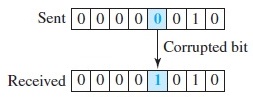
**Chapter 6: Data Link Layer**

1. **Introduction**

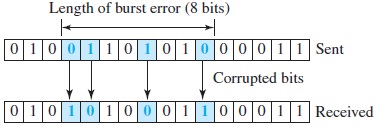
The data link layer divides the stream of bits received from the network layer into manageable data units called frames. Specific responsibilities of the data link layer include framing, addressing, flow control, error control, and media access control.

**Types of Errors**

1. Single-Bit Error



1. Burst Error



**Redundancy**

The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

1. **Block coding**

In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword.

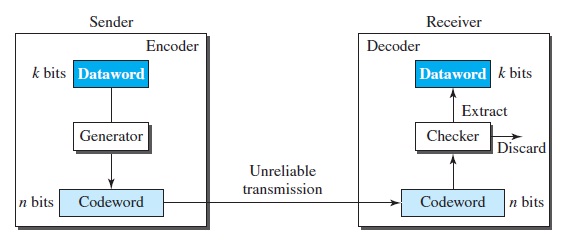


Figure 1. Process of error detection in Block coding

Let us assume that k = 2 and n = 3. Table 1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataword | Codeword | Dataword | Codeword |
| 00 | 000 | 10 | 101 |
| 01 | 011 | 11 | 110 |

Table 1. A code for error detection

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

**Hamming Distance**

The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits.

For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is d(00000, 01101) = 3.

**Minimum Hamming Distance**

In a set of codewords, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of codewords.

1. **Parity-Check Code**

In this code, a k-bit dataword is changed to an n-bit codeword where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s, we discuss the even case. The minimum Hamming distance for this category is dmin = 2, which means that the code is a single-bit error-detecting code.

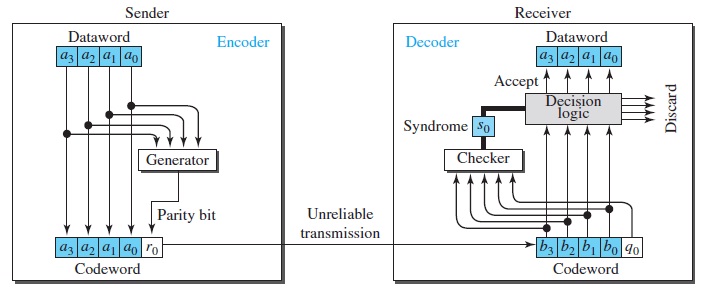


Figure 2. Encoder and decoder for a simple parity check code

The encoder uses a generator that takes a copy of a 4-bit dataword (a0, a1, a2, and a3) and generates a parity bit r0. The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even. This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even.

The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the syndrome, is just 1 bit. The syndrome is 0 when the number of 1s in the received codeword is even; otherwise, it is 1. The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no detectable error in the received codeword; the data portion of the received codeword is accepted as the dataword; if the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

1. **Cyclic Redundancy Check**

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

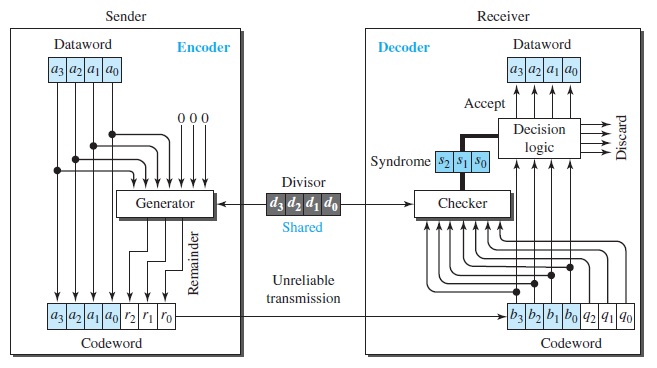


Figure 3. CRC encoder and decoder

In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding n − k (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n − k + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r2r1r0) is appended to the dataword to create the codeword.

The decoder receives the codeword (possibly corrupted in transition). A copy of all n bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of n − k (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded.

1. **Data Link Layer Protocols**
   1. **Stop and Wait ARQ**

The Stop and Wait ARQ is derived from the Stop and Wait Protocol which relies on acknowledgements (ACKs) from the receiver to maintain a consistent flow of data during transmission. It adds error control mechanism to the Stop and Wait Protocol.

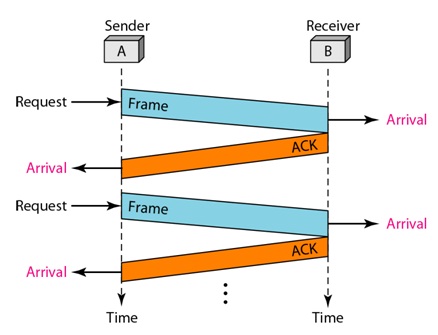


Figure 4. Stop and Wait Protocol

An error is said to occur during transmission of data frames when either the frames get lost or they get corrupted. When a receiver receives a corrupted frame, it silently discards it and sends no acknowledgements to the sender. Since the sender receives no ACKs in both the cases, it assumes that the frame was not received by the receiver and after a time out period, the sender re-sends the frames. The sender knows which frame to send because it keeps a copy of the last sent frame.

Additionally, the sender also uses sequence numbers to number each frame that it sends. This helps the receiver in identifying duplicate frames. The sequence number alternates between 0 and 1. The ACKs from the receiver contain the sequence number of the next frame that it expects from the sender. This is called the acknowledgement number. If the receiver has received frame with sequence number 0, it sends 1 as the ACK and vice versa.

Another case in which an error is said to occur is when the ACK from the receiver is lost. The sender will not know if the frame was lost or the ACK was lost. Hence the sender re-sends after the time out.

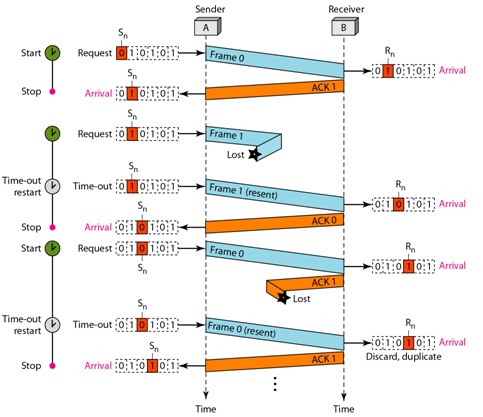


Figure 5. Stop and Wait ARQ

Figure 5 shows an example of Stop-and-Wait ARQ. Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

**Efficiency of Stop and Wait ARQ**

Since the sender can send only one frame before receiving an ACK, there will be only frame at a time in the channel connecting the sender and the receiver. As a result, the capacity of the channel is not fully utilized. Also, the sender can send no other frame before it receives an ACK for the previous frame. Hence the Stop and Wait ARQ is not efficient.

* 1. **Go-Back N ARQ**

The Go-Back N ARQ overcomes the disadvantage of the Stop and Wait ARQ by allowing the sender to send more than one frame before receiving ACKs. This keeps the channel busy and increases efficiency.

The mechanism is similar to the Stop and Wait ARQ except this protocol allows more than one frame to be transmitted at a time. The sender keeps a copy of all the last sent frames until it receives ACKs for those. This protocol uses the concept of a sliding window which defines the number of frames that a sender can send at a time. Also, the frames need to assigned sequence numbers. If the header of a frame allows m bits to be used for the sequence number, the sequence numbers can range from 0 to 2m – 1. For example, if m = 2, the sequence numbers can range from 0 to 3. The sequence numbers repeat after the end of the range is reached. So, the sequence numbers are:-

0, 1, 2, 3, 0, 1, 2, 3, 0, ……..

The maximum size of the sliding window can be 2m – 1. Here, the maximum size can be 3. The following depicts an instance of the sliding window:-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ………. | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | ……… |

Frames already acknowledged Frames sent but Frames that cannot be sent

not acknowledged

1. Send window before sliding

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ………. | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | ……… |

1. Send window after sliding

Figure 6. Send window

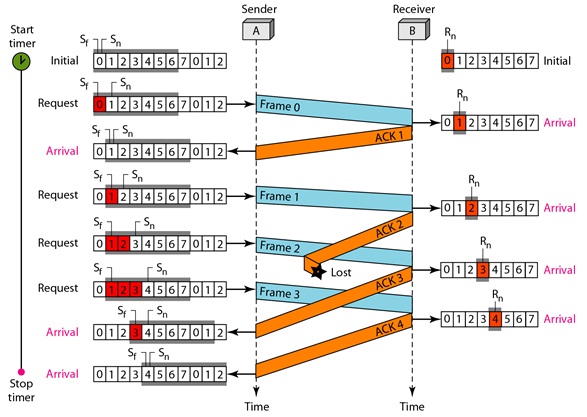


Figure 7. Go back N protocol

Figure 7 shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order. The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames 1, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with sending frame 3.

* 1. **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.

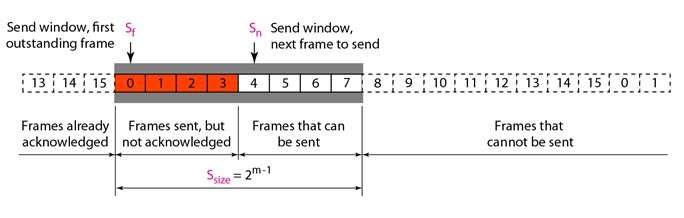


Figure 8. Send window for Selective Repeat ARQ

The send window maximum size can be 2m-1. For example, if m = 4, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the Go-Back-N Protocol). The smaller window size means less efficiency in filling the pipe.

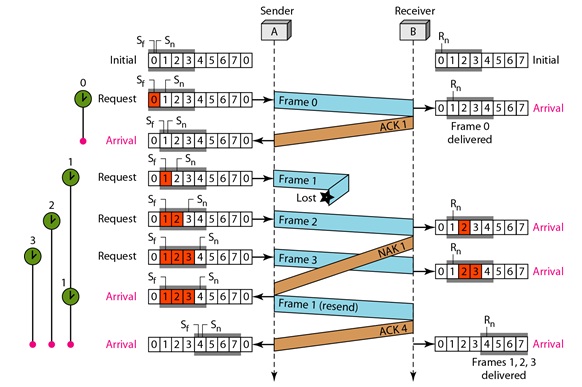


Figure 9. Selective Repeat flow diagram

Figure 9 illustrates the concept of Selective Repeat ARQ. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window.

Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.