Data Link Layer

Introduction
Synchronization
Error Detection
Flow Control
Error Control (via Retransmission)

Introduction
Main Task of the data link layer:
- Provide error-free transmission over a link
Introduction

- The PDU at the Data Link Layer (DL-PDU) is typically called a Frame. A Frame has a header, a data field, and a trailer.

Example

<table>
<thead>
<tr>
<th>01111110</th>
<th>Address</th>
<th>Control</th>
<th>Data (&gt;=0)</th>
<th>Checksum</th>
<th>01111110</th>
</tr>
</thead>
<tbody>
<tr>
<td>8bits</td>
<td>8bits</td>
<td>8bits</td>
<td>&gt;=0</td>
<td>16bits</td>
<td>8bits</td>
</tr>
</tbody>
</table>

Header  \[\rightarrow\]  Trailer

Framing

- Problem: Identify the beginning and the end of a frame in a bit stream.
- Solution (bit-oriented Framing): A special bit pattern (flag) signals the beginning and the end of a frame (e.g., "01111110")

<table>
<thead>
<tr>
<th>01111110</th>
<th>Data</th>
<th>01111110</th>
</tr>
</thead>
</table>

- Problem:
  - The sequence '01111110' must not appear in the data of the frame.
**Bit-Oriented Framing and Bit Stuffing**

- 'Bit stuffing': If the sender detects five consecutive '1' it adds a '0' bit into the bit stream. The receiver removes the '0' from each occurrence of the sequence '111110'.

<table>
<thead>
<tr>
<th>Original bit sequence:</th>
<th>0110111111111111111100</th>
</tr>
</thead>
<tbody>
<tr>
<td>After stuffing bits at sender:</td>
<td>0110111110111110111110100</td>
</tr>
<tr>
<td>After stuffing bits are removed by receiver:</td>
<td>0110111111111111111100</td>
</tr>
</tbody>
</table>

- Note: The flags itself are not bit-stuffed.

---

**Error Control**

Two basic approaches to handle bit errors:

- Error-correcting codes
  - Used if retransmission of the data is not possible
  - Data are encoded with sufficient redundancy to correct bit errors
  - **Examples**: Hamming Codes, Reed Solomon Codes, etc.

- Error-detecting codes plus retransmission
  - Used if retransmission of corrupted data is feasible
  - Receiver detects error and requests retransmission of a frame.
Error Detection Techniques

Error Detection Techniques:
- Parity Checks
- Cyclic Redundancy Check

Parity Checks

General Method:
- Append a parity bit to the end of each character in a frame such that the total number of '1' in a character is:
  - even (even parity) or
  - odd (odd parity)

Example: With ASCII code, a parity bit can be attached to an 7-bit character
- ASCII "G" = 1 1 1 0 0 0 1
- with even parity =
- with odd parity =
Cyclic-Redundancy Codes (CRC)

General Method:
- The transmitter generates an n-bit check sequence number from a given k-bit frame such that the resulting (k+n)-bit frame is divisible by some number.
- The receiver divides the incoming frame by the same number.
- If the result of the division does not leave a remainder, the receiver assumes that there was no error.

Cyclic-Redundancy Codes (CRC)

- CRC is used by all advanced data link protocols, for the following reasons:
  - Powerful error detection capability
  - CRC can be efficiently implemented in hardware

- We discuss the CRC method in five easy steps:
  1. Prerequisites
  2. CRC Encoding Method
  3. Example
  4. Error Detection with CRC
  5. Capabilities of CRC
Step 1: Prerequisites

- Modulo-2 Arithmetic:
  
<table>
<thead>
<tr>
<th>+</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Examples of polynomials based on Modulo-2 operations:
  1. \((x+1)(x+1) =\)
  2. \((x^2+1)(x^3+x+1) =\)
  3. If \(F(x)\) contains \((x+1)\) as a factor, then \(F(1) = 0\)

Step 2: CRC Encoding Method

- Let us view each block of data as a polynomial with binary coefficients:
  \(1 0 1 1 0 1\) is viewed as \(x^5 + x^3 + x^2 + 1\)

Define:
- \(M(x)\) Data block is a polynomial (= Message, Frame)
- \(P(x)\) "Generator Polynomial" which is known to both sender and receiver (degree of \(P(x)\) is \(n\)
**Step 2: CRC Encoding Method**

(I) Append $n$ zeros to $M(x)$, i.e., $M(x) x^n$

(II) Divide $M(x) x^n$ by $P(x)$ and obtain:
$$M(x) x^n = Q(x) P(x) + R(x)$$

(III) Set $T(x) = M(x) x^n + R(x)$. $T(x)$ is the encoded message

Note: $T(x)$ is divisible by $P(x)$. Therefore, if the received message does not contain an error then it can be divided by $P(x)$.

■ Exercise: Encode the frame 1 1 0 1 0 1 1 0 1 1

---

**Step 3: Encoding**

$$M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$$
- Generator polynomial is $P(x) = x^4 + x + 1$
- Degree of $P(x)$ is 4

(I) $M(x) x^4 = x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4$

(II) $M(x) x^4 = Q(x) P(x) + R(x)$
$$= (x^9 + x^8 + x^3 + x) P(x) + x^3 + x^2 + x$$

(III) $T(x) = M(x) x^4 + R(x)$
$$= x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + x$$

Transmitted Bit Sequence: 1 1 0 1 0 1 1 0 1 1 1 1 0
Step 4: Error Detection with CRC

- Errors can be expressed as Error Polynomials

- For example,

  Sent Message: 1 0 1 1 1 0 1
  Received Message: 1 1 1 1 0 0 1

  Error: 0 1 0 0 1 0 0

- In the example, the Error Polynomial \( E(x) \) is given by:

\[
E(x) = x^5 + x^2
\]

Step 5: Error Detection Capability of CRC

Note:

- Errors will not be detected by CRC if

\[
E(x) \div P(x) = B(x) + 0,
\]

i.e., \( E(x) \) contains \( P(x) \) as a factor

- Observe the following trade-off:
  - Large values of \( n \) introduce a lot of overhead, but improve the error detection capability.
Step 5: Error Detection Capability of CRC

1. All single-bit errors are detected, if \( P(x) \) has a factor with at least 2 terms:
   \[ E(x) = x^i \]
   If \( P(x) = (x+1)P'(x) \), then \( x^i \) cannot contain \((x+1)\)

2. All double-bit errors are detected if \( P(x) \) has a factor with at least 3 terms
   \[ E(x) = x^i + x^j = x^j(x^{k+1}) \quad (\text{for } k = i-j, j<i) \]

3. If \( P(x) \) does not divide \((x^k+1)\) for any \( k \) up to the max. frame length then all double errors can be detected.
   \[ P(x) = x^{15} + x^{14} + 1 \text{ will not divide } E(x) = x^k + 1 \text{ for any } k \leq 32,768 \]

Step 5: Error Detection Capability of CRC

1. If \( P(x) \) contains \((x+1)\) as a factor then all errors with “odd” number of bits are detected.
   - CRC detects 50% of all errors

2. A CRC with \( n \) check bits will detect all burst errors of length \( \leq n \) bits.
   Burst Error of Length \( k \): \[ E(x) = x^i(x^{k-1}+...+1) \]
Additional Facts on CRC

- CRC can be efficiently implemented in hardware by a set of XOR gates and a shift register.
- The following generator polynomials are widely used:
  
  **CRC-12:** \( P(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1 \)
  
  **CRC-16:** \( P(x) = x^{16} + x^{15} + x^2 + 1 \)
  
  **CRC-CCITT:** \( P(x) = x^{16} + x^{12} + x^5 + 1 \)
  
  **CRC-32:** \( P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \)

Flow Control

- Flow Control is a technique for speed-matching of transmitter and receiver. Flow control ensures that a transmitting station does not overflow a receiving station with data.
- We will discuss two protocols for flow control:
  - Stop-and-Wait Protocol
  - Sliding Window Protocol
- For the time being, we assume that we have a perfect channel between sender and receiver (no errors)
Stop-and-Wait Flow Control

- Simplest form of flow control
- In Stop-and-Wait flow control, the receiver indicates its readiness to receive data for each frame

Operations:

1. **Sender**: Transmit a single frame
2. **Receiver**: Transmit acknowledgment (ACK)

Analysis of Stop-and-Wait
Analysis of Stop-and-Wait

- **Transmission delay** is the time that the sender needs to transmit a frame.
- Transmission delay is dependent on the size of a frame and the maximum data rate.

**Example:**

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Data rate of network</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bit</td>
<td>1 Mbps</td>
</tr>
</tbody>
</table>

Transmission delay = 1000 bit / 1 Mbps = 1 msec

Analysis of Stop-and-Wait

- **Propagation delay** is the time that a transmitted bit needs to travel from sender to the receiver.
- Propagation delay is only dependent on the speed of the transmission medium and the distance between sender and receiver.
  
  Speed of light: 300,000 km/sec, 
  Speed in guided media (approx.): 200,000 km/sec

**Example:**

Distance = 1000 km

Propagation delay = 1000 km / (200,000 km/sec) = 5 msec
Analysis of Stop-and-Wait

Notation:

- \( C \) = Channel capacity in bps
- \( I \) = Propagation delay
- \( H \) = Number of bits in a frame header
- \( D \) = Number of data bits in a frame
- \( F \) = Total length of a frame \((F = D + H)\)
- \( A \) = Total length of an ACK frame
- \( F/C \) = Transmission delay for a frame

\[
a = \frac{\text{propagation delay}}{\text{transmission delay}} = \frac{I}{F/C}
\]
Analysis of Stop-and-Wait

- Transmission of a frame (in Stop-and-Wait):

![Diagram showing the transmission timeline from Sender to Receiver]

- Efficiency of a protocol is the maximum fraction of time when the protocol is transmitting data.

Efficiency of Stop-and-Wait Flow Control:

\[
\text{efficiency} = \frac{D/C}{F/C + A/C + 2I} = \frac{D}{F + A + 2IC}
\]

- Assuming that \(H\) and \(A\) are negligible we obtain:

\[
\text{normalized efficiency} = \frac{D}{D + 2IC} = \frac{1}{1+2a}
\]
**Exercise 1**

**Satellite Link**
- Roundtrip propagation delay is 270 ms
- Data rate is 56 kbps
- Frame length is 4,000 bits (including header)
- Lengths of ACK frames are negligible

- What is the value of “a”?
- What is the efficiency of the satellite link if Stop-and-Wait is used?

**Exercise 2**

**Local Area Network (LAN)**
- Data rate is 10 (100) Mbps
- Propagation rate is 200,000 km/sec
- Length of the LAN is 10 km
- Frame size is 500 bits (including header)
- Length of ACK frames are negligible

- What are the values of “a” for the 10 and 100 Mbps LANs?
- How efficient are these LANs?
- What is the minimum frame size in the LANs to reach an efficiency of at least 80%?
### Sliding Window Flow Control

- **Major Drawback of Stop-and-Wait Flow Control:**
  - Only one frame can be in transmission at a time
  - This leads to inefficiency if \( a > 1 \)

- **Sliding Window Flow Control**
  - Allows transmission of multiple frames
  - Assigns each frame a \( k \)-bit sequence number
  - Range of sequence number is \([0..2^k-1]\), i.e., frames are counted modulo \( 2^k \)

### Operation of Sliding Window

- **Sending Window:**
  - At any instant, the sender is permitted to send frames with sequence numbers in a certain range
  - The range of sequence numbers is called the *sending window*
Operation of Sliding Window

Receiving Window:
- The receiver maintains a *receiving window* corresponding to the sequence numbers of frames that are accepted.

![Diagram showing receiving window]

Operations at the sender:
- Send a single packet (with sequence number "0"):
### Operation of Sliding Window

**Operations at the sender:**

Receive a single acknowledgement (ACK 1)

---

**Operations at the receiver:**

Receive a single packet (with sequence number "0"): 
**Operation of Sliding Window**

**Operations at the receiver:**

Transmit a single acknowledgement ("ACK 1"): 

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6
```

**How is “flow control” achieved?**

- Receiver can control the size of the sending window
- By limiting the size of the sending window data flow from sender to receiver can be limited

**Interpretation of ACK N message:**

- Receiver acknowledges all packets until (but not including) sequence number N
**Example**

<table>
<thead>
<tr>
<th>Transmitter</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td><strong>ACK4</strong></td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
</tbody>
</table>

**Example Continued**

<table>
<thead>
<tr>
<th>Transmitter</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
<tr>
<td><strong>ACK3</strong></td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
<td>0 1 2 3 4 5 6 7 0 1 2 3</td>
</tr>
</tbody>
</table>
Analysis of Sliding Windows

Define:

- We use the same parameters for as in Stop-and-Wait

- To simplify notation we set:
  \[ \frac{F}{C} = 1 \]
  \[ I = a \]
  (Normalization)

- \( W \) = Maximum window size (identical for sender and receiver)
**Analysis of Sliding Windows**

- If the window size is sufficiently large the sender can continuously transmit packets:
  - \( W \geq 2a+1 \): Sender can transmit continuously
    - normalized efficiency = 1
  - \( W < 2a+1 \): Sender can transmit \( W \) frames every \( 2a+1 \) time units
    - normalized efficiency = \( \frac{W}{1+2a} \)

**ARQ Error Control**

- Two types of errors:
  - Lost frames
  - Damaged Frames

- Most Error Control techniques are based on (1) Error Detection Scheme (e.g., Parity checks, CRC), and (2) Retransmission Scheme

- Error control schemes that involve error detection and retransmission of lost or corrupted frames are referred to as **Automatic Repeat Request (ARQ)** error control
ARQ Error Control

- All retransmission schemes use all or a subset of the following procedures:
  - Receiver sends an acknowledgment (ACK) if a frame is correctly received
  - Receiver sends a negative acknowledgment (NAK) if a frame is not correctly received
  - The sender retransmits a packet if an ACK is not received within a timeout interval
  - All retransmission schemes (using ACK, NAK or both) rely on the use of timers

Note: Once retransmission is used, a sequence number is required for every data packet to prevent duplication of packets

- Both ACKs and NAKs can be sent as special frames, or be attached to data frames going in the opposite direction (Piggybacking)

For piggybacking
ARQ Schemes

- The most common ARQ retransmission schemes:
  - Stop-and-Wait ARQ
  - Go-Back-N ARQ
  - Selective Repeat ARQ
- The protocol for sending ACKs in all ARQ protocols are based on the sliding window flow control scheme

Stop-and-Wait ARQ

- Stop-and-Wait ARQ is an addition to the Stop-and-Wait flow control protocol:
  - Frames have 1-bit sequence numbers ($SN = 0$ or $1$)
  - Receiver sends an ACK ($1-SN$) if frame $SN$ is correctly received
  - Sender waits for an ACK ($1-SN$) before transmitting the next frame with sequence number $1-SN$
  - If sender does not receive anything before a timeout value expires, it retransmits frame $SN$
Stop-and-Wait ARQ

- Lost Frame

Stop-and-Wait ARQ

- Lost ACK

Timeout
**Go-Back-N ARQ**

- Go-Back-N uses the sliding window flow control protocol. If no errors occur the operations are identical to Sliding Window.

- **Operations:**
  - A station may send multiple frames as allowed by the window size.
  - Receiver sends a NAK $i$ if frame $i$ is in error. After that, the receiver discards all incoming frames until the frame in error was correctly retransmitted.
  - If sender receives a NAK $i$ it will retransmit frame $i$ and all packets $i+1, i+2, ..., i+N$ which have been sent, but not been acknowledged.

---

**Diagram:**

- Lost Frame

- Frames 4, 5, 6 are retransmitted.

- Frames 5 and 6 are discarded.
**Go-Back-N ARQ**

- **Lost ACK**
  - Frames 0-4 are retransmitted

**Scenario 1:**
A transmits frame $i$, and B detects error in frame $i$, but has received frames $i-1$, $i-2$, ... correctly

$\Rightarrow$ B sends NAK$i$

**Scenario 2:**
- Frame $i$ is lost or B does not recognize frame $i$
- Assume that A sends frame $i+1$ and B receives it

$\Rightarrow$ B sends NAK$i$, or A will timeout and retransmit frame $i$ and all subsequent frames
Details Go-Back-N ARQ

Scenario 3: B receives frame i and sends $\text{ACK}(i+1)$ which is lost

$\Rightarrow$ B may send an $\text{ACK}(i+k)$ later which also acknowledges all frames $<i+k$ ($\text{ACKs}$ are “cumulative”)

or

A retransmits frame i and all subsequent frames

Scenario 4: NAKi is lost

$\Rightarrow$ A will eventually time out

Example of Go-Back-N ARQ

- In Go-back-N, if frames are correctly delivered, they are delivered in the correct sequence
- Therefore, the receiver does not need to keep track of “holes” in the sequence of delivered frames

<table>
<thead>
<tr>
<th>Frames waiting for ACK/NAK</th>
<th>Frames received</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1</td>
</tr>
<tr>
<td>A B</td>
<td>A B</td>
</tr>
<tr>
<td>frame 1 is correct, send ACK 2</td>
<td></td>
</tr>
<tr>
<td>2 3 4</td>
<td>1</td>
</tr>
<tr>
<td>A B</td>
<td>A B</td>
</tr>
<tr>
<td>frame 2 is in error, send NAK 2</td>
<td></td>
</tr>
<tr>
<td>2 3 4</td>
<td>1</td>
</tr>
<tr>
<td>A B</td>
<td>A B</td>
</tr>
<tr>
<td>retransmit frame 2,3,4</td>
<td></td>
</tr>
</tbody>
</table>
Selective-Repeat ARQ

- Similar to Go-Back-N ARQ. However, the sender only retransmits frames for which a NAK is received.

- **Advantage over Go-Back-N:**
  - Fewer Retransmissions.

- **Disadvantages:**
  - More complexity at sender and receiver
  - Each frame must be acknowledged individually (no cumulative acknowledgements)
  - Receiver may receive frames out of sequence

---

### Example Diagram

**Lost Frame**

- Only Frame 4 is retransmitted

- Frames 5 and 6 are buffered
Example of Selective-Repeat ARQ

Frames waiting for ACK/NAK

Frames received

- Receiver must keep track of ‘holes’ in the sequence of delivered frames

- Sender must maintain one timer per outstanding packet

Example:

1. Frames 1 and 3 are received correctly, ACK2 is sent to retransmit frame 2.
2. Frames 2 and 4 are received correctly.
3. Frame 5 is received correctly, ACK5 is sent.

Analysis of ARQ Protocols

- What is the efficiency of the discussed ARQ protocols?

- A number of assumptions:

  - ACKs and NAKs are never lost, and frames are not dropped.
  - Sizes of ACKs, NAKs, and frame headers are negligible.
Analysis of Stop-and-Wait ARQ

Parameters:

- \( U \) = efficiency
- \( T_{t} = F/C \) transmission delay of a frame
- \( I \) = propagation delay
- \( a = I/T_{t} \)
- \( P \) = Probability that a frame is in error

Without Errors (\( P=0 \)):

\[ U = \frac{T_{t}}{T_{t} + 2I} \]

With Errors:

- Probability that \( k \) transmission attempts are needed to successfully transmit a frame

\[ P[k \text{ attempts needed}] = P^{k-1} \cdot (1-P) \]

- Expected number of attempts (=\( E[A] \)):

\[ E[A] = \sum_{i=1}^{\infty} i \cdot P^{i-1} \cdot (1-P) = \frac{1}{1-P} \]

- Expected efficiency with errors

\[ U = \frac{T_{t}}{E[A] \cdot (T_{t} + 2I)} = \frac{(1-P) \cdot T_{t}}{T_{t} + 2I} = \frac{1-P}{1+2a} \]
**Analysis of Go-Back-N ARQ**

**Without Errors**
- \( W > 2I + T \) 
  \[ U = 1 \]
- \( W < 2I + T \) 
  \[ U = \frac{W}{1+2a} \]

**With Errors**
- \( W > 2I + T \) 
  \[ U = 1 - P \]
- \( W < 2I + T \) 
  \[ U = \frac{W \cdot (1 - P)}{1+2a} \]

**Analysis of ARQ Protocols - Results**

\( P = 10^{-3} \)
Example DL Protocol: XModem

- XModem is a simple data link protocol which used to be popular for communications over modems
- Xmodem uses Stop-and-Wait ARQ

<table>
<thead>
<tr>
<th>SOH</th>
<th>NUM</th>
<th>CNUM</th>
<th>Data</th>
<th>CKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>&lt;=128</td>
<td>1</td>
</tr>
</tbody>
</table>

- SOH: frame delimiter (start of header, “00000001”)
- NUM: 8-bit sequence number
- CNUM: bitwise complement of NUM
- Data: up to 128 bytes
- CKS: checksum = sum of the data field without overflow bit

XModem

- NAK: “00010101” sent every 15 sec when data is not received
- ACK: “00000110” sent if CKS is correct
- EOT: “00000100” end of transmission

- Variations of XModem:
  - XModem-CRC: uses CRC
  - YModem: allows to send 1,024 bytes of data in a frame. Also: allows batch file transfers
  - ZModem