COMP3203 Final Exam Summary

William Findlay

December 14, 2018

Contents

1 Units

- unit chart

prefix	base 10	base 2
pico	10^{-12}	2^{-40}
nano	10^{-9}	2^{-30}
micro	10^{-6}	2^{-20}
milli	10^{-3}	2^{-10}
—	10^{0}	2^{0}
kilo	10^{3}	2^{10}
mega	10^{6}	2^{20}
giga	10^{9}	2^{30}
tera	10^{12}	2^{40}
peta	10^{15}	2^{50}

- $Hz \implies$ cycles per second
  - $GHz \implies 10^9$ cycles per second
  - etc.

2 Formulas

2.1 Frequency

$$f = \frac{1}{T}$$

2.2 Period

$$T = \frac{1}{f}$$

2.3 Wavelength

$$\lambda = vT$$
$$\lambda = \frac{v}{f}$$

2.4 Bandwidth

$$B = vT$$

2.5 Delay

$$D = D_P + D_T + D_Q$$

2.5.1 Propagation

$$D_P = \frac{\text{distance}}{\text{speed of light}}$$

2.5.2 Transmit

$$D_T = \frac{\text{packet size}}{\text{bandwidth}}$$

2.5.3 Queue

$$D_Q = \sum_{\text{nodes}} (\text{buffering} + \text{switching})$$

2.5.4 Round Trip Time

$$RTT = 2D$$

- how long does it take a packet to go **there and back**

2.6 Overhead

$$T_O = \frac{h}{p} \qquad \text{where } h = \text{overhead bits}, p = \text{message bits}$$

- **extra** over **what we want**

3 Error Checking

- VRC
- LRC
- **CRC**
  - this guy is usually used
  - use in tandem with ARQ
- checksum

4 ARQ

- automatic repeat request
- handle errors by requesting they be resent
- use in tandem with error detection
  - **CRC**
  - checksum
- main parts
  - **ACKS**
  - **NAKS**
  - **timers**

4.1 Sliding Window

- number frames sequentially
- window of either fixed or variable size
  - see TCP section

4.1.1 Go Back N

- go back to the beginning of the window and resend everything
- $w - i = N$

4.1.2 Selective Reject

- **only** resend the **damaged frame**
- need **sorting logic**
  - frames may be out of order

4.2 Stop and Wait

- like sliding window with a **window size** $= 1$

5 Multiaccess

- problem of **shared channels**
  - who gets a turn?
  - how do we make sure things get to the right place?
- point-to-point is easy (by contrast)

5.1 LANs

- local area network
- shared channel

5.1.1 Switched LANs

- *interconnection* by *transmission*
- **complex**
  - routing tables
  - hierarchical addressing

5.1.2 Broadcast LANs

- information *received by all*
- **simple**
  - no routing
  - flat addressing scheme
- MAC (medium access control)
- **used more often**

5.2 Uncoordinated Access Control

- sucks
- $P(\text{exactly one talks}) = np(1-p)^{n-1}$

5.3 MAC Protocol

- Medium Access Control
- **dynamic**
- on demand
- must **minimize** collisions
- two classes

  - random access
  - scheduling

MAC vs Static

MAC $\implies$ dynamic, on demand

Static $\implies$ separate dedicated channels

5.3.1 Centralized

- *one* **master node**
  - makes decisions for slaves nodes
- *dependent* on **master**
  - what if it fails?
  - less efficient

5.3.2 Distributed

- all nodes **equivalent**
- make a decision together
  - *distributed* fashion

5.4 How Does MAC Work?

i) **measure** prop time
ii) **coordinate** access
iii) **select** a winner

5.4.1 Measure

- ping
- $T_{prop} = \frac{d}{v}$

5.4.2 Coordinate

```
def coordinateTwoHosts(A,B):
  A.listen(channel)

  if channel not busy:
    A.transmit(m)
    while no message from B:
      A.listen(channel)
    if time > T_prop:
      break
    else:
      A.retransmit(m)

  repeat for B
```

5.4.3 Select a Winner

- let T_A = time for a collision detected by A
- let T_B = time for a collision detected by B
- A wins $\iff T_A < T_B$

- loser is quiet until winner completes
- winner is quiet after transmission for RTT

5.5 MAC Efficiency

$$E = \frac{1}{1 + 2\frac{T_{prop}}{L}}$$

6 Ethernet

- **broadcast network**
  - every node can hear every other
- when collision occurs
  - stop sending
  - wait to retransmit

6.1 Limitations

- very large packet size as bandwidth increases
- MAC is technology dependent
  - are measurements accurate?
  - measurements may differ between hosts
- but it is **realistic**
  - uptime is important

6.2 Backoff Protocols

- **queue** of nodes **waiting to transmit**
- keep track of number of attempts
- define $P(x)$
  - probability you transmit on attempt x
  - decreasing in x

6.2.1 Implementation

- station i has bck_i
- set it to 0
- if queue not empty
  - attempt transmission with $p(bck_i)$
    - fails $\implies bck_i++$
    - succeeds $\implies bck_i := 0$
- if queue was empty, don't change bck_i

6.3 Collision-Free Protocols

6.3.1 Bitmap

- **contention period** = 8 slots
- station i inserts one bit into ith slot
- after N slots, each station knows who wants to transmit
- transmit **in order**
- a station i is out of luck if it becomes ready **just after** slot i passes

6.3.2 Tree Splitting

- nodes are leaves
- recursive
- keep taking left **subtree** until **one node** in **contention**
  - that node wins
  - take right subtree if applicable
  - walk back up to root

6.3.3 Binary Countdown

- assume **all addresses** are **same length**
- node writes its bit from **highest to lowest order**
  - if I have a 0 and somebody else has a 1
    - I drop out
  - otherwise
    - I stay in
- last man standing wins

7 Wireless

7.1 Cellular

- organized into **cells**
  - hexagons
- **neighboring** cells $\implies$ **different frequency bands**

7.2 Ad Hoc

- temporary connection
- **decentralized**
- model with a **Unit Disk Graph**
  - **points** and **circles for range**
  - $G = (V, E)$ where
    - vertices are nodes
    - edges are nodes that can each each other
    - asymmetric ranges $\implies$ directed graph

7.3 Bluetooth

7.3.1 Formation

- **master nodes**
- **slave nodes**
  - bridge nodes (a special slave)

Rules

1. **master** only next to **slaves** (and **bridges**)
2. **slaves** only next to a **master**
3. each master's **piconet** can have **max 7 slaves**
4. bridge between **TWO** masters **ONLY**