Network Algorithms

Wireless Protocols

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

Biggest advantage:
No wires ☺
=> fast installation
=> cheaper

Biggest disadvantage:
No wires ☺
=> attenuation
=> interference
=> energy supply

Big Question
To send or not to send?
Received signal power from sender

\[ \frac{P_u}{d(u,v)^\alpha} \geq \beta \]

Path-loss exponent

Noise

\[ N + \sum_{w \in V \setminus \{u\}} \frac{P_w}{d(w,v)^\alpha} \]

Received signal power from all other nodes (=interference)

Distance between two nodes

Minimum signal-to-interference ratio
Radio Network Model

All n nodes within transmission range. A node can only send xor receive, not both! Message can only be received if exactly one node sends at that time.

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.
Radio Network Model

Assume n nodes within transmission range.

I can:

- send XOR
- receive
- reach all other nodes
Radio Network Model

I can:

- send XOR receive

clique: reach all other nodes

two or more simultaneous transmissions collide
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!

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!

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

How to achieve leader election without CD in Radio Network Model?

- Assume that nodes have unique IDs
Leader Election without CD

How to achieve leader election without CD in Radio Network Model?

- Assume that nodes have unique IDs

- So, e.g., node with ID x should send x in round x, and we take max?

- Takes forever…

- What about randomization?
Slotted Aloha

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

ALOHAnet developed at Uni Hawaii 😊

- How long does it take until a node sends alone?
- 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
Leader Election without CD: Slotted Aloha

**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 then, how can the leader know its role?
Leader Election without CD: Slotted Aloha

**Slotted Aloha**

```markdown
repeat
  transmit with probability 1/n
until one node has transmitted 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 how can the node that sent the leader ID know the leader knows? (Needed so that protocol terminates!)

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).
Leader Election without CD: Slotted Aloha

**Slotted Aloha**

```markdown
repeat
    transmit with probability 1/n
until one node has transmitted 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 how can the node that sent the leader ID know the leader knows? (Needed so that protocol terminates!)

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).

**But assumption: Nodes perfectly synchronized wrt slots!**

How does it work in asynchronous environment?
Leader Election without CD: Unslotted Aloha

**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, …., n\}? Slots could be divided accordingly then!

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!

With and without collision detection, with and without asynchronous wakeup (nodes wakeup up at arbitrary times), ...?
Non-Uniform Initialization without CD

If nodes could choose IDs in \( \{1, \ldots, n\} \), they could transmit one by one without interference! How to achieve?

**Repeated Aloha**

\[
i = 1
\]

\[
\text{repeat}
\]

transmit with probability \( \frac{1}{n} \)

\[
\text{if node } v \text{ transmitted alone, } v \text{ gets ID } i \text{ (ACKed), then leaves: } i++, n--
\]

\[
\text{until all nodes have an ID}
\]

- Expected runtime?
- Other problems?
Non-Uniform Initialization without CD

If nodes could choose IDs in \{1, ..., n\}, they could transmit one by one without interference! How to achieve?

**Repeated Aloha**

\[
i = 1
\]

\[
\text{repeat}
\]

\[
\text{transmit with probability } 1/n
\]

\[
\text{if node } v \text{ transmitted alone, } v \text{ gets ID } i \text{ (ACKed), then leaves: } i++, n--
\]

\[
\text{until all nodes have an ID}
\]

Each ID assignment takes expected time \(e\)

\[
\Rightarrow \text{Total expected time } n*e = O(n)
\]

Problem also: nodes need to known \(n\)!
How to do fast uniform initialization? Let’s do with Collision Detection first!
Uniform Initialization with CD

Idea: Nodes find their unique «value» adaptively!

- Each node $v$ maintains and extends string $b_v$
- Initially, string is empty: $b_v = «»$
- As long as node did not send alone:
  - extend ID strings with random bits $r$
  - otherwise gets next available global identifier: $m$
- Try to divide remaining nodes with same bit string in two groups, one with $r=0$ and one with $r=1$
- Eventually, each node has its unique string $b_v$

(Note: do not transform unique string to ID but use global counter $m$: only then IDs between 1 and $n$ independently of «execution tree»)
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 \( m \); \( m++ \)
8. Else recursively split remaining nodes:
9. RandomizedSplit(\( b_0 \)), RandomizedSplit(\( b_1 \))
Successful: split nodes into 2 non-empty subsets

Analysis?

After getting an ID, node does not participate anymore! How does he know? Leader election first…
Split successful if nodes are divided in two non-empty sets.
How many splits are there?
Uniform Initialization with CD

Split successful if nodes are divided in two non-empty sets. How many splits are there? Always exactly n-1: we have n non-empty leaves and n-1 inner nodes.

So how long does it take to make a successful split (two non-empty sets)?
Split successful if nodes are divided in two non-empty sets.
How many splits are there?
Always exactly \( n-1 \): we have \( n \) non-empty leaves and \( n-1 \) inner nodes.

So how long does it take to make a successful split (two non-empty sets)?
Let RV \( X \) denote the size of the set.

\[
P[0 < X < k] = 1 - P[X=0] - P[X=k] = 1 - \frac{1}{2^k} - \frac{1}{2^k} \geq 0.5
\]

So per split \( O(1) \) rounds, so overall runtime \( O(n) \).
Uniform Initialization with CD

For node $v$:

**Main()**:  
1. $m = 0$ (* already identified nodes *)  
2. $b_v = 0$ (* 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 $b_v \neq b$, 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($b_1$)

Problems though:  
- How can successful sender know it is alone, and has ID?
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 \( b_v \neq b \), append \( r \) to my string \( b_v \), i.e., \( b_v := b_v \cdot r \)
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(\( b_1 \))

Problems though:
- How can successful sender know it is alone, and has ID?
  E.g., as with ALOHA: elect leader first that sends all ACKs, coming up soon 😊
Uniform Initialization with CD

Problems though:

- How can successful sender know it is alone, and has ID?
  E.g., as with ALOHA: elect leader first that sends all ACKs, coming up soon 😊
- How to do it without CD?
Uniform Initialization with CD

Problems though:
- How can successful sender know it is alone, and has ID?
  E.g., as with ALOHA: elect leader first that sends all ACKs, coming up soon 😊
- How to do it without CD?
  Need to distinguish whether sending set $S=\emptyset$ or $|S|>0$. Do leader election first and then do a trick! How?

```
For node $v$:
Main():
1. $m = 0$ (* already identified nodes *)
2. $b_v = «»$ (* current bitstring of node $v$ *)
3. RandomSplit($b_v$)

RandomSplit($b$):
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 $b_v \neq b$, 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. RandomSplit($b_0$), RandomSplit($b_1$)
```
Uniform Initialization with CD

For node $v$:

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

**RandomizedSplit($b$)**
1. Repeat
2. If $b_y = b$
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 $b_y = b$, append $r$ to my string $b_y$, i.e., $b_y := b_y r$
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($b_1$)

Emulate CD without CD:
Need to distinguish whether sending set $S=\{}$ or $|S|>0$.

Assume leader: how to emulate CD?
Emulate CD without CD:
Need to distinguish whether sending set $S=\{\}$ or $|S|>0$.

Assume leader: how to emulate CD?
Idea: divide each time slot $t$ in two slots, $t.A$ and $t.B$.
(Note: doubles runtime of algorithm.)
Uniform Initialization with CD

For node v:

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

RandomSplit(b)
1. Repeat
2. If \( b_0 = b \)
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 \( b_0 > b \), append \( r \) to my string \( b \), i.e., \( b_v = b_r \)
7. If single node u transmitted in slot \( r \), gets global ID \( m_m++ \)
8. Else recursively split remaining nodes:
9. RandomizedSplit(b0), RandomizedSplit(b1)

Emulate CD without CD:
Need to distinguish whether sending set \( S = \{\} \) or \(|S| > 0\).

Assume leader: how to emulate CD?
Idea: divide each time slot \( t \) in two slots, \( t.A \) and \( t.B \).
(Note: doubles runtime of algorithm.)
A-slot transmissions as usual (protocol spec), in B-slot leader always transmits.
Uniform Initialization with CD

**Main()**
1. \( m = 0 \) (* already identified nodes *)
2. \( b_v = « » \) (* 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 \( r \) and listen in \( 1+r \mod 2 \)
5. Until there was at least one transmission in both slots (two groups, requires collision detection)
6. If \( b_v \neq b \), append \( r \) to my string \( b_v \), i.e., \( b_v \leftarrow b_v \cdot r \)
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(\( b1 \))

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

| \( |S| \) | \( S \) |
|---|---|
| 0 | \( S=\{ \} \) |
| 1, \( S = \{ \ell \} \) | \( |S| = 1 \) |
| 1, \( S \neq \{ \ell \} \) | \( |S| = 1 \) |
| \( |S| \geq 2 \) | \( |S|>0 \) |
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

<table>
<thead>
<tr>
<th>noise / silence</th>
<th>successful transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
</tr>
</tbody>
</table>

Four cases can be distinguished.
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 $O(n)$.

But how to elect a leader in the uniform model?
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 $c$.

Slotted Aloha

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

Example: Leader election with slotted Aloha in $O(1)$ on expectation. But how long whp.?
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 $c$.

**Slotted Aloha**

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

Example: Leader election with slotted Aloha in $O(1)$ on expectation. But how long whp.? log(n) many rounds: probability of not having a leader after log n rounds:

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

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

Idea for uniform model?
Uniform Leader Election (without CD)

Ideas? Remember: uniform = do not know n

Plan: Decrease probability over time!
- For example, decrease exponentially: $p = \frac{1}{2^k}$
- At some point, cumulative probability will be constant
- Then constant probability of single transmission
- But make sure do not decrease too fast from then on...
Uniform Leader Election (without CD)

Ideas? Remember: uniform = do not know n

Plan: Decrease probability over time!
- For example, decrease exponentially: \( p = \frac{1}{2^k} \)
- At some point, cumulative probability will be constant
- Then constant probability of single transmission
- But make sure do not decrease too fast from then on...

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

Decay Election
Uniform Leader Election (no CD)

Decay Election

for k = 1, 2, 3, ... do
  for i = 1 to c*k do
    transmit with probability \( p = \frac{1}{2^k} \)
    if v is only sender then v becomes leader
  end for
end for

Analysis.

Repetitions needed

At the beginning: \( p \) too high and many collisions
Uniform Leader Election (no CD)

**Decay Election**

for $k = 1, 2, 3, \ldots$ do
  for $i = 1$ to $c \cdot k$ 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 log $n$ many rounds 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.
Uniform Leader Election With High Probability

Decay Election

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

Can we do faster?
Like in $\log n$ time?
Or even $\log\log n$ time?
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

Analysis.
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

~ half of the nodes will never transmit again

learn via ACK

---

**Analysis.**

1. Number of active nodes decreases monotonically, but always $\geq 1$.
2. Define successful round (SR): at most half of active nodes transmit
3. Assume $k \geq 2$ and let $X$ be number of active nodes. Then

$$Pr[1 \leq X \leq \left\lceil \frac{k}{2} \right\rceil] \geq \frac{1}{2} - Pr[X = 0] = \frac{1}{2} - \frac{1}{2^k} \geq \frac{1}{4}. $$

Constant probability for successful round!

1. So runtime $\log n$: $\log n$ SR suffice for leader election on expectation.
2. Even holds whp. (Chernoff bounds)
Transmit or Keep Silent

Transmit or Keep Silent solves uniform leader election without CD in $O(\log n)$ rounds, whp.

Even faster possible?
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 😊
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 😊

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: $i := 2^{i-2}$
7: $u := 2^i$
8: while $l + 1 < u$ do
9: $j := \lfloor \frac{i+u}{2} \rfloor$
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
Super Fast Uniform Leader Election (with CD)

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: \( \hat{i} := 2^{i-2} \)
7: \( u := 2^i \)
8: while \( \hat{l} + 1 < u \) do
9: \( j := \lceil \frac{\hat{l} + u}{2} \rceil \)
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 is fast: 
\( 1/2^2, 1/2^4, 1/2^8, 1/2^{16}, \ldots \)
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, \( n \approx 2^i \)

Phase 3 finds constant approx of \( n \) with random walk until single node transmits
Super Fast Uniform Leader Election (with CD)

Guess, guess, walk

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

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

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

Phase 1 is fast:
- \(1/2^2, 1/2^4, 1/2^8, 1/2^{16}, \ldots\)
- 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, \( n \approx 2^i \)

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

Runtime?
Super Fast Uniform Leader Election (with CD)

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: \( \hat{i} := 2^{i-2} \)
7: \( u := 2^i \)
8: while \( l + 1 < u \) do
9: \( j := \lfloor \frac{l+u}{2} \rfloor \)
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 is fast:
\( 1/2, 1/2^2, 1/2^3, \ldots \)

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

Phase 3 finds constant approx of \( n \) with random walk:
so also \( \log \log n \) time

As we will see, so also \( \log \log n \) time

Reduce quickly:
so \( \log \log n \) time

Binary search in log interval

so also \( \log \log n \) time
Guess Guess Walk

Guess-Guess-Walk elects leader
- with probability at least $1 - \frac{\log \log(n)}{\log(n)}$
- 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

Guess-Guess-Walk elects leader
- with probability at least $\frac{1}{\log \log(n)}$ / $\log(n)$
- 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):

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

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

Proof. The nodes transmit with probability $1/2^j < 1/2^{\log n - \log \log n} = \frac{\log n}{n}$. 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 \( P[X > 1] \leq 1/4 \).

Proof. Markov’s inequality yields

\[
Pr[X > 1] = Pr \left[ X > \frac{2^k}{n} E[X] \right] < Pr \left[ X > \frac{2^k}{2^v} E[X] \right] < Pr \left[ X > 4E[X] \right] < \frac{1}{4}.
\]
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

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

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

Comments:
- With a more detailed analysis, we can increase the success probability to $1 - 1/\log(n)$
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:

**Aynchronous 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 $w = \frac{c}{p} \ln(n)$ nodes in each time slot, for some constant $c$.
- First batch of $w$ nodes will transmit with probability $p$ (uniform = independent of $n$)
- Probability that exactly one of them transmits in first time slot (event $E_1$)

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

- ... for $w = \frac{n}{\log(n)}$ 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

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

for $w = \frac{n}{\log(n)}$, 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