**Data Link Layer:**

**UNIT-III**

**Data Link Control**
Outline

- Framing
  - Byte vs. bit oriented protocols
- Data Link Control
  - Flow control
  - Error control
- Protocols
Framing

- Process of wrapping data with certain info before sending out

A frame typically consists of

- Flag: indication for start and end of a frame
- Header: source/destination addresses, as well as other control information
- Data from the upper layer
- Trailer: error detection/correction code
Byte vs. Bit Oriented

- Framing in byte-oriented protocols

- Framing in bit-oriented protocols

![Diagram of byte-oriented framing](chart1)

- Framing in byte-oriented protocols

![Diagram of bit-oriented framing](chart2)
Byte Stuffing

- Process of adding extra byte whenever there is an escape or a flag character in the data

Data from upper layer

Frame sent

Flag  Header  ESC  Flag  ESC  ESC  Trailer  Flag

Stuffed

Extra 2 bytes

Frame received

Flag  Header  ESC  Flag  ESC  ESC  Trailer  Flag

Unstuffed

Flag  ESC

Data to upper layer
Bit Stuffing

- Process of adding extra bit to ensure flag sequence does not appear in the data.
Flow Control and Error Control

- **Flow control**
  - A set of procedures that tells the sender how much data can be sent before waiting for acknowledgment

- **Error control**
  - Includes both error detection and correction
  - Allows receiver to inform sender of lost or duplicate frames
  - Mostly based on Automatic Repeat Request (ARQ)
Data Link Protocols

Protocols

For noiseless channel
- Simplest
- Stop-and-Wait

For noisy channel
- Stop-and-Wait ARQ
- Go-Back-N ARQ
- Selective Repeat ARQ
Protocols for Noiseless Channel

- Assuming channel is error free
  - Not realistic...
- No need for error control
"Simplest" Mechanism

- **Assuming**
  - Noiseless channel
  - Unlimited buffer and speed for the receiver
"Simplest" : Pseudo Code

- **Sender**

```
1 while(true) // Repeat forever
2 {
3  WaitForEvent(); // Sleep until an event occurs
4  if(Event(RequestToSend)) //There is a packet to send
5    {
6      GetData();
7      MakeFrame();
8      SendFrame(); //Send the frame
9    }
10 }
```

- **Receiver**

```
1 while(true) // Repeat forever
2 {
3  WaitForEvent(); // Sleep until an event occurs
4  if(Event(ArrivalNotification)) //Data frame arrived
5    {
6      ReceiveFrame();
7      ExtractData();
8      DeliverData(); //Deliver data to network layer
9    }
10 }
```
"Simplest": Flow Diagram

Sender

Request → Frame

Receiver

Request → Frame → Request → Frame → Request → Frame

Arrival

Time

Time
Stop-and-Wait Mechanism

- Still noiseless channel
- Receiver has limited buffer
  - Requires flow control
- Sender sends one frame at a time and wait for an acknowledgment
Stop-and-Wait: Overview
Stop-and-Wait: Pseudo Code

Sender side

```plaintext
while(true)  // Repeat forever
    canSend = true  // Allow the first frame to go
    {
        WaitForEvent();  // Sleep until an event occurs
        if(Event(RequestToSend) AND canSend)
            {
                GetData();
                MakeFrame();
                SendFrame();  // Send the data frame
                canSend = false;  // Cannot send until ACK arrives
            }
        WaitForEvent();  // Sleep until an event occurs
        if(Event(ArrivalNotification)  // An ACK has arrived
            {
                ReceiveFrame();  // Receive the ACK frame
                canSend = true;
            }
    }
```
Stop-and-Wait: Pseudo Code

- Receiver side

```java
1  while(true)            // Repeat forever
2  {
3    WaitForEvent();    // Sleep until an event occurs
4    if(Event(ArrivalNotification))  // Data frame arrives
5      {
6        ReceiveFrame();
7        ExtractData();
8        Deliver(data);        // Deliver data to network layer
9        SendFrame();        // Send an ACK frame
10      }
11  }
```
Stop-and-Wait: Flow Diagram
Noisy Channel

- Realistic
  - Error can and will happen
  - Require error control

- Mechanisms:
  - Stop-and-Wait ARQ
  - Go-Back-N ARQ
  - Selective Repeat ARQ
Stop-and-Wait ARQ

- Sender keeps a copy of sent frame until successful delivery is ensured.
- Receiver responds with an ack when it successfully receives a frame.
- Both data and ack frames must be numbered.
- When sender does not receive an ack within certain time, it assumes frame is lost, then retransmits the same frame.
Stop-and-Wait ARQ

Sender
- Network
- Data link
- Physical
  - Receive frame
  - Send frame

Data frame
- seqNo

Receiver
- Network
- Data link
- Physical
  - Receive frame
  - Send frame

ACK frame
- ackNo

Event: Request from network layer
Repeat forever
Algorithm for sender site

Event: Notification from physical layer

Time-out Event:

Event: Notification from physical layer
Repeat forever
Algorithm for receiver site
Flow Diagram: Normal Operation

Sender

\[ S = 0 \]

\[ S = 1 \]

Receiver

\[ R = 0 \]

\[ R = 1 \]

\[ R = 0 \]

Frame 0

ACK 1

Frame 1

ACK 0

Deliver

Deliver

Time

Time
Thinking Corner

Why data frames need to be numbered?
Flow Diagram: Lost Frame

Sender

Timeout

$S = 0$

$S = 1$

Receiver

$R = 0$

$R = 1$

Time

Timeout

Frame 0

ACK 1

Frame 1

ACK 0

Deliver
Flow Diagram: Lost ACK

<table>
<thead>
<tr>
<th>Time</th>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>S = 0</td>
<td>R = 0</td>
<td>R = 1</td>
</tr>
<tr>
<td>S = 1</td>
<td>R = 0</td>
<td>R = 0</td>
</tr>
<tr>
<td>S = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S = 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timeout

*Frame 0*

*Frame 1*

*ACK 0*

*ACK 1*

Frame 0 expected; discard
Thinking Corner

- Why ACK frames need to be numbered?
Flow Diagram: Delayed ACK

Sender

Receiver

$S = 0$

$R = 0$

$R = 1$

$R = 0$

Timeout

Frame 0

$S = 1$

$R = 1$

Frame 1

ACK 1

Frame 0 expected; discard

Timeout

Frame 1

ACK 0

$R = 0$

Deliver

$R = 1$

Deliver
Bidirectional Transmission

- Data are transferred both ways
- ACK are "piggybacked" with data frames
Example

- Assuming a communication system where:
  - Stop-and-Wait ARQ is used
  - Bandwidth of the link is 1 Mbps
  - Propagation delay is 10 ms
  - One-way data flow

- Questions
  - What is the bandwidth-roundtrip-delay product?
  - If the system data frames are 1000 bits in length, what is the utilization percentage of the link?
Improving Link Utilization

- Previous example demonstrates major disadvantage of Stop-and-Wait ARQ
- Prefer to send more frames before waiting for ACK
- Example:
  - Recalculate the link utilization if we allow up to 15 frames to be sent before waiting for an ACK
Go-Back-N ARQ

- Allows multiple frames to be sent before waiting for ACK
  - These frames must be numbered differently
  - Frame numbers are called Sequence numbers
- Frames must be received in the correct order
- If a frame is lost, the lost frame and all of the following frames must be retransmitted
Sequence Numbers

- Frame header contains $m$ bits for sequence number
- That allows up to $2^m$ different frame numbers
- How big should $m$ be?
Sending Window

- Sending more than one frame at once requires the sender to buffer multiple frames
  - Known as "sending window"
  - Any of these frames in the window can be lost
"Sliding" Window

- Once the first frames in the window is ACKed
  - ACKed frames are removed from the buffer
  - More frames are transmitted
  - Result: The window slides to the right

![Diagram showing sliding window](attachment:image.png)

Window size = 7

- a. Before sliding
- b. After sliding two frames
Receiving Window

- Receiver expects one frame at a time

\[ \begin{array}{cccccccc}
5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \\
\end{array} \]

a. Before sliding

\[ \begin{array}{cccccccc}
5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \\
\end{array} \]

b. After sliding
Send vs. Receive Windows

a. Sender window

b. Receiver window
Go-Back-N: Window Sizes

- For $m$-bit sequence numbers
- Send window size: at most $2^{m-1}$
  - Up to $2^{m-1}$ frames can be sent without ACK
- Receive window size: 1
  - Frames must be received in order
Go-Back-N: Normal Operation
Go-Back-N: Lost Frame

ACKs are cumulative
Lost ACK: Window Size < $2^m$
What is a problem if send window is greater than $2^{m-1}$?
Lost ACK: Window Size = $2^m$
Thinking Corner

- Stop-and-Wait is a special case of Go-Back-N.
  - What is the send window size in Stop-and-Wait?
Selective Repeat ARQ

- Go-Back-N always discards out-of-order frames
  - Losing one frame may result in retransmission of multiple frames
  - Very inefficient in noisy link
- Selective Repeat ARQ allows frames to be received out of order
  - Therefore, receive window > 1
Send and Receive Windows

- Sender and receiver share window space equally

- For $m$-bit sequence numbers
  - Send window: up to $2^{m-1}$
  - Receive window: up to $2^{m-1}$
Send Window

Send window, first outstanding frame

Send window, next frame to send

Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent

$S_{\text{size}} = 2^{m-1}$
Receive Window

Frames already received

Frames that can be received and stored for later delivery. Colored boxes, already received

$R_{\text{size}} = 2^{m-1}$

Frames that cannot be received

Receive window, next frame expected

$R_n$
Negative ACK

- Used by receiver to indicate missing frame
Selective Repeat: Window Size

![Diagram of Selective Repeat Protocol]

- **Sender**
  - Time-out
  - Frame 0
  - Frame 1
  - Correctly discarded

- **Receiver**
  - Frame 0
  - Frame 1

**a. Window size = 2^{m-1}**

**b. Window size > 2^{m-1}**
Data Link Control Protocols: 

HDLC and PPP
HDLC

- **High-level Data Link Control**
- Bit-oriented protocol
- Support both
  - Point-to-point links
  - Multipoint links
Normal Response Mode

- "NRM" mode
- Used in both point-to-point and multi-point link

**Diagram:**

*a. Point-to-point*

*b. Multipoint*
Asynchronous Balance Mode

- "ABM"
- Supports only point-to-point links
- Each station is both primary and secondary
HDLC Frames

- **Information frame (I-frame)**

- **Supervisory frame (S-frame)**

- **Unnumbered frame (U-frame)**
Control Field Format

- **N(S)** – Frame sequence number
- **N(R)** – Ack sequence number
- **P/F**
  - Poll (primary → secondary)
  - Final (secondary → primary)
## U-Frame Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Response</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 001</td>
<td>SNRM</td>
<td></td>
<td>Set normal response mode</td>
</tr>
<tr>
<td>11 011</td>
<td>SNRME</td>
<td></td>
<td>Set normal response mode, extended</td>
</tr>
<tr>
<td>11 100</td>
<td>SABM</td>
<td>DM</td>
<td>Set asynchronous balanced mode or <strong>disconnect mode</strong></td>
</tr>
<tr>
<td>11 110</td>
<td>SABME</td>
<td></td>
<td>Set asynchronous balanced mode, extended</td>
</tr>
<tr>
<td>00 000</td>
<td>UI</td>
<td>UI</td>
<td>Unnumbered information</td>
</tr>
<tr>
<td>00 110</td>
<td></td>
<td>UA</td>
<td>Unnumbered acknowledgment</td>
</tr>
<tr>
<td>00 010</td>
<td>DISC</td>
<td>RD</td>
<td>Disconnect or <strong>request disconnect</strong></td>
</tr>
<tr>
<td>10 000</td>
<td>SIM</td>
<td>RIM</td>
<td>Set initialization mode or <strong>request information mode</strong></td>
</tr>
<tr>
<td>00 100</td>
<td>UP</td>
<td></td>
<td>Unnumbered poll</td>
</tr>
<tr>
<td>11 001</td>
<td>RSET</td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td>11 101</td>
<td>XID</td>
<td>XID</td>
<td>Exchange ID</td>
</tr>
<tr>
<td>10 001</td>
<td>FRMR</td>
<td>FRMR</td>
<td>Frame reject</td>
</tr>
</tbody>
</table>
Connection and Disconnection

- **Connection establishment**
  - U-frame (SABM)
  - U-frame (UA)

- **Data transfer**

- **Connection release**
  - U-frame (DISC)
  - U-frame (UA)
Data Transfer: No Error

Node A

I-frame (data frame 0)

Flag | Control | Data | FCS
---|---|---|---
000 |  |  |

I-frame (data frame 1)

Flag | Control | Data | FCS
---|---|---|---
010 |  |  |

I-frame (data frame 0)

Flag | Control | Data | FCS
---|---|---|---
002 |  |  |

I-frame (data frame 1)

Flag | Control | Data | FCS
---|---|---|---
012 |  |  |

I-frame (data frame 2)

Flag | Control | Data | FCS
---|---|---|---
022 |  |  |

S-frame (RR), an ACK 3

Flag | Control | FCS
---|---|---
10RR | 3 |  

Node B

Time

Time
Data Transfer: With Error
**PPP**

- **Point-to-Point Protocol**
- Byte-oriented protocol
- Most common protocol for point-to-point access
  - Dial-up access
  - ADSL
  - GPRS/EDGE/3G
PPP Frame Format

- Escape byte: 01111101
- Control field uses HDLC's U-frame format
  - No flow or error control
Multiplexing in PPP

Network layer

Data from different networking protocols

Data

Data link layer

NCP

OSI CP

IPCP

AP

CHAP

PAP

LCP

Flag Address Control Protocol FCS Flag

LCP: Link Control Protocol
AP: Authentication Protocol
NCP: Network Control Protocol

LCP: 0xC021
AP: 0xC023 and 0xC223
NCP: 0x8021 and ....
Data: 0x0021 and ....
PPP Stack

- Link Control Protocol (LCP)
- Authentication Protocol (AP)
- Network Control Protocol (NCP)
LCP: Link Control Protocol

- Responsible for establishing, maintaining, configuring, and terminating links
PPP Authentication

- Two protocols are supported:
  - Password Authentication Protocol (PAP)
  - Challenge Handshake Authentication Protocol (CHAP)
PAP

Authenticate-request

Authenticate-ack or authenticate-nak

Authenticate-request

Authenticate-ack

Authenticate-nak

PAP packets

Flag Address Control CO2316 Payload (and padding) FCS Flag
CHAP
NCP: Network Control Protocol

- A set of control protocols to allow data from the network layer to be encapsulated into a PPP frame
- One common protocol: IPCP (Internetwork Protocol Control Protocol)
  - Allow negotiation at the network layer
Data Encapsulation

IP packet

Header

User data

Flag Address Control 0x0021 Payload (and padding) FCS Flag
Example: PPP Session
Example: PPP Session (cont'd)