**Link Layer and Local Area Networks (LANs)**

**Link Layer: Introduction**

For the Link Layer, we focus within a Subnet.

Terminology

* **Nodes:** Hosts and Routers
* **Links**: Communication channels that connect adjacent nodes along the path: Wired links, Wireless links, LANs
* **Layer-2 packet**: Frame, encapsulates datagram

The **Data-Link Layer** has responsibility of transferring datagram from one nod to a **physically adjacent** node over a link.

**Link Layer: Context**

Datagrams are transferred by different link protocols over different links:

* Ethernet on first link, frame relay on intermediate links, 802.11 on last link.

Each link protocol provides different services e.g. may or may not provide Reliable Data Transfer over links

Travel analogy:

* Tourist = **datagram**
* Transport segment (location) = **communication link**
* Transportation mode (train, bus, airplane) = **link layer protocol**
* Travel agent = **routing algorithm**

**Link Layer: Services**

**Frame, link access**

* Encapsulates datagram into a frame, adding header and trailer (last bytes at end of block for error checking).
* Channel access if it’s a shared medium
* MAC addresses used in frame headers to identify the SOURCE/DEST (different from IP addresses)

**Reliably delivery between adjacent nodes**

* Recall reliable delivery in previous topics
* RDT service is seldom used low bit-error link
* High error rates in wireless links.

**Flow Control**

* Pacing between adjacent sending and receiving nodes

**Error Detection**

* Errors caused by signal attenuation (reduction of signal strength during transmission) and noise.
* Receiver detects presence of errors: signals sender for retransmission or drops the frame.

**Error Correction**

* Receiver identifies and corrects bit errors without resorting to re-transmission.

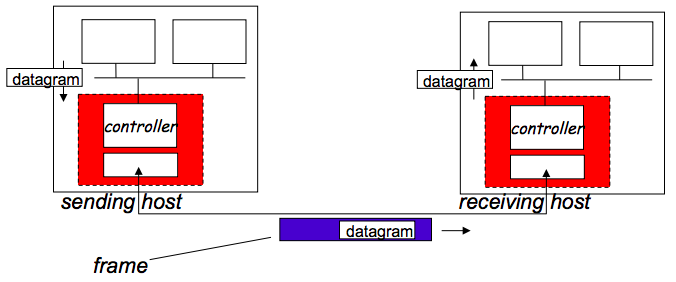
**Half-Duplex and Full-Duplex**

* Half-Duplex: nodes at both ends of link can retransmit, but not at the same time.

Where is the link layer implemented?

* In each and every host.
* Link layer is implemented in a **network interface card (adaptor)** or on a chip.
  + Ethernet card, 802.11 card, Ethernet chipset.
  + Implements the link, physical layer.
* Attaches into a host’s system bus
* Combination of hardware, software, firmware

Adaptors communicating:

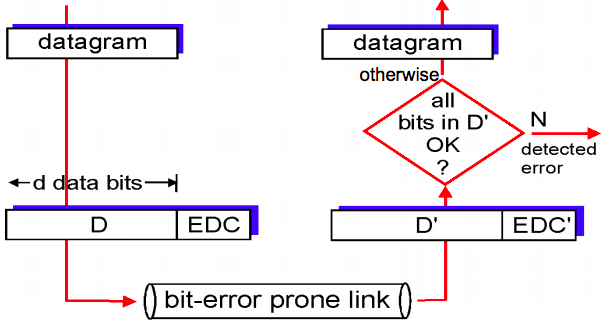


**Sending side:**

* Encapsulates datagram in frame
* Adds error checking bits, rdt, flow control etc.

**Receiving side:**

* Looks for errors, rdt, flow control
* Extracts datagram, passes to upper layer at receiving side.

**Error Correction**

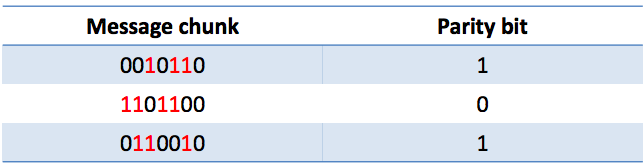
**EDC = Error Detection and Correction bits** (redundancy)  
**D = Data protected by error checking, may include error fields**

Error detection is not 100% reliable:

* Protocol may miss some errors (rare though)
* Larger EDC field yields better detection and correction

**Simple Parity – Sender**

* For every d\_bits add a parity bit:
  + Parity bit = 1 if the number of one’s are odd
  + Parity bit = 0 if the number of one’s are even
* Example (where d=7):

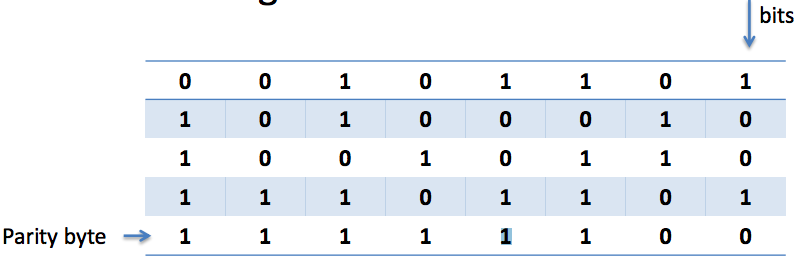
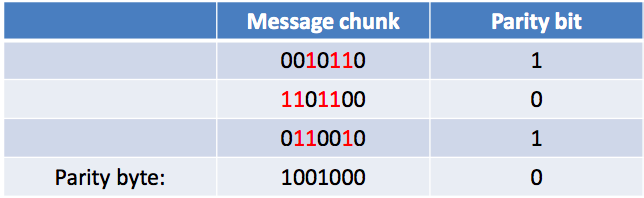
 result = 0010110**1**1101100**0**0110010**1**

**Simple Parity – Receiver**

* For each block of size d\_bits, count # of ones and compare with the following bit parity.
* If an ODD number of bits get flipped, it will be detected
* Cost: One extra bit for every d\_bits (in this example, 21 🡪 24 bits)

**Two Dimensional Parity**

* On top of Parity Bits, add an extra PARITY BYTES + compute parity on columns too.
* Can detect 1,2,3-bit (and some 4-bit) errors.



* Exactly ONE-BIT has been flipped in the example: which one is it?

In practise, bit errors occur in bursts.

We’re willing to trade computational complexity for space efficiency.

* Make the detection routine more complex, to detect error bursts, without tons of extra data.

Insight: We need hardware to interface with the network 🡪 do the computation there.

**Cyclic Redundancy Check (CRC)**

CRC is an error-detecting code to detect accidental changes to raw data.

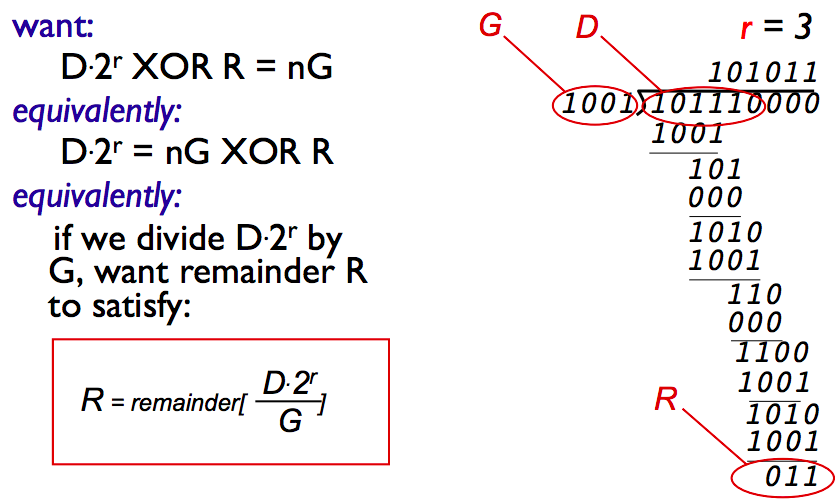
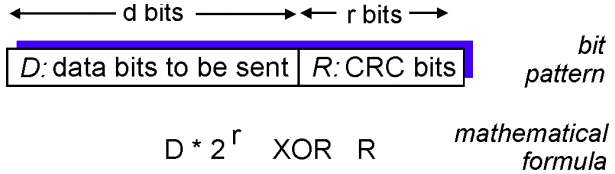
**D** = data bits, viewed as a binary number.

**G** = choose **R+1 bit pattern** (generator)

Goal: choose **R** CRC bits such that:

* **<D,R>** exactly divisible by G (modulo 2)
* Receiver knows G, **divides <D,R> by G**. **If non-zero remainder, error detected!**
* Can detect all burst errors less than R+1 bits.
* Widely used in practise (Ethernet, 802.11 WiFi, ATM)

CRC example:

 CRC Formula

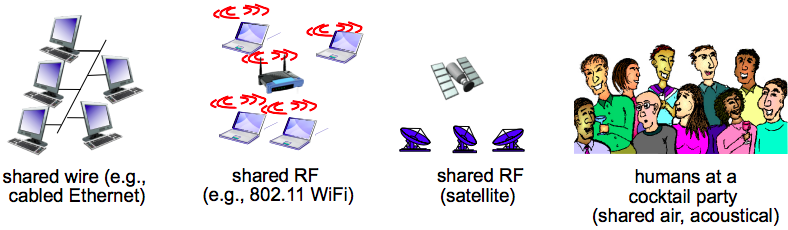
Summary on Modulo-2 arithmetic:

* Addition and subtraction are identical and both are equivalent to XOR
  + 1011 XOR 0101 = 1110
  + 1011 – 0101 = 1110
  + 1011 + 0101 = 1110
* Multiplication by 2k is essentially a left-shift by K-bits
  + 1011 x 22 = 101100 (shifted two positions to the left)

**Multiple access links / Multiple Access Protocol**

Two types of “links”

* **Point-to-point**: PPP for dial-up access | Point-to-point link between Ethernet switch, host
* **Broadcast Link (shared wire or medium)**: old-fashioned Ethernet, upstream HFC, 802.11 wireless LAN

****

Broadcast links have a Multiple Access Problem.

**Multiple Access Problem**: How to coordinate access of multiple sending / receiving nodes to a shared broadcast channel?

* With a single shared broadcast channel, two or more simultaneous transmissions by nodes can cause interference
  + **Collision** occurs if node receives two or more signals at the same time.
  + All frames involved in the collision are lost and the broadcast channel is wasted during the collision interval.

**Multiple Access Protocol** is used to determine how nodes share a channel / determine when nodes can transmit.

* Communication about channel sharing must use the channel itself, NO out-of-band channel is allowed for coordination.
* It is used by both wired and wireless local area network and satellite network.

An ideal multiple access protocol

* Given a broadcast channel of **rate R bps:**  
  1. When one node wants to transmit, it can send at rate R

2. When M nodes want to transmit, each can send at average rate R/M

3. Fully decentralised: no special node to coordinate transmissions / no synch of clocks, slots

4. Simple

**MAC Protocols classification**Medium Access Control is the lower sublayer of the data link layer, which provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network over a shared medium.

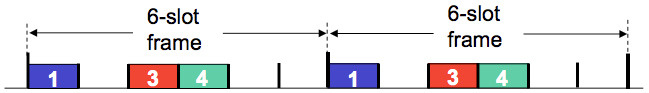
Three broad classes:

1. **Channel Partitioning**: Divide channel into smaller “pieces” (time slots, frequency, code)

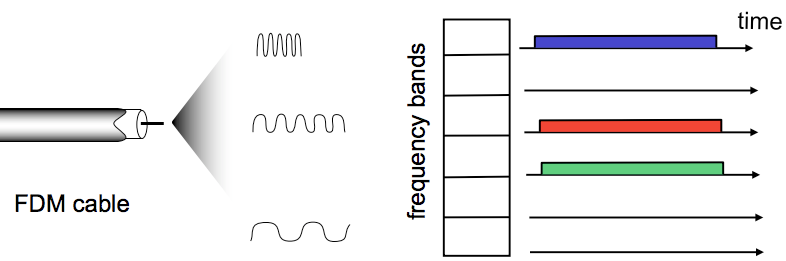
Allocate piece to node for exclusive use.

1. **Random Access**: Channel not divided, allow collisions / “recover” from collisions
2. **Taking Turns**: Nodes take turns, but nodes with more to send can take longer turns.

Channel partitioning MAC protocols: **TDMA (Time Division Multiple Access)**

* Access to channels in “rounds”
* Each station gets a fixed length slot (length = pkt transmission time) in each round
* Unused slots go idle
* Example of TDMA (6 station LAN, slots 1-3-4 have a pkt, slots 2-5-6 are idle)  
  

Channel partitioning MAC protocol: **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 of FDMA (6 station LAN, slots 1-3-4 have a pkt, slots 2-5-6 are idle):

**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, CMSA, CMSA/CD, CMSA/CA

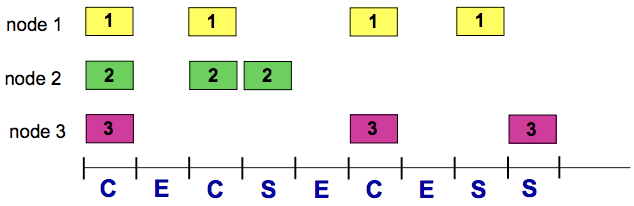
**Slotted ALOHA**

**Operation**: When a node obtains a fresh frame, it transmits in the next slot

* If no collision: the node can send the new frame in the next slot
* If collision: the node retransmits the frame in each subsequent slot with probability P until success

**Assumptions**:

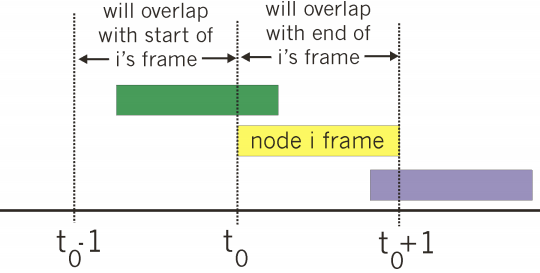
* All frames are the same size / Time is divided into equal sized slots (time to transmit 1 frame)
* Nodes start to transmit only at slot beginning.
* Nodes are synchronised
* If two or more nodes transmit in the slot, all nodes detect collision.



|  |  |
| --- | --- |
| PROS | CONS |
| * Single active node can continuously transmit at full rate of channel * Highly decentralised: only slots in nodes need to be in sync * Simple | * Collisions, wasting slots * Idle slots * Nodes maybe able to detect collisions in less than the time to transmit a packet * Clock synchronisation |

**Efficiency**: Long-run fraction of successful slots.

**At Best**: Channel used for useful transmissions **37%** of the time. **(max efficiency 1 / e = 0.37)**



**Pure Un-Slotted ALOHA**

Simpler, no synchronisation.

When the first frame arrives, transmit immediately.

**Collision probability will increase: frame sent at t0 collides  
 with other frames sent in [ t0 – 1 , t0 + 1 ]**

**Max Efficiency = 1 / (2\*e) = 0.18** or 18% of the time for useful transmissions

**CSMA (Carrier Sense Multiple Access)**

With CSMA, listen before you transmit.

* If channel senses idle, transmit the entire frame
* If channel senses busy, defer transmission (human analogy: don’t interrupt others)

**Collisions can still occur** because propagation delay means two nodes may not hear each other’s transmissions.

* Distance and Prop Delay play a role in determining collision probability.

**CSMA reduces but does NOT eliminate transmissions**.

**CSMA / CD (Collision Detection)**

Carrier Sense Multiple Access with Collision Detection is a media access control method, using a carrier-sensing scheme in which a transmission station detects collisions by sensing transmissions from other stations while transmitting a frame. When this collision condition is detected, the station stops transmitting that frame, transmits a jam signal and then waits for the random time interval before trying to resent a frame. CSMA/CD is a modification of pure CSMA, improving CSMA performance by terminating the transmission as soon as a collision is detected, shortening the time required before a retry can be attempted.

With CSMA / CD:

* Collisions are DETECTED within a short time
* Colliding transmissions are aborted, reducing channel wastage

Collision Detection

* Easy in wired LANs: measure signal strengths, compare transmitted, received signals
* Difficult in wireless LANs received signal strength overwhelmed by local transmission strength

Human analogy: the polite conversationalist

**Procedure #1 for CSMA / CD** **- Initiating a transmission:**

This procedure is used to initiate a transmission. The procedure is complete when the frame is transmitted successfully or a collision is detected during transmission.

1. Is my frame ready for transmission? If yes, go to the next point.
2. Is medium idle? If not, wait until it becomes ready
3. Start transmitting and monitor for collision during transmission
4. Did a collision occur? If so, go to collision detected procedure
5. Reset transmission counters and end frame transmission.

**Procedure #2 for CSMA / CD** **– Collision detected procedure:**

This procedure is used to resolve a detected collision. The procedure is complete when retransmission is initiated or aborted due to numerous collisions.

1. Continue transmission (with a jam signal instead of frame header/data/CRC) until minimum packet time is reached to ensure that all receivers detect the collision.
2. Increment retransmission counter
3. Was the maximum number of transmission attempts reached? If so, abort transmission.
4. Calculate and wait the random **binary (exponential back-off period** based on number of collisions  
   - After mth collision, the NIC (network interface card) chooses K at random from { 0 , 1 , 2 … , 2m – 1 }.  
   - NIC waits K \* 512 bit times, then returns to MAIN PROCEDURE #1
5. Re-enter the MAIN PROCEDURE #1

Analogy: Dinner party, where guests talk to each other through a shared medium (the air). Before speaking, each guest politely waits for the current speaker to finish. If two guests start speaking at the same time, both stop and wait for short, random periods of time (in Ethernet, this time = microseconds). The hope is that by choosing a random period of time, both guests will not choose the same time to try to speak again, thus avoiding another collision.

**FOR CSMA/CD TO WORK, WE NEED RESTRICTIONS ON MIN FRAME SIZE and MAX DISTANCE**: **BECAUSE TRANSMISSION / PROPAGATION DELAY CAN AFFECT COLLISION PROBABILITY.**

**GREAT RESOURCE FOR SUMMARY ON MAC PROTOCOLS**

**https://hewlettpackard.github.io/wireless-tools/Linux.Wireless.mac.html**

**“Taking Turns” MAC protocols**

**Channel partitioning MAC protocols**:

* Share channel *efficiently and fairly* at high load
* Inefficient at low load: delay in channel access , I/N bandwidth allocated even if only 1 active node

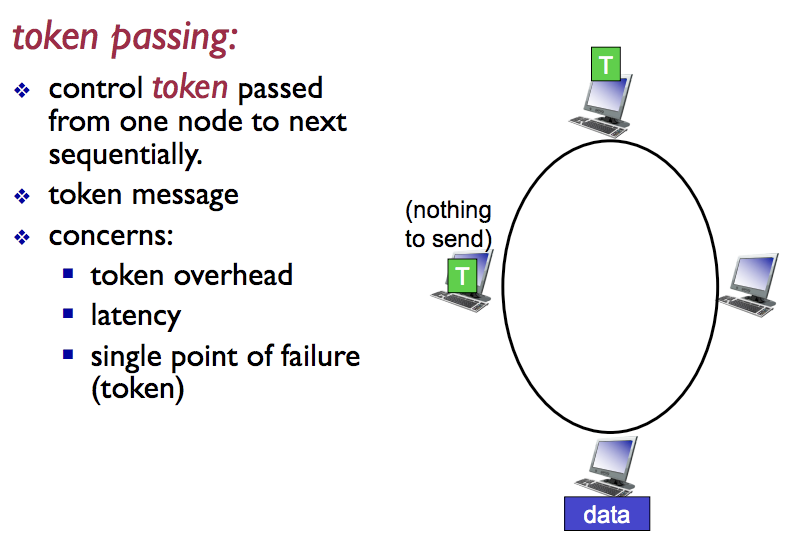
**Random access MAC protocols:**

* Efficient at low load: single node can fully utilise channel
* High load: collision overhead

Taking Turns Protocol are the best of both worlds

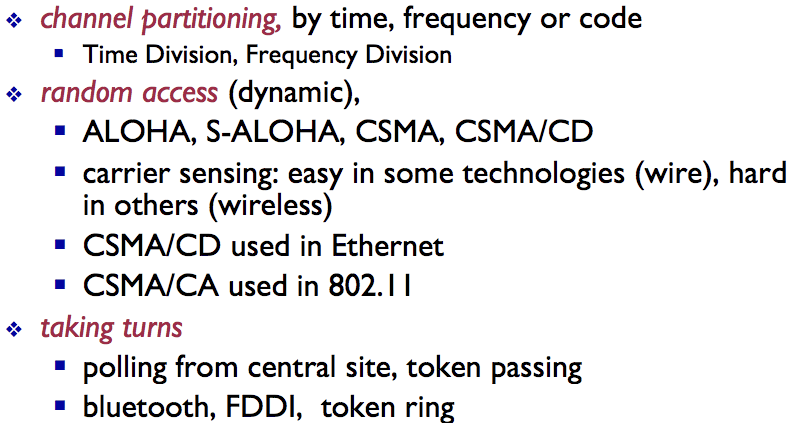
**Taking Turns Protocol: POLLING**

Polling is the third major channel access mechanism, after TDMA and CSMA/CD. The base station retains total control over the channel, but the frame content is no longer fixed, allowing variable sized packets to be sent. The base station sends a specific packet (a poll packet) to trigger the transmission by the node. The node just waits to receive a poll packet, and upon receiving it, sends what it has to transmit.



**IMPORTANT FOR EXAM: OF ALL 3 TYPES OF MAC PROTOCOLS, WHICH OF THE 4 IDEAL PROPERTIES DO EACH SATISFY?**

**Summary of MAC protocols**



**NOTE: LOOK MORE INTO** [**https://en.wikipedia.org/wiki/Carrier-sense\_multiple\_access\_with\_collision\_detection**](https://en.wikipedia.org/wiki/Carrier-sense_multiple_access_with_collision_detection) **and** [**https://en.wikipedia.org/wiki/Cyclic\_redundancy\_check**](https://en.wikipedia.org/wiki/Cyclic_redundancy_check)

**For overall brief:** [**https://hewlettpackard.github.io/wireless-tools/Linux.Wireless.mac.html**](https://hewlettpackard.github.io/wireless-tools/Linux.Wireless.mac.html)

**Not enough notes on CRC especially**