Data Link Layer

• In general, the functions of the data link layer include:
  □ Providing a well-defined service interface to the network layer (framing)
  □ Dealing with transmission errors (error control)
  □ Regulating the flow of data so that slow receivers are not swamped by fast senders (flow control)

• To accomplish these goals, packets from the network layers are encapsulated into frames:

Data Link Layer

• Topics:
  □ Introduction
  □ Error detection
  □ Error correction
  □ Framing
  □ Stop-and-wait protocols
  □ Sliding window protocols
  □ Protocol performance
  □ Example data link layer protocols

• In this chapter, we are concerned with single sender - single receiver data link protocols. We will study data link protocols for multipoint links in chapter 4.

• Assigned reading: Tanenbaum, 3.1 - 3.5
Data Link Layer

- Services provided to the network layer:
  - **unacknowledged connectionless service**
    No ACKs and hence no retransmissions (at the data link level). Appropriate for low-error rate environments and real-time traffic (e.g., digitized voice). Used in many LANs.
  - **acknowledged connectionless service**
    Each frame individually acknowledged. Retransmissions occur for negative ACKs (NACKs) or missing ACKs (timeout by sender).
  - **acknowledged connection-oriented service**
    Connection established before frames sent. Frames contain sequence numbers. Guarantees that each frame is passed to network layer exactly once at receiving side.

Functions of Data Link Layer

To accomplish the general functions of handling errors and regulating data flow, there are three specific things the DLL does.

- Framing
  - partition stream into units called frames
  - perform error detection/correction on frames
  - retransmission on frame basis

- Error Control
  - both detection and correction schemes may be used
  - what is done in response to detection depends on protocol

- Flow Control
  - prevents sender from “swamping” the receiver
  - normally built into the protocol

- At what other layer are these functions often found?

- Why the redundancy?
Framing

- Deciding, at the receiving end, where frames start and stop and filling idle periods

- Four types of framing are used
  - byte count
    - frame length given in header field
  - character-based framing:
    - special characters (flag bytes) used with byte stuffing
  - bit-oriented framing:
    - special bit sequences (flag bits) used with bit stuffing
  - violations of physical layer encoding
    - Ex. high-high or low-low in Manchester encoding
    - No character- or bit-stuffing is required

Length Count

- Header contains length of frame

- Problems
  - count may become corrupted
  - receiver has no way to tell where next frame starts

- Rarely used alone, often used in conjunction with another framing method, as a double-check

- Example:

CSC 335-Turner, Page 5
Character-Oriented Framing

- special characters
  - DLE STX - start of text
  - DLE ETX - end of text
  - SYN - synchronous idle, between frames

- Ex. IBM BISYNC

- character stuffing
  - required if delimiters appear in data
  - soln: insert extra DLE before each DLE in data
  - extra “stuffed” DLE’s extracted at receiving end

Character-Oriented Framing Example

- Simplified example of method used in PPP

```
<table>
<thead>
<tr>
<th>FLAG</th>
<th>Header</th>
<th>Payload field</th>
<th>Trailer</th>
<th>FLAG</th>
</tr>
</thead>
</table>

(a)

Original characters

A B ESC FLAG
A B ESC ESC
A ESC B ESC ESC FLAG
A ESC B ESC ESC ESC

After stuffing

A ESC B ESC ESC ESC FLAG B
A ESC B ESC ESC ESC ESC ESC B
```

CSC 335-Turner, Page 7

Figures reproduced, with permission, ©2011 Prentice Hall.
Bit-Oriented Framing

- Disadvantage of character-oriented framing:

- Start- and stop-frame pattern: 0 1 1 1 1 1 1 0

- Bit stuffing
  - sender stuffs 0 after every sequence of five 1's
  - when receiver sees five 1's followed by 0, extracts the 0
  - boundary between two frames can be unambiguously recognized

- Example:
  1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0

- Why do we have to stuff after five 1's, even if followed by a 0?

Error Correction and Detection

- Examples sources of errors
  - thermal noise
  - impulse noise: e.g., lightning
  - signal distortion
  - crosstalk
  - sender, receiver losing synchronization

- Two ways of handling errors
  - error detection
    - error is detected in frame
    - frame is retransmitted
    - ex. parity checks, cyclic redundancy codes
  - error correction
    - sender transmits redundant information
    - receiver uses this to correct errors
    - not as widely used in communications as error detection
    - ex. Hamming code

- Both error correction and error detection are widely used in computer networks. Examples?
Parity Checks

- Add an extra bit to a string of bits in order to make the total number of 1's even (even parity) or odd (odd parity).

- Example: ASCII characters contain 7 bits plus parity bit
  ASCII 'a' represented by:  _ _ 1 1 0 0 0 1

- Advantages
  - detects any single bit error
  - in fact, detects any error involving odd number of bits

- Disadvantages
  - only 50% chance of detecting burst error
  - defn: n-bit burst error: bits 1 and n inverted, others may or may not be inverted

- Single parity bit often used in computer memories.

Horizontal and Vertical Parity Checks

- A variation is to arrange (send) data as two-dimensional arrays, with a parity bit for each column and row.

  1 0 0 1 0 1 0
  0 1 1 1 0 1 0
  1 1 1 0 0 0 1
  1 0 0 0 1 1 1
  0 0 1 1 0 0 1

- Provides more detection capability in that any set of error bits confined to a single row or column will be detected

- Can even be used to correct single bit errors

- Example of errors that will be missed?

- Other drawback?

CSC 335-Turner, Page 11
Cyclic Redundancy Codes (CRC)

- Basic idea: treat string of bits as coefficients of polynomial, using modulo 2 arithmetic according to rules of algebraic field theory.

- Ex. $1\ 0\ 1\ 0\ 0\ 1$ represents $x^5 + x^3 + 1$

- Addition and subtraction are both equivalent to exclusive-or

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
+ & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
- & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}
\]

Cyclic Redundancy Codes (CRC)

- Method
  - divide string (frame) by a generator polynomial \( G(x) \)
  - tag the remainder (called a checksum) onto the frame when it is transmitted (equates to subtracting the remainder)
  - at the receiver, divide the entire frame by \( G(x) \)
  - a non-zero remainder indicates an error
CRC Example

FRAME: 1 0 1 0 0 0 1 1 0 1
\(G(x): 1 1 0 1 0 1\)
Compute Checksum as follows:

When Will A CRC Code Fail?

- \(D(x) \equiv \text{data}\)
- \(G(x) \equiv \text{generator polynomial}\)
- \(T(x) \equiv \text{transmitted bits}\)
- \(R(x) \equiv \text{received bits}\)
- \(E(x) \equiv \text{error bits}\)
- Know: \(R(x) = T(x) + E(x)\)
- What is the relationship between \(G(x)\) and \(R(x)\) for the code to fail?
- Hence, what is the relationship between \(G(x)\) and \(E(x)\) for the code to fail?
**Example of CRC Failure**

- From earlier example:
  - $G(x) = x^5 + x^4 + x^2 + 1$
  - Frame: 1 0 1 0 0 0 1 1 0 1
  - Transmitted: 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0
  - Received: 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1

**Use of CRC Codes**

- CRC codes are the most common error detection scheme used in communications

- Detection capabilities in detecting errors depends on $G(x)$.
  - all single errors if $G(x)$ contains two or more terms
  - all double errors if $x$ does not divide $G(x)$, and
    - $G(x)$ does not divide $x^k + 1$, for any $k < K$, where $K$ is the frame length
  - all odd errors if $G(x)$ contains $x + 1$ as a factor
  - all burst errors of length $r$ or less, where $r$ is degree of $G(x)$
    - burst errors of length $r + 1$ will be missed with probability $1/2^{r-1}$
    - burst errors of length $r + 2$ or more will be missed with probability $1/2^r$

- Why are checksums generally placed at the end of the frame?
**Example CRC Codes**

- CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - Has $x + 1$ as prime factor.
  - Used when character length is 6 bits. Why?

- CRC-16 = $x^{16} + x^{15} + x^2 + 1$

- CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$
  - Have $x + 1$ as prime factor.
  - Used when character length is 8 bits.
  - Catch all single, double errors, odd errors.
  - Catch all burst errors of length 16 or less.
  - Catch 99.997% of burst errors of length 17.
  - Catch 99.998% of all burst errors of length 18 or more.

- CRC-32 =
  - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11}$
  - $+ x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$
  - 99.99999997% of all burst errors of length 34 or more.

**Error Correcting Codes**

- Frame consists of $m$ data bits, $r$ check bits. $n = m + r$. An $n$-bit unit is referred to as a codeword.

- The number of bits by which two codewords differ is called the Hamming distance.

- The minimum distance found (by comparing all pairs of codewords) is the Hamming distance of the code.

- To detect $d$-bit errors, need (code) distance of $b = d + 1$. Ex. codewords with single parity bit have distance of 2.

- To correct $d$-bit errors, need (code) distance of $b = 2d + 1$. We correct to closest codeword.

- Error correcting codes seldom used in communications, except space.

- There are MANY error correcting codes, and they are used in a growing number of applications, such as CDs, modems, and wireless networks.
Correcting 1-Bit Errors

- Theory
  - \( m \) message bits, \( r \) check bits, \( n = m + r \)
  - each of the \( 2^m \) legal codewords has \( n + 1 \) bit patterns dedicated to it. WHY?

- \((n + 1)2^m \leq 2^n\)
  - divide both sides by \( 2^m \) to obtain
    \[(m + r + 1) \leq 2^r\]
  - so this is a lower bound on number of check bits
  - 11 data bits, how many check bits?
  - 16 data bits, how many check bits?
  - 32 data bits, how many check bits?

Hamming Codes

The Hamming code attains the bound presented on the prior slide.

- Features and technique:
  - it is a block code, meaning that it divides a bit stream into fixed-length blocks.
  - the bits are numbered 1 to \( n \)
  - bits numbered as powers of 2 are check bits
  - the value of each check bit \( 2^k \) depends on the parity of the bits whose label contains that \( 2^k \) when written as the sum of powers of 2
  - to find incorrect bit, determine if check bits are correct, adding \( 2^k \) to a counter if check bit \( 2^k \) is of the wrong parity
  - The sum of the incorrect check bits is called the error syndrome, which identifies the incorrect bit
Hamming Code Example

Data = 101010111010: 12 data bits, 5 check bits

Hamming codes can be used to correct burst error. How?

Convolution Codes

Convolution codes simply encode sequences of input bits to produce output bits.

- Features:
  - No natural message size or encoding boundary (it is not a block code)
  - Output depends on input plus current state
  - **Code rate**: ratio of input bits to output bits
  - **Constraint length**: number of prior bits on which an output depends

- Example:
  - Code rate \( r = \frac{1}{2} \)
  - Constraint length \( k = 7 \):
Reed-Solomon and LDPC Codes

While Hamming codes operate on individual bits, Reed-Solomon codes operate on \( m \)-bit symbols.

- Reed-Solomon code features
  - defined as polynomials operating over finite fields
  - for an \( m \)-bit symbol, codewords are \( 2^m - 1 \) symbols long
  - strong error correction properties, especially for burst errors

- Low-Density Parity Check (LDPC) codes:
  - Linear block codes
  - Output bits are formed from a small fraction of input bits, represented by a matrix with a low density of 1s
  - Received codewords are decoded with an approximation algorithm
  - Outperform most other codes. Included in 10 Gbps Ethernet and latest 802.11

Data Link Protocols

- Elementary protocols:
  - Unrestricted simplex
  - Simplex stop-and-wait, error-free channel
  - Simplex stop-and-wait, noisy channel

- Sliding window protocols:
  - Go-back-n
  - Selective repeat

- Assumptions:
  - Independent physical, data link, and network layers
  - Sender has an infinite supply of data to send
  - Machines don’t crash

- Implementation:
Unrestricted Simplex Protocol

- Assumptions
  - Data transmitted in only one direction
  - Transmitting and receiving network layers always ready
  - Negligible processing time
  - Infinite buffer space
  - No lost or damaged frames
- Of course, unrealistic.

Simplex Stop-and-Wait Protocol

- Assumptions
  - Data transmitted in only one direction
  - No lost or damaged frames
- Main concern: sender flooding receiver
  - Solution: receiver sends dummy frame as acknowledgement
  - Frames travel both directions, information only one
  - Half-duplex physical channel suffices
Simplex Protocol for Noisy Channel

- Assumptions
  - Frames may be damaged or lost
  - Receiver can (almost always) detect damaged frames using checksum

- We could use previous protocol
  - Problem occurs when what happens?
  - Result?
  - Solution?

Positive Acknowledgement with Retransmission

- Still simplex data transmission
  - 1-bit sequence number used
  - Receiver expects particular sequence number next
  - Frames with wrong number are rejected as duplicates

- Handles damaged and lost frames

- Potential problem?

SOLUTION?
Protocol Improvements

• The resulting stop-and-wait protocol will handle all problems except CRC failures (discussed earlier)

• Nonetheless, this protocol has a serious potential performance problem. What?
  □ inefficient use of bandwidth (stop and WAIT)

• Solution:
  □

• Other improvements:
  □ use the same channel for data in both directions
  □ acks, naks, data, other control frames all transmitted in both directions. Sending short control frames is wasteful, however. The solution?

• Advantages
  □
  □

Sliding Window Protocols

• Analogy: Lazy Susan

• Definitions
  □ sequence numbers range from 0 to $2^n - 1$
  □ at any instant of time, the sender maintains list of consecutive sequence numbers corresponding to frames it is permitted to send; these frames fall within the sending window
  □ the receiver maintains a receiving window corresponding to frames it is permitted to accept
  □ windows need not be same size

• 1-bit sliding window protocol is equivalent to?
### Protocol Operation

- **Sender**
  - □ sequence numbers within sender’s window represent frames sent but not yet acknowledged
  - □ receipt of ack bumps lower edge of window
  - □ sender keeps copies of all frames currently in the window, in case it needs to retransmit them
  - □ when window grows to maximum size, sender cannot accept packets from network layer until window shrinks

- **Receiver**
  - □ sequence numbers within receiver’s window represent frames that may be accepted
  - □ frames with sequence numbers outside window are dropped
  - □ received frame (with seq. number equal to lower edge of window) is accepted and window is shifted
  - □ window size remains fixed

---

**Protocol Operation Example**

![Diagram](image-url)

(a) (b) (c) (d)

CSC 335-Turner, Page 34

*Figures reproduced, with permission, ©2011 Prentice Hall.*
Sliding Window Variations

- **Go-back-n** - receiver discards all subsequent frames (subsequent sequence numbers) following an error, forcing the sender to go back to the damaged/lost frame, send it and all subsequent frames.

- **Selective repeat** - receiver stores correct frames following a bad one. Sender retransmits only the bad frame, receiver acknowledges the highest received in order.

- For go-back-n, receiver window size is \( 1 \).

- For selective repeat, window size is \( 1 \).

- Which is preferred if propagation delay is large but error rate is low? Why?

- Which is preferred if error rate is high and propagation delay is low? Why?

---

CSC 335-Turner, Page 35
Go-Back-N

- Ack of frame $j$ automatically acknowledges frames before $j$, perhaps freeing multiple buffers at once.

- Need one timer per outstanding frame, implemented in software with one hardware timer.

- Even though we have $2^n$ distinct sequence numbers, only $2^n - 1$ frames may be outstanding at any one time. Why?
Selective Repeat

- Uses less bandwidth and more buffer space than go-back-n.

- Sender
  - window starts at 0 and grows to $X$.

- Receiver
  - window remains fixed at $X$.
  - buffer for each sequence number in window
  - empty/full bit flag for each buffer
  - frame arrives, seq number checked to see if in window, check buffer to see if we already have it; if not, store

- Must be more careful in use of sequence numbers than in go-back-n...
Selective Repeat

- Solution?

- Other features of Protocol 6 (Tanenbaum)
  - after in-sequence data frame arrives, start timer. send explicit ack if no reverse traffic before timer fires
  - send negative ack (nak) if frame arrives damaged or out of order; send only one nak per lost frame

- Are naks most useful when variance of round-trip delay is large or small? WHY?

Protocol Performance

- System properties
  - $C =$ channel capacity in bps
  - $I =$ interrupt/service time + propagation delay

- Frame format
  - $D =$ number of data bits per frame
  - $H =$ number of bits in the frame header
  - $F = D + H$ (total frame length)
  - $A =$ number of bits in an ACK frame

- Error probabilities
  - $E =$ P(bit being in error)
  - $L =$ P(frame or its ACK is lost or damaged)
  - $P_1 =$ P(data frame is lost or damaged)
  - $P_2 =$ P(ACK frame is lost or damaged)
Protocol Performance (cont.)

- Protocol parameters
  - $W =$ window size
  - $T =$ timeout interval

- Performance metrics
  - $R =$ mean number of retransmissions per data frame
  - $U =$ channel utilization

Stop-and-Wait with No Errors

- At time $(F/C + A/C + 2I)$, the sender has processed the ACK

- Bandwidth occupied by one frame
  
  $= C(F/C + A/C + 2I)$
  
  $= F + A + 2CI$

- $D$ bits of data are actually sent

- So, utilization is?
Stop-and-Wait with Errors

- Lost frame uses $F + CT$ bits of transmission capacity
- $R$: mean number of retransmissions per frame
- So, total capacity used by a frame is $R(F + CT) + (F + A + 2CI)$
- Probability that the frame and ACK arrive intact is $(1 - P_1)(1 - P_2)$
- Therefore, $L = 1 - (1 - P_1)(1 - P_2)$
- Probability that exactly $k$ attempts are needed is $(1 - L)L^{k-1}$
- Expected number of transmissions per frame is:
  $$\sum_{k=1}^{\infty} (k(1 - L)L^{k-1}) = \frac{1}{1 - L} = R + 1$$
  $$R = \frac{1}{1 - L} - 1 = \frac{L}{1 - L}$$
- So, utilization is?

Sliding Window with No Errors

- In order to simplify analysis, assume:
  - Acks are piggybacked and can be ignored.
  - Interrupt processing time is negligible, so $I = \tau$, the one-way propagation delay
- Sender can send for $WF/C$ seconds before it must stop and wait.
- Ack of first frame arrives at time $F/C + 2I$.
- Case 1: Large window (sender may transmit continuously)
  $$WF/C \geq F/C + 2I$$
  $$W \geq 1 + 2CI/F$$
  Hence, $U =$?
- Case 2: Small window (sender must stop and wait)
  - $W \leq 1 + 2CI/F$
  - Sender can transmit $W$ frames in time $F/C + 2I$.
  - Therefore, $U =$?
Sliding Window with Errors

- Only consider selective repeat here.
- From before, $R = \frac{L}{1 - L}$
- So, to receive $W$ frames, have to send $\frac{W}{L}$ frames

Case 1: $W \geq 1 + 2CI/F$
- Transmission is still continuous, but extra frames must be sent to correct damaged ones
- $U = ?$

Case 2: $W \leq 1 + 2CI/F$
- For small windows, efficiency drops by the same percentage
- Therefore, $U = ?$

Protocol Performance Examples

- Boundary between large and small windows has been $W = 1 + 2CI/F$
- Reason:
- Examples of CI:
  - 10 Mbps over 1km, $CI \approx 50$ bits
  - 64 kbps over 3000 km, $CI \approx 960$ bits
  - (satellite) 64 kbps with $I = 270$ msec, $CI \approx 17000$ bits
Protocol Performance Examples

ARPANET DLC

- History
  - No true data link layer; instead, part of subnet protocol
  - IMP-to-IMP protocol was designed before sliding window concept was developed

- Features
  - byte-oriented DLC
  - 8 logically independent full duplex channels
  - each channel uses its own stop-and-wait protocol
  - each incoming packet assigned to lowest free logical channel

- Error recovery
  - only positive acks
  - return frames update acks for all 8 channels
  - retransmission of frames not ack’ed by the time their “turn” comes up

- Disadvantages
  - delivers frames out of order (usually reordered at hosts)
  - makes the job of higher layers difficult
**HDLC etc.**

- **History**
  - started with IBM's SDLC (Synchronous DLC)
  - ANSI changed it to ADCCP
  - ISO changed it to HDLC (High-level DLC)
  - CCITT changed HDLC to LAP (Link Access Procedure) and LAPB

- **Major features**
  - bit-oriented, using bit stuffing
  - use CRC-CCITT for checksum
  - arbitrarily long data
  - use 3-bit or 7-bit sliding window
  - uses ACKs and NAKs
  - HDLC and ADCCP allow selective repeat, SDLC and LAPB do not

- **Uses**
  - LAPB used in X.25 (lower 3 layers in most public networks)
  - LAPD used for D channel in ISDN

- **Limitations:** many variations with different options for each

**The Data Link Layer in the Internet**

Much Internet access is through multipoint lines in LANs

- Point-to-point communication occurs primarily in two situations
  - Connecting routers
  - Connecting homes to the internet
  - Usual situation:
    - User connects with modem
    - Home PC temporarily becomes an internet host

CSC 335-Turner, Page 51

*Figures reproduced, with permission, ©2011 Prentice Hall.*
Point-to-Point Protocol (PPP)

- Constructed by IETF to improve SLIP (RFCs 1661, 1662, 1663, ...)
- Provides:
  - framing with error detection
  - Link Control Protocol to manage lines
  - Network Control Protocol to negotiate network layer options (incl. dynamic allocation of IP addresses)
- Frame format
  - like HDLC only byte-oriented
  - Address: all stations accept 11111111
  - Control: 00000011 – unnumbered
  - Protocol: IP, IPX (Novell), Appletalk, LCP, NCP, ...
  - Payload max: 1500 bytes. WHY?
  - 16- or 32-byte checksum

Asymmetric Digital Subscriber Loop (ADSL)

- Overall operation:
  - PC to DSL modem uses IP on top of Ethernet
  - DSL modem to DSLAM uses ADSL with (possibly) ATM, ATM AAL5, and/or PPP
- Asynchronous Transfer Mode (ATM):
  - virtual-circuit datagram network
  - uses fixed-length cells of 48 + 5 bytes
- AAL5
  - Performs a mapping of data to a sequence of ATM cells
  - AAL5 frame format: