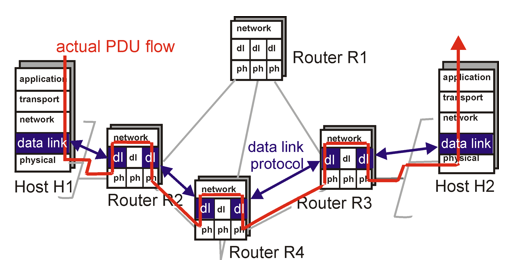
# 5. The Data Link Layer

## Link Layer Goals

* understand principles behind link layer services:
  + error detection, correction
  + sharing a broadcast channel: multiple access
  + link layer addressing
  + local area networks: Ethernet, VLANs
  + reliable data transfer, flow control
* instantiation and implementation of various link layer technologies
* resources:
  + <https://www.uotechnology.edu.iq/dep-eee/lectures/4th/Communication/Information%20theory/6.pdf>

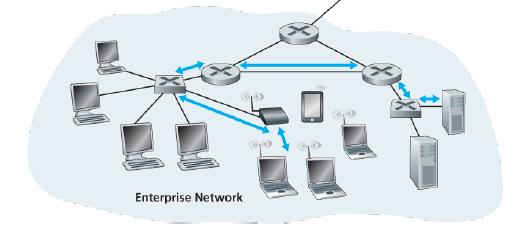
## Link Layer: setting the context

* two **physically connected** devices:
  + host-router, router-router, host-host
  + protocol data unit: **frame**



## Link Layer: introduction

* important terminology…
  + **nodes** = any device that runs a link-layer protocol; includes hosts, routers, switches, and Wi-Fi access points.
  + **links** = communication channels that connect adjacent nodes along the communication path
    - **wired links, wireless links, LANs**
* **Data-link layer** has responsibility of transferring datagram from one node to another,  
  *physically adjacent* node over a link.
* **Link layer analogy:** 
  + A tourist (or **datagram**) is travelling from Princeton, NJ, to Lausanne, Switzerland.
  + The travel agent (or **router/routing protocol**) is planning this trip for the tourist; decides that it’s best for tourist to take limo to JFK airport, then a plane to Geneva airport, and finally a train to arrive at Lausanne.
  + Notice how each transportation mode (or **link-layer protocol**) is different, but provides same service of transporting the tourist from one location to another
  + Also notice how each of the three segments (or **links**) of the trip is “direct” between two “adjacent” locations.

****  
*Six link-layer hops between wireless host and server*

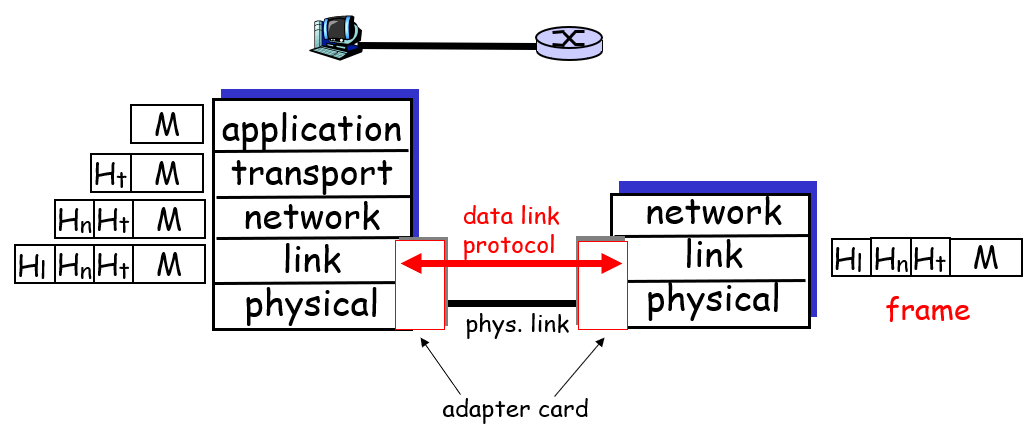
## Link Layer: Services

* Although the basic service of any link layer is to move a datagram from one node to an adjacent one over a single comm link, details of the provided service can vary. Possible services include…
  + **Framing:** 
    - Encapsulating datagram into frame before transmission over the link.
    - Frame consists of a data field, in which the network-layer datagram is inserted, and a number of header fields.
    - The structure of the frame is specified by the link-layer protocol.
  + **Link access:**
    - A **medium access control (MAC) protocol** specifies the rules by which a frame is transmitted onto the link.
    - For point-to-point links with single sender/receiver, the MAC protocol is nonexistent—sender can send a frame whenever link is idle
    - When multiple nodes access a single broadcast link, the MAC protocol serves to coordinate frame transmissions of the many nodes.
  + **Reliable delivery:**
    - Like TCP, a link-layer protocol can also provide reliable delivery through ACKs and retransmissions
    - Often required for links that are prone to high error rates, such as a wireless link, with the goal of correcting an error locally
    - Usually not used for low bit-error links, such as fiber, coax, and twisted-pair links, because of the unnecessary overhead
  + **Flow control:** 
    - pacing between sender and receivers
  + **Error detection:**
    - link-layer hardware in a receiving node can incorrectly decide that a bit in a frame is 0 when it is actually 1, and vice versa;
    - errors caused by signal attenuation, electromagnetic noise, etc.
    - receiver detects presence of errors by performing error check and signals sender for retransmission or drops frame
  + **Error correction:**
    - receiver identifies where exactly the errors in the frame are and corrects them without resorting to transmission

Source: IBM\_10:60:99 (00:09:6b:10:60:99)

## Link Layer: Implementation

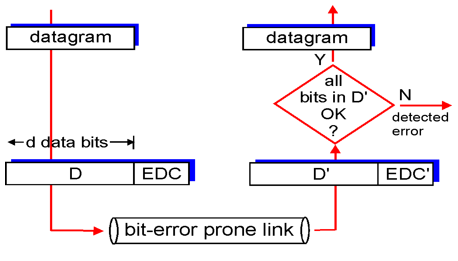
* implemented in network interface controller (NIC)
  + e.g. PCMCIA card, Ethernet adapter
* typically includes: RAM, DSP chips, host bus interface, and link interface



## Error Prevention

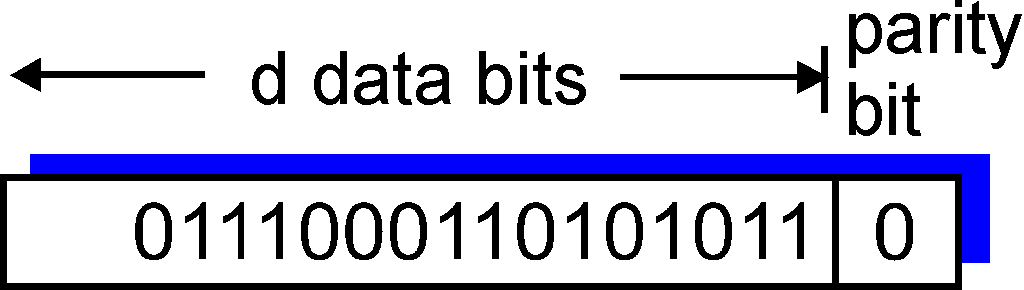
|  |  |  |
| --- | --- | --- |
| **Source of Error** | **What Causes It** | **How to Prevent or Fix** |
| **White Noise** | Movement of electrons | Increase signal strength |
| **Impulse Noise** | Sudden increases in electricity (e.g., lightning) | Shield or move the wires |
| **Cross-talk** | Multiplexer guard bands too small or wires too close together | Increase the guard bands or move or shield the wires |
| **Echo** | Poor (misaligned) connections | Fix the connections or tune equipment |
| **Attenuation** | Gradual decrease in signal over distance | Use repeaters |
| **Intermodulation noise** | Signals from several circuits combine | Move or shield the wires |

## Error Detection

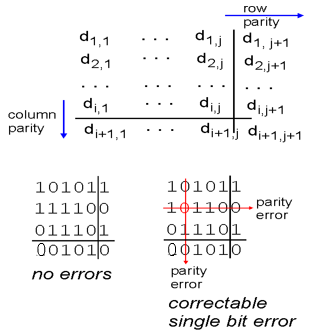
* D = Data protected by error checking, may include header fields
* EDC = Error Detection and Correction bits (redundancy means sending some extra redundant bits with the data for error detection)
* Error detection not 100% reliable
  + protocol may miss some errors, but rarely
  + larger EDC field yields better detection and correction
* Single bit error = only one bit in the data unit changed from 0 to 1 or vice-versa.
* Burst error = 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1

## Parity Checking

* A redundant bit called a **parity bit (pb)** is added to every data unit so that the total number of 1’s in the unit (including the parity bit) becomes even (or odd), as per the chosen scheme.
* **Single bit parity:** detect single bit errors

  
*This is meant to be an odd scheme, so a pb of 0 is added to preserve the odd number of 1’s*

* **Two dimensional bit parity:** detect *and correct* single bit errors

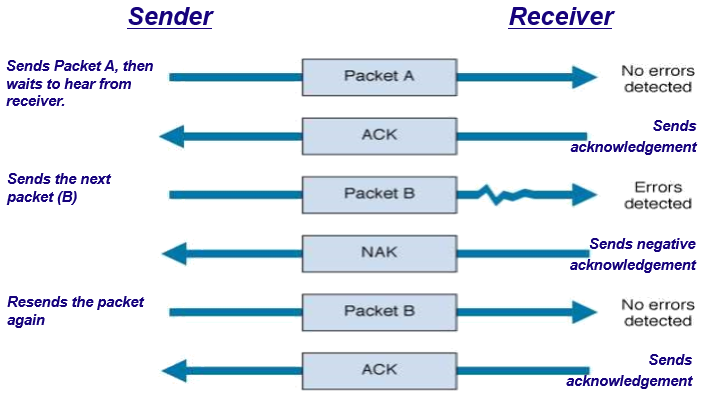
  
*This is an even scheme, so a pb of 0 is added to preserve even number of 1’s.  
But we see for that 0 in row 2, column 2, the number of 1’s is odd, so it should be 1!*

## Error Correction

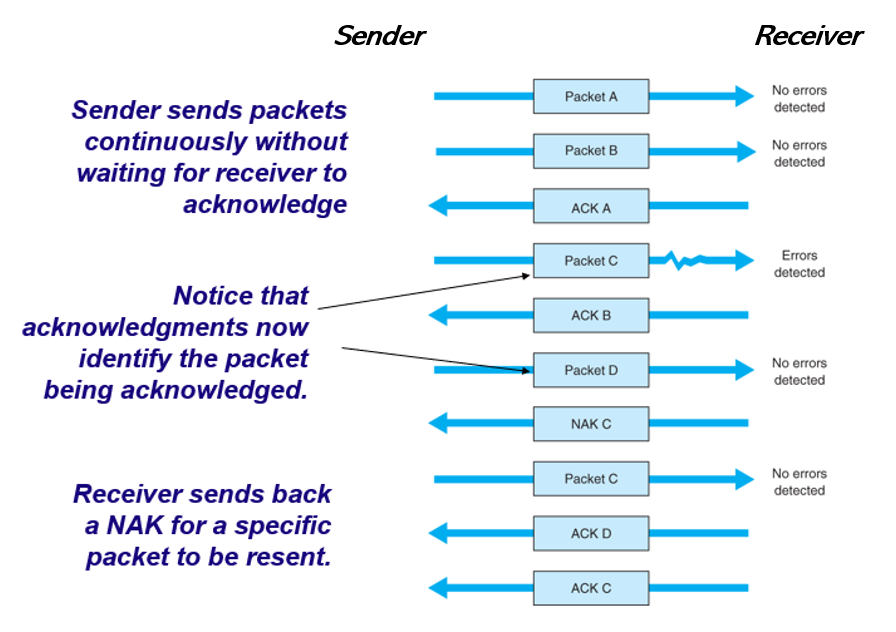
* Once detected, the error must be corrected. Error correction techniques include…
  + **Retransmission** (or, **backward error correction**)
    - Simplest, most effective, least expensive, most commonly used
    - Corrected by retransmission of the data: receiver, when detecting an error, asks the sender to retransmit the message
    - Often called **Automatic Repeat reQuest (ARQ)**
  + **Forward Error Correction (FEC)**
    - Receiving device can correct incoming messages without retransmission
    - E.g. two-dimensional parity bit checking, cyclic redundancy check, Hamming code.

## Automatic Repeat reQuest (ARQ)

* Process of requesting a data transmission be resent
* Main ARQ protocols
  + **Stop and Wait ARQ** (A half-duplex technique)
    - Sender sends a message and waits for acknowledgment, then sends the next message
    - Receiver receives the message and sends an acknowledgement, then waits for the next message

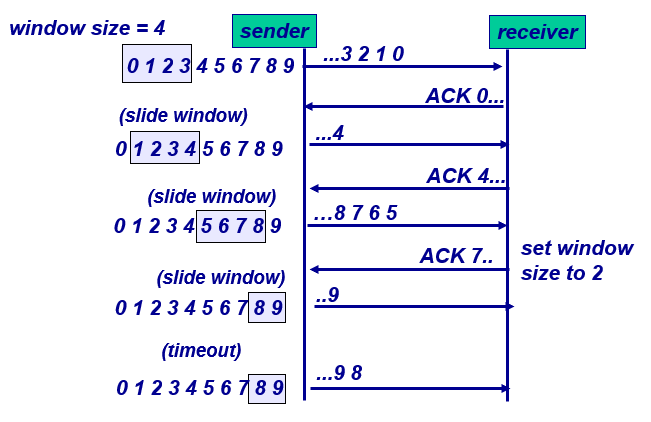


* + **Continuous ARQ** (A full duplex technique)
    - Sender continues sending packets without waiting for the receiver to acknowledge
    - Receiver continues receiving messages without acknowledging them right away



## Flow Control with ARQ

* Ensuring that sender is not transmitting too quickly for the receiver
  + Stop-and-wait ARQ
    - Receiver sends an ACK or NAK when it is ready to receive more packets
  + Continuous ARQ:
    - Both sides agree on the size of the “sliding window” :
      * Number of messages that can be handled by the receiver without causing significant delays

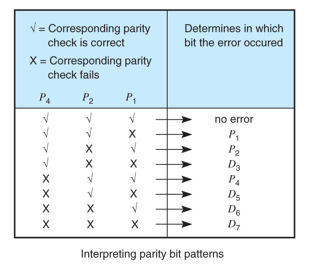
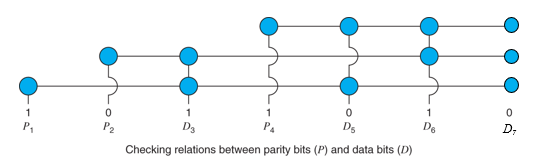


## Forward Error Correction

* Receiving device can correct incoming messages itself (without retransmission)
* Requires extra corrective information
  + Sent along with the data
  + Allows data to be checked and corrected by the receiver
  + Amount of extra information: usually 50-100% of the data
* Used in the following situations:
  + One way transmissions (retransmission not possible)
  + Transmission times are very long (satellite)
  + In this situation, relatively insignificant cost of FEC

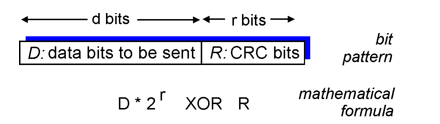
## Hamming Code

* Each data bit figures into three EVEN parity bit calculations
* If any one bit (parity or data) changes 🡪 change in data bit can be detected and corrected
* Only works for one bit errors



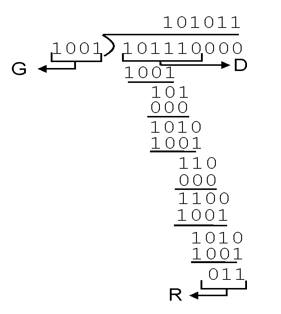
## Checksumming: Cyclic Redundancy Check

* view data bits, D, as a binary number
* choose bit pattern (generator), G
* goal: choose r CRC bits, R, 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 bits
* widely used in practice (ATM, HDCL)



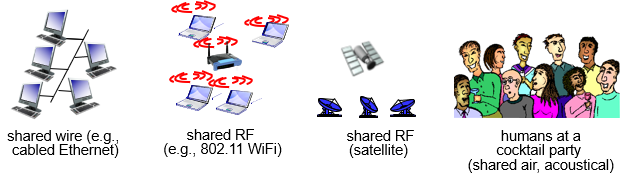
## CRC Example

* Want: D.2r XOR R = nG
* *equivalently:* D.2r = nG XOR R
* *equivalently:* if we divide D.2r by G, want reminder R
* *R =* remainder[D⋅2r/G]



## Multiple access links

* Three types of “links”:
  + **point-to-point** (single wire)
    - PPP for dial-up access, replaces SLIP
    - point-to-point link between Ethernet switch, host
  + **broadcast** (shared wire or medium)
    - classic Ethernet
    - upstream HFC (hybrid-fiber-coaxial)
    - 802.11 Wireless LAN (a.k.a. Wi-Fi)
  + **switched** (many wires, many points?)
    - switched Ethernet
    - ATM (asynchronous transfer mode), deprecated in favor of IP



## Multiple access protocols

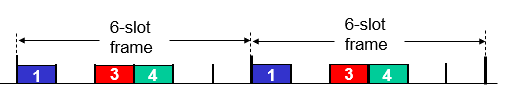
* single shared communication channel
* two or more simultaneous transmissions by nodes results in interference
  + only one node can send successfully at a time
* *multiple access protocol:*
  + distributed algorithm that determines how stations share channel, i.e., determine when station can transmit
  + communication about channel sharing must use channel itself!
  + what to look for in multiple access protocols:
    - synchronous or asynchronous
    - information needed about other stations
    - robustness (e.g., to channel errors)
    - performance

## 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
    - examples include
      * TDMA: time division multiple access
      * FDMA: frequency division multiple access
      * CDMA: code division multiple access
  + 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
    - tightly coordinate shared access to avoid collisions

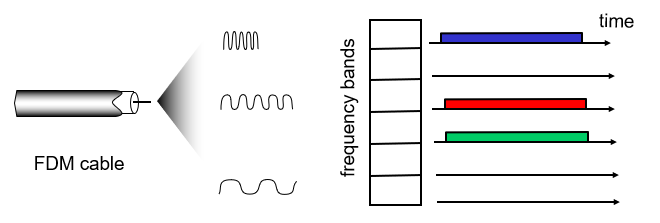
## Channel partitioning protocols: TDMA

* 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



## Channel partitioning protocols: FDMA

* 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



## Channel partitioning protocols: CDMA

* Exploits spread spectrum (DS or FH) encoding scheme
  + unique “code” assigned to each user; i.e., code set partitioning
  + used mostly in wireless broadcast channels (cellular, satellite, etc)
* All users share the same frequency, but each user has own “chipping” sequence/code
* Chipping sequence like a mask: used to encode the signal
  + encoded signal = (original signal) × (chipping sequence)
* **Decoding:** inner product of encoded signal and chipping sequence (note, the inner product is the sum of the component-by-component products)
* To make CDMA work, chipping sequences must be chosen orthogonal to each other (i.e., inner product = 0)

## Random access protocols

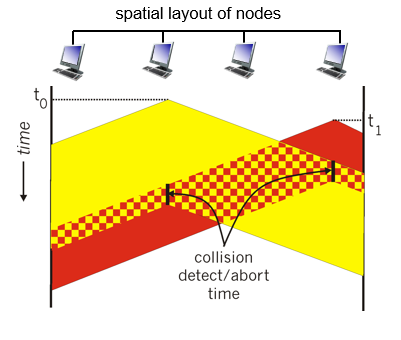
* When node has packet to send
  + transmit at full channel data rate R.
  + no prior coordination among nodes
* two or more transmitting nodes 🡪 “collision”,
* random access MAC protocol specifies:
  + how to detect collisions
  + how to recover from collisions (e.g., via delayed retransmissions)
* Examples of random access MAC protocols:
  + slotted ALOHA and ALOHA
  + CSMA, CSMA/CD, CSMA/CA

## CSMA: Carrier Sense Multiple Access

* **CSMA**: listen before transmitting!
* If channel sensed idle: transmit entire frame
* If channel sensed busy: defer transmission
  + Persistent CSMA: retry immediately with probability p when channel becomes idle (may cause instability)
  + Non-persistent CSMA: retry after random interval
* human analogy: don’t interrupt others!

## CSMA Collisions

* **collisions *can* still occur:** propagation delay means two nodes may not hear each other’s transmission
* **collision** = entire packet transmission time wasted
* **note:** distance & propagation delay play role in in determining collision probability



## CSMA/CD (collision detection)

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

## Ethernet CSMA/CD Algorithm

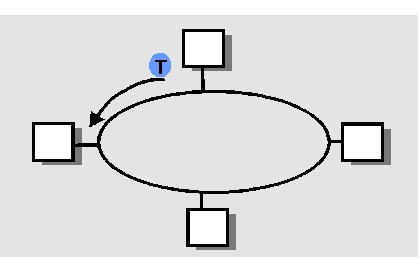
1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters *binary (exponential) backoff*:
   1. after *m*th collision, NIC chooses K at random from {0,1,2, …, 2m-1}. NIC waits K·512 bit times, returns to Step 2
   2. longer backoff interval with more collisions

## Comparing 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
  + look for best of both worlds!

## Taking Turns protocol

* Polling:
  + master node “invites” slave nodes to transmit in turn
  + typically used with “dumb” slave devices
  + Request to Send, Clear to Send msgs
  + 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)



## Summary of MAC protocols

* What do you do with a shared media?
  + **Channel Partitioning, 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)
    - CSMA/CD used in Ethernet
  + **Taking Turns**
    - polling from a central site, token passing