, it should acknowledge it.
- **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.
- **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 it's acknowledgement is lost in transit.

There are three types of techniques available which Data-link layer may deploy to control the errors by Automatic Repeat Requests (ARQ):

- **Stop-and-wait ARQ**

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. In Stop-and-Wait ARQ, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.



Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. The ACK frame for this protocol has a sequence number field. In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

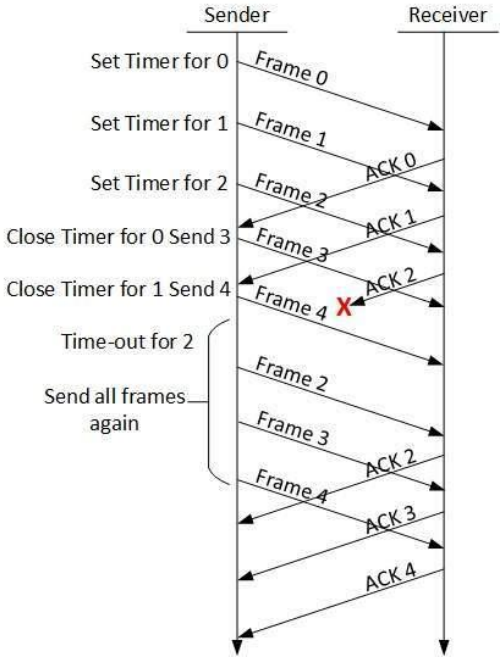
In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic. When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver sends an ACK containing the number 5.

- **Go-Back-N ARQ**

Stop and wait ARQ mechanism does not utilize the resources at their best. When the acknowledgement is received, the sender sits idle and does nothing.

In Go-Back-N ARQ method, both sender and receiver maintain a window. The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.

When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not received any ACK for a particular frame, it retransmits all the frames from that particular frame (including that frame as well) for which it has not received any positive ACK.



In Go-Back-N ARQ, frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows *m* bits for the sequence number, the sequence numbers range from 0 to *2m* - 1. For example, if *m* is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

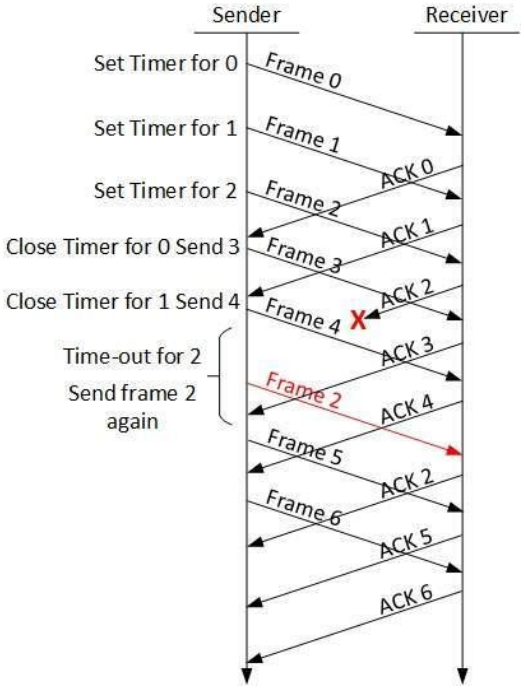
In other words, the sequence numbers are modulo-2^m.

- **Selective Repeat ARQ**

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher

probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ.

In Selective-Repeat ARQ, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged. The sender in this case, sends only packet for which NACK is received.



The Selective Repeat Protocol also uses two windows: a send window and a receive window both of the same size. The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered.

PIGGYBACKING

In all practical situations, the transmission of data needs to be bi-directional. This is called as full-duplex transmission. We can achieve this full duplex transmission *i.e.* by having two separate channels-one for forward data transfer and the other for separate transfer *i.e.* for acknowledgements. A better solution would be to use each channel (forward & reverse) to transmit frames both ways, with both channels having the same capacity. If A and B are two users. Then the data frames from A to B are intermixed with the acknowledgements from A to B.

One more improvement that can be made is piggybacking. The concept is explained as follows:

In two way communication, whenever a data frame is received, the received waits and does not send the control frame (acknowledgement) back to the sender immediately. The receiver waits until its network layer passes in the next data packet. The delayed acknowledgement is then attached to this outgoing data frame. This technique of temporarily delaying the acknowledgement so that it can be hooked with next outgoing data frame is known as *piggybacking.*

The major advantage of piggybacking is better use of available channel bandwidth.

The disadvantages of piggybacking are:

1. Additional complexity.

2. If the data link layer waits too long before transmitting the acknowledgement, then retransmission of frame would take place.