Distributed Wireless Networks

Thanks to Yvonne-Anne Pignolet from ABB research for basis of slides!
Wireless Networks

Big advantage: no wires! Network setup easy and fast.
Wireless Networks

Big advantage: no wires! Network setup easy and fast.

Big challenge: no wires! Interference, attenuation, energy supply.
Wireless Networks

Big advantage: no wires! Network setup easy and fast.

Big challenge: no wires! Interference, attenuation, energy supply.

The big question is: to send or not to send?

Avoid interference!
Radio Network Model

I can:

send XOR
receive
reach all other nodes

Our model today.
Radio Network Model

I can:

- send XOR
- receive
- reach all other nodes
- only receive if exactly one node transmits

One antenna.
Collision Detection (CD)

In a system with CD, can distinguish busy from idle, i.e., whether one or more nodes send at the same time (interference), from nobody transmitting.
Fundamental Wireless Problems

Leader Election

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...
Fundamental Wireless Problems

Nice to have: Leader could then also coordinate the future slots efficiently...

Leader Election

How long does it take until one node can transmit alone?

Initialization

How to assign IDs \{1, 2, \ldots, n\}? Slots could be divided accordingly then!

Nice to have: given that, nodes can send one-by-one! Fair and efficient.
Fundamental Wireless Problems

Leader Election

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...

Initialization

How to assign IDs \{1, 2, ….., n\}? Slots could be divided accordingly then!

Many problem variants:

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Fundamental Wireless Problems

**Leader Election**

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...

**Initialization**

How to assign IDs \{1, 2, \ldots, n\}? Slots could be divided accordingly then!

**Now:** How to do leader election w/o CD in Radio Network model?

**Many problem variants:**

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Fundamental Wireless Problems

**Leader Election**

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...

**Initialization**

How to assign IDs \{1, 2, \ldots, n\}? Slots could be divided accordingly then!

Assume nodes have unique IDs!

**Now:** How to do leader election w/o CD in Radio Network model?

Many problem variants:

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Fundamental Wireless Problems

**Leader Election**

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...

**Initialization**

How to assign IDs \{1, 2, \ldots, n\}? Slots could be divided accordingly then!

Assume nodes have unique IDs!

**Now:** How to do leader election w/o CD in Radio Network model?

Many problem variants:

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?

Assume nodes start at the same time!
Fundamental Wireless Problems

Leader Election

How long do nodes need to to reach all nodes?
Two or more collide?

Initialization

How to assign IDs \{1, 2, \ldots, n\}?
Slots could be divided accordingly then!

Many problem variants:

Idea: node with ID x sends at time x!
Problem: long time till first node transmits (say IDs are MAC addresses)?

Assume nodes have unique IDs!

Now: How to do leader election w/o CD in Radio Network model?

Assume nodes start at the same time!

asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
The Classic Solution: Slotted Aloha

**Slotted Aloha**

```
repeat
    transmit with probability 1/n
until one node has transmitted alone
```

Simple model: $n$ is known (non-uniform)!
The Classic Solution: Slotted Aloha

**Slotted Aloha**

repeat
  transmit with probability $1/n$
until one node has transmitted alone

Simple model: $n$ is known (non-uniform)!

How long does it take until a node sends alone?
The Classic Solution: Slotted Aloha

**Slotted Aloha**

repeat
transmit with probability $1/n$
until one node has transmitted alone

Simple model: $n$ is known (non-uniform)!

How long does it take until a node sends alone?

And how does node know that it sent alone??
The Classic Solution: Slotted Aloha

**Slotted Aloha**

repeat
  transmit with probability $1/n$
until one node has transmitted alone

Simple model: $n$ is known (non-uniform)!

How long does it take until a node sends alone?

And how does node know that it sent alone??

**Random Variable X**

$x$ is the RV denoting the number of nodes transmitting in a given time slot
Slotted Aloha

repeat
transmit with probability $1/n$
until one node has transmitted alone

Probability that exactly one node sends:

$$Pr[X = 1] = n \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \approx \frac{1}{e}.$$ 

So expected time complexity: $e$
Slotted Aloha

repeat
transmit with probability 1/n
until one node has transmitted alone

Probability that exactly one node sends:

\[ Pr[X = 1] = n \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \approx \frac{1}{e}. \]

So expected time complexity: e

But: The problem is that the leader does not know he was alone and won?!
Slotted Aloha

repeat
    transmit with probability $1/n$
until one node has transmitted alone

Probability that exactly one node sends:

$Idea: nodes just send ACK to leader!$

$$Pr[X = 1] = n \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \approx \frac{1}{e}$$

So expected time complexity: $e$

But: The problem is that the leader does not know he was alone and won?!
Slotted Aloha

repeat
  transmit with probability $1/n$
until one node has transmitted alone

Probability that exactly one node sends:

$$Pr[X = 1] = \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \approx \frac{1}{e};$$

Nice: constant!

But the ACKs may collide as well!

So we need a leader to let the leader know he is the leader?! ARGH! 😊

But: The problem is that the leader does not know he was alone and won?!
The Solution: Distributed ACK

Distributed ACK

After leader successfully sent alone: all other nodes know the leader now, so the nodes start sending the ID of the leader with $1/n$. 
The Solution: Distributed ACK

Distributed ACK

After leader successfully sent alone:
all other nodes know the leader now, so the nodes start sending the ID of the leader with $\frac{1}{n}$.

So the leader will learn soon...
But termination still unclear: How can the node that sent the leader ID know the leader knows?
The Solution: Distributed ACK

Distributed ACK

After leader successfully sent alone:
all other nodes know the leader now, so the nodes start sending the ID of the leader with $1/n$.

So the leader will learn soon...
But termination still unclear: How can the node that sent the leader ID know the leader knows?

OK now easy: every other node already knows and can shut up, and the leader can send an acknowledgement to this node (e.g., leave next round reserved).
Remark: Synchronous Time Slots Not Needed

Slotted Aloha

repeat
  transmit with probability $1/n$
until one node has transmitted alone

Protocol also works without time slots: just send when needed

«Pure ALOHA» reduces success probability by a factor of two:

Slotted ALOHA

success if nobody else sends here!

Pure ALOHA

success if nobody else sends here!
Fundamental Wireless Problems

Leader Election

How long does it take until one node can transmit alone? Leader could also coordinate slots in future...

Initialization

How to assign IDs \{1, 2, \ldots, n\}? Slots could be divided accordingly then!

Many problem variants:

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Fundamental Wireless Problems

**Leader Election**

How long does it take until one node can transmit and coordinate slots?

**Initialization**

How to assign IDs \{1, 2, …, n\}? Slots could be divided accordingly then!

---

**Many problem variants:**

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Fundamental Wireless Problems

I can:
- send / receive
- reach all nodes

Two or more collide

Easy: just do repeated ALOHA / leader election!

How long does it take until one node can transmit alone and coordinate slots?

What about this problem: Ideas?

Initialization

How to assign IDs \(\{1, 2, \ldots, n\}\)?

Slots could be divided accordingly then!

Many problem variants:

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Repeated Aloha

i = 1
repeat
    transmit with probability 1/n
    if node v transmitted alone, v gets ID i (ACKed), then leaves: i++, n--
until all nodes have an ID
Repeated Aloha

\[ i = 1 \]

repeat
\[
\begin{align*}
\text{transmit with probability } &\frac{1}{n} \\
\text{if node v transmitted alone, v gets ID } &i \text{ (ACKed), then leaves: } i++, \ n-- \\
\text{until all nodes have an ID}
\end{align*}
\]

Competition among \( n, n-1, n-2, n-3, \ldots \) nodes!
Repeated Aloha

\[ i = 1 \]
\[
\text{repeat}
\]
\[ \begin{align*}
\text{transmit with probability } & \frac{1}{n} \\
\text{if } & \text{node } v \text{ transmitted alone, } v \text{ gets ID } i \text{ (ACKed), then leaves: } i++, n--
\end{align*} \]
\[
\text{until all nodes have an ID}
\]

Runtime: Aloha takes time \( e \), so \( n \cdot e \): linear in \( n \).

Competition among \( n, n-1, n-2, n-3, \ldots \) nodes!
Repeated Aloha

\[ i = 1 \]
\[ \textbf{repeat} \]
\[ \quad \text{transmit with probability } \frac{1}{n} \]
\[ \quad \textbf{if} \ \text{node } v \ \text{transmitted alone, } v \ \text{gets ID } i \ \text{(ACKed), then leaves: } i++, \ n-- \]
\[ \textbf{until} \ \text{all nodes have an ID} \]

Runtime: Aloha takes time \( e \), so \( n \cdot e \): linear in \( n \).

Can we do faster? And can we get rid of the assumption that we need to know \( n \)?

Competition among \( n \), \( n-1 \), \( n-2 \), \( n-3 \), ... nodes!
Repeated Aloha

\begin{align*}
i &= 1 \\
\text{repeat} \\
&\quad \text{transmit with probability } 1/n \\
&\quad \text{if node } v \text{ transmitted alone, } v \text{ gets ID } i \text{ (ACKed), then leaves: } i++, n-- \\
\text{until all nodes have an ID}
\end{align*}

Runtime: Aloha takes time \( e \), so \( n \cdot e \): linear in \( n \).

Can we do faster? And can we get rid of the assumption that we need to know \( n \)?

Let’s do with CD first!

Competition among \( n, n-1, n-2, n-3, \ldots \) nodes!
Idea: let nodes determine their ID in a binary manner!
Idea: let nodes determine their ID in a binary manner!

Initially: nodes start with empty bitstring, \( n \) unknown.
Idea: let nodes determine their ID in a binary manner!

Initially: nodes start with empty bitstring, \( n \) unknown.

Each node chooses random bit \( b_1 \in \{0,1\} \) and sends in slot \( b_1 \) and listens in slot \( 1 - b_1 \).
Idea: let nodes determine their ID in a binary manner!

Initially: nodes start with empty bitstring, n unknown.

Each node chooses random bit $b_1 \in \{0,1\}$ and sends in slot $b_1$ and listens in slot $1-b_1$.

If at least one transmission in both slots, continue to next round. Otherwise repeat this step!
Idea: let nodes determine their ID in a binary manner!

Initially: nodes start with empty bitstring, n unknown.

Each node chooses random bit \( b_1 \in \{0,1\} \) and sends in slot \( b_1 \) and listens in slot \( 1 - b_1 \).

If at least one transmission in both slots, continue to next round. Otherwise repeat this step!

Requires CD: idle or busy?
Idea: let nodes determine their ID in a binary manner!

Now repeat: Each node with $b_1$ chooses next random bit $b_2 \in \{0,1\}$ to append, and sends in slot $b_1 b_2$ and listens in slot $b_1 (1 - b_2)$.
Idea: let nodes determine their ID in a binary manner!

Now repeat: Each node with $b_1$ chooses next random bit $b_2 \in \{0,1\}$ to append, and sends in slot $b_1 b_2$ and listens in slot $b_1 (1 - b_2)$.

Again: if at least one transmission in both slots: append random bit 0 or 1, otherwise repeat step.
Idea: let nodes determine their ID in a binary manner!

... repeat until alone!
Idea: let nodes determine their ID in a binary manner!

No need to know n! But simultaneous start and collision detection.

... repeat until alone!
Uniform Initialization with CD

For node v:

Main():
1. \( m = 0 \) (* already identified nodes *)
2. \( b_v = \langle \rangle \) (* current bitstring of node v *)
3. RandomSplit\((b_v)\)

RandomizedSplit\((b)\)
1. Repeat
2. If \( b_v = b \)
3. choose \( r \in \{0,1\} \) at random (* next bit in string *)
4. in next two time slots: transmit in \( r \), listen in \( 1+r \mod 2 \)
5. until there was at least one transmission in both slots (two groups, requires collision detection)
6. If \( b_v = b \), append \( r \) to my string \( b_v \), i.e., \( b_v := b_v r \)
7. If single node \( u \) transmitted in slot \( r \), gets global ID \( m \); ++
8. Else recursively split remaining nodes:
9. RandomizedSplit\((b_0)\), RandomizedSplit\((b_1)\)
Uniform Initialization with CD

For node v:

Main():
1. \( m = 0 \) (* already identified nodes *)
2. \( b_v = \langle \rangle \) (* current bitstring of node v *)
3. RandomSplit\((b_v)\)

RandomizedSplit\((b)\)
1. Repeat
2. If \( b_v = b \)
3. choose \( r \in \{0,1\} \) at random (* next bit in string *)
4. in next two time slots: transmit in \( r \), listen in \( 1+r \) mod 2
5. until there was at least one transmission in both slots (collision detection)
6. If \( b_v = b \), append \( r \) to my string \( b_v \), i.e., \( b_v := b_v r \)
7. If single node u transmitted in slot \( r \), gets global ID \( m \); \( m++ \)
8. Else recursively split remaining nodes:
9. RandomizedSplit\((b_0)\), RandomizedSplit\((b_1)\)

Note: do not transform unique string to ID but use global counter \( m \): only then IDs between 1 and \( n \)
Analysis

How many splits needed?
Analysis

How many splits needed?

n-1: n leaves and n inner nodes in splitting tree.
Analysis

- How many splits needed?
  - n-1: n leaves and n inner nodes in splitting tree.

- How long until successful (non-empty) split?
Analysis

How many splits needed?

n-1: n leaves and n inner nodes in splitting tree.

How long until successful (non-empty) split?

Let RV X denote the size of the set.

\[ P[0 < X < k] = 1 - P[X=0] - P[X=k] = 1 - \frac{1}{2^k} - \frac{1}{2^k} \geq \frac{1}{2} \]

So per split \( O(1) \) rounds, so overall runtime \( O(n) \).
Analysis

How many splits needed?

n-1: n leaves and n inner nodes in splitting tree.

How long until successful (non-empty) split?

Let RV $X$ denote the size of the set (total $k$ nodes).

$$P[0 < X < k] = 1 - P[X=0] - P[X=k] = 1 - \frac{1}{2^k} - \frac{1}{2^k} \geq \frac{1}{2}$$

So per split $O(1)$ rounds, so overall runtime $O(n)$.

We solved a much more general problem than leader election in the same time as repeated Aloha!
Problems Solved? Really?
Problems Solved? Really?

How does node know it is alone?
Problems Solved? Really?

How does node know it is alone?

Could do leader election first who sends ACKs. Coming up soon...
Problems Solved? Really?

How does node know it is alone?

Could do leader election first who sends ACKs. Coming up soon...

How to do without collision detection??
Need a mechanism to distinguish between $S=\emptyset$ and $|S|>0$. 
Problems Solved? Really?

There is a neat trick: given a leader, I can emulate CD in a scenario without CD!

How does node know it is alone?

Could do leader election first who sends ACKs. Coming up soon…

How to do without collision detection??
Need a mechanism to distinguish between $S=\emptyset$ and $|S|>0$. 

\[
\begin{array}{c}
\text{} \\
\text{} \\
\text{} \\
00 \\
01 \\
10 \\
11 \\
000 \\
001 \\
\end{array}
\]
Trick: Emulate CD w/o CD

assume: leader
Trick: Emulate CD w/o CD

\[ t \quad t+1 \]

assume: leader
Trick: Emulate CD w/o CD

For the emulation, double each time slot (runtime $\cdot 2$).
Trick: Emulate CD w/o CD

For the emulation, double each time slot (runtime \( \cdot 2 \)).

In the a-time-slots nodes transmit as usual.

Assume:

- In the time slots, nodes transmit as usual.
- For the emulation, double each time slot (runtime \( \cdot 2 \)).
Trick: Emulate CD w/o CD

For the emulation, double each time slot (runtime $\cdot 2$).

In the a-time-slots nodes transmit as usual.

In the b-time-slots nodes transmit as usual but also the leader!

assume: leader
Trick: Emulate CD w/o CD

Result: If idle in A-slot and busy in B-slot: $|S|=0$. If busy in both slots, $|S|>0$. So can distinguish whether sending set $S=\{}$ or $|S|>0$!

| $|S|$       | nodes in $S$ transmit | nodes in $S \cup \{\ell\}$ transmit |
|------------|------------------------|---------------------------------------|
| $|S|=0$    | $\times$               | $\checkmark$                          |
| $|S|=1, S=\{\ell\}$ | $\checkmark$             | $\checkmark$                          |
| $|S|=1, S \neq \{\ell\}$ | $\checkmark$             | $\times$                              |
| $|S| \geq 2$ | $\times$               | $\times$                              |

assume: leader

In the a-time-slots nodes transmit as usual.

In the b-time-slots nodes transmit as usual but also the leader!
Uniform Initialization (no CD)

1. Elect a leader
2. Divide every slot of the protocol with CD into two slots
   a) In the first slot, the nodes $S$ transmit according to the protocol
   b) In the second slot, the nodes $S$ from a) and the leader transmit
3. Distinguish the cases according to the table

   noise / silence : ✗
   successful transmission: ✓

| $|S| = 0$       | nodes in $S$ transmit | nodes in $S \cup \{\ell\}$ transmit |
|----------------|-----------------------|-------------------------------------|
| $|S| = 1, S = \{\ell\}$ | ✓                     | ✓                                   |
| $|S| = 1, S \neq \{\ell\}$ | ✓                     | ✗                                   |
| $|S| \geq 2$      | ✗                     | ✗                                   |

Leader brings CD to any protocol! E.g., RFID protocols. Overhead: factor 2 runtime.
Result

From our $O(n)$ runtime result w/ CD, by emulation we obtain:

**Uniform Init. w/o CD**

Given leader election, even without knowing the number of nodes and **without collision detection**, nodes can be initialized in time $O(n)$. 
Result

From our $O(n)$ runtime result w/ CD, by emulation we obtain:

**Uniform Init. w/o CD**

Given leader election, even without knowing the number of nodes and without collision detection, nodes can be initialized in time $O(n)$.

But how to elect leader if $n$ is unknown?
Result

From our $O(n)$ runtime result w/ CD, by emulation we obtain:

**Uniform Init. w/o CD**

Given leader election, even without knowing the number of nodes and without collision detection, nodes can be initialized in time $O(n)$.

But how to elect leader if $n$ is unknown?

Also: how long does it take to elect leader with high probability, not just on expectation?
Leader Election With High Probability

**With High Probability (whp.)**

An event happens with high probability if it occurs with $p \geq 1 - 1/n^c$ for some constant $c$. 
With High Probability (whp.)

An event happens with high probability if it occurs with $p \geq 1 - 1/n^c$ for some constant $c$.

Remember:

**Slotted Aloha**

```
repeat
    transmit with probability 1/n
until one node has transmitted alone
```
With High Probability (whp.)

An event happens with high probability if it occurs with $p \geq 1 - 1/n^c$ for some constant $c$.

Remember: Leader election with slotted Aloha in $O(1)$ on expectation. How long does it take w.h.p.?

Remember:

**Slotted Aloha**

repeat
  transmit with probability $1/n$
until one node has transmitted alone
With High Probability (whp.)

An event happens with high probability if it occurs with $p \geq 1 - \frac{1}{e}$ for some constant $c$.

Slotted Aloha

repeat
transmit with probability $1/n$
until one node has transmitted alone

Remember:

Leader election with slotted Aloha in $O(1)$ on expectation. How long does it take w.h.p.?

$log(n)$ many rounds: probability of not having a leader after $log n$ rounds:

$$\left(1 - \frac{1}{e}\right)^{e \ln n} = \left(1 - \frac{1}{e}\right)^{e \cdot c' \ln n} \leq \frac{1}{e^{\ln n \cdot c'}} = \frac{1}{n^{c'}}.$$
Slotted Aloha

Slotted Aloha solves leader election without CD but if \( n \) is known in \( O(\log n) \) rounds, whp.
Slotted Aloha

Slotted Aloha solves leader election without CD but if \( n \) is known in \( O(\log n) \) rounds, whp.

What if \( n \) is unknown?
Where do we stand?

**Slotted Aloha**

Slotted Aloha solves leader election without CD but if \( n \) is known in \( O(\log n) \) rounds, whp.

- **What if \( n \) is unknown?**
  
  Idea: nodes should just start with high sending probability, and decrease it over time till successful (single transmission).
Where do we stand?

**Slotted Aloha**

Slotted Aloha solves leader election without CD but if \( n \) is known in \( O(\log n) \) rounds, whp.

**What if \( n \) is unknown?**

Idea: nodes should just start with high sending probability, and decrease it over time till successful (single transmission).

It is easy to see: if the sum of sending probabilities is constant, then a leader will be elected with constant probability!
Where do we stand?

**Slotted Aloha**

Slotted Aloha solves leader election without CD but if \( n \) is known in \( O(\log n) \) rounds, whp.

What if \( n \) is unknown?

Idea: nodes should just start with high sending probability, and decrease it over time till successful (single transmission).

But how? Tradeoff: **Decrease too slow**: takes long until constant cumulative probability, if **decrease too fast** I may miss the „sweet spot“!

It is easy to see: if the sum of sending probabilities is constant, then a leader will be elected with constant probability!
Leader Election With Unknown \( n \)

**Decay Election**

\[
\text{for } k = 1, 2, 3, \ldots \text{ do} \\
\quad \text{for } i = 1 \text{ to } c \times k \text{ do} \\
\quad \quad \text{transmit with probability } p = \frac{1}{2^k} \\
\quad \quad \text{if } v \text{ is only sender then } v \text{ becomes leader} \\
\quad \text{end for} \\
\text{end for}
\]

- **Repetitions needed**
- **At the beginning:** \( p \) maybe too high and many collisions
Leader Election With Unknown n

Decay Election

for k = 1,2,3,... do
  for i= 1 to c*k do
    transmit with probability p=1/2^k
    if v is only sender then v becomes leader
  end for
end for

Analysis: After k \approx \log n rounds, p \approx 1/n, then we have c*k = c * \log n many repetitions with constant cumulative probability! We know that slotted Aloha solves leader election in this situation in \log n rounds, whp. So overall \log(n) \cdot \log(n) rounds, with high probability.
Decay Election

Decay Election solves uniform leader election without CD in $O(\log^2 n)$ rounds, whp.
Where do we stand?

Decay Election

Decay Election solves uniform leader election without CD in $O(\log^2 n)$ rounds, whp.

Much faster distributed algorithms exist!
For now, assume CD again. But $n$ still unknown!
**Where do we stand?**

**Decay Election**

Decay Election solves uniform leader election without CD in $O(\log^2 n)$ rounds, whp.

---

**Transmit or Keep Silent**

An algorithm based on «knock-out»/tournament:

```
repeat
    transmit with probability $p=1/2$
    if at least one node transmitted then
        everybody who did not: quit protocol
until single node transmits
```
Where do we stand?

Decay Election

Decay Election solves uniform leader election without CD in $O(\log^2 n)$ rounds, whp.

Much faster distributed algorithms exist!
For now, assume CD again. But $n$ still unknown!

Transmit or Keep Silent

An algorithm based on «knock-out»/tournament:

\begin{verbatim}
repeat
  transmit with probability $\frac{1}{2}$
  if at least one node transmitted then
    everybody who did not: quit protocol
until single node transmits
\end{verbatim}

\begin{verbatim}
learn via ACK
\end{verbatim}

\begin{verbatim}
~ half of the nodes will never transmit again
\end{verbatim}
Fast Uniform Leader Election (with CD)

**Transmit or Keep Silent**

repeat
   transmit with probability $p=1/2$
   if at least one node transmitted then
      everybody who did not: quit protocol
until single node transmits
Fast Uniform Leader Election (with CD)

Transmit or Keep Silent

repeat
    transmit with probability $p=1/2$
    if at least one node transmitted then
        everybody who did not: quit protocol
until single node transmits

Define **successful round**: at most half of the active nodes transmit. I.e., problem divided in half.
Fast Uniform Leader Election (with CD)

Transmit or Keep Silent

repeat
transmit with probability $p = 1/2$
if at least one node transmitted then
   everybody who did not: quit protocol
until single node transmits

Define **successful round**: at most half of the active nodes transmit. I.e., problem divided in half.

Successful round is likely: constant probability. Accordingly, $\log n$ rounds in total, even w.h.p. (Chernoff bounds).
Fast Uniform Leader Election (with CD)

Transmit or Keep Silent

repeat
    transmit with probability $p=1/2$
    if at least one node transmitted then
        everybody who did not: quit protocol
until single node transmits

Define **successful round**: at most half of the active nodes transmit. I.e., problem divided in half.

Successful round is likely: constant probability. Accordingly, log $n$ rounds in total, even w.h.p. (Chernoff bounds).

Transmit or Keep Silent solves uniform leader election without CD in $O(\log n)$ rounds, whp.
Super Fast Uniform Leader Election (with CD)
Super Fast Uniform Leader Election (with CD)

Idea:
1. Get a very rough estimation of the number of nodes \( n \) very fast
2. Get a more accurate estimation of the number of nodes (binary search)
3. Random walk to find constant approximation

At any point: stop if leader found 😊
Phase 1: super-exponential decrease until nobody transmits: $1/2^2, 1/2^4, 1/2^8, 1/2^{16}, ...$ finds raw estimate of $n \approx 2^i, i \approx 2^k$, i.e., of $2^{2^k}$

Phase 2:

1. $i := 1$
2. repeat
3. $i := 2 \cdot i$
4. transmit with probability $1/2^i$
5. until no node transmitted
6. $l := 2^{i-2}$
7. $u := 2^i$
8. while $l + 1 < u$ do
9. $j := \left\lceil \frac{l + u}{2} \right\rceil$
10. transmit with probability $1/2^j$
11. if no node transmitted then
12. $u := j$
13. else
14. $l := j$
15. end if
16. end while

{End of Phase 2}

Phase 3:

17. $k := u$
18. repeat
19. transmit with probability $1/2^k$
20. if no node transmitted then
21. $k := k - 1$
22. else
23. $k := k + 1$
24. end if
25. until exactly one node transmitted

{End of Phase 3}
Guess, guess, walk

Phase 1:
1: $i := 1$
2: repeat
3: $i := 2 \cdot i$
4: transmit with probability $1/2^i$
5: until no node transmitted
   {End of Phase 1}

Phase 2:
6: $l := 2^i - 2$
7: $u := 2^i$
8: while $l + 1 < u$ do
9:   $j := \lceil \frac{l + u}{2} \rceil$
10: transmit with probability $1/2^j$
11: if no node transmitted then
12:   $u := j$
13: else
14:   $l := j$
15: end if
16: end while
   {End of Phase 2}

Phase 3:
17: $k := u$
18: repeat
19:   transmit with probability $1/2^k$
20:   if no node transmitted then
21:     $k := k - 1$
22:   else
23:     $k := k + 1$
24:   end if
25: until exactly one node transmitted

Phase 1 super-exponential decrease until nobody transmits: $1/2^2, 1/2^4, 1/2^8, 1/2^{16}, ...$
finds raw estimate of $n \approx 2^i$, $i \approx 2^k$, i.e., of $2^{2^k}$

Phase 2 gets better estimate with binary search, $n \approx 2^j$
Guess, guess, walk

Phase 1:

1: $i := 1$
2: repeat
3: $i := 2 \cdot i$
4: transmit with probability $1/2^i$
5: until no node transmitted
{End of Phase 1}

Phase 2:

6: $l := 2^{i-2}$
7: $u := 2^i$
8: while $l + 1 < u$ do
9: $j := \lceil \frac{l+u}{2} \rceil$
10: transmit with probability $1/2^j$
11: if no node transmitted then
12: $u := j$
13: else
14: $l := j$
15: end if
16: end while
{End of Phase 2}

Phase 3:

17: $k := u$
18: repeat
19: transmit with probability $1/2^k$
20: if no node transmitted then
21: $k := k - 1$
22: else
23: $k := k + 1$
24: end if
25: until exactly one node transmitted

Phase 1 super-exponential decrease until nobody transmits: $1/2^2, 1/2^4, 1/2^8, 1/2^{16}, \ldots$
finds raw estimate of $n \approx 2^i$, $i \approx 2^k$, i.e., of $2^{2^k}$

Phase 2 gets better estimate with binary search, $n \approx 2^j$

Phase 3 finds constant approx of $n$ with random walk until single node transmits
Guess, guess, walk

Phase 1:

1. $i := 1$
2. repeat
3. $i := 2 \cdot i$
4. transmit with probability $1/2^i$
5. until no node transmitted
   {End of Phase 1}

Phase 2:

6. $l := 2^{i-2}$
7. $u := 2^i$
8. while $l + 1 < u$
9. $j := \left\lfloor \frac{l + u}{2} \right\rfloor$
10. transmit with probability $1/2^j$
11. if no node transmitted then
12. $u := j$
13. else
14. $l := j$
15. end if
16. end while
   {End of Phase 2}

Phase 3:

17. $k := u$
18. repeat
19. transmit with probability $1/2^k$
20. if no node transmitted then
21. $k := k - 1$
22. else
23. $k := k + 1$
24. end if
25. until exactly one node transmitted

Phase 1 super-exponential decrease until nobody transmits: $1/2^2, 1/2^4, 1/2^8, 1/2^{16}, \ldots$
finds upper estimate of $n^2 = 2^{2k}$, i.e., of $2^{2k}$

Phase 2 gets better estimate with binary search:
$n \approx 2^j$

Phase 3 finds constant approx of $n$ with random walk until single node transmits

Reduce quickly, so log log $n$ time
Binary search in log interval, so also log log $n$ time
As we will see, also log log $n$ time
Phase 1: super-exponential decrease until nobody transmits:
\[ \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots \]
finds raw estimate of \( n \approx 2^i \), i.e., of \( 2^{2^k} \)

Phase 2: better estimate with binary search:
\[ n \approx 2^j \]

Phase 3: finds constant approx of \( n \) with random walk until single node transmits.

Analysis: see lecture notes.
Backup Slides
Guess-Guess-Walk elects leader
- with probability at least $1 - \log\log(n)/\log(n)$
- in time $O(\log\log n)$.

Why?
Guess-Guess-Walk elects leader
- with probability at least $1 - \log\log(n)/\log(n)$
- in time $O(\log\log n)$.

Why?

We will show:
1. In Phase 1+2, we fail in each round (go to next phase too early/late) with probability $1/\log(n)$, so union bound over all rounds: overall we fail with probability $\log\log(n)/\log(n)$
2. Phase 1 terminates after $O(\log\log n)$ rounds
3. Phase 2 is a binary search on interval of size $O(\log n)$, hence terminates in $O(\log\log n)$ time: after the phase, our estimate of $\log n$ is at most $\log\log n$ away from the true value
4. Phase 3 requires $O(\log\log n)$ slots to elect a leader with probability $1 - 1/\log(n)$. 
Guess-Guess-Walk elects leader
- with probability at least $1 - \log\log(n) / \log(n)$
- in time $O(\log\log n)$.

Why?

We can show that with a small $\log\log(n)$ additive deviation from ideal sending probability, we almost always have busy or idle (CD detects)!

$$If \ j > \log n + \log \log n, \ then \ \Pr[X > 1] \leq \frac{1}{\log n}.$$  

$$If \ j < \log n - \log \log n, \ then \ P[X = 0] \leq \frac{1}{n}.$$
Lemma 1: For large enough $j$ when sending with probability $1/2^j$, unlikely to have many senders:

If $j > \log n + \log \log n$, then $P[X > 1] \leq 1/\log(n)$.

Proof. The nodes transmit with probability $1/2^j < 1/2^{\log n + \log \log n} = \frac{1}{n \log n}$. The expected number of nodes transmitting is $E[X] = \frac{n}{n \log n}$. Using Markov’s inequality (see Theorem 13.21) yields $Pr[X > 1] \leq Pr[X > E[X] \cdot \log n] \leq \frac{1}{\log n}$. \qed
Lemma 2: For small $j$, many nodes will send:

If $j < \log n - \log \log n$, then $P[X=0] \leq \frac{1}{n}$.

Proof. The nodes transmit with probability $\frac{1}{2^j} < \frac{1}{2^{\log n - \log \log n}} = \frac{\log n}{n}$. Hence, the probability for a silent time slot is $(1 - \frac{\log n}{n})^n = e^{-\log n} = \frac{1}{n}$. □
Lemma 3: Let \( v \) be such that \( 2^{v-1} < n \leq 2^v \), i.e., \( v \approx \log n \). If \( k > v+2 \), then \( P[X > 1] \leq \frac{1}{4} \).

Proof. Markov’s inequality yields

\[
Pr[X > 1] = Pr\left[X > \frac{2^k}{n} E[X]\right] < Pr\left[X > \frac{2^k}{2^v} E[X]\right] < Pr[X > 4E[X]] < \frac{1}{4}.
\]
Lemma 4: If $k < v - 2$, then $P[X=0] \leq 1/4$.

Proof. A similar analysis is possible to upper bound the probability that a transmission fails if our estimate is too small. We know that $k \leq v - 2$ and thus

$$Pr[X = 0] = \left(1 - \frac{1}{2^k}\right)^n < e^{-\frac{n}{2^k}} < e^{-\frac{2v-1}{2^k}} < e^{-2} < \frac{1}{4}.$$
Lemma 5: If $-2 \leq k \leq v+2$, then the probability that exactly one node transmits is constant.

Proof. The transmission probability is $p = \frac{1}{2^{v+\Theta(1)}} = \Theta(1/n)$, and the lemma follows with a slightly adapted version of Theorem 13.1.

\[\square\]
Lemma 6: With probability $1 - 1/\log(n)$, a leader is found in Phase 3 in $O(\log \log n)$ time.

Proof. For any $k$, because of Lemmas 13.13 and 13.14, the random walk of the third phase is biased towards the good area. One can show that in $O(\log \log n)$ steps one gets $\Omega(\log \log n)$ good transmissions. Let $Y$ denote the number of times exactly one node transmitted. With Lemma 13.15 we obtain $E[Y] = \Omega(\log \log n)$. Now a direct application of a Chernoff bound (see Theorem 13.22) yields that these transmissions elect a leader with probability $1 - \frac{1}{\log n}$. \qed
Guess-Guess-Walk elects leader
- with probability at least $1 - \frac{\log\log(n)}{\log(n)}$
- in time $O(\log\log n)$.

Comments:
- With a more detailed analysis, we can increase the success probability to $1 - \frac{1}{\log(n)}$
Even Faster Uniform Leader Election?

Uniform Lower Bound

Any uniform protocol which elects leader with probability at least $1 - 1/2^t$ must run for at least $t$ rounds.

Why?
Uniform Lower Bound

Any uniform protocol which elects leader with probability at least $1 - 1/2^t$ must run for at least $t$ rounds.

Why?

Claim already holds for $n=2$ nodes. (Hence also for $n>2$: if a network with $n>2$ nodes could find a leader quicker with higher probability then so could $n=2$ nodes: one node could simulate the rest of the network.)

For two nodes, they must reach a situation where exactly one transmits. Uniform, so nodes use same $p$ in each round (cannot adapt: same feedback). The corresponding probability is at most

$$P[X=1] = 2 \times p \times (1-p) \leq 1/2$$

Thus after time $t$, the election probability is at most $1 - 1/2^t$.

For $t=\log\log(n)$, Guess-Guess-Guess-Walk almost tight!
**Leader Election with Asynchronous Wakeup?**

**Recall:**

*Asynchronous Wakeup:* node start executing algorithm at different times

Assume uniform and anonymous: nodes do not know n, do not have an identifier, **execute the same code.**

**Observations:**

- At some point the nodes must transmit.
- Look at first time slot where some nodes transmit. They will do so with probability p, independent of n. (**Uniform**)

**Strategy for adversary to make runtime high?**
Leader Election with Asynchronous Wakeup?

Wakeup Lower Bound

Any uniform protocol has time complexity $\Omega(n/\log n)$ for leader election if nodes wake up arbitrarily.

Analysis.
Leader Election with Asynchronous Wakeup?

**Wakeup Lower Bound**

Any uniform protocol has time complexity $\Omega(n/\log n)$ for leader election if nodes wake up arbitrarily.

**Analysis.**

- Assume first transmission at time $t$, with probability $p$, independent of $n$.
- Adversary wakes up $w = c/p \ln(n)$ nodes in each time slot, for some constant $c$.
- First batch of $w$ nodes will transmit with probability $p$ \((\text{uniform} = \text{independent of } n)\).
- Probability that exactly one of them transmits in first time slot (event $E_1$)

$$P[E_1] = w \cdot p \cdot (1-p)^{w-1} < 1/n^c$$

- ... for $w = n/\log(n)$ it is polynomially small (whp. unsuccessful).
- Nodes cannot distinguish between noise and idle: so is the same for every batch of nodes that wakes up! (No CD: do not know whether too many or too few nodes.)
- So we have $n/w$ many time slots. Probability that none of those works out

$$P[E] = (1-P[E_1])^{n/w} > 1-1/n^c$$

for $w = n/\log(n)$, so does not work out whp.
Summary

**Leader Election**
How long does it take until one node can transmit alone?
- $e$ in expectation, knowing $n$
- $O(\log n)$ whp, (without) knowing $n$, no CD
- $O(\log \log n)$ without knowing $n$, with CD, with probability $1 - \log \log n / \log n$
- $1 - 1/\log n$ election probability lower bound for $\log \log n$

**Initialization**
How to assign IDs $\{1, 2, \ldots, n\}$?
- $O(n)$ with RandomizedSplit

**Asynchronous Wakeup**
How long for leader election if nodes wakeup up at arbitrary times?
- $\Omega(n/\log n)$ without IDs and without knowing $n$
End of Lecture