Data Link Layer:
Overview, operations

Chapter 3

Outlines

1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

1. Data Link Layer Functions

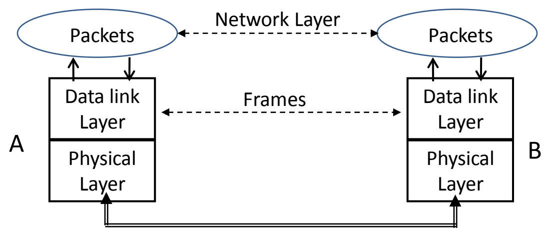
- Provides a *well-defined service interface* to the network layer.
- Determines how the bits of the physical layer are grouped into frames *(framing).*
- Deals with transmission errors *(CRC and ARQ).*
- Flow control: regulates the flow of frames.
- Performs general link layer management. (seq #, protocols, address etc)

Outlines

1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

2. Data Link Services

- Network layer has bits,
- Says to data link layer:
  - "send these to this other network layer",
- Data link layer sends bits to other data link layer,
- Other data link layer passes them up to network layer.



2.1. Types of Services

- Acknowledged/Unacknowledged
  - Receiver returns acknowledgement (ACK) for each transmitted frame.
- Connection oriented/Connectionless
  - Setup a logical connection before transmitting frames.

Outlines

1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

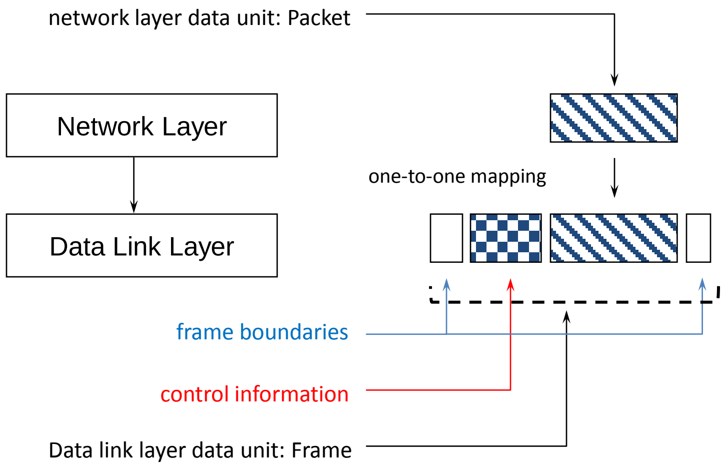
3. Framing

- Data link breaks physical layer stream of bits into *frames*

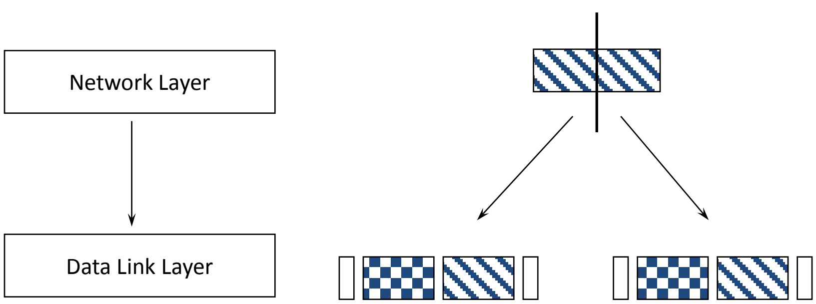
  ...0101101001010011010100 10...

- The data unit at the data link layer is the "frame",
- Issues:
  1. Frame creation
  2. Frame delineation
     - ✓ How does receiver detect boundaries?

3.1. Frame Creation



3.1. Frame Creation *(cont ́d)*

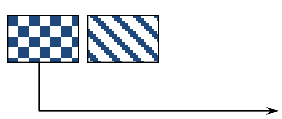


3.2. Frame Delineation

- How to tell when a new frame starts:
  - Character count
  - Frame tags with character stuffing
  - Frame tags with bit stuffing

3.2. Frame Delineation

Delineation by character count

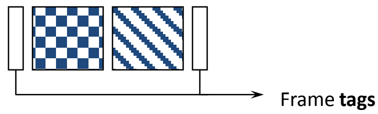


Control Field containing character count

- Character count lists the number of characters in the data field of the frame
- Problem: corrupted control fields

3.2. Frame Delineation

Frame tagging with character stuffing



- Use starting and ending characters (**tags**) to mark boundaries of frame.
- Problem: What if tag character occurs in the data or control portions of the frame?

3.2. Frame Delineation

Character stuffing

- Insert extra escape characters when a tag appears in data field

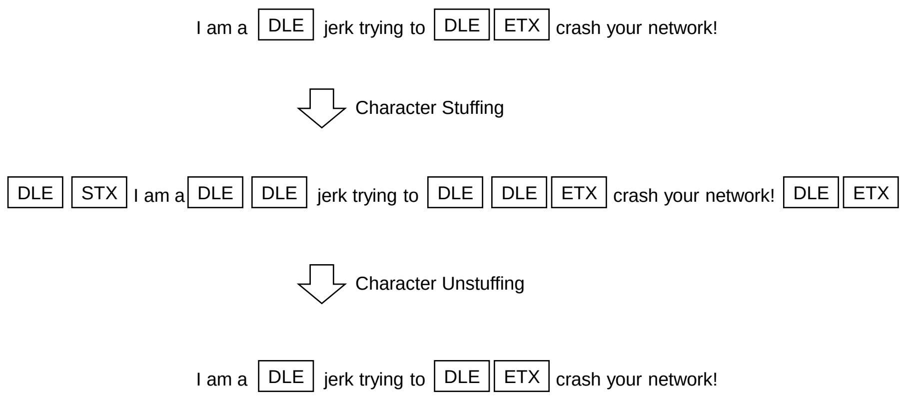
DLE STX Start Tag

DLE ETX End Tag

DLE DLE Character Stuffed DLE code

3.2. Frame Delineation

Character Stuffing Example



3.2. Frame Delineation

Frame tagging with bit stuffing

- Bit strings may be used instead of character sequences to delineate frames
- More efficient

3.2. Frame Delineation

Bit stuffing

- Each frame begins with a start and end bit sequence, e.g., 01111110
- When sender's data link layer sees five 1's in a row, it stuffs a zero bit
- The receiver "unstuffs" a zero after five consecutive 1's.

At the sender: 1 1 1 1 1 1 → 1 1 1 1 1 0 1

1 1 1 1 1 0 → 1 1 1 1 1 0 0

3.2. Frame Delineation

Bit Stuffing Example

001100101111111100010100

Sender ⇩ Bit Stuffing

01111110001100101111101100010100011111110

Receiver ⇩ Bit Unstuffing

001100101111111100010100

Outlines

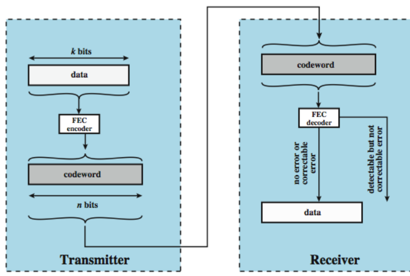
1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

4. Error Detection/Correction

- No physical link is perfect
- Bits will be corrupted
- We can either:
  - detect errors and request retransmission
  - or correct errors without retransmission

4. Error Detection/Correction

1. Original data (k bits)
2. Sender: apply error control technique
   Insertion/field addition (n bits)
3. Receiver: receives message
   recalculates
4. Comparison: received and recalculated



4.1. Error Detection

Parity bit technique

- Append a single parity bit to a sequence of bits.
- If using “odd” parity, the parity bit is chosen to make the total number of 1’s in the bit sequence odd.
- If “even” parity, the parity bit makes the total number of 1’s in the bit sequence even.

Example:

Transmitted Sequence	Parity	Parity Bit
00010101	even	1
01111	even	0
11111111	odd	1
10011	odd	0

3.2. Error Control

Polynomial Codes

- Can detect errors on large chunks of data,
- Has low overhead,
- More robust than parity bit,
- Requires the use of a "code polynomial",
  - Example: $x^2 + 1$

3.2. Error Control

Cyclic Redundancy Check

- CRC: Example of a polynomial code
- Procedure (at the sender):

1. Let **r** be the degree of the code polynomial **C(x)**.
   (Both sender and receiver know the code polynomial)
   Append **r** zero bits to the end of the message bit string.
   Call the entire bit string **S(x).**
2. Divide **S(x)** by the code polynomial **C(x)** using modulo 2 division.
3. Subtract the remainder from **S(x)** using modulo 2 subtraction. (call resulting polynomial **t(x)**.)
4. Transmit the checksummed message **t(x)**.

3.2. Error Control

Background

- **S(x)** = f(x)**C(x)** + remainder
- **S(x)** - remainder = f(x)**C(x)** = t(x)
  - sender transmits t(x)
  - note that t(x) is divisible by **C(x)**
  - if the received sequence at the receiver is not divisible by C(x), error has occurred

3.2. Error Control

Generating a CRC : *Example*

Message: 1011 → $1 \times x^3 + 0 \times x^2 + 1 \times x^1 + 1 \times x^0$
$= x^3 + x + 1$

MSB
Most Significant Bit

LSB
Low Significant Bit

Code Polynomial C(x): $x^2 + 1$ (101)

Step 1: Compute S(x)

r = 2

S(x) = 101100 $(x^5 + x^3 + x^2)$

3.2. Error Control

Generating a CRC *Example (cont'd)*

Step 2: Modulo 2 divide

```
                         1001
C(x)  ---->  101 |  101100   <----  S(x)
                    101
                    ---
                     001
                     000
                     ---
                      010
                      000
                      ---
                       100
                       101
                       ---
                        01   <----  Remainder
```


3.2. Error Control

Generating a CRC *Example(cont'd)*

Step 3: Modulo 2 subtract the remainder from **S(x)**

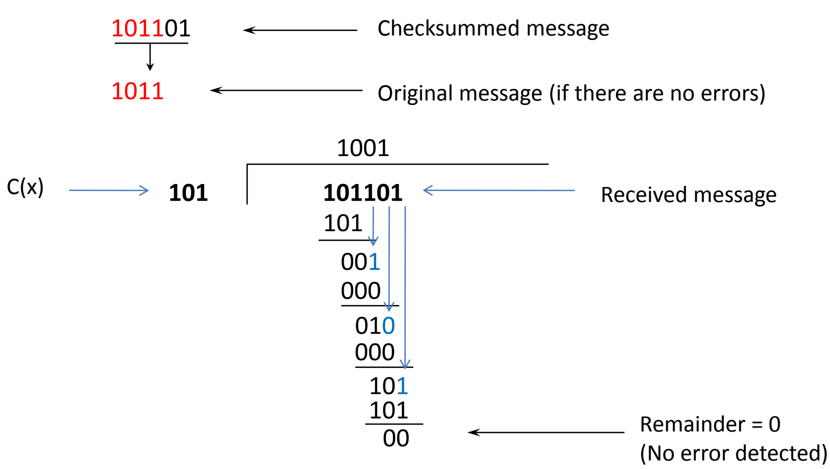
```
 101100
-    01
 ------
 101101
```

Checksummed Message **t(x)**

3.2. Error Control

- Procedure (at the receiver)
  - Divide the received message by the code polynomial **C(x)** using modulo 2 division. If the remainder is zero, there is no error detected.

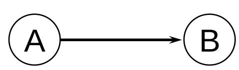
Decoding a CRC *Example*



Outlines

1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

5. Flow Control



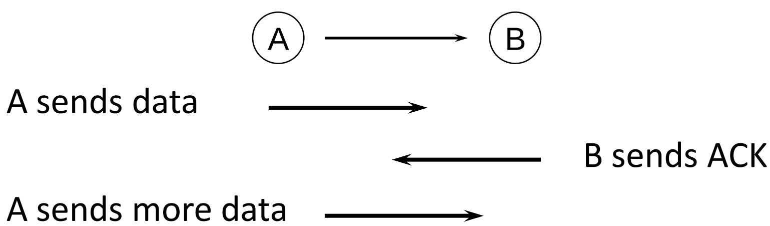
If A sends at a faster rate than B can receive, bits will be lost

We need flow control!

Some Flow Control Algorithms

- Simplex protocols : Stop and Wait
  - Data only flows in one direction
  - Acknowledgement stream may flow in the other direction
- Full duplex protocols
  - Sliding window with Go Back N
  - Sliding window with Selective Repeat

5.1. Stop-and-Wait



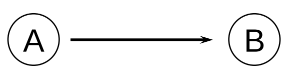
- The receiver sends an acknowledgement frame telling the sender to transmit the next data frame.
- The sender waits for the ACK, and if the ACK comes, it transmits the next data frame.

5.1. Stop-and-Wait

- What if the frame will be lost?
  - The sender waits the acknowledgement within a "timeout" delay and then retransmits again the same frame.

- What if the ACK will be lost?
  - The sender waits the acknowledgement within a "timeout" delay and then retransmits again the same frame.
  - The receiver discard the duplicated frame (seq nb) and send again the same ACK.

5.2. Windowed Flow Control



A sends packets 1, 2, 3 →

← B sends ACK for 1, 2

A sends packets 4, 5 →

← B sends ACK for 3, 4, 5

5.2. Full Duplex Flow Control Protocols

Data frames are transmitted in both directions

Sliding Window
Flow Control Protocols

5.2.1. Sliding Window Protocols: *Definitions*

Sequence Number: Each frame is assigned a sequence number that is incremented as each frame is transmitted

Sender's Window: Keeps sequence numbers of frames that have been sent but not yet acknowledged

Sender Window size: The number of frames the sender may transmit before receiving ACKs

Receiver's Window: Keeps sequence numbers of frames that the receiver is allowed to accept

Receiver Window size: The maximum number of frames the receiver may receive out of order

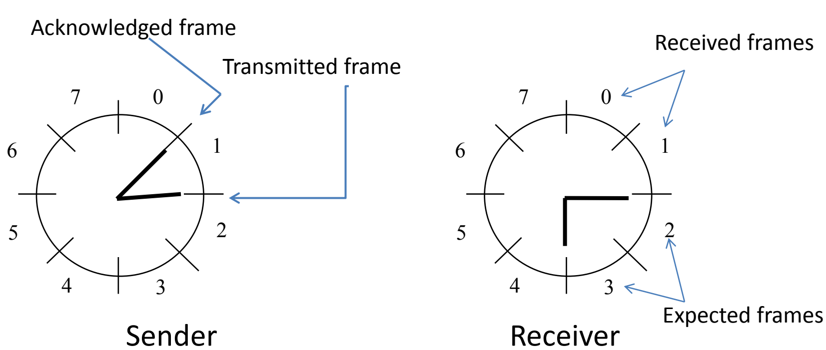
5.2.1. Sliding Window Protocols:

General Remarks

- The sending and receiving windows do not have to be the same size
- Any frame which falls outside the receiving window is discarded at the receiver
- Unlike the sender's window, the receiver's window always remains at its initial size

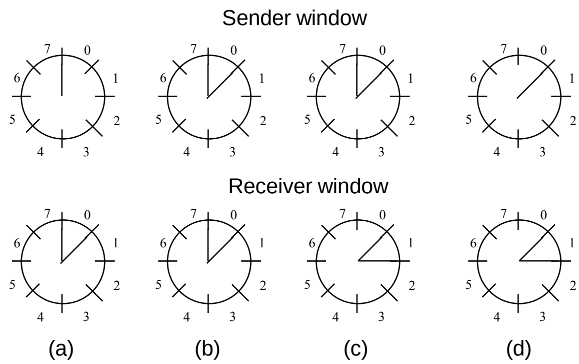
5.2.2. Simple Sliding Window (Window Size = 2)

A sliding window with a maximum window size of 2 frames



Window for a 3-bit sequence number (000=0, 001=1,010= 2 … 7)

Sliding Window example (w=1)



(a) Initial state, no frames transmitted
(b) Sender transmits frame 0
(c) Receiver receives frame 0 and ACKs
(d) Sender receives ACK

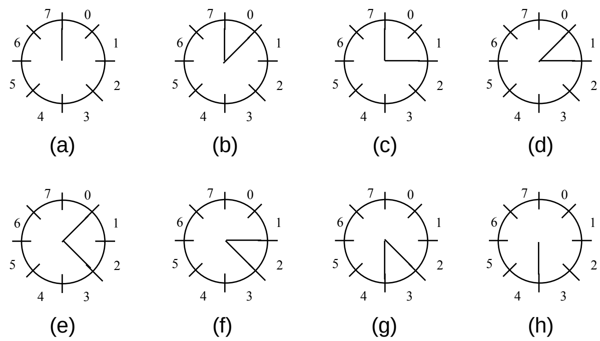
This protocol behaves identically to stop and wait.

Sliding Window with Window Size *W*

With a window size of 1, the sender waits for an ACK before sending another frame.

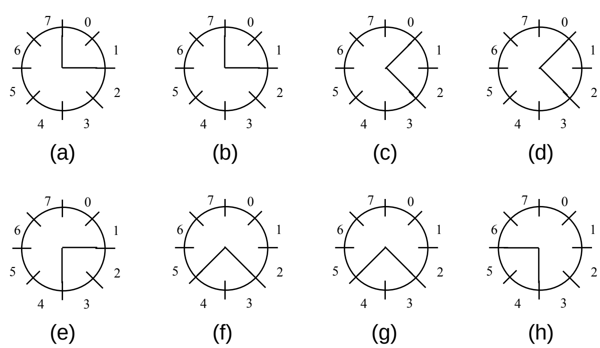
With a window size of **W**, the sender can transmit up to **W** frames before "being blocked".

Sender-Side Window (window Size W=2)



(a) Initial window state
(b) Send frame 0
(c) Send frame 1
(d) ACK for frame 0 arrives
(e) Send frame 2
(f) ACK for frame 1 arrives
(g) ACK for frame 2 arrives, send frame 3
(h) ACK for frame 3 arrives

Receiver-Side Window (Window Size W=2)



(a) Initial window state
(b) Nothing happens
(c) Frame 0 arrives, ACK frame 0
(d) Nothing happens
(e) Frame 1 arrives, ACK frame 1
(f) Frame 2 arrives, ACK frame 2
(g) Nothing happens
(h) Frame 3 arrives, ACK frame 3

What about Errors?

What if a data or acknowledgement frame is lost when using a sliding window protocol?

Two Solutions:

- Go Back N
- Selective Repeat

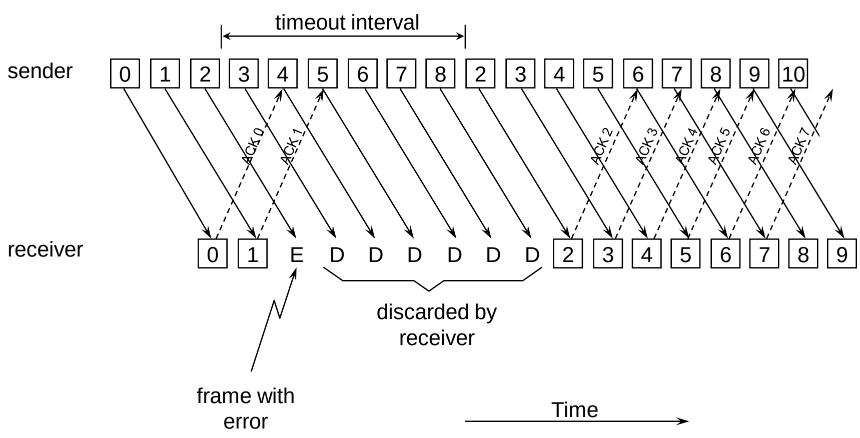
What about Errors? *Cont'd*

- One very important note about acknowledgement
  - Ack for frame n = I am expecting frame n+1 (not "I received fame n")

Sliding Window with Go Back N

- When the receiver notices a missing or erroneous frame, it simply discards all frames with greater sequence numbers and sends no ACK
- The sender will eventually time out and retransmit all the frames in its sending window

5.3. Go Back N



5.3. Go Back N *(cont'd)*

Go Back N can recover from erroneous or missing packets

But...

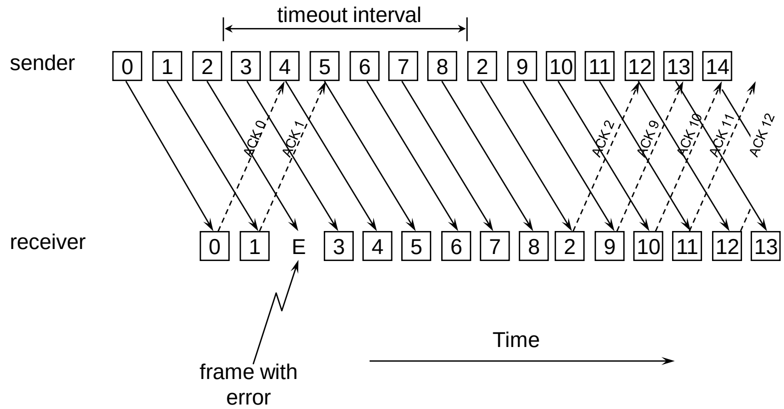
It is wasteful. If there are a lot of errors, the sender will spend most of its time retransmitting useless information

5.4. Sliding Window with Selective Repeat

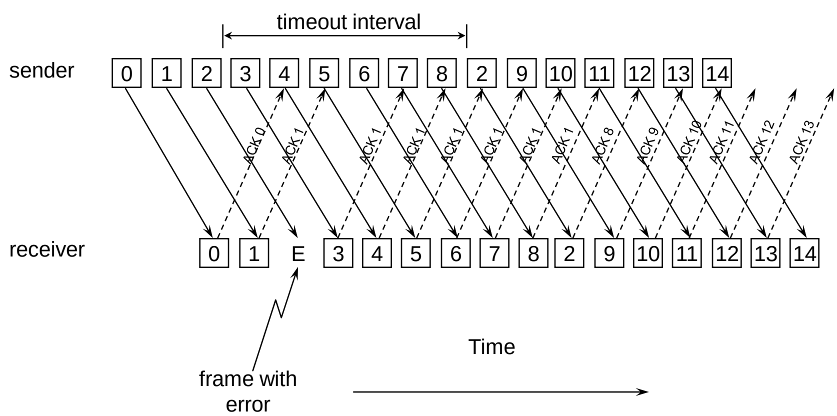
The sender retransmits only the frame with errors

- The receiver stores all the correct frames that arrive following the bad one. (Note that this requires a significant amount of buffer space at the receiver.)
- When the sender notices that something is wrong, it just retransmits the one bad frame, not all its successors.

5.4. Selective Repeat



5.4. Selective Repeat: a variant



5.4. Selective Repeat: *cont΄d*

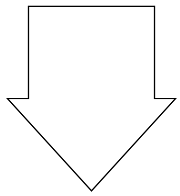
- In this scheme, every time a receiver receives a frame, it sends an acknowledgement which contains the sequence number of the next frame expected

Outlines

1. Data Link Layer Functions
2. Data Link Services
3. Framing
4. Error Detection/Correction
5. Flow Control
6. Medium Access

9. Medium Access

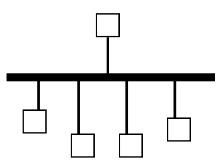
Many users typically share a single link or a single medium

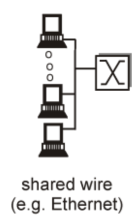


How do you give them all access?

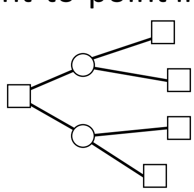
9.1. Possible Media

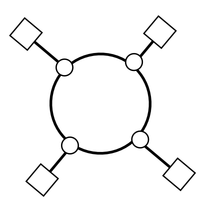
- Broadcast or shared channels





- Point-to-point links





9.2. Multiple Access Protocols

If more than one host sends at the same time, there is a collision

Need algorithm to share the channel:

Multiple access protocol

distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit

9.2. MAC Protocols: a taxonomy

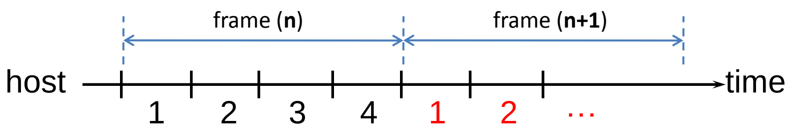
Three broad classes:

- Channel Partitioning
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - “recover” from collisions
- “Taking turns”
  - Nodes take turns, but nodes with more to send can take longer turns

9.2.1. Channel Partitioning MAC protocols
(Fixed Assignment Schemes)

Example: Time Division Multiple Access (TDMA)

It allows several users to share the same frequency channel by dividing the signal into different time slots. The users transmit one after the other using his own time slot.

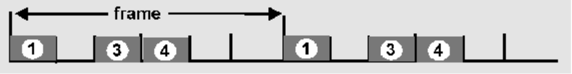


Channel capacity is assigned even to users who have nothing to send

9.2.1. 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,
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle.

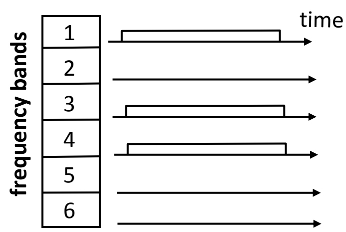


- inefficient with low duty cycle users and at light load.

9.2.1. 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



9.2.2. Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "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:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

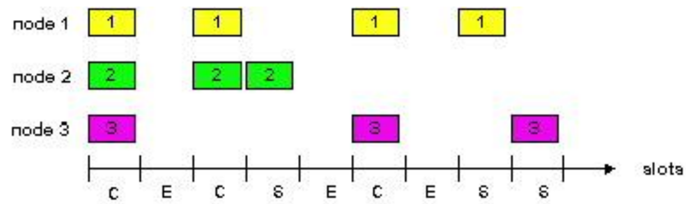
Assumptions

- all frames same size
- time is divided into equal size slots, time to transmit 1 frame
- nodes start to transmit frames only at beginning of slots
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

Operation

- when node obtains fresh frame, it transmits in next slot
- no collision, node can send new frame in next slot
- if collision, node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

CSMA (Carrier Sense Multiple Access)

CSMA: listen before transmit:

If channel sensed idle: transmit entire frame

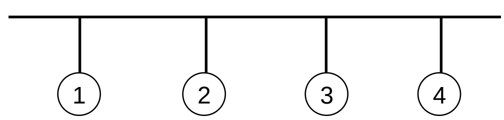
- If channel sensed busy, defer transmission

- Human analogy: don't interrupt others!

CSMA/CD (Collision Detection)

Send when you have a packet to send. If collision, retransmit

Example: Ethernet



Listen before transmission
if busy, wait
if idle, transmit

Listen during transmission
if collision, abort
and retransmit

CSMA/CD (Collision Detection)

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

- Collision Detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: receiver shut off while transmitting
- human analogy: the polite conversationalist

9.2.3. “Taking Turns” MAC protocols

Polling:
- master node “invites” slave nodes to transmit in turn
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)

Token passing:
- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)