Computer Networks: The Data Link Layer

1. Data Link Layer Functions and Services

The **Data Link Layer** is Layer 2 of the OSI model and is responsible for transferring data between adjacent network nodes. It takes the packets from the Network Layer and encapsulates them into **frames** for transmission across the physical medium.

i. Data Link Layer Design Issues

This layer addresses several crucial challenges to ensure reliable and efficient data transfer:

- **Framing:** Breaking down the stream of raw bits from the Physical Layer into discrete units called frames.
- **Error Control:** Managing transmission errors so that corrupted or lost frames can be reliably retransmitted.
- **Flow Control:** Regulating the amount of data the sender can transmit to prevent overwhelming a slow receiver.

ii. Service Types to the Network Layer

The Data Link Layer can provide different quality of service levels to the Network Layer, depending on the protocol used:

- **Unacknowledged Connectionless Service:** Sends independent frames without establishing a connection or waiting for an acknowledgment. This is simple, fast, and typically used when the error rate is low. It offers no flow or error control.
- **Acknowledged Connectionless Service:** Transmits independent frames with acknowledgments to confirm receipt, but no formal connection setup is required.
- **Acknowledged Connection-Oriented Service:** Requires a formal connection to be established, and guarantees that every frame is delivered correctly and in the proper serial order. This includes built-in flow control and error control mechanisms. Protocols like HTTP, FTP, and SMTP rely on this service (or similar features in the Transport Layer).

iii. Flow Control and Error Control Overview

Flow Control refers to a set of procedures used to restrict the amount of data the sender can transmit before waiting for an acknowledgment. This ensures that a fast transmitter does not flood a slow receiver with data, preventing buffer overflow at the destination.

Error Control is essential because data can be corrupted during transmission due to noise. This layer implements procedures to detect and correct errors, giving confidence that the transmitted and received data are identical. The network layer relies on this service for reliable end-to-end data transfer.

---

2. Framing: Delimiting the Bit Stream

Framing is the translation of the Physical Layer's raw bits into discrete logical units called frames. A frame consists of a **Header**, a **Payload field** (containing the network layer packet), and a **Trailer**.

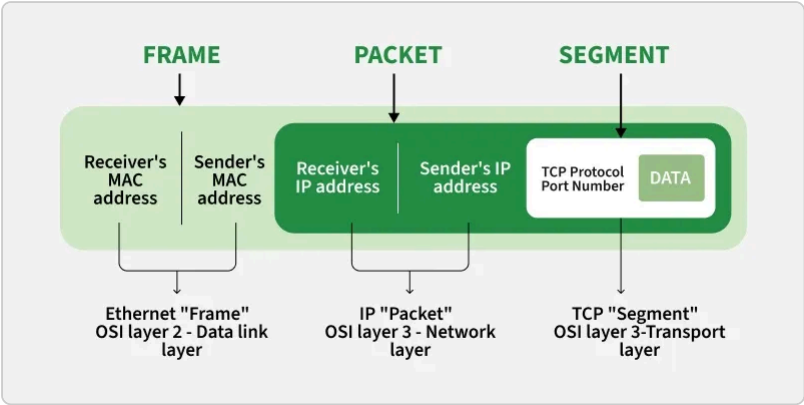


Figure 1: Relationship between Packets and Frames.

The **Header** typically contains the source/destination address and data type, while the **Trailer** usually contains error detection information. Frames can be either **Fixed Length** or **Variable Length**.

i. Types of Frames and Key Concepts

The core problem in framing is how the receiver detects the start and end boundaries of a frame within a continuous stream of bits. Four primary methods are used to solve this boundary problem:

- Character Count
- Flag Byte with Byte Stuffing
- Starting and Ending Flag with Bit Stuffing
- Physical Layer Encoding Violations

ii. Character Count & Flag Byte with Byte Stuffing

In the **Character Count** method, a field in the header explicitly specifies the number of characters (or bytes) in the frame. The receiver uses this count to determine the end of the frame. The main drawback is that if the count field is corrupted by a transmission error, the receiver loses synchronization and cannot accurately determine the boundaries of subsequent frames.

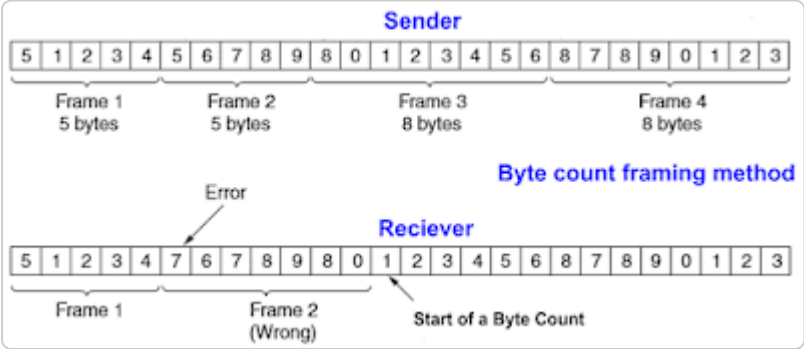


Figure 2: Problem with the Character Count Framing Method.

The **Flag Byte with Byte Stuffing** method starts and ends each frame with a special character known as a **Flag Byte**. The problem is that the actual data payload may accidentally contain the same bit pattern as the Flag Byte. To prevent this from interfering with the framing, the sender inserts a special **Escape Byte (ESC)** just before any accidental Flag or ESC byte in the data. The receiver then removes the extra ESC bytes (unstuffing) before passing the data to the Network Layer.

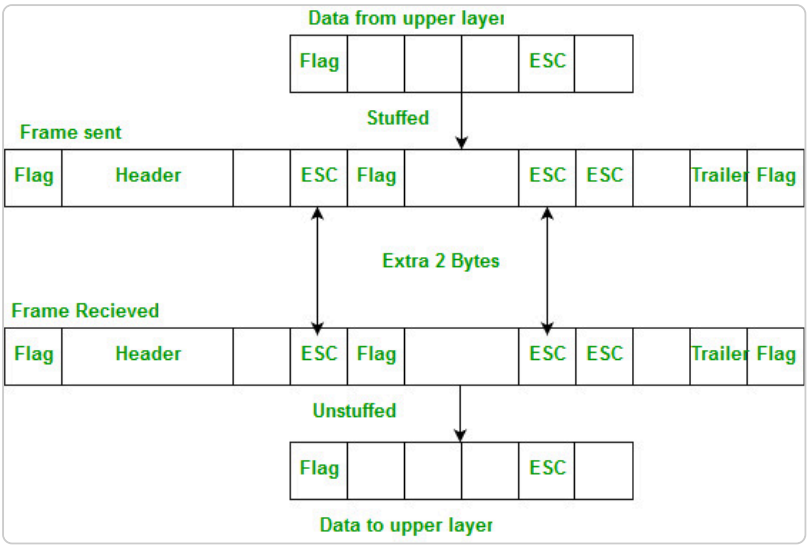


Figure 3: Byte Stuffing and Unstuffing Process.

iii. Bit Stuffing

Bit Stuffing allows frames to contain an arbitrary number of bits (not restricted to full bytes). Each frame begins and ends with the special bit pattern **01111110** (the Flag). The rule is:

Whenever the sender's Data Link Layer encounters **five consecutive 1s** in the data, it automatically **stuffs a 0 bit** into the outgoing bit stream. This ensures the Flag pattern cannot be accidentally created within the data itself.

The receiver automatically **destuffs** the extra 0 bit when it encounters five consecutive 1s followed by a 0 bit. This mechanism is completely transparent to the Network Layer.

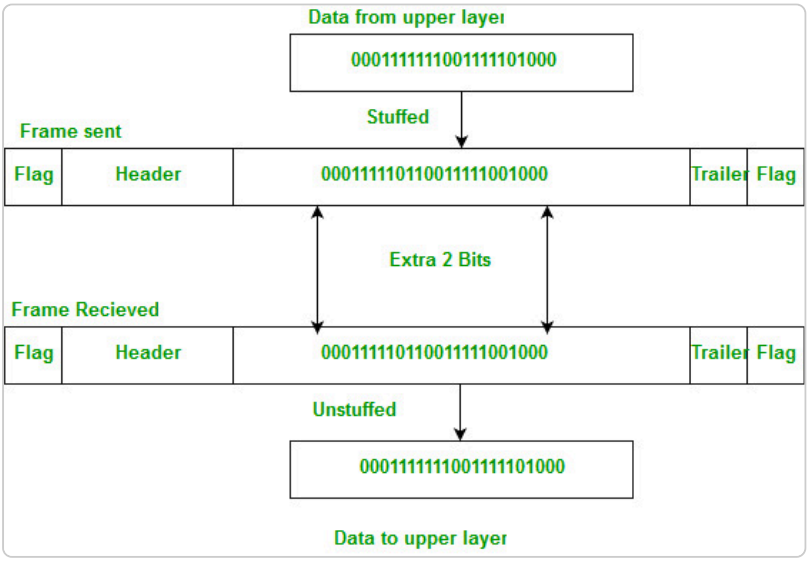


Figure 4: Bit Stuffing Process to prevent Flag corruption.

---

3. Error Control Mechanisms

Error control involves **Error Detection** and, in more advanced cases, **Error Correction** to ensure reliable transmission. An error is a condition where the output information does not match the input information, typically caused by noise changing a 0 bit to a 1, or vice versa.

i. Error Detection Techniques

Error detection uses redundancy to add extra bits to the data, allowing the receiver to check if corruption occurred during transit. The main methods include:

Method	Principle	Advantage/Limitation
Parity Check (VRC)	Adds one extra bit (Parity Bit) to a block of data to make the total number of 1s either even or odd.	Simplest method, but can only detect an odd number of errors (fails if two bits are flipped).
Checksum	Treats the data unit as blocks, adds them using 1's complement arithmetic, and appends the 1's complement of the sum (the checksum) to the data.	Used in higher layers (like IP and TCP) for error checking on the header and data.
Cyclic Redundancy Check (CRC)	A powerful technique based on polynomial division (not fully detailed here, but highly effective).	Excellent error detection capabilities, widely used in Data Link Layer protocols.

The core logic of a parity check can be represented using the XOR operation:

$$Parity\ Bit\ (P) = D_1 \oplus D_2 \oplus D_3 \oplus \ldots\ (where \oplus\ is\ XOR)$$

ii. Checksum Protocol Logic

To compute the **Checksum**, the sender treats the data unit as a sequence of blocks. It adds the blocks using one's complement arithmetic and creates an additional block (the Checksum) of the same size. The complete message (data + checksum) is transmitted. The receiver repeats the addition and checks the final sum:

$$Checksum = 1's\ Complement\ of\ (\sum Data\ Blocks)$$

If the final sum at the receiver is zero, the data is assumed to be correct.

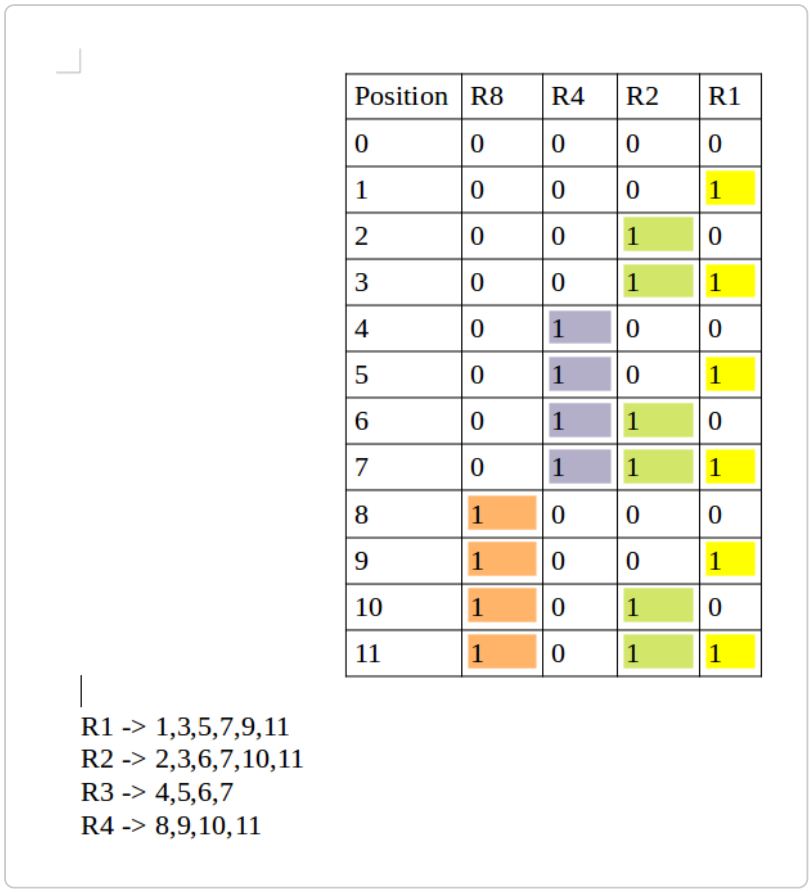
iii. Error Correction: Hamming Code

Error Correction involves not just detecting an error, but also identifying the exact location of the error so the original, error-free data can be reconstructed. The most common technique for this is the **Hamming Code**, given by R.W. Hamming.

The Hamming Code works by adding special **Parity bits** to the original **Data bits** based on the following rules:

- All bit positions that are a power of 2 (1, 2, 4, 8, etc.) are designated as Parity bits (P).
- All other bit positions are Data bits (D).
- Each Parity bit is calculated based on the data bits whose binary position includes a 1 in the same position as the Parity bit.

For a common 7-bit Hamming Code (4 data bits + 3 parity bits):



Position	R8	R4	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

Figure 5: Hamming Code Bit Positions and Structure.

Study hard! This note covers all the key concepts for the Data Link Layer.