MEPL

IBPS SO PREPRATION

Data Link Layer

SHOBHIT BANDHU

15

Data-link Layer Introduction

Introduction

Data Link Layer is second layer of OSI Layered Model. This layer is one of the most complicated layers and has complex functionalities and liabilities. Data link layers hides the details of underlying hardware and represents itself to upper layer as the medium to communicate.

Data link layer works between two hosts which are directly connected in some sense. This direct connection could be point to point or broadcast. Systems on broadcast network are said to be on same link. The work of data link layer tends to get more complex when it is dealing with multiple hosts on single collision domain.

Data link layer is responsible for converting data stream to signals bit by bit and to send that over the underlying hardware. At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals assembles them in a recognizable frame format, and hands over to upper layer.

Data link layer has two sub-layers:

- **Logical Link Control:** Deals with protocols, flow-control and error control
- **Media Access Control:** Deals with actual control of media

Functionality of Data-link Layer

Data link layer does many tasks on behalf of upper layer. These are:

- **Framing:**

  Data-link layer takes packets from Network Layer and encapsulates them into Frames. Then, sends each Frame bit-by-bit on the hardware. At receiver's end Data link layer picks up signals from hardware and assembles them into frames.

- **Addressing:**

  Data-link layer provides layer-2 hardware addressing mechanism. Hardware address is assumed to be unique on the link. It is encoded into hardware at the time of manufacturing.

- **Synchronization:**

  When data frames are sent on the link, both machines must be synchronized in order to transfer to take place.

- **Error Control:**

  Sometimes signals may have encountered problem in transition and bits are flipped. These error are detected and attempted to recover actual data bits. It also provides error reporting mechanism to the sender.

- **Flow Control:**

  Stations on same link may have different speed or capacity. Data-link layer ensures flow control that enables both machines to exchange data on same speed.

- **Multi-Access:**

  Hosts on shared link when tries to transfer data, has great probability of collision. Data-link layer provides mechanism like CSMA/CD to equip capability of accessing a shared media among multiple Systems

Error Detection and Correction

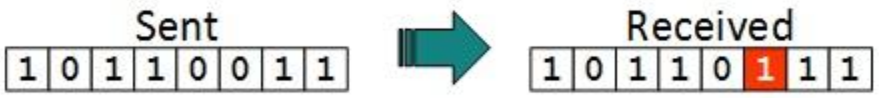
Introduction

There are many reasons such as noise, cross-talk etc., which may help data to get corrupted during transmission. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing. So, upper layers expect error-free transmission between systems. Most of the applications would not function expectedly if they receive erroneous data. Applications such as voice and video may not be that affected and with some errors they may still function well.

Data-link layer uses some error control mechanism to ensure that frames *data bit streams* are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.

There may be three types of errors:

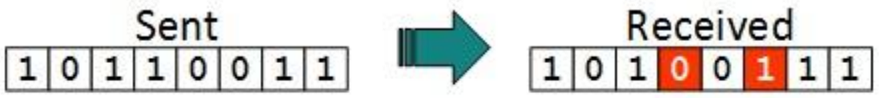
- **Single bit error:**



[*Image: Single bit error*]

In a frame, there is only one bit, anywhere though, which is corrupt.

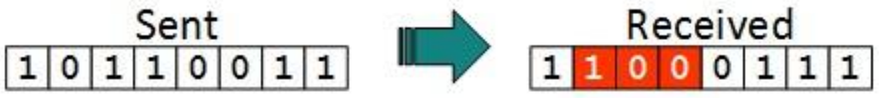
- **Multiple bits error:**



[*Image: Multiple bits error*]

Frame is received with more than one bits in corrupted state.

- **Burst error:**



[*Image: Burst error*]

Frame contains more than1 consecutive bits corrupted.

Error control mechanism may involve two possible ways:

- Error detection
- Error correction

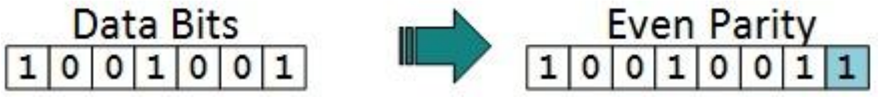
Error Detection

Errors in the received frames are detected by means of Parity Check and CRC *Cyclic Redundancy Check*. In both scenarios, few extra bits are sent along with actual data to confirm that bits received at other end are same as they were sent. If the checks at receivers end fail, the bits are corrupted.

Parity Check

One extra bit is sent along with the original bits to make number of 1s either even, in case of even parity or odd, in case of odd parity.

The sender while creating a frame counts the number of 1s in it, for example, if even parity is used and number of 1s is even then one bit with value 0 is added. This way number of 1s remain even. Or if the number of 1s is odd, to make it even a bit with value 1 is added.



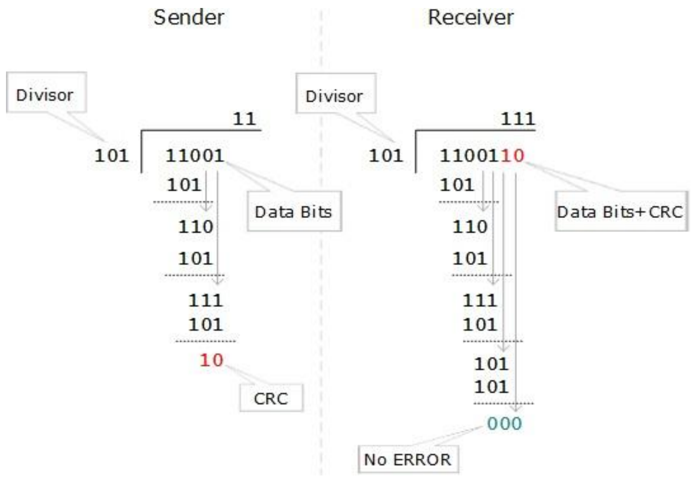
[*Image: Even Parity*]

The receiver simply counts the number of 1s in a frame. If the count of 1s is even and even parity is used, the frame is considered to be not-corrupted and is accepted. If the count of 1s is odd and odd parity is used, the frame is still not corrupted.

If a single bit flips in transit, the receiver can detect it by counting the number of 1s. But when more than one bits are in error it is very hard for the receiver to detect the error

Cyclic Redundancy Check

CRC is a different approach to detect if the frame received contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polyno mials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a codeword. The sender transmits data bits as code-words.



[*Image: CRC in action*]

At the other end, the receiver performs division operation on code-words using the same CRC divisor. If the remainder contains all zeros the data bits are accepted, otherwise there has been some data corruption occurred in transit.

Error Correction

In digital world, error correction can be done in two ways:

- **Backward Error Correction:** When the receiver detects an error in the data received, it requests back the sender to retransmit the data unit.
- **Forward Error Correction:** When the receiver detects some error in the data received, it uses an error-correcting code, which helps it to auto-recover and corrects some kinds of errors.

The first one, Backward Error Correction, is simple and can only be efficiently used where retransmitting is not expensive, for example fiber optics. But in case of wireless transmission retransmitting may cost too much. In the latter case, Forward Error Correction is used.

To correct the error in data frame, the receiver must know which bit *location of the bit in the frame* is corrupted. To locate the bit in error, redundant bits are used as parity bits for error detection. If for example, we take ASCII words 7*bitsdata*, then there could be 8 kind of information we need. Up to seven information to tell us which bit is in error and one more to tell that there is no error.

For m data bits, r redundant bits are used. r bits can provide 2r combinations of information. In m+r bit codeword, there is possibility that the r bits themselves may get corrupted. So the number of r bits used must inform about m+r bit locations plus no-error information, i.e. m+r+1.

$$2^r >= m+r+1$$

Data-link Control and Protocols

Introduction

Data-link layer is responsible for implementation of point-to-point flow and error control mechanism.

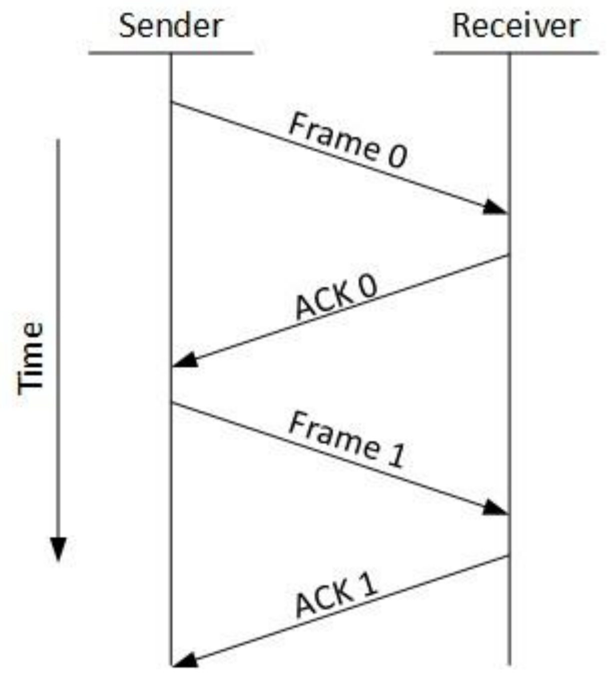
Flow Control

When data frames *Layer*−2 *data* is sent from one host to another over a single medium, it is required that the sender and receiver should work on same speed. That is, sender sends at a speed on which the receiver can process and accept the data. What if the speed *hardware*/*software* of the sender or receiver differs? If sender is sending too fast the receiver may be overloaded *swamped* and data may loss.

Two types of mechanism can be deployed in the scenario to control the flow:

- **Stop and Wait**

  This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received.



[*Image: Stop and Wait Protocol*]

- **Sliding Window**

  In this flow control mechanism both sender and receiver agrees on the number of data-frames after which the acknowledgement should be sent. As we have seen, stop and wait flow control mechanism wastes resources, this protocol tries to make use of underlying resources as much as possible.

Error Control

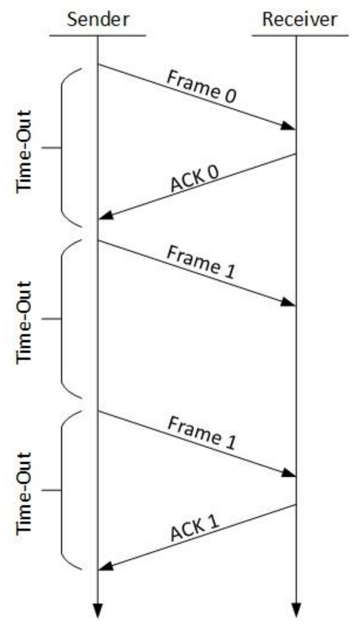
When data-frame is transmitted there are probabilities that data-frame may be lost in the transit or it is received corrupted. In both scenarios, the receiver does not receive the correct data-frame and sender does not know anything about any loss. In these types of cases, both sender and receiver are equipped with some protocols which helps them to detect transit errors like data-frame lost. So, either the sender retransmits the data-frame or the receiver may request to repeat the previous data-frame.

Requirements for error control mechanism:

- **Error detection:** The sender and receiver, either both or any, must ascertain that there's been some error on 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 in the timeout period, the sender retransmit the frame, thinking that the frame or it's acknowledge 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



[*Image: Stop and Wait ARQ*]

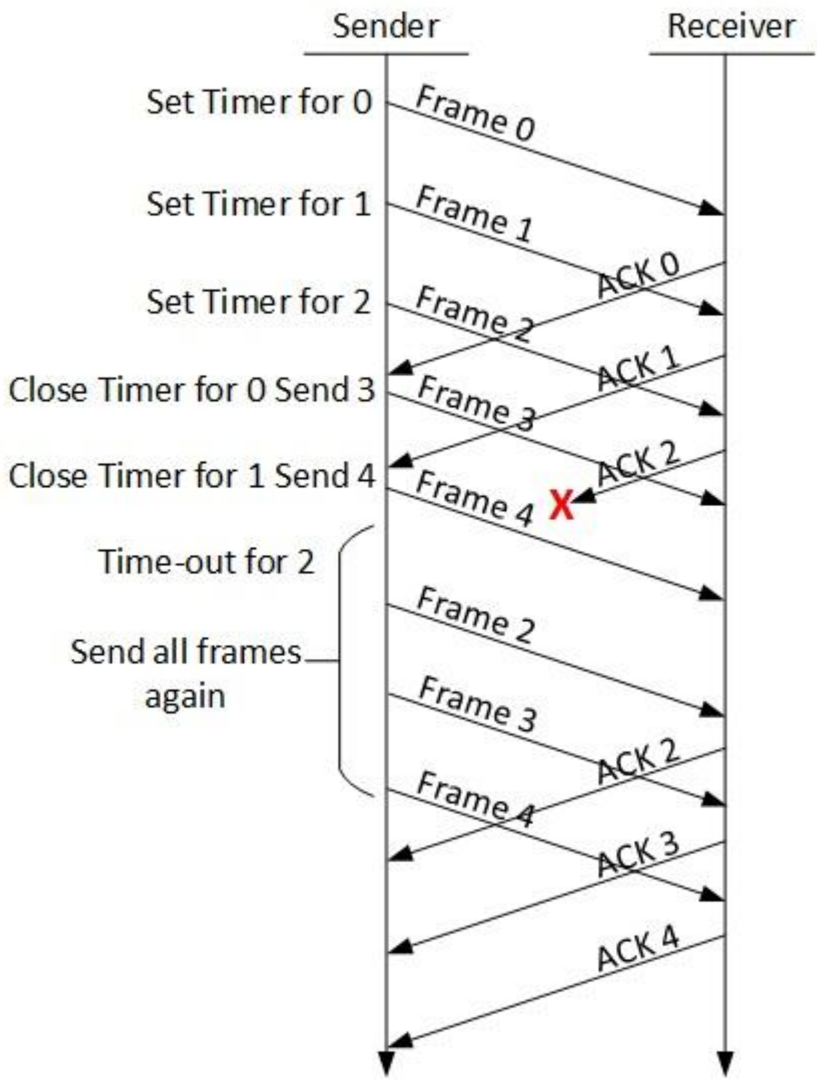
The following transition may occur in Stop-and-Wait ARQ:

- The sender maintains a timeout counter.
- When a frame is sent the sender starts the timeout counter.
- If acknowledgement of frame comes in time, the sender transmits the next frame in queue.

    - If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.
    - If a negative acknowledgement is received, the sender retransmits the frame.

- Go-Back-N ARQ

Stop and wait ARQ mechanism does not utilize the resources at their best. For the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintains a window.



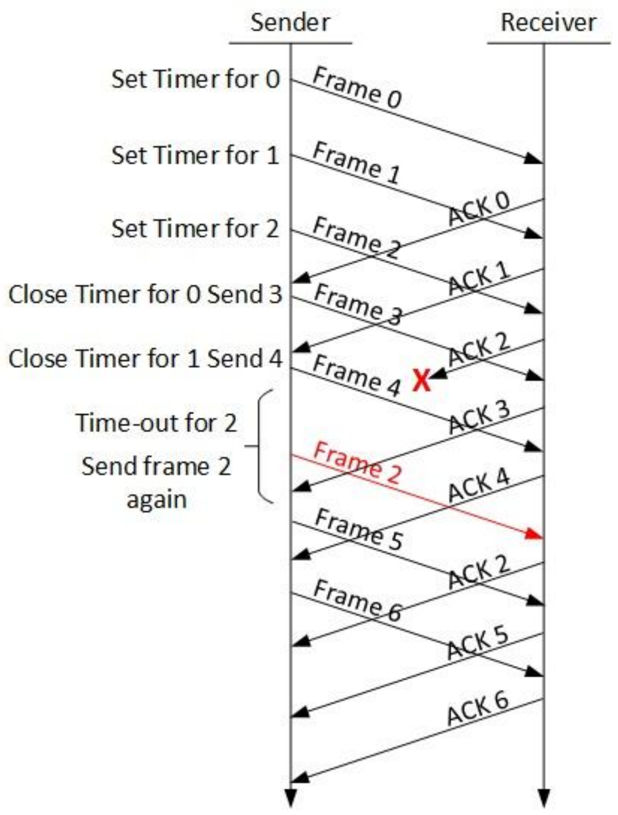
[*Image: Go-Back-N ARQ*]

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 ACK. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

- Selective Repeat ARQ

In Go-back-N ARQ, it is assumed that the receiver does not have any buffer space for its window size and has to process each frame as it comes. This enforces the sender to retransmit all the frames which are not acknowledged.



[*Image: 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.