How do we share a channel?

• Ideas:
  – Division via time, frequency and codes
  – Statistical multiplexing
  – Randomized access protocols
    • Aloha
    • CSMA variants
    • Classic Ethernet
    • Wireless
  – Contention-free protocols
    • Token ring
    • DQDB
Frequency Division Multiple Access

- Simultaneous transmission in different frequency bands
  - Analog: Radio/TV, AMPS cell phones (800MHz)
  - Also called Wavelength DMA (WDMA) for fiber

  ![Diagram showing frequency bands and guard bands]

  "Speaking at different pitches"
Time Division Multiple Access

- Timeslice given frequency band between users
  - Digital: used extensively inside the telephone network
  - T1 (1.5Mbps) is 24 x 8 bits/125us; also E1 (2Mbps, 32 slots)

  "Speaking at different times"

- Advantage: lower delay; Disadvantage: synchronization
Code Division Multiple Access

- Give each user a different code (right)
  - Send +ve or –ve code for 1/0
  - All users send at once
  - Uses bandwidth for N users
    - “chip rate” >> data rate
  - Mixes time and frequency

- Codes are orthogonal to each other
  - Can correlate for one code
    - This will ignore the rest

- Widely used for 3G mobile phones

Four “4 chip” orthogonal codes
Statistical Multiplexing

- Static partitioning schemes are not suited to data communications where peak rate >> average rate.

- If we share on demand we can support more users
  - Based on the statistics of their transmissions
  - Occasionally we might be oversubscribed
  - This is called statistical multiplexing

- Statistical multiplexing is heavily used in data networks
  - But not to capture (self-similar) packet bursts at small timescales!
Example

- One user sends at 1 Mbps and is idle 90% of the time.
  - 10 Mbps channel; 10 users if statically allocated

- What are the likely loads if we share on demand?

![Graph](attachment:attachment.png)
Example continued

• For 10 users, \( \text{Prob(need 10 Mbps)} = 10^{-10} \) Not likely!
• For 40 users, \( \text{Prob(>10 active users)} = 0.15\% \), which is low

• We can support 4X users!
• But: important caveats …
ALOHA

- Wireless links between the Hawaiian islands in the 70s
- Want distributed allocation
  - no special channels, or single point of failure

- Aloha protocol:
  - Just send when you have data!
  - There will be some collisions of course …
  - Detect errored frames and retransmit a random time later

- Simple, decentralized and works well for low load
  - For many users, analytic traffic model, max efficiency is 18%
Carrier Sense Multiple Access

• We can do better by listening before we send (CSMA)
  – good defense against collisions only if “a” is small (LANs)

• “a” parameter: number of packets that fit on the wire
  – $a = \text{bandwidth} \times \text{delay} / \text{packet size}$
  – Small ($<<1$) for LANs, large ($>>1$) for satellites
What if the Channel is Busy?

• 1-persistent CSMA
  – Wait until idle then go for it
  – Blocked senders can queue up and collide

• non-persistent CSMA
  – Wait a random time and try again
  – Less greedy when loaded, but larger delay

• p-persistent CSMA
  – If idle send with prob p until done; assumed slotted time
  – Choose p so p * # senders < 1; avoids collisions at cost of delay
CSMA with Collision Detection

- Even with CSMA there can still be collisions. Why?

- For wired media we can detect all collisions and abort (CSMA/CD):
  - Requires a minimum frame size ("acquiring the medium")
  - B must continue sending ("jam") until A detects collision
Classic Ethernet

- IEEE 802.3 standard wired LAN (1-persistent CSMA/CD)
- Classic Ethernet: 10 Mbps over coaxial cable
  - baseband signals, Manchester encoding, preamble, 32 bit CRC
- BUT: Newer versions are much faster
  - Gigabit Ethernet (1 Gbps)
- Modern equipment isn’t one long wire
  - We cover switches later
# Ethernet Frames

<table>
<thead>
<tr>
<th>Preamble (8)</th>
<th>Source (6)</th>
<th>Dest (6)</th>
<th>Len (2)</th>
<th>Payload (var)</th>
<th>Pad (var)</th>
<th>CRC (4)</th>
</tr>
</thead>
</table>

- Min frame 64 bytes, max 1500 bytes
- Max length 2.5km, max between stations 500m (repeaters)
- Addresses unique per adaptor; globally assigned
- Broadcast media
Binary Exponential Backoff

- Build on 1-persistent CSMA/CD
- On collision: jam and exponential backoff
  - Jamming: send 48 bit sequence to ensure collision detection
- Backoff:
  - First collision: wait 0 or 1 frame times at random and retry
  - Second time: wait 0, 1, 2, or 3 frame times
  - Nth time (N<=10): wait 0, 1, ..., $2^N$-1 times
  - Max wait 1023 frames, give up after 16 attempts
  - Scheme balances average wait with load
Classic Ethernet Performance

• Much better than Aloha or CSMA!
  – Works very well in practice

• Source of protocol inefficiency: collisions
  – More efficient to send larger frames
    • Acquire the medium and send lots of data
  – Less efficient as the network grows in terms of frames
    • recall “a” = delay / (frame size * transmission rate)
    • “a” grows as the path gets longer (satellite)
    • “a” grows as the bit rates increase (Fast, Gigabit Ethernet)
Wireless Multiple Access

Wireless is more complicated than wired …

1. Cannot detect collisions
   - Transmitter swamps co-located receiver
2. Different transmitters have different coverage areas
   - Asymmetries lead to hidden/exposed terminal problems
Hidden Terminals

- A and C can both send to B but can’t hear each other
  - A is a hidden terminal for C and vice versa
- CSMA will be ineffective – want to sense at receiver
Exposed Terminals

- B, C can hear each other but can safely send to A, D

- Compare to spatial phones:

  reuse in cell
CSMA with Collision Avoidance

• Since we can’t detect collisions, we avoid them
  – CSMA/CA as opposed to CSMA/CD
  – Not greedy like Ethernet

• When medium busy, choose random backoff interval
  – Wait for that many idle timeslots to pass before sending
  – Remember p-persistence … a refinement

• When a collision is inferred, retransmit with binary exponential backoff (like Ethernet)
  – Use CRC and ACK from receiver to infer “no collision”
  – Again, exponential backoff helps us adapt “p” as needed
Aside: RTS / CTS for hidden terminals

1. B stimulates C with Request To Send (RTS)
2. A hears RTS and defers to allow the CTS
3. C replies to B with Clear To Send (CTS)
4. D hears CTS and defers to allow the data
5. B sends to C
802.11 Wireless LANs

- Dominant standard with a many PHY/MAC options/features
- Wireless plus wired infrastructure
- Avoids collisions with CSMA/CA; RTS/CTS largely unused
- Much PHY processing for high-rate physical links
CSMA Story

- Aloha
- Classic Ethernet
- 802.11
- Cable modems
- RFID
  - Explore this in Homework 2
Contestation-free Protocols

• Collisions are the main difficulty with random schemes
  – To improve efficiency/scalability, many schemes grant ongoing bandwidth and use random schemes for request traffic

• Q: Can we avoid collisions altogether?
• A: Yes. By taking turns or with reservations
  – Token Ring / FDDI, DQDB

• More generally, what else might we want?
  – Deterministic service, priorities/QOS, reliability
Token Ring (802.5)

- Token rotates permission to send around node
- Sender injects packet into ring and removes later
  - Maximum token holding time (THT) bounds access time
  - Early or delayed token release
  - Round robin service, acknowledgments and priorities
- Monitor nodes ensure health of ring
FDDI (Fiber Distributed Data Interface)

- Roughly a large, fast token ring
  - 100 Mbps and 200km vs 4/16 Mbps and local
  - Dual counter-rotating rings for redundancy
  - Complex token holding policies for voice etc. traffic

- Token ring advantages
  - No contention, bounded access delay
  - Support fair, reserved, priority access

- Disadvantages
  - Complexity, reliability, scalability
DQDB (Distributed Queue Dual Bus)

- Two unidirectional buses that carry fixed size cells
  - Cells are marked busy/free and can signal a request too
- Nodes maintain a distributed FIFO queue
  - By sending requests they are reserving future access
DQDB Algorithm

• Two counters per direction (UP, DN)
  – RC (request count), CD (countdown)

• Consider sending downstream (DN):
  – Always have RC count UP requests, minus free DN cells if larger than zero
  – This is a measure of how many others are waiting to send
  – To send, copy RC to CD, decrement CD for each free DN cell, send when zero
  – This waits for earlier requests to be satisfied before sending

• Highly scalable, efficient, but not perfectly fair