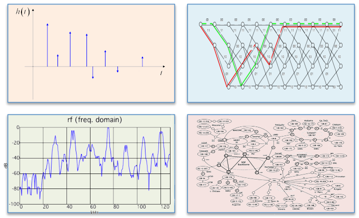


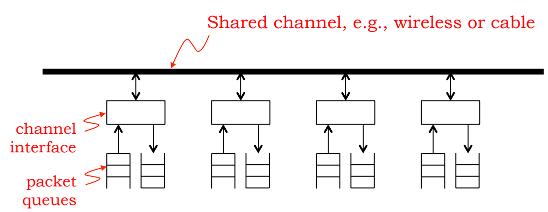
INTRODUCTION TO EECS II
DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2011
Lecture #17

- Shared medium → Media access (MAC) protocol
- Time division multiple access (TDMA)
- Contention protocols: Aloha

Shared Media Networks

Shared Communications Channels



- Basic idea in its simplest form: avoid collisions between transmitters
- Wanted: a communications protocol ("rules of engagement") that ensures "good performance"
- Nodes may all hear each other perfectly, or not at all, or partially (more on this later, in the "Abstraction" slide)

"Good Performance": What are the Metrics?

- High utilization
  - Channel capacity is a limited resource → use it efficiently
  - Ideal: use 100% of channel capacity in transmitting packets
  - Waste: idle periods, collisions, protocol overhead
- Fairness
  - Divide capacity equally among requesters
  - But not every node is requesting all the time…
- Bounded wait
  - An upper bound on the wait before successful transmission
  - Important for isochronous communications (e.g., voice/video)
- Dynamism and scalability
  - Accommodate changing number of nodes, ideally without changing implementation of any given node
- Not all protocols do well on all these metrics

Channel Sharing Protocols

- Protocol ≡ "rules of engagement" for good performance
  - Known as media access control (MAC) or multiple access control
- Time division
  - Share time "slots" between requesters
  - Prearranged: time division multiple access (TDMA)
  - Not prearranged: contention protocols (e.g., Alohanet). These are interesting because each node operates independently
- Frequency division
  - Give each transmitter its own frequency, receivers choose "station"
  - Cf. lab for PSet 7 – use different carrier frequencies & recv filters
- Code division
  - Uses unique orthogonal pseudorandom code for each transmitter
  - Channel adds transmissions to create combined signal
  - Receiver listens to one "dimension" of combined signal using dot product of code with combined signal
  - Not covered in 6.02

Utilization

- Utilization measures the throughput of a channel:

$$U_{channel} = \frac{\text{total throughput over all nodes}}{\text{maximum data rate of channel}}$$

- Example: 10 Mbps channel, four nodes get throughputs of 1, 2, 2 and 3 Mbps. So utilization is (1+2+2+3)/10 = 0.8.
- $0 \le U \le 1$. Utilization can be less than 1 if
  - The nodes have packets to transmit (nodes with packets in their transmit queues are termed *backlogged*), but the protocol is inefficient.
  - There is insufficient *offered load*, i.e., there aren't enough packets to transmit to use the full capacity of the channel.
- With backlogged nodes, perfect utilization is easy: just let one node transmit all the time! But that wouldn't be fair…

Fairness

- Many plausible definitions. A standard recipe:
  - Measure throughput of nodes = x_i, over a given time interval
  - Say that a distribution with lower standard deviation is "fairer" than a distribution with higher standard deviation.
  - Given number of nodes, N, fairness F is defined as

$$F = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \sum_{i=1}^{N} x_i^2}$$

- $1/N \le F \le 1$, where $F=1/N$ implies single node gets all the throughput and $F=1$ implies perfect fairness.
- We'll see that there is often a tradeoff between fairness and utilization, i.e., fairness mechanisms often impose some overhead, reducing utilization

Abstraction for Shared Medium

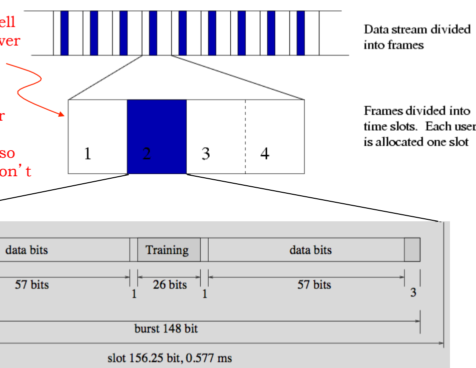
- Time is divided into *slots* of equal length
- Each node can start transmission of a packet only at the beginning of a time slot
- All packets are of the same size and hence take the same amount of time to transmit, equal to some integral multiple of time slots.
- If the transmissions of two or more nodes overlap, they are said to *collide* and none of the packets are received correctly. Note that even if the collision involves only part of the packet, the entire packet is assumed to be lost.
- Transmitting nodes can detect collisions, which usually means they'll retransmit that packet at some later time.
- Each node has a queue of packets awaiting transmission. A node with a non-empty queue is said to be *backlogged*.
- Depending on context, nodes may hear each other perfectly (eg, Ethernet), or not at all (e.g., satellite ground stations), or partially (e.g., WiFi devices or cell phones). For now, assume all nodes want to send packets to a fixed "master" (eg, base station)

Time Division Multiple Access (TDMA)

- Suppose that there is a centralized resource allocator and a way to ensure time synchronization between the nodes – for example, a cellular base station.
- For *N* nodes, give each node a unique index in the range [0,N-1]. Assume each slot is numbered starting at 0.
- Node *i* gets to transmit in time slot *t* if, and only if, *t mod N* = *i*. So a particular node transmits once every *N* time slots.
- No packet collisions! But unused time slots are "wasted", lowering utilization. Poor when nodes send data in bursts or have different offered loads.

TDMA for GSM Phones

First slot is used in cell phones to contact tower for slot assignment. Tower can determine appropriate timing advance for each user (accounts for varying distance from tower) so that transmissions won't overlap at the tower.

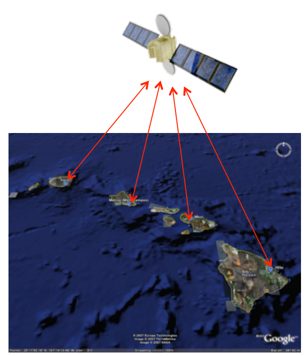


http://en.wikipedia.org/wiki/Time_division_multiple_access

Contention Procotols: Aloha (Simplest Example)

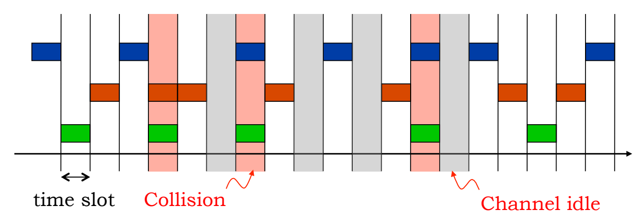
To improve performance when there are burst data patterns or skewed loads, use a contention protocol where allocation is not pre-determined.

Alohanet, designed by Norm Abramson et al. (Hawaii), was a satellite-based data network connecting computers on the Hawaiian islands. One frequency was used to send data to the satellite, which rebroadcast it on a different frequency to be received by all stations.



Stations could only "hear" the satellite, so had to decide independently when it was their turn to transmit.

Success, Idleness, Collisions



- Throughput = *Uncollided* packets per time interval
- Utilization of the above pic = Throughput / Channel Rate = 12/20 = .6

Slotted Aloha

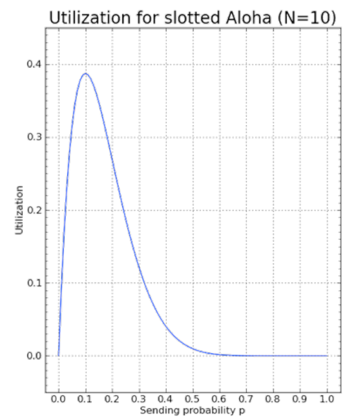
- Aloha protocol followed by each of N nodes:

  if a node is backlogged, it sends a packet in the next time slot with probability p.

- Assume (for now) each packet takes exactly one time slot to transmit (*slotted* Aloha)
- Utilization when all nodes backlogged? The probability that exactly one node sends a packet.
  - prob(send a packet) = p
  - prob(don't send a packet) = $1-p$
  - prob(only one sender) = $p(1-p)^{N-1}$
  - There are $(N \text{ choose } 1) = N$ ways to choose the one sender

$$U_{\text{slotted Aloha}} = Np(1-p)^{N-1}$$

Maximizing Utilization



To determine maximum: set $dU/dp = 0$, solve for p.

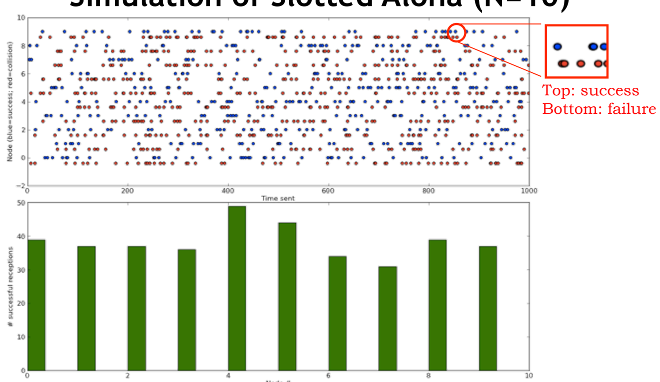
Result: $p = 1/N$, so

$$U_{\max} = (1 - \frac{1}{N})^{N-1}$$

As $N \to \infty$, $U_{\max} \to \frac{1}{e} \approx 37\%$

$$\ln\left((1-\frac{1}{N})^{N-1}\right) = (N-1)\ln(1-\frac{1}{N})$$
$$= (N-1)(-\frac{1}{N} - \frac{1}{2N^2} - \frac{1}{3N^3} - \ldots)$$
$$= -1 + \frac{1}{N} + \frac{1}{6N^2} + \frac{1}{12N^3} + \ldots$$
$$= -1 \text{ as } N \to \infty$$

Simulation of Slotted Aloha (N=10)



Utilization = .38, Fairness = .98

```
python PS6_2.py -r -n 10 -p .1 -t 1000
```

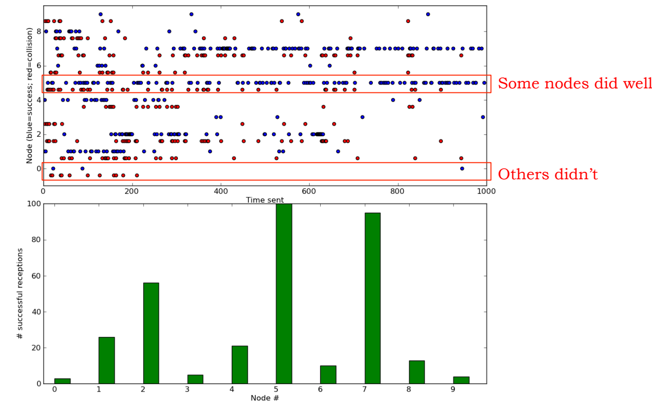

Stabilization: Selecting the Right p

- 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

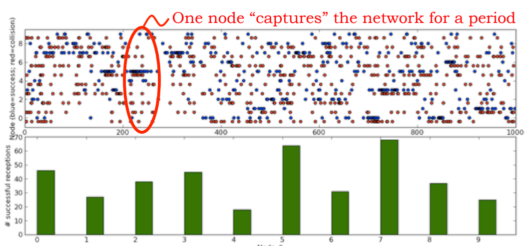


Utilization = .33, Fairness = .47

`python PS8_3.py -r -n 10 -t 1000`

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} << $1/max(N)$ seems to work best

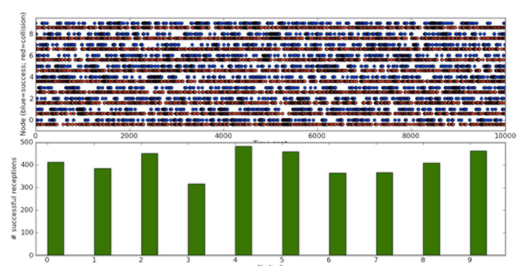


Utilization = .4, Fairness = .87

`python PS8_3.py -r -n 10 -t 1000 --pmin=.05`

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 PS8_3.py -r -n 10 -t 10000 --pmin=.05 --pmax=0.8`