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**

2.7 Orthogonality

- take inner product of two vectors
- add them in mod 2
  - $0 \implies$ orthogonal
  - $1 \implies$ **not** orthogonal

---

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 $\textbf{window size} = 1$

---

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** $= N$ 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

7.2.1 Traversal

- **compass routing**
  - draw line $\vec{st}$
  - pick smallest angle edge sv
  - draw new line $\vec{vt}$
  - **doesn't always complete**
- **face routing**
  - draw line $\vec{st}$
  - **LHR** or **RHR**
  - pick a face which crosses the line
  - walk it until you are **about to cross**, then **flip face** to next

7.2.2 Gabriel Test

- if **A** and **B** are **in range**
- draw a **circle** with **radius AB**
  - if there is some C inside the circle, remove link AB
  - make link AC, CB instead
- **removes all edge crossings**
- edges **preserved** are called **Gabriel edges**

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**

7.3.2 Joining Two Piconets

- **roles** of master and slave can **switch**
  - done by *changing frequencies*
- a slave will act as a **bridge**

---

8 GPS

8.1 Three Techniques

8.1.1 TOA

- **time of arrival**

- compute distance from
  - three **anchor nodes**
  - use **ping time** to do this

8.1.2 TDOA

- **time difference of arrival**
- difference in arrival from **two anchors**
  - $|t_1 - t_2|$
- if **speed in medium is known**
  - time $\implies$ distance
  - $v = d/t$

8.1.3 AOA

- **angle of arrival**
- sensor node **determines directions**
  - from **two anchors**
- if **two sensors** on **same line**
  - use a **third anchor**

8.2 Satellites

- three satellites
  - (a_1, b_1, c_1)
  - (a_2, b_2, c_2)
  - (a_3, b_3, c_3)
- three unknowns
  - (x, y, z)
- solve linear system

$$
\begin{aligned}
(x-a_1)^2 + (y-b_1)^2 + (z-c_1)^2 &= r_1^2 = c^2t_1^2 \\
(x-a_2)^2 + (y-b_2)^2 + (z-c_2)^2 &= r_2^2 = c^2t_2^2 \\
(x-a_3)^2 + (y-b_3)^2 + (z-c_3)^2 &= r_3^2 = c^2t_3^2
\end{aligned}
$$

---

9 Routing

- **routing**
  - algorithm to **deliver packets**
  - two problems
    - selection
      - routing table
    - delivery
- **route discovery**
  - algorithm to **discover a route**
  - **precedes** routing

9.1 Distance Vector

- **source** distance **is 0**
  - all **other nodes** ∞
- **for all edges**

  - if distance to t can be shortened by taking the edge...
  - update to new lower value
- at ith iteration
  - all shortest paths of **length at most i edges**
- initially

$$D[i,j] = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } j \text{ is a neighbor of } i \\ \infty & \text{otherwise} \end{cases}$$

- then

$$D[i,k] = \min_{j,k \neq i} \{w[i,j] + D[j,k]\}$$

9.2 Link State Protocol

- router **responsible for neighbors**
- make an LSP packet
  - ID of creator
  - list of neighbors and cost
  - sequence number
  - TTL for the packet
- LSP **transmitted** to **all routers**
  - **flooding**
- every router has a complete map
- **updating/calculation** uses **Dijkstra's**

9.3 MST

- two standard algorithms
  - Prim
  - Kruskal

9.3.1 Kruskal

- start with nodes separated
  - keep adding smallest edge that doesn't create a cycle
  - we are done when all vertices are in the tree
- time complexity
  - # of times we change group label is at most $\log n$
  - limited by how fast we can sort the edges
    - $\boxed{(|E| \log |E|)}$

9.3.2 Prim

- **(p)**rim's = **(p)**ick a node
  - pick smallest edge from that vertex that reaches an unvisited vertex
  - add that edge, now imagine the two vertices as one meta-vertex
  - repeat until we have reached the last vertex
- time complexity
  - $\boxed{O(|E| \log |E| + |V| \log |E|)}$

9.4 Dijkstra

- add all vertices to a min-heap of $d(v)$ Q
  - initialize $d(s)$ is 0
  - initialize all other $d(v)$ to ∞
- pop s and update weight of all neighbors v of s as $d(s) + wt(s, v)$
  - keep track that you popped it
- pop the lowest and repeat the above step for the lowest
- continue until Q contains no more vertices
- time complexity
  - $\boxed{O(|E| \log |E| + |V| \log |E|)}$

---

10 IP

10.1 IPv4

10.1.1 Classes

- five classes
  - A 7N 24H
  - B 14N 16H
  - C 21N 8H
  - D multicast
  - E experiments

Class	Net ID	Host ID
A	7 bits	24 bits
B	14 bits	16 bits
C	21 bits	8 bits

Figure 1: Division of bits in class A, B, and C IP classes.

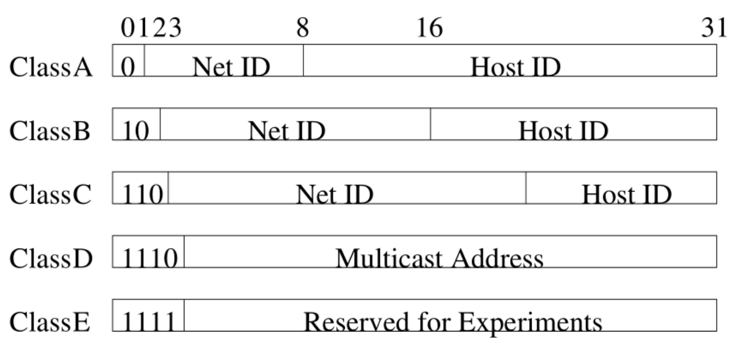


Figure 2: Breakdown of the five IP classes.

10.1.2 Subnetting

- solution when addresses can't be made "less flat"
- **only router** should be **aware**
- two levels to **three levels**

  - **net ID**
  - **subnet ID**
  - **host ID**
- single IP, **several physical networks**
  - subnets
  - near each other physically

10.1.3 Subnet Masks

- 24 `1`'s
- followed by 8 `0`'s
- bitwise **AND** of **address** and **subnet mask**
  - yields **subnet number**

10.1.4 Header

- **no options** $\implies$ **20 bytes**
- IHL is header length in *words*
  - a word is **32-bit**
- TOS
  - type of service
- TOS subfield
  - route selection
  - subnet service
  - queuing
- precedence subfield
  - degree of urgency
- options
  - security
  - timestamping
  - source routing
  - route recording

10.2 IPv6

- need **more address space**
- hexadecimal digits
  - 16 bit blocks
- `block1:block2:block3:block4:block5:block6:block7:block8`
- `block1:block2::block7:block8`
  - in this example blocks 3-6 inclusive are all 0's
- `128.33.87.51` = `::00FF:128.33.87.51`
- prefixes
  - `001` global unicast
  - `010` provider based

10.2.1 Neighbor Discovery

- each router periodically sends advertisements on subnets
- indicate
  - IP address
  - gateway functionality
  - link layer address
  - network prefixes
  - etc.

- *automatically identify* routers in the **subnet**

10.3 DHCP

- **automate** IP configuration
- **host broadcasts** *discover message* to **physical network**
- **server** responds with *offer message*
- host accepts an offer
- server sends DHCP ACK and assigns the IP
  - for a period of time T and two thresholds
  - $T_1, T_2 = T/2, 7T/8$
- T_1 is period before it asks to **renew lease**
- T_2 is is period before it starts broadcasting to **any server** on network
- T is period before it must **start from beginning**

10.4 ARP

- **address resolution protocol**
- translates IP understood locally
- allows hosts to build tables
- updates every 15 minutes

10.5 RARP

- **reverse address resolution protocol**
- obsolete
- request IP address from a computer network

---

11 TCP

11.1 How it Works (Sliding Window)

- **byte oriented**
  - sender writes bytes into
  - receiver reads from
- variable Max Segment Size
  - decides when it has enough bytes ($= MSS$)
  - or sending process requests packets
  - timer can trigger transmissions

11.1.1 Opening Connection

- A sets SYN bit and register a SEQ#
- B sets SYN bit and registers a SEQ#
  - acknowledges with A's SEQ# + 1
- A acknowledges with B's SEQ# + 1
- this is important because
  - A informs B of its starting number
  - B acknowledges and informs A of its own starting number
  - A acknowledges B's starting number
- in this way, they can anticipate what the other will do
- a timer makes sure that if an expected response is not received, they will retry

11.1.2 Closing Connection

- A sets FIN bit with SEQ#
- B responds with its own FIN bit
- A acknowledges

11.1.3 Sliding Window (Important)

- **credit allocation scheme**
- each byte transmitted has a sequence number
- sender includes SEQ# of first byte
- receiver acknowledges
  - $(A = i, W = j)$
  - all bytes up to $i - 1$ are acknowledged
  - next byte has SEQ# i
  - grant permission to send next $i, i+1, \ldots, j-1$ bytes
- keep track of a congestion window w locally
  - if $W < w$, increase it by sending out a packet
  - if $W \geq w$, wait for ACK and reduce W (and increase w)

11.2 Building Statistics

11.2.1 Average Round Trip Time

- $RTT(i)$ = round trip time for ith segment
- $ARTT(k)$ = **average** round trip time for first k segments

$$ARTT(k+1) = \frac{1}{k+1} \sum_{i=1}^{k+1} RTT(i)$$

- observe we have coefficient $\frac{1}{k+1}$ for all terms
- or, recursively

$$ARTT(k+1) = \frac{k}{k+1} ARTT(k) + \frac{1}{k+1} RTT(K+1)$$

- observe we have coefficient $\frac{k}{k+1}$ and $\frac{1}{k+1}$

11.2.2 Smoothed Round Trip Time

- $SRTT(k)$ = smoothed RTT estimate
- defined by recursion

$$SRTT(K=1) = \alpha SRTT(k) + (1-\alpha) RTT(k+1)$$

- α further from $k \implies$ less weight assigned

11.2.3 Traffic Variance

- calculate error then calculate standard deviation

$$AERR(k+1) = RTT(k+1) - ARTT(k)$$

$$ADEV(k+1) = \frac{1}{k+1} \sum_{i=1}^{k+1} |AERR(i)|$$

11.2.4 RTT Variance Estimation

- **Jacobson's**
  - works for no retransmissions
- **Exponential RTO Backoff**
  - good for when a retransmission occurs
- **Karn's** (best of both worlds)
  - do not use measured RTT for retransmitted segments
  - calculate backoff RTO when retransmission occurs
  - use backoff RTO until we get an ACK for a new segment
    - then start using Jacobson's again

11.3 Equilibrium Model

- **loss** $\implies$ decrease w
  - $w = \frac{w}{2}$
- **no loss** $\implies$ increase w
  - $w = w + \frac{1}{w}$

---

12 Sample Test

1

A system has an n-layer protocol hierarchy. Applications generate messages of length M Bytes. At each level of the layers, an h-Byte header is added.

1.1

[3 pts] What fraction of the network bandwidth is filled with headers? (Give the formula.)

$$\text{overhead} = \frac{nh}{nh + M}$$

1.2

[3 pts] Now assume $M = 20h$. What should the max number n of layers be so that the fraction in previous Question 1 does not exceed 10 % of the total?

$$\begin{aligned}
\text{overhead} &= \frac{nh}{nh + M} \\
10\% &\geq \frac{nh}{nh + 20h} \\
\frac{1}{10} &\geq \frac{n}{n + 20} \\
(n + 20)\frac{1}{10} &\geq n \\
(n + 20)\frac{1}{10} &\geq n \\
\frac{n}{10} + 2 &\geq n \\
n + 20 &\geq 10n
\end{aligned}$$

$$20 \geq 9n$$
$$n \leq \frac{20}{9}$$

1.3

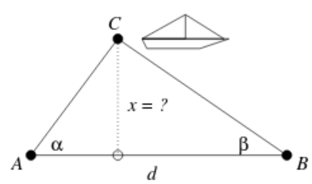
Two CDMA users are assigned the 9-bit vectors $A = 110011011, B = 100101111$, respectively. Are they orthogonal? (Prove or disprove!) **Hint:** Recall $0 \rightarrow -1$ and $1 \rightarrow +1$.

Take inner product of vectors in mod 2.

$$\begin{aligned}\langle \vec{A}, \vec{B} \rangle \mod 2 &= 1 + 0 + 0 + 0 + 0 + 1 + 0 + 1 + 1 \mod 2 \\ &= 0 && \iff \text{orthogonal}\end{aligned}$$

2

You are observing a ship from two base stations A, B. Assume that at this time of observation $\alpha = \pi/3, \beta = \pi/4$ and $d = 1000\ m$.

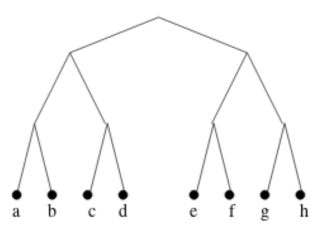


Derive a formula for the unknown distance x (You are not required to evaluate the trigonometric functions of $\pi/3$ and $\pi/4$).

$$x = d\frac{\tan\alpha\tan\beta}{\tan\alpha + \tan\beta}$$
$$x = 1000\,\mathrm{m}\frac{\tan\frac{\pi}{3}\tan\frac{\pi}{4}}{\tan\frac{\pi}{3} + \tan\frac{\pi}{4}}$$

3

Ethernet stations a, b, c, d, e, f, g, h contend for a channel. Assume a, e, f, g, h become ready at once and that they use the tree resolution protocol to resolve contentions.



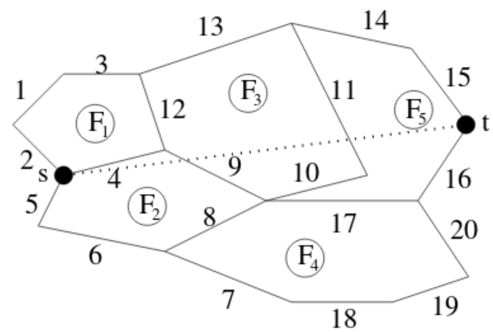
for each contention slot give in the table below the winning stations.

Slot	Station
1	a e f g h
2	a
3	e f g h
4	e f

Slot	Station
5	e
6	f
7	g h
8	g
9	h

4

The links and faces of a planar wireless network are labeled as depicted in the Figure below. Moreover there is a source node s and a destination node t.



4.1

Apply the face routing algorithm with the left-hand rule (on a face) to give a path from s to t. In the table below name the face and the edges of that face being traversed. **Your answers must list all the links traversed and the paths formed must arise from the corresponding routing algorithm!**

Face	List of Edges Being Traversed
F_2	4
F_3	12,13
F_5	14, 15

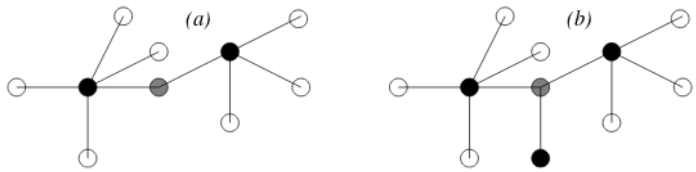
4.2

Apply the compass routing algorithm to give a path from s to t.

4, 9, 10, 11, 14, 15

5

In the networks below empty (gray, black) bullets are pure slaves, bridges, masters, respectively. According to Bluetooth formation rules, which of the two networks are bluetooth networks, which are not and why?



5.1 A

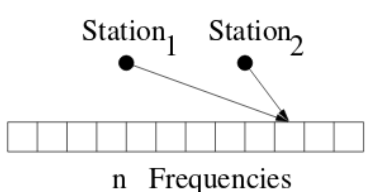
- **valid**
  - all piconets have slavecount ≤ 7
  - all piconets have slaves
  - no adjacent masters
  - no adjacent slaves
  - bridge connects **two** piconets by their master nodes

5.2 B

- **invalid**
- the good
  - all piconets have slavecount ≤ 7
  - no adjacent masters
  - no adjacent slaves
- the bad
  - bridge connects **three** piconets by their master nodes
  - should be **TWO**
  - one piconet has **no slaves**

6

There are $n \geq 2$ possible frequencies and 2 synchronous wireless stations. Each station is using frequency-hoping to select at random (with probability $1/n$) one of these frequencies. What is the probability that the stations select the same frequency? (Give explanation of your answer.)



Define event A is both stations select same frequency.

$$\begin{aligned} P(A) &= n\frac{1}{n}\frac{1}{n} \\ &= \frac{1}{n} \end{aligned}$$

7

n sensors all having range equal to 1, form a unit line graph arranged on a line such that the ith sensor has x-coordinate equal to x_i, for $i = 1, 2, \ldots, n$. Further, assume $x_i = i + (-1)^i$, for all $i = 1, 2, \ldots, n$.

7.1

Give the values x_1, x_2, x_3.

$$\begin{aligned} x_1 &= 1 + (-1)^1 = 1 - 1 = 0 \\ x_2 &= 2 + (-1)^2 = 2 + 1 = 3 \\ x_3 &= 3 + (-1)^3 = 3 - 1 = 2 \end{aligned}$$

7.2

[2 pts] Is the unit line graph a connected graph? Give a precise explanation of your answer.

In order for the graph to be connected, we need some node with an x-coordinate of 1 at a bare minimum.

Is it possible to have such an x-coordinate?

$$\begin{aligned} 1 &= i + (-1)^i \\ i &= 0 \end{aligned}$$

Our $i \in \{1, 2, \ldots, n\} \implies i \neq 0$. Therefore it is not a connected graph.