# COMP 790-088
## Networked and Distributed Systems

# The Data Link Layer

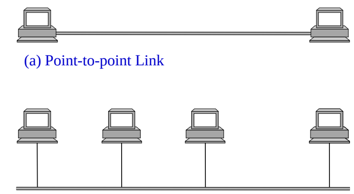
*Jasleen Kaur*

**August 31, 2009**



# The Data Link Layer
## What Issues Will We Focus On?

- Simplest way to create a network of nodes
  - » Connect the nodes directly with a physical medium
- Point-to-point links:
  - » Choice of Media
  - » Encoding
  - » Framing
  - » Error detection
  - » Error recovery
- Shared Media:
  - » Media access control



(a) Point-to-point Link

(b) Shared Media

# Error Detection
## What Causes Errors?

- Bit errors occur due to electrical interference or thermal noise
  - » Detect whether errors have occurred in the data frames received
    - ❖ Notify sender (for retransmission)
    - ❖ Or correct errors (based on error-correcting codes)
    - ❖ Or simply drop packet (to avoid wasting processing resources)
- Basic idea of error detection:
  - » Use *k* redundant bits to enable receiver to detect errors in a packet of size *n*
- Goals:
  - » **Strong error detection properties** (detect different types of errors)
  - » **Small overhead** (*k* vs. *n*)
  - » Efficient to implement





# Naïve Approach
## Packet Duplication

- Just append a duplicate copy of the frame

  **0 1 0 1 0 0 1 1   0 1 0 1 0 0 1 1**

- Error detection:
  - » Errors that corrupt same positions in both frames will go undetected
- Overhead:
  - » 100 % overhead ($k = n$)
- Efficiency?

# One-dimensional Parity
## Smarter Scheme

- Insert an extra bit to every set of 7 bits to balance the number of "1"s

```
0 1 0 1 0 0 1 | 1
```

- Error detection:
  - » Catches all 1-bit errors
  - » Can it detect 2-bit errors?
- Overhead:
  - » 14 % ($k$ = 1, $n$ = 7)





# Two-dimensional Parity
## Improving Efficiency

- Compute parity also for each bit position across each of the bytes in a frame:
  - » Add parity bit for the parity byte also

```
0 1 0 1 0 0 1 | 1
1 1 0 1 0 0 1 | 0
1 0 1 1 1 1 0 | 1
0 0 0 1 1 1 0 | 1
0 1 1 0 1 0 0 | 1
1 0 1 1 1 1 1 | 0
0 1 0 1 1 0 0 | 1
-----------------
1 0 1 0 1 1 1 | 1
-----------------
```

- Error detection:
  - » Catches all 1/2-bit errors
  - » Can it detect 3-bit errors?
  - » Can it detect 4-bit errors?
  - » Can 2-D parity be used to correct all 1-bit errors?
  - » Can it be used to correct 2-bit errors?
- Overhead:
  - » 30 % ($k$ = 15, $n$ = 49)

# Internet Checksum
## Low-cost Error Detection

- Add up all 16-bit words and send the sum
  - » Use 16-bit ones complement arithmetic

```
(-5) + (-3)
  = -(0101) + (-(0011))
  = 1010 + 1100
  = 0110 (+1 carry)
  = 0111
  = -(1000) = -8
```

- Error detection:
  - » Weak (eg: one increment, one decrement)
  - » Not used at data link layer (used by UDP, TCP, IP)
- Overhead:
  - » Only $k = 16$ redundant bits for any size $n$
- Easy to implement in software





# Cyclic Redundancy Check (CRC)
## Strongest Error Detection

- Represent $n$-bit message as $n$-1 degree polynomial
  - » e.g., MSG=10011010 as $M(x) = x^7 + x^4 + x^3 + x^1$
- Let $k$ be the degree of some divisor polynomial, $C(x)$
  - » e.g., $C(x) = x^3 + x^2 + 1$
- Transmit polynomial $P(x)$ that is evenly divisible by $C(x)$
  - » shift left $k$ bits, i.e., $M(x)x^k$
  - » subtract remainder of $M(x)x^k$ / $C(x)$ from $M(x)x^k$
- Receiver polynomial $P(x) + E(x)$
  - » $E(x) = 0$ if no errors
- Divide $(P(x) + E(x))$ by $C(x)$; remainder zero if:
  - » $E(x)$ was zero (no error), or
  - » $E(x)$ is exactly divisible by $C(x)$

# Cyclic Redundancy Check (CRC)

## Selecting *C(x)*

- Can catch:
  - All single-bit errors, as long as the $x^k$ and $x^0$ terms have non-zero coefficients.
  - All double-bit errors, as long as $C(x)$ contains a factor with at least 3 terms
  - Any odd number of errors, as long as $C(x)$ contains the factor $(x + 1)$
  - Any 'burst' error (i.e., sequence of consecutive error bits) for which the length of the burst is less than $k$ bits.
  - Most burst errors of larger than $k$ bits can also be detected

- CRC algorithm is easily implemented in hardware
  - CRC-32 commonly used by link layer protocols ($k = 32$)







# Error Detection

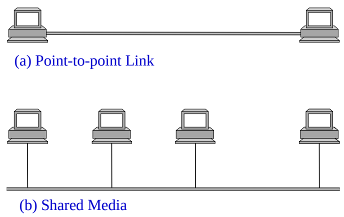
## Summary

- Goals:
  - **Strong error detection properties** (detect different types of errors)
  - **Small overhead** ($k$ vs. $n$)
  - Efficient to implement

- Ideas used:
  - Redundancy: error detection implemented at multiple layers
    - Why?
  - Software-easy implementations at higher layers
    - Checksum
  - Hardware-easy implementations at lower layers
    - CRC-32

# The Data Link Layer
## What Issues Will We Focus On?

- Simplest way to create a network of nodes
  - » Connect the nodes directly with a physical medium
- Point-to-point links:
  - » Choice of Media
  - » Encoding
  - » Framing
  - » Error detection
  - » Error recovery
- Shared Media:
  - » Media access control








# Shared Access Networks
## Interconnecting Large Number of Hosts

- One way of scaling up point-to-point links:
  - » Connect multiple hosts to the same “link”
- Media access control protocols:
  - » Arbitrate access if multiple hosts wish to transmit at the same time
- Protocol objectives (for a shared channel of capacity $R$):
  - » Efficiency:
    - ❖ When only one node has data to send, it gets throughput of $R$
  - » Fairness:
    - ❖ When $M$ nodes want to send, each gets throughput of $R/M$
  - » Support for low-latency applications
  - » Decentralized protocol (no single point of failure)
  - » Simple and inexpensive to implement

# Shared Access Networks

## Types of Media Access Protocols

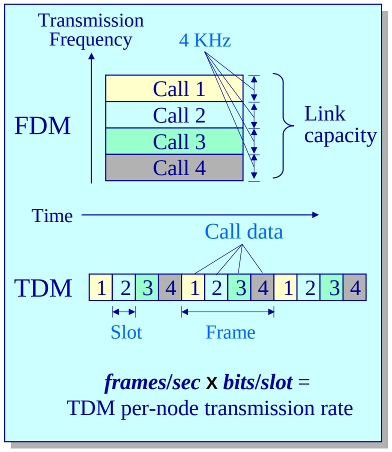
- Channel Partitioning protocols
  - » e.g.: TDM, FDM, CDMA
- Random access protocols
  - » e.g.: Ethernet, 802.11
- Taking turns protocols
  - » e.g.: Token ring



# Channel Partitioning Protocols

## Static Access Opportunity

- Channel is "statically" partitioned among nodes
  - » Transmission divided into "slots"
  - » Slots allocated to nodes
    - ❖ Frequency division multiplexing (FDM)
    - ❖ Time division multiplexing (TDM)
- Slots are "idle" if not used by owning node
  - » No sharing of slots!
  - » Fair and decentralized, but not efficient



# Random Access Protocols

## Non-static Channel Assignment

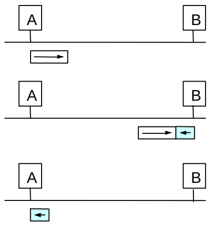
- No fixed pre-determined schedule of transmission
  - » Transmit, if data available
  - » Detect if “collision” occurs
- Typically, CSMA/CD
  - » Carrier Sense (CS): listen before transmitting
  - » Collision Detection (CD): listen while transmitting and stop on detecting collision
  - » Multiple Access (MA): Multiple nodes plugged into common “bus” medium
- Example protocols:
  - » Slotted Aloha
  - » Ethernet (multiple-access bus version)
  - » 802.11 (wireless)





# Ethernet : Media Access Protocol

## Shared Bus Version




- If line is idle
  - » Send immediately (efficiency)
  - » Upper bound message size: 1500 B (latency)
- If line is busy
  - » Wait until idle and transmit immediately (efficiency)
  - » Is this really efficient when multiple nodes waiting?
    - ❖ Aloha: wait until idle and transmit with probability *p*
- How to guarantee that collision is detected (before frame is completely transmitted)?
  - » Impose max propagation delay (51.2 $\mu$s), minimum frame size (64 B)
  - » If collision, then what?

# Ethernet : Media Access Protocol

## If Collision, Then What?

- Exponential back-off
  - » 1st collision: randomly transmit after: 0 or 51.2 $\mu$s
  - » 2nd collision: randomly transmit after: 0, 51.5, 102.4, 153.6 $\mu$s
  - » $n^{th}$ collision: transmit after $k$ * 51.2 $\mu$s,
    for randomly select $k = 0, \ldots, 2^n - 1$
    - ❖ $n$ capped at 10
  - » Give up after 16 tries

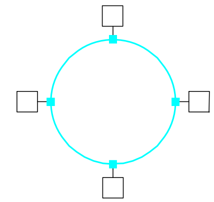
- How does randomization help here?
- Why increase back-off range with time?
- What fairness properties can you expect?
- What is the efficiency of this protocol?





# Token Ring: Media Access Protocol

## Interconnecting Large Number of Hosts

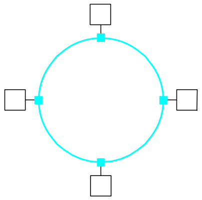


- Structure:
  - » Set of nodes connected in a ring structure
    - ❖ Data always flows in one direction
  - » Each node receives frames and forwards it
    - ❖ Destination saves a copy as frame passes by

- Basic Idea for Media Access:
  - » A special “token” frame denotes right to transmit
    - ❖ Node that currently holds the token can transmit
  - » When a nodes with data to transmit sees a token, it forwards data instead of token
    - ❖ When data frame reaches back to sender, it removes it and reinserts token

# Token Ring: Media Access Protocol

## Properties



- Basic Idea:
  - » A special "token" frame denotes right to transmit
    - ❖ Node that currently holds the token can transmit
  - » When a nodes with data to transmit sees a token, it forwards data instead of token
    - ❖ When data frame reaches back to sender, it removes it and reinserts token
- How to guarantee fairness?
- How to guarantee low latency?
- How to provide failure resistance?

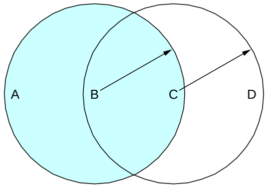




# 802.11 Media Access Protocol

## Challenges in a Wireless Setting

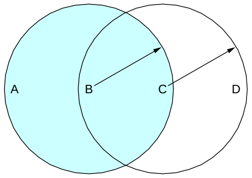
- Basic framework for Media Access:
  - » Carrier-sense: like Ethernet, listen to medium and wait if busy
  - » Collision avoidance: unlike Ethernet, no collision detection
- Complication with collision detection:
  - » Expensive
  - » Hidden node problem
    - ❖ Both A and C transmit, unaware of collision




- Complication with carrier-sense:
  - » Exposed node problem
    - ❖ If B is transmitting to A, C will refrain from transmitting to D
      - ◆ Even though it can!

# 802.11 Media Access Protocol
## MACAW

- MACAW Protocol:
  - » **M**ultiple **A**ccess With **C**ollision **A**voidance for **W**ireless LANs
  - » Basic philosophy: don't do collision detection, but *collision avoidance*
- Sender, receiver exchange control frames before a data frame
  - » Sender sends RTS (Request to Send), with length of data frame
  - » Receiver replies CTS (Clear to Send), with echo of frame length
  - » If sender gets CTS frame, it sends data frame
    - ❖ If not, does exponential back-off before trying again
  - » Receiver sends ACK after receiving data frame
- What should other nodes do?
  - » If they see a CTS frame?
  - » If they see RTS, but not CTS?
- Properties: Fairness? Efficiency? Latency?
  - » Zig-zag decoding: using collisions








# Media Access Control
## Summary

- Protocol objectives (for a shared channel of capacity *R*):
  - » Efficiency: when only one node has data to send, it gets throughput of *R*
  - » Fairness: when *M* nodes want to send, each gets throughput of *R/M*
  - » Support for low latency applications
  - » Decentralized protocol (no single point of failure)
  - » Simple and inexpensive to implement
- Ideas used:
  - » **Virtualization:** *helps guarantee fairness*
    - ❖ Partitioning (TDM/FDM), round-robin access opportunity (Token Ring)
  - » **Fine-grained multiplexing:** *helps achieve low latency*
    - ❖ Fixed slot size (TDM/FDM), Limited frame size (Ethernet, 802.11, Token Ring)
  - » **Randomization:** *helps achieve fairness in many cases*
    - ❖ Exponential back-off on collision (Ethernet, 802.11)
  - » **Signaling:** *helps achieve efficiency in multi-node settings*
    - ❖ Channel-grabbing protocol (MACAW)