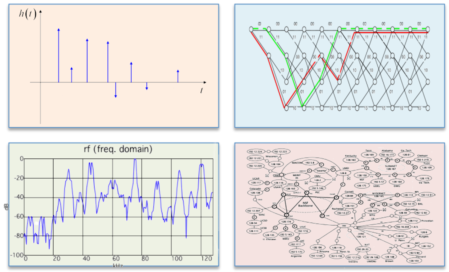


INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Spring 2011
Lecture #13

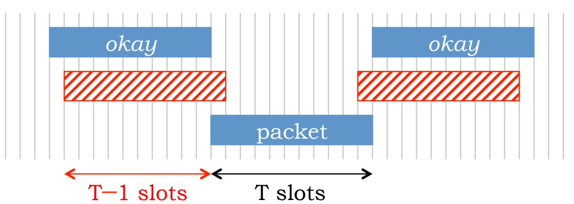
- unslotted Aloha
- carrier sense, contention windows
- code division glimpse

Slotted Aloha Summary

- Assumptions
  - Time is divided into slots
  - Transmissions begin on slot boundaries
  - Packets are 1 slot long
- Choose to transmit packet with probability p
  - Each node uses collisions/success to adjust p
  - Adjust p between p_{min} (avoid starvation) and p_{max} (avoid capture)
  - On collision $p \leftarrow max(p_{min}, p/2)$, on success $p \leftarrow min(p_{max}, 2*p)$
- Utilization = throughput achieved/maximum data rate
  - U = prob(exactly one transmission in a slot)
    = $N*p*(1-p)^{N-1}$, where N is number of backlogged nodes
  - Maximized when $p = 1/N$, $U_{max} = (1 - 1/N)^{N-1}$
  - As $N \rightarrow \infty$, $U_{max} \rightarrow 1/e \approx 37\%$
  - While each node's p is never exactly $1/N$, the goal is to have its average value over modest intervals be approximately $1/N$.

Unslotted Aloha

- Packets take T time slots to transmit
  - As slots get smaller and T grows, approximates transmission at arbitrary times.
- Collisions are no longer "perfect"
  - Any overlap between multi-slot packets is a collision
  - Larger window of vulnerability to other transmissions



Any other packet transmitted in these 2T−1 time slots will collide with target packet

Utilization in Unslotted Aloha

Probability of no transmission for 2T−1 slots:

$$(1-p)^{2T-1}$$

Probability of a sender experiencing no collisions:

$$\geq \left[p(1-p)^{2T-2}\right]\left[(1-p)^{2T-1}\right]^{N-1} = p(1-p)^{(2T-1)N-1}$$

$\geq$ for nodes that try to send new packet while busy with last one!

Utilization = throughput/maximum rate:

$$U_{\text{unslotted Aloha}} \geq \frac{Np(1-p)^{(2T-1)N-1}}{1/T} = TNp(1-p)^{(2T-1)N-1}$$

U_{max} for Unslotted Aloha

Maximization with respect to p:

$$\log(\ldots) = \text{const} + \log(p) + \left[(2T-1)N-1\right]\log(1-p)$$

Derivative:

$$\frac{1}{p} + \frac{(2T-1)N-1}{1-p}, \text{ which equals 0 at } p = \frac{1}{(2T-1)N}$$

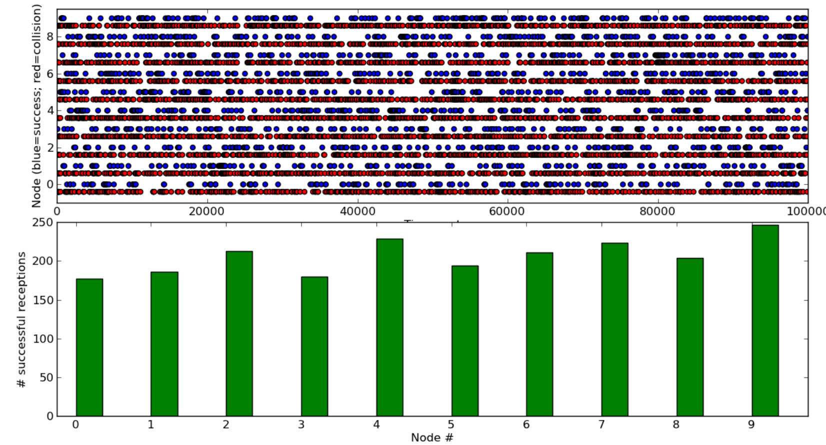
Plugging back into U:

$$U_{max} = \frac{T}{2T-1}\left(1 - \frac{1}{(2T-1)N}\right)^{(2T-1)N-1}$$

For large N: $U_{max} \approx \left(\frac{T}{2T-1}\right)\frac{1}{e}$ For large N, T: $U_{max} \approx \frac{1}{2e}$

Half the utilization of slotted Aloha; makes sense: twice the window of vulnerability

Simulation of Unslotted Aloha



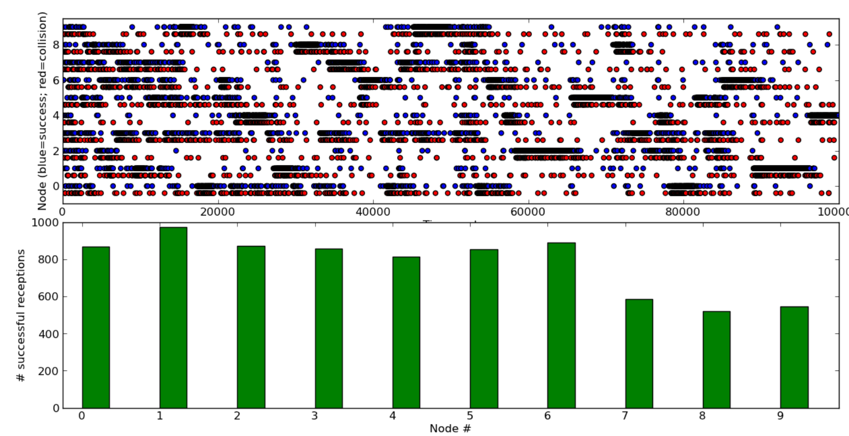
Utilization = .21, Fairness = .99

```
python PS6_3.py -r -n 10 -t 100000 --pmin=.005 --pmax=0.8 -s 10
```


Carrier Sense

- Reduce collisions with on-going transmissions by transmitting only if channel appears not to be busy.
- For large T (slots/packet) if channel is busy this cycle, the same sender will probably be transmitting more of their packet next cycle
- When the channel is idle, there's no chance of interrupting an on-going transmission.
- That leaves the possibility of colliding with another transmission that starts at the same time – a one slot window of vulnerability, not 2T-1 slots.
- Expect collisions to drop dramatically, utilization to be quite a bit better, although a "wasted" slot is now necessary
- Busy = detect energy on channel. On wireless channels, transmitters turn on carrier to transmit (we'll learn more about this after break), hence the term "carrier sense".

Simulation of Carrier Sense



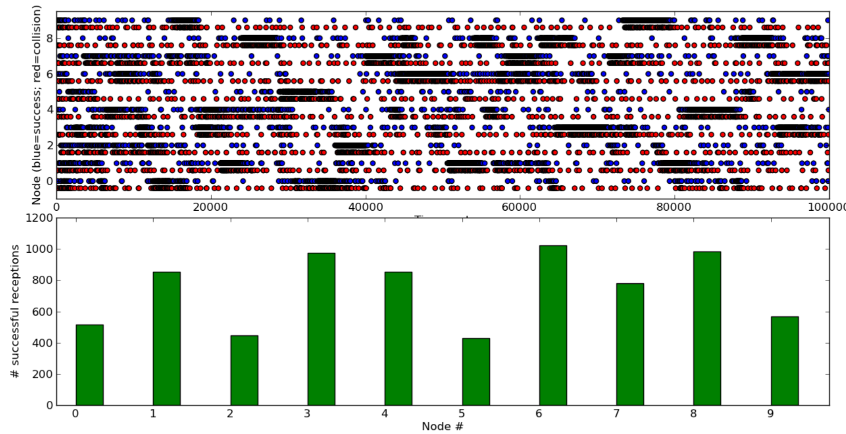
Utilization = .78, Fairness = .96

```
python PS6_4.py -r -n 10 -t 100000 --pmin=.005 --pmax=0.8 -s 10
```

Contention Windows

- Contention Window: parameter is some integer *CW*
- When node wants to transmit, it picks a random number *r* uniformly in [1,*CW*] and sends after the r^{th} idle slot from the current time.
- If transmission succeeds: $CW \leftarrow max(CW_{min}, CW/2)$
  If transmission collides: $CW \leftarrow min(CW_{max}, CW*2)$
- Node is guaranteed to attempt a transmission within CW slots. With the earlier scheme, there was always the chance (though exponentially decreasing) that a node may not transmit within some fixed number of time slots.

Simulation of Contention Windows



Utilization = .74, Fairness = .92

```
python PS6_5.py -r -n 10 -t 100000 -W 256 -s 10
```


Summary of MAC Protocols

- Goal of MAC protocols is to maximize utilization and fairness
- TDMA is a good choice when nodes are all (or mostly) backlogged
  - Round-robin sharing provides known communication capacity and bounded wait
  - It's precisely fair, 100% utilization if all nodes have packets
  - Poor choice when traffic is bursty or if some nodes have a higher offered load than others
- Contention protocols dynamically adapt to changing traffic
  - Distributed protocol (each node makes its own decisions based on transmission experience) avoids cost of centralized controller having to know which nodes have packets to send
  - Parameter (p, CW) that controls when packets are sent is adjusted so that prob(sending packet) is lowered when collisions are detected and raised when transmissions are successful.

Summary (cont'd.)

- Slotted Aloha – based on very simple rule: transmit with probability p.
  - Dynamic adjustment of p "stabilizes" the protocol.
    - Use binary exponential backoff to adjust p downward
  - Utilization maximized when p = 1/(number of backlogged nodes)
  - For large numbers of backlogged nodes $U \approx 1/e \approx 37\%$
  - For fairness: $p_{min} \le p \le p_{max}$
- Unslotted Aloha – packets take multiple time slots to send, models transmissions at arbitrary times
  - Gets half of the max utilization of slotted Aloha due to doubled window of vulnerability to collisions
  - Carrier sense avoids collisions from packets once transmission has started → much better utilization
  - Fairness still requires bounds on p

Code Division Multiple Access (CDMA)

- Two vectors are orthogonal if their dot products are 0. Here's a set of 4 mutually orthogonal vectors:
  - ```
    V1: (1,  1,  1,  1)
    V2: (1,  1, -1, -1)
    V3: (1, -1,  1, -1)
    V4: (1, -1, -1,  1)
    ```
- Assign each transmitter a particular orthogonal vector (Vi) it will use to encode its transmissions (called the "chip code"). With vectors shown above we can support 4 transmitters.
  - If message bit is 0, transmit –Vi
  - If message bit is 1, transmit Vi

  } 1 message bit → len(Vi) "chips"
- Channel will sum the transmitted values:
  - ```
    send 00 using V1: -1 -1 -1 -1 -1 -1 -1 -1
    send 01 using V2: -1 -1  1  1  1  1 -1 -1
    send 11 using V3:  1 -1  1 -1  1 -1  1 -1
    send 10 using V4:  1 -1 -1  1 -1  1  1 -1
    channel:           0 -4  0  0  0  0  0 -4
    ```


CDMA Receiver

- At receiver take groups of len(V) bits and form dot product with Vi for desired channel.
  - If result is negative, message bit is 0
  - If result is positive, message bit is 1

```
channel:            0 -4  0  0  0  0  0 -4

receive using V1:   1  1  1  1  1  1  1  1
dot product:                -4          -4
message bits:                0           0

receive using V2:   1  1 -1 -1  1  1 -1 -1
dot product:                -4           4
message bits:                0           1

receive using V4:   1 -1 -1  1  1 -1 -1  1
dot product:                 4          -4
message bits:                1           0
```


Asynchronous CDMA

- Use N orthogonal vectors to multiplex N transmitters (e.g., use a NxN Walsh/Hadamard Matrix)
- Scheme described above works for synchronous CDMA when all symbols are transmitted starting at same moment. For example this works fine for a cell tower transmitting to mobile phones.
- But hard to synchronize mobile phone transmissions, so use asynchronous CDMA:
  - Can't create transmissions that are truly orthogonal if they start at different times
  - Approximate orthogonality with longer uncorrelated pseudo-random sequences (called pseudo-noise or PN). "pseudo" implies that sequence can be reconstructed at receiver given a known starting point.
  - Assuming equal signal strengths from each transmitter at receiver, if we decode bits using a particular PN sequence synchronized with desired transmitter, we'll get desired signal plus some uncorrelated noise from other transmitters.