Issues of DLC in DLL

- Framing (Frame synchronization)
- Flow control (already covered)
- Error detection and error control (covered partially)
- Addressing (SAP and MAC address)
- Control and management of the data link

Link Layer: Implementation

- Implemented in “adapter”
  - e.g., PCMCIA card, Ethernet card
  - typically includes: RAM, CPU/DSP chips, host bus interface, and link interface

Framing

- a technique for identifying the beginning and the end of a frame in a bit stream (frame synchronization)

- Three Methods:
  1. Character-oriented (so called “character stuffing”)
  2. Bit-oriented (so called “bit stuffing”)
  3. Based upon “encoding violation”
**Character-oriented Framing**

Frame ➔ a sequence of 8-bit characters starting with **DLE** STX and ending with **DLE** ETX

**Example:**

```
DLE STX A DLE B ... DLE ETX
```

*character stuffed* (Check with IRA)

---

**Bit-oriented Framing**

Frame ➔ a sequence of bits starting with “01111110” and ending with “01111110”

**Example:**

```
0111110 110110 011111 10...0111110 110110 01111110
```

*transmitted frame*

*5 1's bit stuffed*

---

**Based on the Encoding Violation**

Frame ➔ indicating the begin and the end by using an invalid code on the physical layer

**Example:** Manchester encoding

- valid “0” ➔
- valid “1” ➔
- 1-bit interval

Two invalid code ➔

*can be used to indicate frame boundaries*

---

**Flow Control**

~ to make sure that a transmitting entity does not overwhelm a receiving entity with data
- W/o flow control, Rxer’s buffer may fill up and overflow (starting dropping incoming data) while it is processing old data

- **Two types of flow control:**
  1. Stop-and-Wait FC
  2. Sliding Window FC
Stop-and-Wait Flow Control Protocol

- Simplest form of FC
- Rxer indicates its readiness to receive data for each frame (Txer and Rxer have only ONE buffer, separately)
- **Operations:**
  1. Sender: Transmits a single frame
  2. Receiver: sends an ACK upon reception (when it is willing to receive the next frame)
  3. Go to 1 until transmission is completed.

Sliding Window FC Protocol

- **Basic principle:**
  - Assigns each frame a k-bit sequence number
  - Frames are numbered as by modulo $2^k$;
    Range of sequence number = $[0, 1, \ldots, 2^k-1]$;
  - Both Txer and Rxer have buffer size more than one
  - Multiple frames can be sent at a time
    ~ provided that the Txing buffer is available
  - Reception of multiple frames can be acknowledged in one message
  - Rxer ACKed by the sequence number of the next frame expected

Sliding Window – in a detail

- To improve the utilization of the channel in the cases of $T_{prop} > T_{frame}$ by allowing multiple frames to be transmitted before receiving ACK(s) (to improve the performance of the stop-and-wait mechanism)
- To keep track of which frames without waiting for any ACKed, each frame is labeled with sequence number.
- **Rule of sliding window:**
  - Txer maintains a list of SEQ numbers that it is allowed to send
  - Rxer maintains a list of SEQ numbers that it is prepared to receive
  - Frames are numbered $(0 \sim 2^k-1)$ modulo $2^k$.
  - The window size is $2^k$, and the SEQ # has a bounded size since it occupies a field in the frame
  - Sender must buffer these frames in case they need to be retransmitted

Explanation – a pipeline approach

- Frames buffered until acknowledged
- Window of frames that may be transmitted
- Frames already transmitted
- Window shrinks from trailing edge as frames are sent
- Window expands from leading edge as acks are received

- Frames already received
- Window of frames that may be accepted
- Frames buffered until acknowledged
- Window shrinks from trailing edge as frames are sent
- Window expands from leading edge as acks are sent
Sending window:
- At any instant, the sender is permitted to send a certain number of frames. The sequence numbers of these frames are said to fall within the “sending window”.

Receiving window:
- The receiver maintains a “receiving window” corresponding to the number of frames it allows to receive.

Note:
- Usually, window size $\leq 2^k$
- For a FDX system, each node must maintain two windows (buffers): one for Txer and the other for Rxer
- Rxer can issue RNR # (receiver not ready) to cut off the flow of frames and send normal ACK to reopen the window.

Error Detection
- Additional bits added by transmitter for error detection code
- Data can be corrupted during transmission
  - Bits lost, bit values changed
- Frame includes additional information to detect or correct even error

Set by sender
Checked by receiver

Error Detection Techniques
- Parity
  - Even*, odd*, two-dimension parity
  - Cannot handle error that changes two bits*
  - Value of parity bit is such that character has even (even parity) or odd (odd parity) number of ones
  - Even number of bit errors goes undetected

- Hamming code
- Checksum (various rules)
  - Can detect multiple bit errors
- Cyclic Redundancy Check (CRC)
  - Can detect more errors (single-bit, odd/even number of bits, burst error, etc.)
Parity Checking

- **Single Bit Parity:**
  - Detect single bit errors

- **Two Dimensional Parity:**
  - Detect and correct single bit errors

Horizontal & Vertical Parity check bits →

- Ex: Memory systems

Checksum

- To *compute* the checksum, the sender
  - treats data as sequence of integers (computed over data)
  - computes and sends arithmetic sum (appended to frame)
  - cannot detect all common errors

- **Example (16-bit checksum)**
  
  Hello world
  
  48 65 6C 6F 20 77 6F 72 6C 64 20 69 73 20 61 20 20 6D 65 73 73 73

  - If the sum grows larger than 16-bit, the carry bits are added into the final sum.

  - Ex: Used in TCP/IP very, BIOS/Boot ROOM

Illustration of Checksum Fail

The receiver will declare the packet has a valid checksum.

The sender will ensure the same checksum.

Cyclic Redundancy Check (CRC)

- **CRC can detect . . .**
  - All single-bit error
  - All double-bit errors, as long as P(x) has at least three 1’s
  - Any odd number of errors, as long as P(x) contains (x+1)
  - Any burst error with *burst length* < length of P(x)
    (i.e., the length of FCS)
  - Most larger burst error (a small set of bits near a single location)
    and vertical error (a specific set of bits near a single location)

- Ex: Ethernet, IEEE802 LANs, ATM header, HDLC/PPP

- Intuitively, CRC can detect more errors than Checksum does.
Error Control

• Automatic Repeat Request (ARQ)
  ~ handle errors: lost frames and damaged frames (erred)
  ~ the retransmission mechanisms that turn an unreliable data link into a reliable one

• Three standardized ARQs
  1. Stop-and-Wait ARQ
  2. Go-Back-N ARQ
  3. Selective-Reject ARQ

* All ARQs are based upon the sliding window flow control scheme.

Stop-and-Wait ARQ

Station A B

Go-Back-N ARQ

Station A B

Selective-Reject ARQ

Station A B

Selective-Repeated ARQ

A just txed frame \(i+1 = 5\) later, B rxd frame 5 out-of-order, and sends a REJ 4
**Multiple Access Links and Protocols**

- Three types of “links“:
  1. **point-to-point** (single wire, e.g. PPP, SLIP)
  2. **broadcast** (shared wire or medium; e.g, Ethernet, Wireless LAN, etc.)
  3. **switched** (e.g., switched Ethernet, ATM, etc.)

**Goal:** efficient, fair, simple, and decentralized

---

**MAC Protocols: a taxonomy**

**Three broad classes:**

- **Channel Partitioning (fixed)**
  - divide channel into smaller "pieces" (e.g., time slots, frequency (carrier), etc.)
  - allocate piece to node for exclusive use
- **Random Access**
  - allow collisions
  - "recover" from collisions - need to be taken care
- **Taking turns**
  - tightly coordinate shared access to avoid collisions

---

**Channel Partitioning MAC protocols - TDMA**

**TDMA: time division multiple access**

- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle → wasted
- **Example:** 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle
- **TDM (Time Division Multiplexing) → at Physical medium**
  - channel divided into N time slots, one per user; inefficient with low duty cycle users and at light load.

---

**Channel Partitioning MAC protocols - FDMA**

**FDMA: frequency division multiple access**

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- **Example:** 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle
- **FDM (Frequency Division Multiplexing):** frequency subdivided.
FDD/FDMA - general scheme

- Example: AMPS

![Diagram showing frequency allocation for FDD/FDMA]

Channel Partitioning - CDMA

CDMA (Code Division Multiple Access)
- unique “code” assigned to each user; i.e., code set partitioning
- used mostly in wireless broadcast channels (cellular, satellite, Wireless LAN, Bluetooth, etc.)
- all users share same frequency, but each user has own "chipping" sequence (i.e., code) to encode data
- encoded signal = (original data) x (chipping sequence)
- decoding: inner-product of encoded signal and chipping sequence
- allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")

CDMA Encode/Decode

Channel (wireless)

CDMA: two-sender interference

senders

Channel (wireless)
Random Access Protocols

- When a node has packet to send
  - transmit at full channel data rate R.
  - no a priori coordination among nodes
- (if) two or more transmitting nodes \( \rightarrow \) "collision"
- Random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - Pure ALOHA
  - Slotted ALOHA
  - CSMA and CSMA/CD (in Ethernet, IEEE 802.3)*
  - CSMA/CA (in IEEE 802.11 Wireless LAN)**

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- pkt needs transmission:
  - send without awaiting for beginning of time slot
- collision probability increases: Ask for no . . .
  - pkt sent at \( t_0 \) collide with other pkts sent in \( [t_0-1, t_0+1] \)

Pure Aloha (cont.)

\[
P(\text{success by given node}) = P(\text{node transmits at } t_0) \cdot \]

\[
P(\text{no other node transmits in } [t_0-1, t_0] \cdot P(\text{no other node transmits in } [t_0-1, t_0] = p \cdot (1-p) \cdot (1-p)
\]

\[
P(\text{success by any of } N \text{ nodes}) = N [p \cdot (1-p) \cdot (1-p)]
\]

... choosing optimum \( p \) as \( N \rightarrow \infty \ldots \)

\[
= 1/(2e) = .18 \text{ (or } 18\%	ext{)}
\]

Slotted Aloha

- time is divided into equal size slots (slot = pkt trans. time)
- node with new arriving pkt: transmit at beginning of next slot
- if collision: retransmit pkt in future slots with probability \( p \), until successful.

Success (S), Collision (C), Empty (E) slots
**Slotted Aloha’s Efficiency**

**Q:** what is max fraction slots successful?  
**A:** Suppose N stations have packets to send  
- each transmits in slot with probability $p$  
- prob. of successful transmission, $S$, is:

by single node: $S = p \cdot (1-p)^{(N-1)}$  
by any of N nodes  
$S = \text{Prob. (only one transmits)}$  
$= N \cdot p \cdot (1-p)^{(N-1)}$  
... choosing optimum $p$ as $n \rightarrow \infty$ ...  
$= \frac{1}{e} = 0.37$ (or 37%) as $N \rightarrow \infty$

*At best: channel use for useful transmissions 37% of time!*  

- Penalty: need to keep synchronized!

---

**CSMA: Carrier Sense Multiple Access**

**CSMA:** listen before transmit:  
- If channel sensed idle: transmit entire pkt  
- If channel sensed busy, defer transmission  
  - Persistent CSMA: retry immediately with probability $p$  
    when channel becomes idle (*may cause instability ?*)  
    ($0 < p < 1$)  
  - Non-persistent CSMA: retry after random interval  
  - "CS" to "human analogy" ~ *don't interrupt/bother others!*

---

**Classes of CSMAs**

- **0-persistence (nonpersistent)**  
  - Listen to channel before transmitting . . .  
    - Idle channel  \(\rightarrow\) send frame immediately  
    - Busy channel  \(\rightarrow\) wait for a random time and sense channel again

- **1-persistent (greedy)**  
  - Listen to channel before transmitting . . .  
    - Idle channel  \(\rightarrow\) send frame immediately  
    - Busy channel  \(\rightarrow\) wait until channel idle (continuously sense)

- **$p$-persistent (used in slotted CSMA)**  
  - Listen to channel before transmitting . . .  
    - Idle channel  \(\rightarrow\) send frame at current slot with probability $p$  
    - Busy channel  \(\rightarrow\) wait and sense at next time slot

---

**CSMA Collisions**

- **collisions can occur:**  
  ~ propagation delay means two nodes may not hear each other’s transmission

- **collision:**  
  ~ entire packet transmission time wasted

- **note:**  
  ~ role of distance and propagation delay in determining collision prob.
CSMA/CD (Collision Detection)

**CSMA/CD**: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- persistent or non-persistent retransmission

- **Collision Detection (CD)**:
  - easy in wired LANs: measure signal strengths/periods, compare transmitted and received signals, etc.
  - difficult in wireless LANs: receiver shut off while transmitting (Collides at Rxer, Txer listening for ACK)
- Human analogy: the polite conversationalist

Visualization - Collision Detection

Channel Utilization for CSMA and ALOHA

"Taking Turns" MAC protocols

- **Channel partitioning MAC protocols**:
  - share channel efficiently at high load
  - inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!
- **Random access MAC protocols**:
  - efficient at low load: single node can fully utilize channel
  - high load: collision overhead

- **"Taking Turns" protocols** (round-robin, 循環賽)
  - look for best of both (above two) worlds!
"Taking Turns" MAC protocols

**Polling protocol:**
- *master* node "invites" (giving permission to) *slave* nodes to transmit in turn (cyclically)
- Request to Send, Clear to Send msgs
  - eliminate collision, improve efficiency
- Concerns (drawbacks):
  - polling overhead
  - Latency (polling delay)
  - single point of failure (if the master fails)

**Token passing/Ring protocol:**
- control *token* passed (exchanged) from one node to next sequentially.
- token message (node hold token in order to transmit its message)
- concerns:
  - token overhead
  - latency
  - single point of failure (token)

---

Token Ring: IEEE 802.4 Standard

- *(a)* C transmits frame addressed to A
  - *(c)* A copies frame as it goes by
- *(b)* Frame is not addressed to B; B ignores it
- *(d)* C absorbs returning frame

---

Token Bus: IEEE802.5 standard

- Token passed in MAC address’ descending order

- Once received the token, the station either starts to transmit or passes the token to the next station within one response window.
- Station 50 passes a token to station 30 via the broadcast bus.
- Although all the stations in the bus can see the token, but only station 30 has the right to use the token.

---

Reservation-Based Protocols

**Distributed Polling:**
- time divided into slots
- begins with N short reservation slots (minislots)
  - reservation slot time equal to channel end-end propagation delay
  - station with message to send posts reservation
  - reservation seen by all stations
- after reservation slots, message transmissions ordered by known priority
Summary of MAC protocols

- What do you do with a **shared** media?
  - **Channel Partitioning** (Fixed)
    - by time, frequency or code
    - Time Division, Code Division, Frequency Division
  - **Random partitioning** (dynamic),
    - ALOHA, S-ALOHA, CSMA, CSMA/CD
    - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - **Taking Turns**
    - polling from a central cite, token passing

---

Point-to-Point Data Link Control

- One sender, one receiver, one link: easier than broadcast link:
  - no Media *Access Control*
  - no need for explicit MAC *addressing*
  - e.g., dialup link, ISDN line, a SONET/SDH link, an X.25 connection
- Popular point-to-point DLC protocols:
  - **PPP** (point-to-point protocol)
  - **HDLC**:
    - High level data link control (Data link used to be considered "high layer" in protocol stack)
- Related RFCs: 1547, 1661, 2157, 1662 (HDLC)

---

PPP Design Requirements [RFC 1547]

- **Packet framing**: encapsulation of network-layer datagram in data link *frame* (to identify the start and the end of a frame)
  - carry network layer data of any network layer protocol data (not just IP) *at the same time*
  - ability to *demultiplex* upwards
- **Bit transparency**: must carry any bit pattern in the data field (cannot forbid the use of certain bit patterns in the network-layer packet)
- **Error detection** (no correction)
- **Connection liveness**: detect link failure at link level, signal link failure to network layer
- **Network layer address negotiation**: endpoint can learn/configure each other’s network address
- **Multiple link types**: e.g., serial/parallel, synchronous/Asynch., low-/high-speed, electrical/optical
- Simplicity

---

PPP Non-Requirements

**Non-goals ~ not to be implemented**

- no error correction/recovery
- no flow control
- No resequencing ~ out of order delivery OK
- no need to support multipoint links (e.g., polling)
  - HDLC can accommodate multiple receivers
  
  Error recovery, flow control, data re-ordering all relegated to higher layers!
PPP Data Frame (RFC 1662)

- **Flag**: delimiter (framing)
- **Address**: does nothing (only one option)
- **Control**: does nothing; in the future possible multiple control fields
- **Protocol**: upper layer protocol to which frame delivered (e.g., PPP-LCP = C021, IP = 21, IPCP = 8021, hexadecimal)

```
01111110 11111111 00000001 01111110
flag protocol info check flag
```

Why bother?

### Byte Stuffing

Asynchronous HDLC, actually

- "Data transparency" requirement: data field must be allowed to include flag pattern "01111110"
  - Q: Is received "01111110" data or flag?
- Sender: adds ("stuffs") extra "01111110" byte after each "01111110" data byte
- Receiver: two "01111110" bytes in a row: discard first byte, continue data reception; if single "01111110" flag byte in data to send

Actually, 7D, data XOR 20

PPP Data Control Protocol

Q: How does the link get initialized when a host/router is first turn on?

A: using **PPP-Link Control Protocol** and **Network Control Protocols**

- Before exchanging network-layer data, data link peers must . . .
  - configure PPP link (max. frame length, authentication)
  - learn/configure network

- for IP: carry IP Control Protocol (IPCP) msgs (protocol field: 8021) to configure/learn IP address
RFC 1661 PPP Link Phases

Begin/end

- dead
- terminating
- open

LCP Negotiation
- Up
- LCP echo and reply
- Maintaining PPP link
- NCP Negotiation
- Exchange network control packets to configure ntwk module
- e.g., IPCP (RFC 1332)
- DECNet (RFC 1762)

PPP begins sending network-layer data