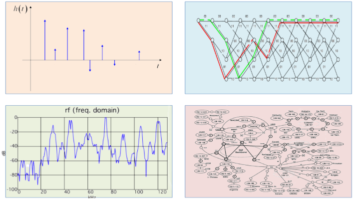
MAC Protocols and Packet Switching

6.02 Fall 2013 Lecture 19



INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

Today's Plan

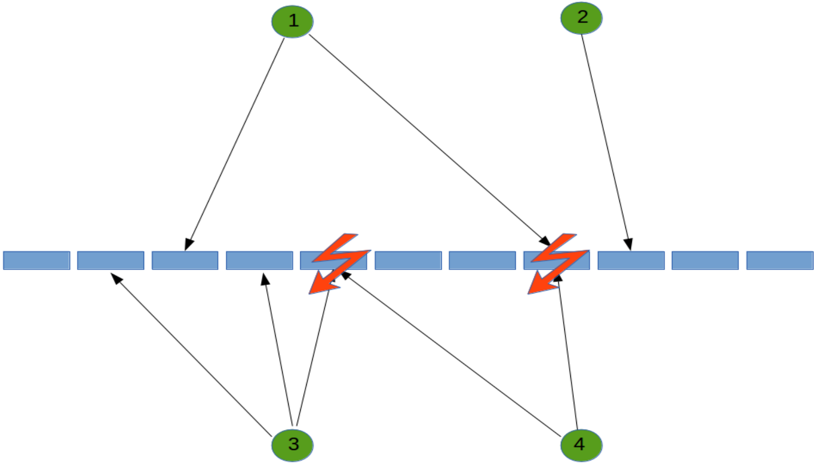
MAC Protocols:

- Randomized Access (Aloha)
- Stabilization Algorithms

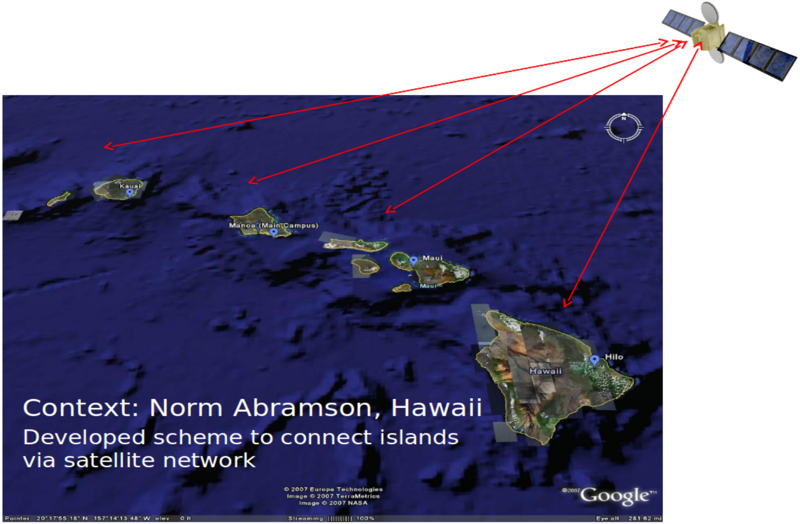
Packet Switching:

- Multi-Hop Networks
- Delays, Queues, and the Little's Law

Competition Between Nodes



The Aloha Protocol



Aloha Context

- Alohanet 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.

Contention Procotols: Aloha (Simplest Example)

- assume it takes one time slot to send one data packet
- each backlogged node sends a packet with probability p

Decentralization via Randomization!

When pn is not large (where n is the number of backlogged nodes), we hope that the probability of successful transmission in a given time slot will be large enough.

Aloha (Simplest Example): Statistical Analysis

- $\mathbf{P}(\text{given node success}) = p(1-p)^{n-1}$
- $\mathbf{P}(\text{success}) = np(1-p)^{n-1}$
- utilization: $U = \mathbf{P}(\text{success}) = np(1-p)^{n-1}$

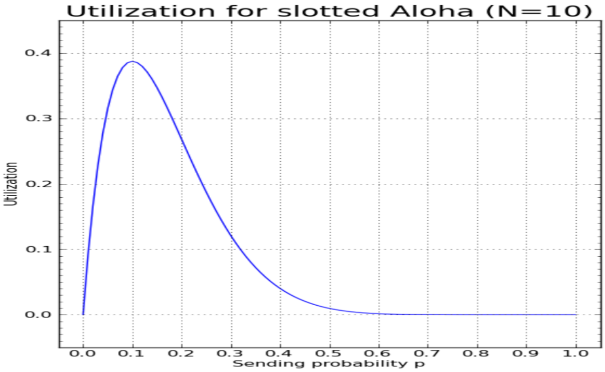
maximizing utilization: $U(p) = np(1-p)^{n-1} \to \max_{p \in [0,1]}$

$$\frac{dU}{dp} = n(1-p)^{n-1} - n(n-1)(1-p)^{n-2} = n(1-p)^{n-2}(1-np).$$

The function $U = U(p)$ has positive derivative when $0 \le p < 1/n$, negative derivative when $1/n < p < 1$. Hence it achieves maximum at $p = 1/n$.

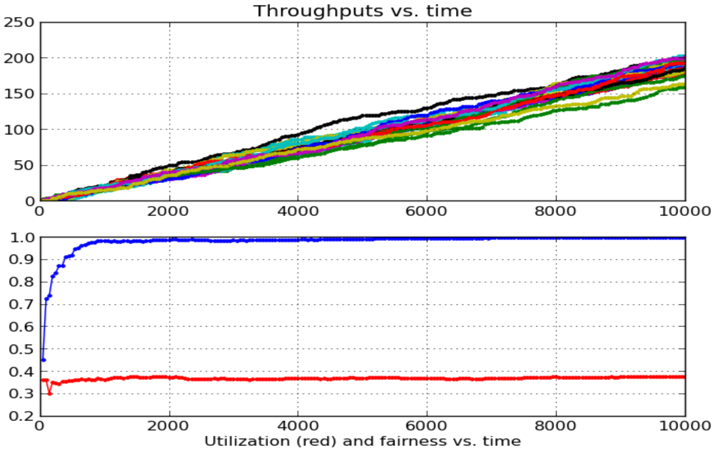
This Magic Number, 0.37

$$U_{\max} = \frac{n}{n}\left(1 - \frac{1}{n}\right)^{n-1} \rightarrow \frac{1}{e} \approx 0.37 \quad \text{as} \quad n \rightarrow \infty$$



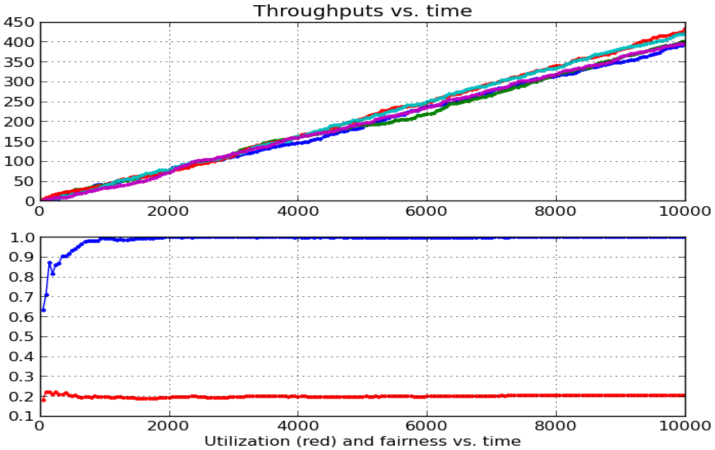
When $p = 0.05$ Is Perfect

BTW, $n = ?$



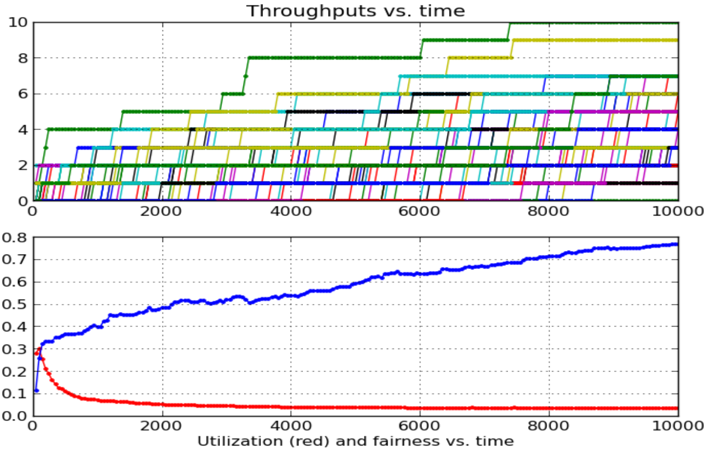
When $p = 0.05$ But Should Be 0.2

BTW, $n = ?$



When $p = 0.05$ But Should Be 0.01

BTW, $n = ?$



Stabilization: Selecting The Right p

The Issue:

- setting $p = 1/n$, where n is the number of backlogged nodes, maximizes utilization
- in many applications, the number of backlogged nodes is constantly varying: **we do not know n!**
- how to dynamically adjust p to achieve maximum utilization?

The Solution:

- detect collisions by listening, or by missing acknowledgement
- each node maintains its own dynamically changing p
- if collision detected (too much traffic), decrease local p
- if success (maybe more traffic possible), increase local p

Stabilization: the process of ensuring operation at, or near, a desired operating point. Stabilizing Aloha means finding a p that maximizes utilization as loading changes.

Stabilization by Exponential Back-Off

Select parameters $\alpha, \beta, p_{\text{min}}, p_{\text{max}}$ such that

$$0 < \alpha < 1, \qquad \beta > 1, \qquad 0 < p_{\text{min}} < p_{\text{max}} < 1$$

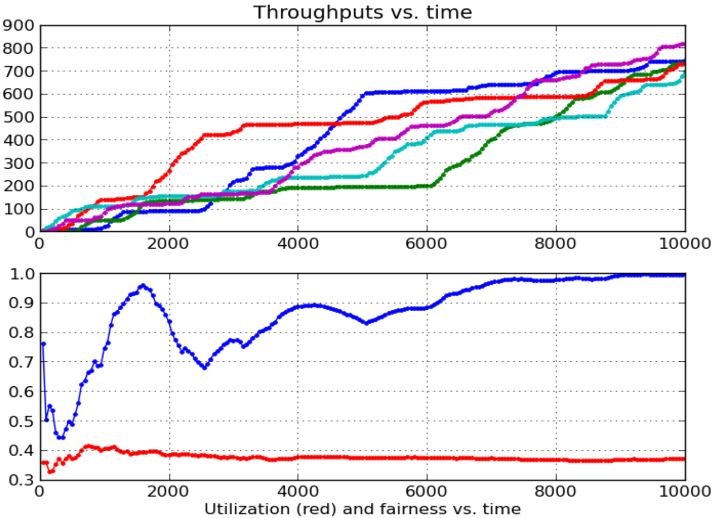
Decreasing p on collision:

$$p_{\text{next}} = \max\{\alpha p, p_{\text{min}}\}$$

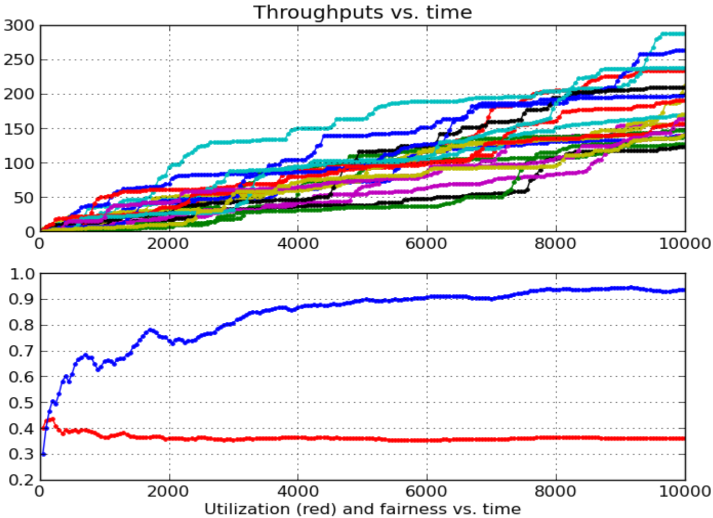
Increasing p on success:

$$p_{\text{next}} = \min\{\beta p, p_{\text{max}}\}$$

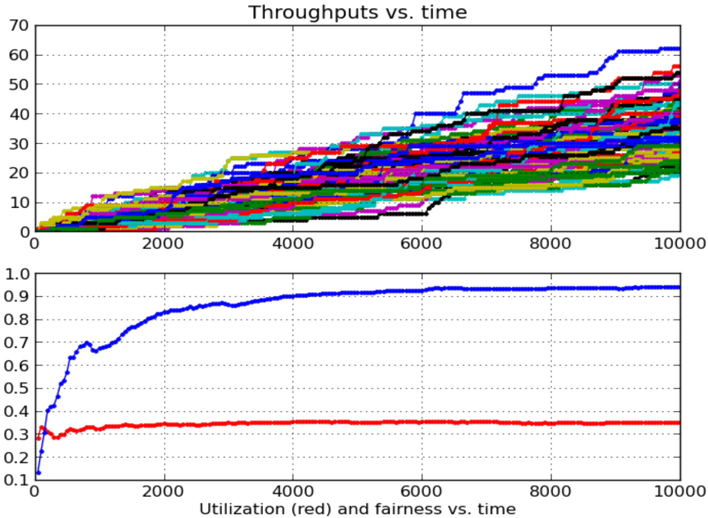
When $p_{\min} = 0.01$, $p_{\max} = 0.4$, $\alpha = 0.4$, $n = 5$



When $p_{\min} = 0.01$, $p_{\max} = 0.4$, $\alpha = 0.4$, $n = 20$



When $p_{\min} = 0.01$, $p_{\max} = 0.4$, $\alpha = 0.4$, $n = 100$



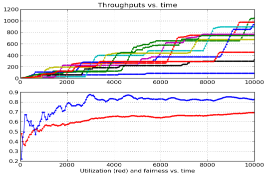
Example: Ethernet Media Access Control

- the network is monitored for transmissions ("carrier sense")
- if an active carrier is detected, transmission is deferred
- if active carrier is not detected, begin frame transmission
- while transmitting, monitor for a collision
- if a collision is detected, transmit "jam sequence"
- wait a random period of time before re-starting transmission
- on repeated collisions, increase random delay
- on success, clear the collision counter used for backoff

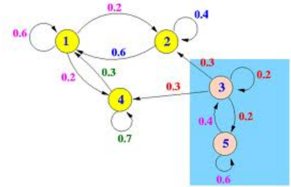
(from http://www.techfest.com/networking/lan/ethernet3.htm)

Analysis Of Stabilization Algorithms

simulation (e.g., PS7)

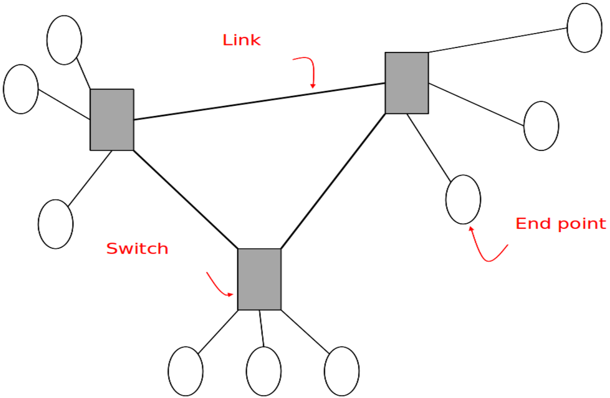


Markov Chains (see 6.041)



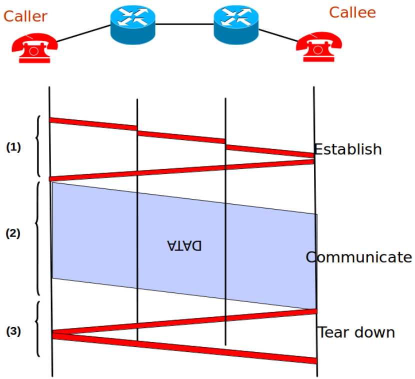
Multi-Hop Networks

Another take on sharing:



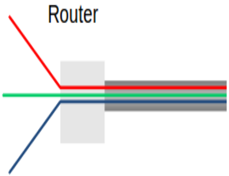
Circuit Switching

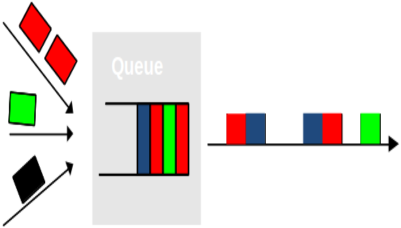
- establish a circuit between end points (e.g. by dialing a phone number)
- communicate using the established path
- tear down the connection (e.g. hang up)



Packet Switching

- packet headers have destination info
- routers have **routing tables** – links to destinations info
- packets wait in link queues, dropped if full





To Survive A Major Attack

WWJC ? Paul Baran in the late 1950s:

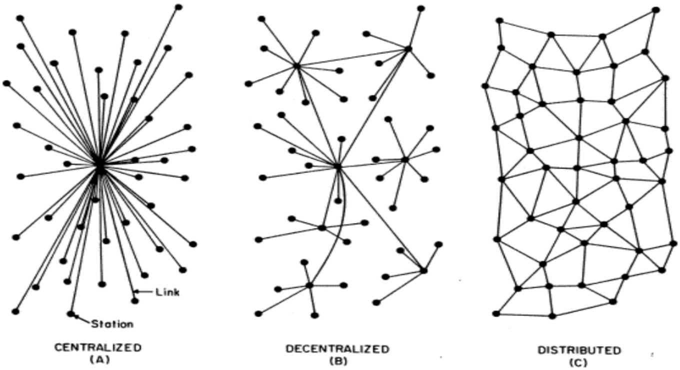
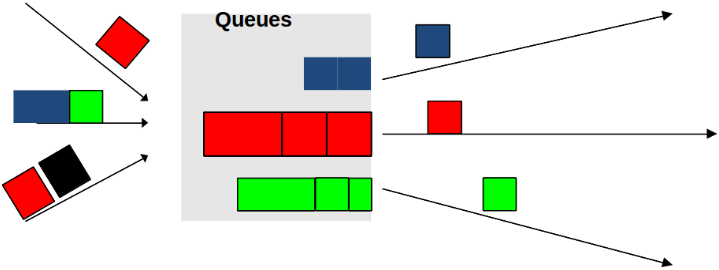


FIG. I — Centralized, Decentralized and Distributed Networks

Queues As A Necessary Evil

- manage packets between arrival and departure
- needed to absorb bursts
- add delay by making packets wait until link is available
- shouldnt be too big



Little's Law

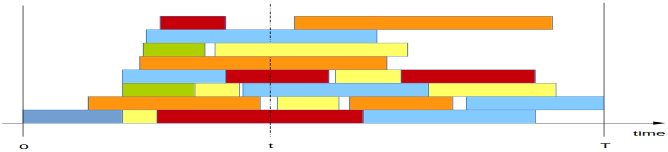
- Q_{avg} – average queue size
- D_{avg} – average packet delay
- R – throughput rate (packets per unit of time)

$$\boxed{Q_{avg} = R \cdot D_{avg}}$$

A true mathematical statement when

- zero queue length at the start and at the end, or
- packet delay counts only between the start and the end, or
- ovbservation time is large compared to the product of maximal queue size and maximal delay

Little's Law: A Proof



- T – length of queue observation (from $t = 0$ to $t = T$)
- N –number of packets observed
- $Q(t)$ – packets in the queue at time t
- D_k delay for the kth packet

Total area:

$$\int_0^T Q(t)dt = \sum_{k=1}^{N} D_k \quad \text{i.e.} \quad \overbrace{\frac{1}{T}\int_0^T Q(t)dt}^{Q_{avg}} = \underbrace{\frac{N}{T}}_{R} \overbrace{\frac{1}{N}\sum_{k=1}^{N} D_k}^{D_{avg}}$$