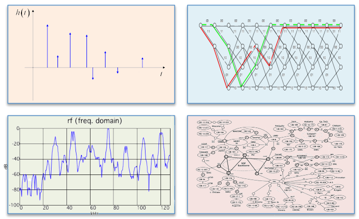


INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Spring 2012
Lecture #18 (More MAC)

- Stabilizing Aloha
- Carrier sense multiple access (CSMA)
- Contention windows (instead of xmit probability)

Simulation of Slotted Aloha (N=10)

Utilization = .38, Fairness = .98

```
python PS7_fixedaloha.py -r -n 10 -p .1 -t 1000
```

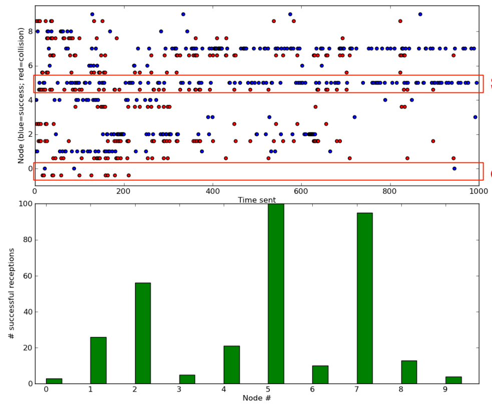

Stabilization: Selecting the Right p

- Setting $p = 1/N$ maximizes utilization, where N is the number of *backlogged* nodes.
- With bursty traffic or nodes with unequal offered loads (aka *skewed loads*), the number of backlogged is constantly varying.
- Issue: how to dynamically adjust p to achieve maximum utilization?
  - Detect collisions by listening, or by missing acknowledgement
  - Each node maintains its own estimate of p
  - If collision detected, too much traffic, so decrease local p
  - If success, maybe more traffic possible, so increase local p
- "Stabilization" is, in general, the process of ensuring that a system is operating at, or near, a desired operating point.
  - Stabilizing Aloha: finding a p that maximizes utilization as loading changes.

Binary Exponential Backoff

- Decreasing p on collision
  - Estimate of N (# of backlogged nodes) too low, p too high
  - To quickly find correct value use multiplicative decrease: $p \leftarrow p/2$
  - k collisions in a row: p decreased by factor of 2^{-k}
  - Binary: 2, exponential: k, back-off: smaller p → more time between tries
- Increasing p on success
  - While we were waiting to send, other nodes may have emptied their queues, reducing their offered load.
  - If increase is too small, slots may go idle
  - Try multiplicative increase: $p \leftarrow min(2*p,1)$
  - Or maybe just: $p \leftarrow 1$ to ensure no slots go idle

Simulation of Stabilized Aloha



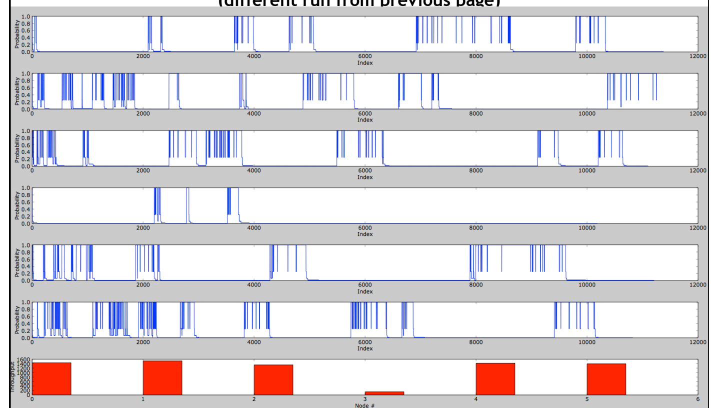
Utilization = .33, Fairness = .47

```
python PS7_stabaloha.py -r -n 10 -t 1000
```

6.02 Spring 2012 — Lecture 18, Slide #5

Node probabilities over time

(different run from previous page)

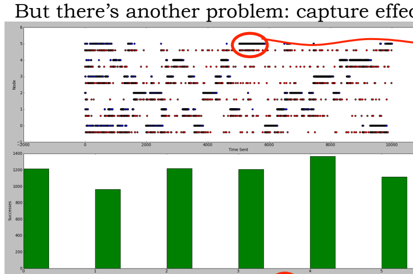


Y-axis is per-node transmission probability
Bottom panel: per-node throughput

6.02 Spring 2012 — Lecture 18, Slide #6

What Went Wrong?

- Starvation
  - Too many successive failures → p very small → no xmit attempts
  - Result: significant long-term unfairness
  - Try a reduction rule with a lower bound: $p \leftarrow max(p_{min}, p/2)$
  - Choosing $p_{min} << 1/max(N)$ works best
- But there's another problem: capture effect



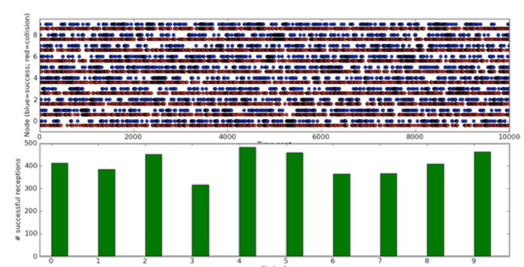
Utilization = .71 Fairness = .99

```
python PS7_stabaloha.py -r -n 6 -t 1000 --pmin=.05
```

6.02 Spring 2012 — Lecture 18, Slide #7

Limiting the Capture Effect

- Capture effect
  - A successful node maintains a high p (avg. near 1)
  - Starves out other nodes for short periods
  - Try an increase rule with an upper bound: $p \leftarrow min(p_{max}, 2*p)$



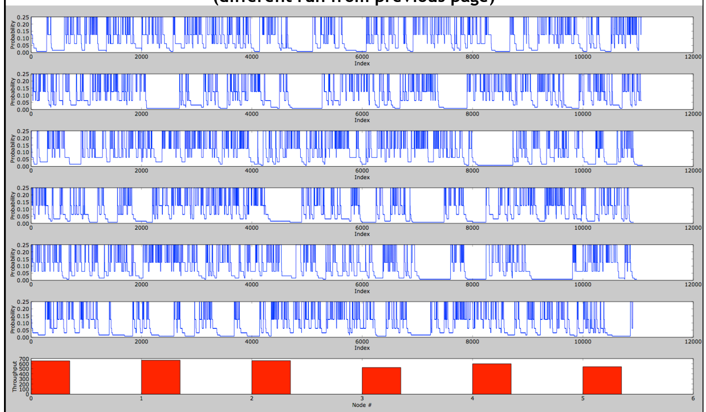
Utilization = .41, Fairness = .99

```
python PS7_stabaloha.py -r -n 10 -t 10000 --pmin=.05 --pmax=0.8
```

6.02 Spring 2012 — Lecture 18, Slide #8

Node Probabilities: pmax=.25, pmin=.01

(different run from previous page)



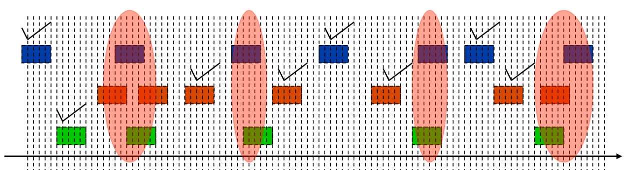
Y-axis is per-node transmission probability
Bottom panel: per-node throughput

Spot Quiz

1. In Aloha, each node maintains a variable, p. What does p represent? [P(x) is "probability of event x"]
   - A. P(node being backlogged)
   - B. P(backlogged node sends in a timeslot)
   - C. P(packet transmission is received correctly)
   - D. P(time slot is kept idle)
2. In ***stabilized*** Aloha, the value of p never goes below pmin. Why should pmin not be too small?
   - A. To increase the utilization
   - B. To avoid extreme unfairness
   - C. To reduce the problems caused by the "capture effect"
   - D. To reduce the number of collisions
3. Slotted Aloha, packet size 1 slot, N backlogged nodes, each node has a ***fixed*** p. Calculate: P(***collision*** in a timeslot)

   Soln: P(collision) = P(≥2 xmits) = 1 – P(no xmit) – P(1 xmit) =
   P(no xmit) = $(1-p)^N$; P(1 xmit) = $Np(1-p)^{(N-1)}$
   Therefore, P(collision) = $1 - (1-p)^N - Np(1-p)^{(N-1)}$.

"Unslotted" Aloha in Pictures: Collisions



Packet is 5 slots long in this picture

- A *collision* occurs when multiple transmissions overlap in time (even partially)
- Throughput = *Uncollided* packets per second
- Utilization = Throughput / Channel Rate

Utilization in Unslotted Aloha

Probability of no transmission for 2T−1 slots:

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

Probability of a sender experiencing no collisions:

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

= 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:

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

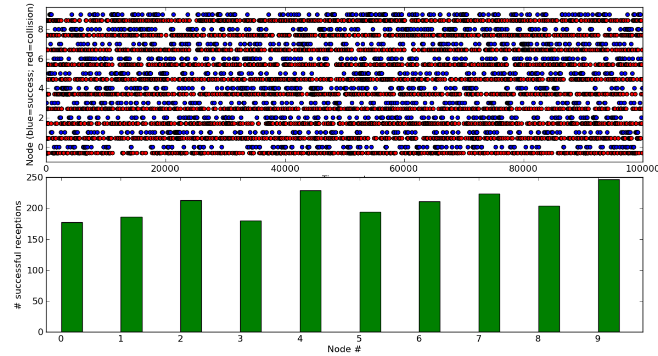
Plugging back into U:

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

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



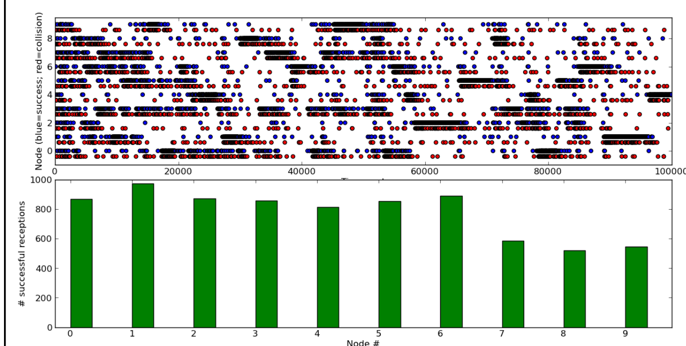
Utilization = .21, Fairness = .99

```
python PS7_stabaloha.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 slot, the same sender will probably be transmitting more of their packet next slot
- 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
- Busy = detect energy on channel. On many channels, transmitters turn on carrier to transmit, hence the term "carrier sense".

Simulation of Carrier Sense



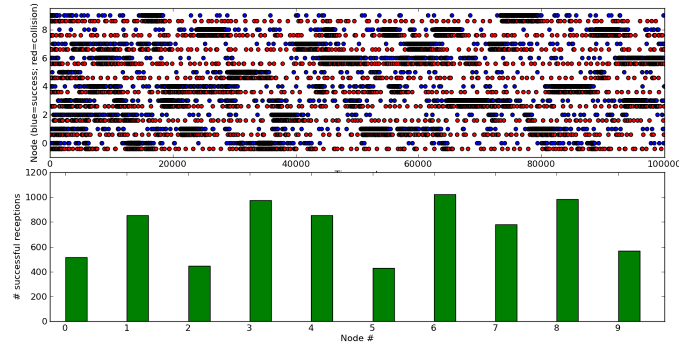
Utilization = .78, Fairness = .96

```
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} <u>idle</u> 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 PS7_cw.py -r -n 10 -t 100000 -W 256 -s 10
```


Summary of MAC Protocols

- Main goals: high utilization and fairness
- TDMA: good when nodes are mostly backlogged
  - Round-robin sharing provides equal rates & bounded wait
  - 100% utilization & fairness = 1.0 if all nodes backlogged
  - Poor choice when traffic is bursty or load is skewed
  - Hard to implement in a fully distributed way (easier with "master", like a base station or access point)
- Contention protocols dynamically adapt to traffic
  - Distributed: avoids central controller having to know which nodes have packets to send
  - Fixed Aloha: max util is 1/e (37%) when N is large
  - Stabilized Aloha: Parameter (*p* or *CW*) that controls when packets are sent is adjusted so that Prob(sending packet) is lowered when collisions are detected and raised when xmissions succeed (multiplicative decrease of *p*)
  - Carrier sense improves throughput (with stabilization)