MAC Theory

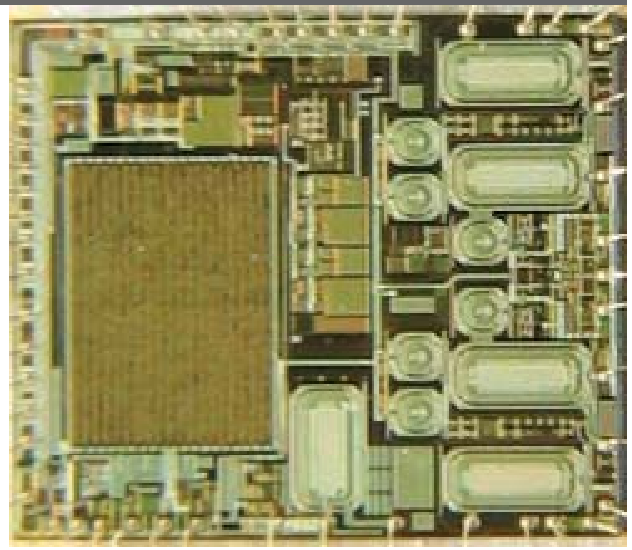
Chapter 7

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Standby Energy [digitalSTROM.org]

- 10 billion electrical devices in Europe
- 9.5 billion are not networked
- 6 billion euro per year energy lost

- Make electricity smart
  - cheap networking (over power)
  - true standby
  - remote control
  - electricity rates
  - universal serial number
  - …



Rating

- Area maturity

- Practical importance

- Theoretical importance

Overview

- Understanding Aloha
- Unknown Neighborhood
- The Broadcast Problem
- CDMA

The best MAC protocol?!?

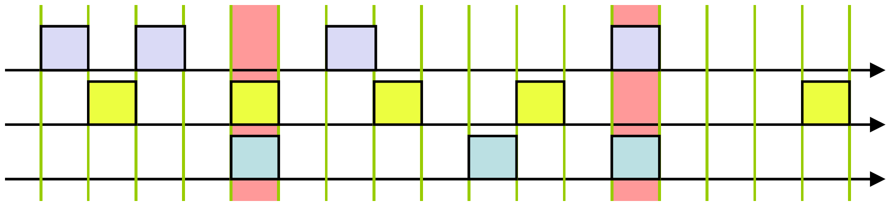
- Energy-efficiency vs. throughput vs. delay
- Worst-case guarantees vs. best-effort
- Centralized/offline vs. distributed/online

- So, clearly, there cannot be a best MAC protocol!

- … but we don’t like such a statement
  - We study some ideas in more detail…

Slotted Aloha

- We assume that the stations are perfectly synchronous
- In each time slot each station transmits with probability *p*.



$$P_1 = \Pr[\text{Station 1 succeeds}] = p(1-p)^{n-1}$$

$$P = \Pr[\text{any Station succeeds}] = nP_1$$

$$\text{maximize } P: \frac{dP}{dp} = n(1-p)^{n-2}(1-pn) \overset{!}{=} 0 \quad \Rightarrow \quad pn = 1$$

$$\text{then, } P = (1-\frac{1}{n})^{n-1} \geq \frac{1}{e}$$

- In Slotted Aloha, a station can transmit successfully with probability at least $1/e$, or about 36% of the time.

Some formula favorites („Chernoff-type“ inequalities)

- How often do you need to repeat an experiment that succeeds with probability *p*, until one actually succeeds? About 1/*p* times.
- Basic insights like this have been formulated in various ways, for instance:

For all p, k, such that $0 < p < 1$ and $k \geq 1$,

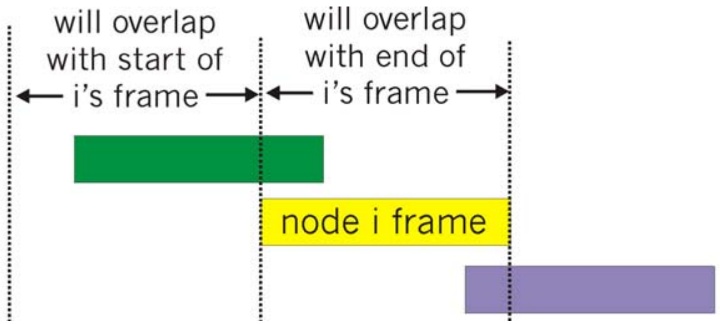
$$1 - p \leq \left(1 - \frac{p}{k}\right)^k .$$

For all n, t, such that $n \geq 1$ and $|t| \leq n$,

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \leq e^t .$$

Unslotted (Pure) Aloha

- Unslotted Aloha: simpler, no (potentially costly!) synchronization
- However, collision probability increases. Why?
- To simplify the analysis, we assume that
  - All packets have equal size.
  - We still have tiny time slots, that is, each packet takes t slots to complete, with $t \rightarrow \infty$.
  - In order to get comparable numbers to the slotted case, assume that a node starts a transmission with probability p/t.
  - Since a transmission can interfere with $2t$-1 starting points of n-1 other nodes, we have:



$$P[\text{transmission succeeds}] \approx \frac{p}{t}\left(1 - \frac{p}{t}\right)^{(2t-1)(n-1)} \approx \frac{p}{t}\left(1 - \frac{p}{t}\right)^{2tn}$$

Unslotted Aloha (2)

- What *p* maximazes this probability?

$$\frac{d}{dp}\frac{p}{t}(1-\frac{p}{t})^{2tn} = \frac{1}{t}(1-\frac{p}{t})^{2tn} - \frac{p}{t}2n(1-\frac{p}{t})^{2tn-1}$$

$$0 \stackrel{!}{=} \frac{1}{t}(1-\frac{p}{t})^{2tn-1} \cdot (1-\frac{p}{t}-2pn)$$

- Hence: $p = \frac{t}{1-2nt} \approx \frac{1}{2n}$

- Plugging *p* back in, we have a successful transmission of any of the *n* stations in time *t* of:

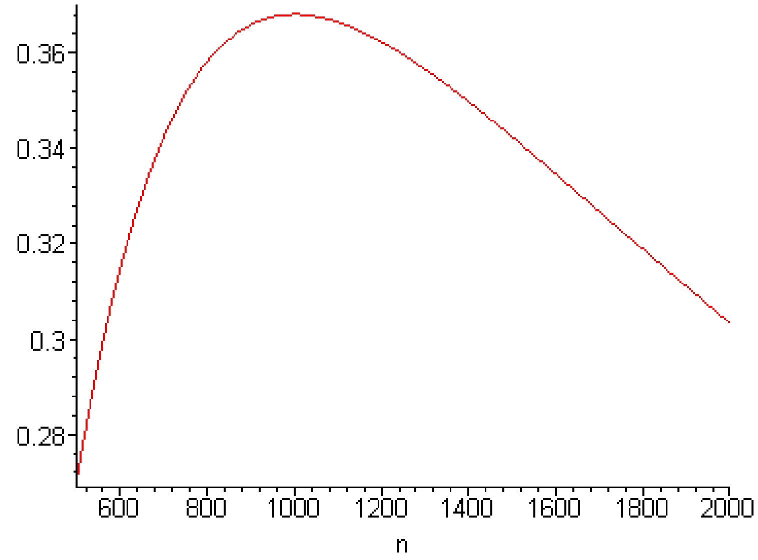
$$P[\text{success}] \approx nt\frac{p}{t}(1-\frac{p}{t})^{2tn} = nt\frac{1}{2nt}(1-\frac{1}{2nt})^{2tn} \approx \frac{1}{2e}$$

- This is the often-quoted factor-2-handicap of unslotted vs. slotted.

Aloha Robustness

- We have seen that round robin has a problem when a new station joins. In contrast, Aloha is quite robust.

- Example: If the actual number of stations is twice as high as expected, there is still a successful transmission with probability 30%. If it is only half, 27% of the slots are used successfully. So nodes just need a good estimate of the number of nodes in their neighborhood.

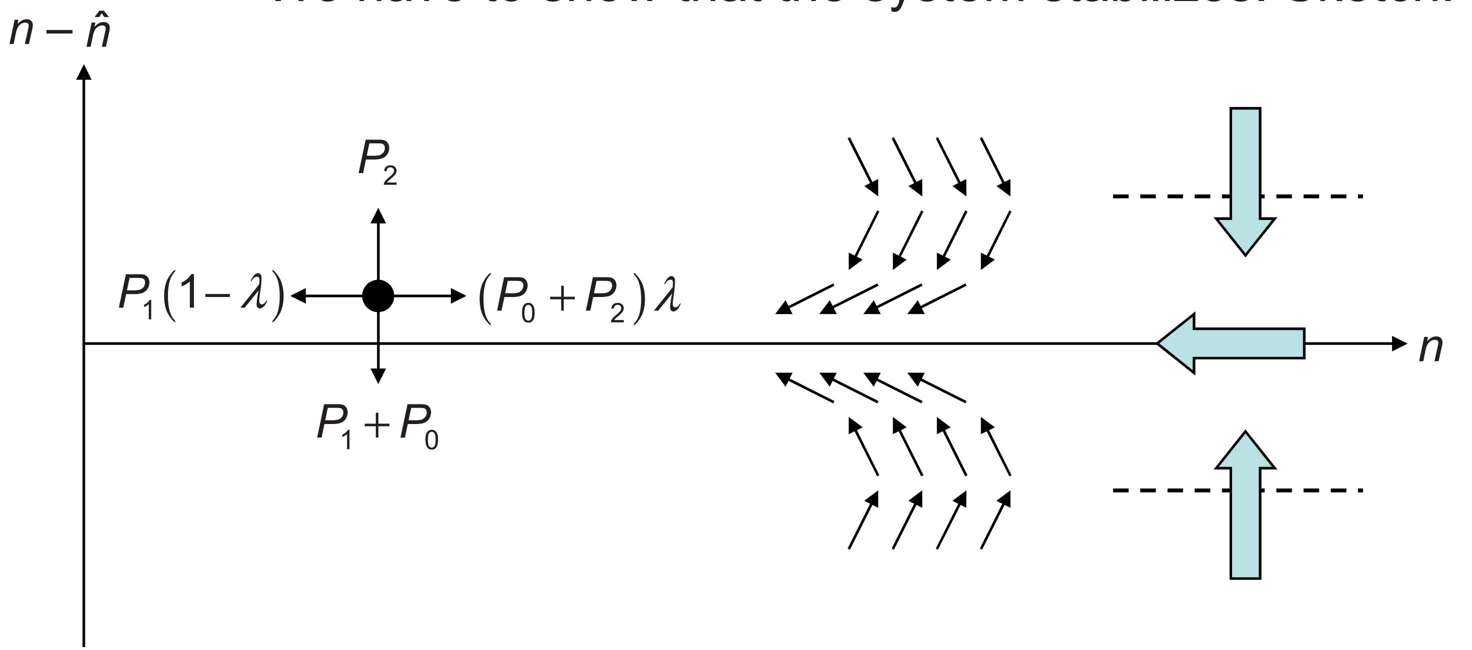


Adaptive slotted aloha

- Idea: Change the access probability with the number of stations
- How can we estimate the current number of stations in the system?
- Assume that stations can distinguish whether 0, 1, or more than 1 stations transmit in a time slot.
- Idea:
  – If you see that nobody transmits, increase p.
  – If you see that more than one transmits, decrease p.
- Model:
  – Number of stations that want to transmit: n.
  – Estimate of n: $\hat{n}$
  – Transmission probability: $p = 1/\hat{n}$
  – Arrival rate (new stations that want to transmit): λ (with $\lambda < 1/e$).

Adaptive slotted aloha 2

We have to show that the system stabilizes. Sketch:



$\hat{n} \leftarrow \hat{n}+\lambda-1$, if success or idle

$\hat{n} \leftarrow \hat{n}+\lambda+\dfrac{1}{e-2}$, if collision

Adaptive slotted aloha Q&A

Q: What if we do not know λ, or λ is changing?

A: Use $\lambda = 1/e$, and the algorithm still works.

Q: How do newly arriving stations know $\hat{n}$?

A: We send $\hat{n}$ with each transmission; new stations do not send before successfully receiving the first transmission.

Q: What if stations are not synchronized?

A: Aloha (non-slotted) is twice as bad.

Q: Can stations really listen to all time slots (save energy by turning off)? Can stations really distinguish between 0, 1, and $\geq$2 sender?

A: Maybe. One can use systems that only rely on acknowledgements.

Unknown Neighborhood?

- We have *n* nodes, all direct neighbors (no multi-hop).
  - However, the value *n* is not known (a.k.a. "uniform" model)
- Time is slotted (as in Slotted Aloha).
  - Synchronous start: All nodes start the protocol at the very same instant.
- In each time slot, a node can either transmit or receive.
  - If exactly one node transmits, all other nodes will receive that message.
  - Without collision detection: More than one transmitting node cannot be distinguished from nobody transmitting. There is just no message that can be received correctly (because of interference).
  - Transmitters cannot know whether they transmitted alone or not.

- What would we want to achieve?
  - Lots of throughput? Fairness between transmitters?
  - Get an exact count of *n*? Get an estimate of *n*?
  - How long does it take until a single node can transmit alone!

Uniform, Sync-Start, Without Collision Detection

- Can a deterministic algorithm work?
  - If nodes just execute the very same algorithm, even two nodes cannot solve the problem because they would always do exactly the same all the time (and none of them would ever receive the transmission of the other).
  - In other words, they need to execute some algorithm that heavily depends on their node ID. Such an algorithm must work for all combinations of possible node ID's. Although this is certainly possible, it's quite difficult. Randomized algorithms are much easier.

- Just transmit with probability $p = 1/n$.
  - Simple; finishes in expected e (2.71) rounds.
  - But not uniform!

Uniform, Sync-Start, Without Collision Detection (2)

- Alternative: In slot k, send with $p = 1/k$.
  - This is uniform (there is no n in the algorithm).
  - But it is also too slow, as it takes n rounds to get to Aloha.

- Better alternative: Send with probability $p = 2^{-k}$ for $e \cdot k$ slots, $k = 1,2, \ldots$
  - At first, p is too high, but soon enough $2^k \approx n$.
  - If we assume (for simplicity) that $2^k = n$, then the probability that any single node transmits alone is $n \cdot 2^{-k} \cdot (1-2^{-k})^{n-1} \approx (1-1/n)^n \geq 1/e$.
  - Since each phase has ek slots, the probability that one of them is successful is $1-(1-1/e)^{ek} \geq 1-e^{-k} \geq 1-1/n$.
  - This last term is known as „with high probability". Hence, with high probability we are successful after $O(\log^2 n)$ steps.

- How does the successful sender know that it's done?

Uniform, Asynchronous Start, Without Collision Detection

- Assume that nodes may wake up in an arbitrary (worst-case) way.
- Also assume that nodes do not have ID's
  - In other words, all nodes must perform the same way, until one node can transmit alone (at which point the others may learn and adapt).

- How long does it take until the first node can transmit alone?
  - If nodes that are awake never transmit (just listen), we will never finish.
  - There must be a first time slot where a node tries to transmit, with probability *p*. Remember that all nodes perform the same protocol!
  - We have the uniform model, hence *p* is a constant, independent of *n*.
  - We trick the algorithm by waking up 4/p·log *n* nodes each step.
  - Using our Chernoff bounds, with high probability at least two newly woken nodes will transmit in each slot. We always have collisions!

- Hence, in this model any algorithm will need at least Ω(n / log n) time!

[Jurdzinski, Stachowiak, 2005]

Uniform, Sync-Start, With Collision Detection

- In each time slot, a node can either transmit or receive.
  - If exactly one node transmits, all other nodes will receive that message.
  - With collision detection: More than one transmitting node can be distinguished from nobody transmitting.
    - There are models where one can estimate the number of transmissions
    - Here we just assume to differentiate between 0, 1, or ≥ 2 transmissions.
    - Transmitters themselves do not know anything about other transmissions.

- Simple Algorithm:

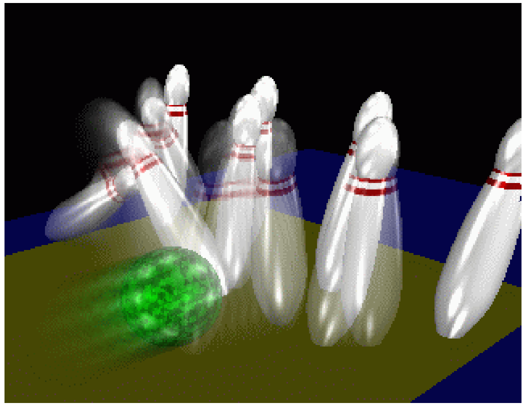
```
repeat
    repeat transmit; throw coin until coin shows head;
    listen
until somebody was transmitting when listening;
```

- After O(log *n*) steps, only a constant number remains in the pool.
- After O(log *n*) more steps, only one remains (with high probability)!

Uniform, Sync-Start, With Collision Detection [Willard 1986]

- The power of collision detection
  - For instance, a transmitter *s* can figure out if it transmitted alone. If *s* was alone (case 1), all but *s* should transmit in the next time slot; if *s* was not alone (0 or $\geq$2), all should remain silent in the next time slot. Using this trick we may elect a "leader".
  - Similarly all can figure out if there was at least one sender.

- Also, we can get a rough estimate of the number of nodes quickly
  - Just reduce the sending probability … … aggressively
  - Indeed, in round k, send with probability $1/2^{2^{k^2}}$, for $k \geq 0$.
  - This becomes interesting if it is about equal to 1/*n*, that is $k \approx \sqrt{\text{loglog } n}$.
  - Now we check all $1/2^{2^i}, i = k^2, \ldots, 0.$
  - This costs loglog *n* time, approximating *n* well (2^{2^i})
  - After this phase only logloglog *n* nodes survive.
  - With so few nodes, loglog *n* tests are enough.
  - The total time is O(loglog *n*).



The best multi-hop MAC protocol?!?

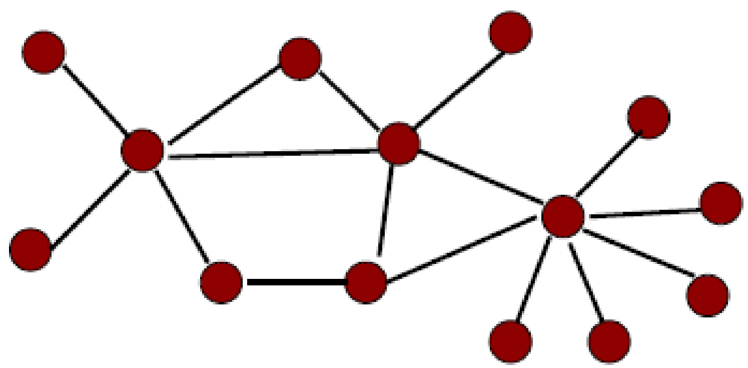
- As in single-hop, there cannot be a best MAC protocol.
  - Energy-efficiency vs. throughput vs. delay
  - Worst-case guarantees vs. best-effort
  - Centralized/offline vs. distributed/online
- Multi-hop challenges?
  - Random topology vs. worst-case graph vs. worst-case UDG vs. …
  - Network layer: local broadcast vs. all-to-all vs. broadcast/echo
  - Transport layer: continuous data vs. bursts vs. …
- We need a simple multi-hop case study
  - The “Broadcasting” Problem

Model

- Network is an undirected graph (arbitrary, not UDG)
  – Nodes do not know topology of graph
- Synchronous rounds
  – Again, nodes can either transmit or receive
- Message is received if exactly one neighbor transmits
  – Without collision detection: That is, a node cannot distinguish whether 0 or 2 or more neighbors transmit

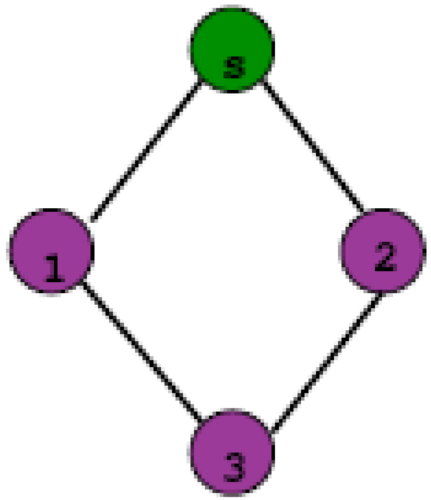
- We study broadcasting problem
  – sort of multi-hop MAC layer, not quite
  – Initially only source has message
  – finally every node has message

- How long does this take?!?



Deterministic Anonymous Algorithms

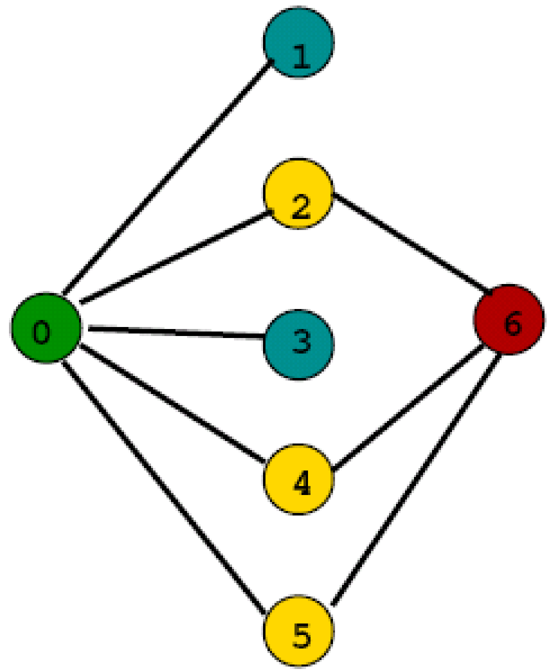
- If nodes are anonymous (they have no node IDs), then one cannot solve the broadcast problem
  - For the graph on the right nodes 1 and 2 always have the same input, and hence always do the same thing, and hence node 3 can never receive the message.
- So, again, the nodes need IDs, or we need a randomized algorithm. We first study the deterministic case!



Deterministic algorithms (not anonymous)

- Consider the following network family:

- *n*+2 nodes, 3 layers
  - First layer: source node (green)
  - Last layer: final node (red)
  - Middle layer: all other nodes (*n*)
  - Source connected to all nodes in middle layer
  - Middle layer consists of golden and blue nodes
  - Golden nodes connect to red node,
    blue nodes don't.

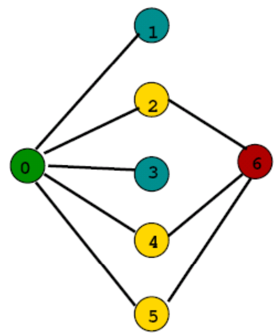


- In one single step all middle nodes know message.
- And…? The problem is that we don't know the golden nodes!

How to choose golden nodes?

- Task:
  - Given deterministic algorithm, i.e., we have sets M_i of nodes that transmit concurrently, first set M_1, then M_2, etc.
  - Choose golden and blue nodes, such that no set M_i contains a single golden node.

- Construction of golden set
  - We start with golden set S being all middle nodes
  - While $\exists$ M_i such that $|M_i \cap S| = 1$ do $S := S \setminus \{M_i \cap S\}$



- Any deterministic algorithm needs at least *n* rounds
  - In every iteration a golden node intersecting with M_i is removed from S; set M_i does not have to be considered again afterwards.
  - Thus after n-1 rounds we still have one golden node left and all sets M_i do not contain exactly one golden node.

Improvement through randomization?

- If in each step a random node is chosen that would not help much, because a single golden node still is only found after about $n/2$ steps. So we need something smarter…

- Randomly select $n^{i/k}$ nodes, for $i = 0, 1,\ldots,k-1$ also chosen randomly.
  - Assume that there are about $n^{s/k}$ golden nodes.
  - Then the chance to randomly select a single golden node is about

$$Pr(\text{success}) = \underbrace{n^{i/k}}_{\text{Positions for golden node}} \cdot \underbrace{n^{s/k-1}}_{\text{Probability for golden node}} \cdot \underbrace{(1-n^{s/k-1})^{n^{i/k}-1}}_{\text{All others are not golden}}$$

  - If we are lucky and $k \approx i+s$ this simplifies to

$$Pr(\text{success}) \approx 1 \cdot \left(1 - \frac{1}{n^{i/k}}\right)^{n^{i/k}} \approx 1/e$$

  - If we choose $k = \log n$ and do the computation correctly, we have polylogarithmic trials to find a single golden node.

Randomized protocol for arbitrary graphs

Broadcast(N, Δ)

$k := 2\lceil \log \Delta \rceil$
$p := \lceil \log(N/\varepsilon) \rceil$
wait till *msg* arrives
for p phases **do**
  wait till $(rnd \ mod \ k) = 0$
  Decay(k, msg)
end for

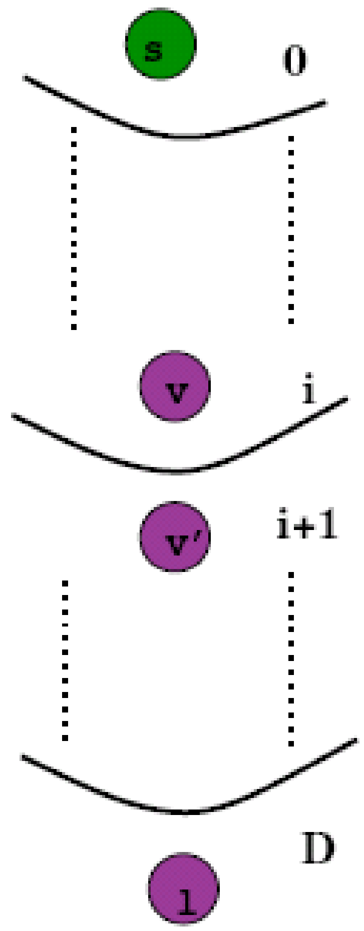
Decay(k, msg)

$coin := heads$
$steps := 0$
while $coin = heads$ and $steps \leq k$ **do**
  send *msg* to neighbours
  flip *coin*
  increment *steps*
end while

- $O(D \cdot \log^2 n)$
- N: upper bound on node number
- Δ: upper bound on max degree
- ϵ: Failure probability, think $\epsilon = 1/N$
- N, Δ, ϵ are globally known
- D: diameter of graph
- Algorithm runs in synchronous phases, nodes always transmit slot number in every message; source sends message in first slot.
- (Note that the Decay algorithm is pretty similar to some of our single-hop algorithms.)

Proof overview

- During one execution of Decay a node can successfully receive a message with probability $p \geq 1/(2e)$
- Iterating Decay $c \cdot \log n$ times we get a very high success probability of $p \geq 1\text{-}1/n^c$
- Since a single execution of *Decay* takes $\log n$ steps, all nodes of the next level receive the message after $c \cdot \log^2 n$ steps (again, with very high probability).
- Having D layers a total of $O(D \cdot \log^2 n)$ rounds is sufficient (with high probability).



Fastest Broadcast Algorithm [Czumaj, Rytter 2003]

- Known lower bound $\Omega(D \cdot \log(n/D) + \log^2 n)$
- Fastest algorithm matches lower bound. Sketch of one case:

for any $k \in \{0, 1, \ldots, \log n\}$, we define

= loglog n

$$\alpha_k = \begin{cases} 2^{-(k+1)} & \text{for } 1 \le k \le \mathcal{LL}(n) , \\ \frac{1}{2 \log n} & \text{for } \mathcal{LL}(n) \le k \le \log n , \\ 1 - \sum_{i=1}^{\log n} \alpha_i & \text{for } k = 0 . \end{cases}$$

Input: Network $\mathcal{N} = (V, E)$.

Randomized sequence $\mathfrak{I} = \langle I_1, I_2, \ldots \rangle$ such that

$\mathbf{Pr}[I_r = k] = \alpha_k \; \forall r \in \mathbb{N}, \forall k \in \{0, 1, 2, \ldots, \log n\}$

for $r = 1$ **to** T **do** { *round number* r }

for each *active* node $v \in V$ *independently* **do**

node v transmits with probability 2^{-I_r}

Node that received message from source

Code Division Multiple Access (CDMA)

- CDMA is a novel Physical/MAC concept.

- Example: Direct Sequence Spread Spectrum (DSSS)
- Each station is assigned an m-bit code (or chip sequence)
- Typically m = 64, 128, ... (in our examples m = 4, 8, …)
- To send 1 bit, station sends chip sequence
- To send 0 bit, station sends complement of chip sequence

- Instead of splitting a 1 MHz band shared between 100 channels into 100 x 10kHz bands, every station can use the whole band, with 100 chips.
  - CDMA does not increase the total bandwidth, but it may simplify the MAC layer at the expense of complicating the physical layer.

CDMA basics 1

Each station s has unique m-bit chipping code S or complement $\bar{S}$

Bipolar notation: binary 0 is represented by -1 (or short: $-$)

Two chips S, T are orthogonal iff $S \cdot T = 0$

$S \cdot T$ is the inner (scalar) product: $S \cdot T = \frac{1}{m} \sum_{i=1}^{m} S_i T_i$

Note: $S \cdot S = 1, S \cdot \bar{S} = -1$

Note: $S \cdot T = 0 \Rightarrow S \cdot \bar{T} = 0$

CDMA basics 2

- Assume that all stations are perfectly synchronous
- Assume that all codes are pairwise orthogonal
- Assume that if two or more stations transmit simultaneously, the bipolar signals add up linearly

- Example
- S = (+ – + – + – + –)
- T = (+ + – – – + + –)
- U = (+ – – + – – + +)
- Check that codes are pairwise orthogonal
- If S, T, U send simultaneously, a receiver receives
  $R = S+T+U = (+3, –1, –1, –1, –1, –1, +3, –1)$

CDMA basics 3

- To decode a received signal R for sender s, one needs to calculate the normalized inner product $R \cdot S$.

- $R \cdot S$ = (+3, –1, –1, –1, –1, –1, +3, –1)·(+ – + – + – + –)/8
  = (+3+1–1+1–1+1+3+1)/8
  = 8/8 = 1 … by accident?

- $R \cdot S = (S+T+U) \cdot S = S \cdot S + T \cdot S + U \cdot S = 1 + 0 + 0 = 1$

- With orthogonal codes we can safely decode the original signals

CDMA: How much noise can we tolerate?

- We now add random noise:
- $R' = R + N$, where N is an m-digit noise vector.

- Assume that chipping codes are balanced (as many “+” as “–”)
- If $N = (\alpha, \alpha, \ldots, \alpha)$ for any (positive or negative) α, then the noise N will not matter when we decode the received signal.

- $R' \cdot S = (R+N) \cdot S = S \cdot S$ +(orthogonal codes)$\cdot S + N \cdot S = 1 + 0 + 0 = 1$

- How much random (white) noise can we tolerate?

CDMA: Construction of orthogonal codes with *m* chips

- Note that we cannot have more than *m* orthogonal codes with *m* chips because each code can be represented by a vector in the *m*-dimensional space, and there are not more than *m* orthogonal vectors in the *m*-dimensional space.
- Walsh-Hadamard codes can be constructed recursively (for $m = 2^k$):

  The set of codes of length 1 is $C_0 = \{(+)\}$.

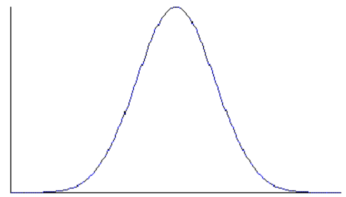
  For each code $(c) \in C_k$ we have two codes $(c\ c)$ and $(c\ \bar{c})$ in C_{k+1}
- Code tree:

  $$C_0 = \{(+)\}$$
  $$C_1 = \{(+\,+),(+\,-)\}$$
  $$C_2 = \{(+\,+\,+\,+),(+\,+\,-\,-),(+\,-\,+\,-),(+\,-\,-\,+)\}$$

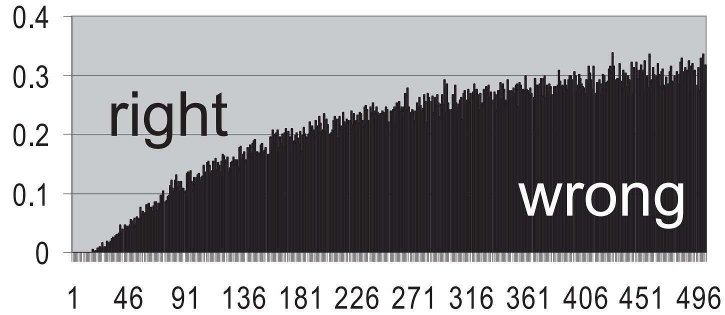
- Note: Random codes are also quite balanced and pretty orthogonal.

CDMA: Random codes

- With k other stations, and m chips
- $m \cdot R \cdot S = m \cdot S \cdot S + m \cdot (k \text{ random codes}) \cdot S = \pm m + X$, where X is the sum of mk random variables that are either +1 or –1.
- Since the random variables are independent, the expected value of X is 0. And better: The probability that X is “far from 0” is “small.”



- Therefore we may decode the signal as follows:
  $R \cdot S > \varepsilon$ □ decode 1; $R \cdot S < -\varepsilon$ □ decode 0. What if $-\varepsilon \le R \cdot S \le \varepsilon$??
- Experimental evaluation (right): For $k = m = 128$ decoding is correct more than 80%. But more importantly: Even if $k > m$ (k=1..500), the system does not deteriorate quickly.



CDMA: Problems

Some of our assumptions were a bit problematic:

A) It is not possible to synchronize chips perfectly. What can be done is that the sender first transmits a long enough known chip sequence on which the receiver can lock onto.

B) Not all stations are received with the same power level. CDMA is typically used for systems with fixed base stations. Then mobile stations can send with the reciprocal power they receive from the base station. (Alternatively: First decode the best station, and then subtract its signal to decode the second best station…)

C) We didn’t discuss how to transmit bits with electromagnetic waves.

CDMA: Summary

- \+ all terminals can use the same frequency, no planning needed
- \+ reduces frequency selective fading and interference
- \+ base stations can use the same frequency range
- \+ several base stations can detect and recover the signal
- \+ soft handover between base stations
- \+ forward error correction and encryption can be easily integrated
- – precise power control necessary
- – higher complexity of receiver and sender

Example: UMTS

Conclusion

- A lot of theoretical research is centered around Aloha-style research, since in the big-Oh world, 36% or 18% throughput is only a constant factor off the optimal, which is considered “negligible”, or “asymptotically optimal”…
- In reality, we would often not be happy with an algorithm that finishes the task in O(f(n)) time, if the hidden constant is huge. Not even if the hidden constant is, ugh, constant.
- What we need is a mix between Aloha, TDMA, and reservation.