Week 9 Lecture 1

Link Layer

# 6.2 Error Detection

Error Detection and Correction Bits (redundancy), data is protected by error checking. Again, this error checking is not 100% reliable. EDC is part of the link layer header. The issue with simple error checking algorithm is that it doesn’t detect errors more than one bit.

For example, if we send two copies of the same message: 101, 101, we can check one bit error like 101, 100. However, we are not able to detect error like 100, 100

# Improvements for Error Detection

## Simple Parity

We add a simple parity bit to the message, for every n bits, add a parity bit.

**Simple Parity for Sender**

**(even parity case)**

Break the message into blocks, then we add 1 if the number of one’s is odd, and 0 if the number of one’s is even. For example:

Original message: 0010110,1101100,0110010

Parity bit for each block: 1, 0, 1,

New message: 00101101,11011000,01100101

So each block has even number of “1”s

**Simple Parity for Receiver**

* For each block of size d: – Count the number of 1’s and compare with following parity bit.
* If an odd number of bits get flipped, we’ll detect it (can’t do much to correct it).

**Issue with Simple Parity**

* If the order of 1s is different but still in the same block, the parity won’t pick up the error.
* If two bits are flipped, we still can’t detect the error

## Two-dimensional Parity

Original message: 0010110,1101100,0110010

Parity bit

Message chunk Parity 0010110 1

1101100 0

0110010 1

Parity byte: 1001000 0

Can detect 1,2,3-bit errors, 4-bit flip over at the four-corner point of a rectangle not detected

Chart, scatter chart

Description automatically generated

## CRC (Cyclic Redundancy Check)

More efficient than adding extra bits: we want to make the space overhead as low as possible and make algorithm more complex. In practice, this is implemented on hardware. This only detects the error, not correcting errors.

Chart, diagram, box and whisker chart

Description automatically generated

**Modulo-2 Arithmetic**

* All calculations are modulo-2 arithmetic
* No carries or borrows in subtraction
* Addition and subtraction are identical, and both are equivalent to XOR
* Multiplication by 2k is essentially a left shift by k bits

**Sender Part**

Step 1: modulo division

Step 2: calculate the reminder

Step 3: sender sends the original bit pattern, followed by the remainder

Note that generator is always 1 bit greater than remainder

Original Bit: 101110000

Generator: 1001

Remainder: 011

Send: D + G = 101110+011

**Receiver Part**

Receiver divides the received frame and divides by G and checks if the remainder is zero.

Receiver Bits: 101110011, Remainder after modulo division: 0

A picture containing text

Description automatically generated

It can detect any continuous block of r bits.

# Multiple Access Links, Protocols

Two types of links:

* **Point to point: dial-up access, PPP**

Collision if note receives two or more signals at the same time when two or more simultaneous transmissions by nodes.

* **Broadcast: ethernet, HFC, wireless LAN**

## An ideal multiple access protocol

Requirements:

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 decentralized:
   1. no special node to coordinate transmissions
   2. no synchronization of clocks, slots
4. simple

# MAC protocols (multiple access protocols)

* channel partitioning
* random access
* “Taking turns”

Channel partitioning

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

Chart, diagram

Description automatically generated

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

Chart, diagram, bar chart

Description automatically generated

**Quiz: Does channel partitioning satisfy ideal properties,**

**2 and 4 are correct.**

* if only one node wants to transmit, it can send at rate R.
* when M nodes want to transmit, each can send at average rate R/M (fairness), only satisfy when M = N (total number of nodes in the network)
* fully decentralized: • no synchronization of clocks, slots • no special node to coordinate transmissions, TDMA needs to create slots, therefore, not satisfied, some synchronisation is required
* simple

## Random Access MAC protocols:

The protocol is able to detect collision, and how to recover from collisions (e.g., via delayed retransmission)

* slotted ALOHA
* ALOHA
* CSMA, CSMA/CD, CSMA/CA

### Slotted Aloha:

* All frames same size
* Time divided into equal size slot
* Nodes start to transmit only at the beginning of a slot
* Nodes are synchronized
* If 2 or more nodes transmit in slot, all nodes detect collision

**Operation:**

When note obtains fresh frame, transmits in next slot

* If no collision: node can send new frame in next slot
* If collision: node retransmit until successfully transmit

Timeline

Description automatically generated

**Pros**

* Very simple
* Highly decentralised, only slots in nodes need to be in sync
* Single active node can continuously transmit at full rate of channel

**Cons**

* Wasteful, the max efficiency is 1/e = 37% (need to know this number for exam)
* Idle slots
* Nodes may be able to detect collision in less than time to transmit packet
* Clock synchronisation

### Aloha (unslotted)

when frame first arrives, transmit immediately

Chart

Description automatically generated

collision probability increases: frame sent at t0 collides with other frames sent in [t0- 1,t0+1]

* max efficiency is 1/2e = 18% (need to know this number for exam)

### CSMA (carrier sense multiple access)/CD (collision detection)

**Listen before transmitting:**

* if channel sensed idle: transmit entire frame
* if channel sensed busy, defer transmission

**However, collision can still occur**

Diagram

Description automatically generated

If we are sending a very small packet, it is very hard for the receiver to detect whether it is a collision. In ethernet, the minimum size of the frame is 64 bytes. (Two 6-byte addresses, 2-byte type, 4-byte CRC, and 46 bytes of data).

If we have the CSMA/CD network length too long, then it is possible that we encounter collision during transmission

Diagram

Description automatically generated

Therefore, the maximum length = (min\_frame\_size)\*(propagation\_speed)/(2\*bandwidth) = (8\*64B)\*(2\*108mps)/(2\*107 bps) = 5120m approx./ This is for a 10Mbps bandwidth

Biggest remaining problem: difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

**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

**Quiz: Does CSMA/CD satisfy ideal properties**

**1,3,4 are correct, 2 the average rate will be less than R/M**

1. if only 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 (fairness)
3. fully decentralized:
   1. no synchronization of clocks, slots
   2. no special node to coordinate transmissions
4. Simple

### “Taking turns”

We can use two approaches to take turns

* Polling: master node “invites” slave nodes to transmit in turn
  + Issues: polling overhead, latency, single point of failure (master)
* Token Passing: control token passed from one node to next sequentially
  + token overhead, latency, single point of failure (token)

**Quiz: Does taking turns satisfy ideal properties?**

1. if only 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 (fairness)
3. fully decentralized:

* no synchronization of clocks, slots
* no special node to coordinate transmissions

1. simple

The master sends token out, therefore, not centralised. It is very simple. If only one node wants to transmit, it needs the token to be released. It has to wait even no other nodes wants to transmit. Therefore, we need to wait for the token to be released, which takes time T, therefore, the average rate is slightly less than R/M. The correct answer is 4 and 2(debatable)