Network Computing and Efficient Algorithms

Wireless Protocals

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Wireless Networks

- **Wireless Networks Models**
  - Geometric graph models
    - unit disk graph.
  - Restricted network graph
    - the total number of neighbors of a node which are not adjacent is small.
- **Biggest Advantage (no wires)**
  - Fast installation
  - Cheaper
- **Biggest Disadvantage (no wires)**
  - Attenuation
  - Interference
  - Energy supply

- **Wireless Networks Models**
  - Geometric graph models
    - unit disk graph.
  - Restricted network graph
    - the total number of neighbors of a node which are not adjacent is small.
- **Biggest Advantage (no wires)**
  - Fast installation
  - Cheaper
- **Biggest Disadvantage (no wires)**
  - Attenuation
  - Interference
  - Energy supply

Assumptions: Clique; Synchronous.
Question: To send or not to send?

Leader Election

Algorithm 12.1 Slotted Aloha() ;

Every node v executes the following code:
repeat
    transmit with probability $1/n$
until one node has transmitted alone

Leader Election

ALGORITHM 12.1 SLOTTED ALOHA() ;

1: **Every node** v executes the following code:
2: **repeat**
3: transmit with probability $1/n$
4: **until** one node has transmitted alone

Theorem 12.2.

Using Algorithm 12.1 allows one node to transmit alone (become a leader) after expected time e.

Leader Election

ALGORITHM 12.1 SLOTTED ALOHA() ;

1: **Every node** v executes the following code:
2: **repeat**
3: transmit with probability $1/n$
4: **until** one node has transmitted alone

Theorem 12.2.

Using Algorithm 12.1 allows one node to transmit alone (become a leader) after expected time e.

Proof. The probability for success, *i.e.*, only one node transmitting is

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

where the last approximation is a result from Theorem 12.29 for sufficiently large n. Hence, if we repeat this process e times, we can expect one success.

(Theorem 12.29: $\lim_{n\to\infty}(1+\frac{t}{n})^n = e^t$)

Leader Election

But then, **How can the leader know its role?**
The nodes start sending the ID of the leader with $1/n$

But **how can the node that sent the leader ID know the leader knows?**
The leader sends an acknowledgement to this node

But then, **How can the leader know its role?**
The nodes start sending the ID of the leader with $1/n$

But **how can the node that sent the leader ID know the leader knows?**
The leader sends an acknowledgement to this node

$\Longrightarrow$ **Distributed ACK**

Initialization

- Sometimes we want the n nodes to have the IDs 1, 2, ... , n. This process is called initialization.
- Initialization can for instance be used to allow the nodes to transmit one by one without any interference.

Initialization

Initialization

- Sometimes we want the n nodes to have the IDs 1, 2, ... , n. This process is called initialization.
- Initialization can for instance be used to allow the nodes to transmit one by one without any interference.

Theorem 12.3.

If the nodes know n, we can initialize them in $O(n)$ time slots.

Initialization

- Sometimes we want the n nodes to have the IDs 1, 2, ... , n. This process is called initialization.
- Initialization can for instance be used to allow the nodes to transmit one by one without any interference.

Theorem 12.3.

If the nodes know n, we can initialize them in $O(n)$ time slots.

Proof. We repeatedly elect a leader using $e.g.$, Algorithm 12.1. The leader gets the nest free number and afterwards leaves the process. We know that this works with probability $1/e$. The expected time to finish is hence $e \cdot n$.

Initialization

- Sometimes we want the n nodes to have the IDs 1, 2, ... , n. This process is called initialization.
- Initialization can for instance be used to allow the nodes to transmit one by one without any interference.

Theorem 12.3.

If the nodes know n, we can initialize them in $O(n)$ time slots.

Proof. We repeatedly elect a leader using $e.g.$, Algorithm 12.1. The leader gets the nest free number and afterwards leaves the process. We know that this works with probability $1/e$. The expected time to finish is hence $e \cdot n$.

For a more realistic scenario, we need a uniform algorithm.

Uniform Initialization with CD

Definition 10.4 (Collision Detection, CD).

Two or more nodes transmitting concurrently is called interference. In a system with collision detection, **a receiver can distinguish interference from nobody transmitting.** In a system without collision detection. a receiver cannot distinguish the two caese.

Uniform Initialization with CD

Definition 10.4 (Collision Detection, CD).

Two or more nodes transmitting concurrently is called interference. In a system with collision detection, **a receiver can distinguish interference from nobody transmitting.** In a system without collision detection. a receiver cannot distinguish the two caese.

The main idea of the algorithm is to **partition nodes iteratively into sets. Each set is identified by a label (a bitstring)**, and by storing one such bitstring, each node knows in which set it currently is. Initially, all nodes are in a single set, identified by the empty bitstring. This set is then partitioned into two non-empty sets, identified by '0' and '1'. In the same way, all sets are iteratively partitioned into two non-empty sets, as long as a set contains more than one node. If a set contains only a single node, this node receives the nest free ID. The algorithm terminates once every node is alone in its set. **Note that this partitioning process iteratively creates a binary tree which has exactly one node in the set at each leaf, and has n leaves.**

Initialization with Collision Detection

Algorithm 12.5 Initialization with Collision Detection ()

```
Every ndoe v executes the following code:
nestId ← 0
myBitstring ← "  "                              ▷ Initialize to empty string
bitstringToSplit ← ["  "]                       ▷ a queue with sets to split
while bitstringToSplit is not empty do
    b ← bitstringToSplit.pop()
    repeat
        if b = myBitString then
            choose r uniformly at random from {0,1}
            in the next two time slots:
            transmit in slot r, and listen in other slot
        else
            it is not my bitsrting, just listen in both slots
    until there was at least 1 transmission in both slots
    if b = myBitstring then
        myBitstring ← myBitsrting + r                ▷ append bit r
    for r ∈ {0,1} do
        if some node u transmitted alone in slot r then
            node u becomes ID nextId and becomes passive
            nextId ← nextId + 1
        else
            bitstringToSplit.push(b + r)
```

Uniform Initialization with CD

THeorem 12.6

Algorithm 12.5 correctly initializes n nodes in expected time $O(n)$

Uniform Initialization with CD

THeorem 12.6

Algorithm 12.5 correctly initializes n nodes in expected time $O(n)$

Proof. A successful split is defined as a split in which both subsets are non-empty. We know that there are exactly $n-1$ successful splits because we have a binary tree with n leaves and $n-1$ inner nodes. Let us now calculate the probability for creating two non-empry sets from a set of size $k \geq 2$ as

$$Pr[1 \leq X \leq k-1] = 1 - Pr[X=0] - Pr[X=k] = 1 - \frac{1}{2^k} - \frac{1}{2^k} \geq \frac{1}{2}.$$

Thus, in expectation we need $O(n)$ splits.

Uniform Initialization with CD

THeorem 12.6

Algorithm 12.5 correctly initializes n nodes in expected time $O(n)$

Proof. A successful split is defined as a split in which both subsets are non-empty. We know that there are exactly $n-1$ successful splits because we have a binary tree with n leaves and $n-1$ inner nodes. Let us now calculate the probability for creating two non-empry sets from a set of size $k \geq 2$ as

$$Pr[1 \leq X \leq k-1] = 1 - Pr[X=0] - Pr[X=k] = 1 - \frac{1}{2^k} - \frac{1}{2^k} \geq \frac{1}{2}.$$

Thus, in expectation we need $O(n)$ splits.

What if we do not have collision detection?

Uniform Initialization with CD

Let us assume that we have a special node l (leader) and let S denote the set of nodes which want to transmit. We now split every time slot from Algorithm 12.5 into two time slots and use the leader to help us distinguish between silence and noise. In the first slot every node from the set S transmits, in the second slot the nodes in $S \cup \{l\}$ transmit. This gives the nodes sufficient information to distinguish the different cases(see Table below).

	nodes in S transmit	nodes in $S \cup \{l\}$ transmit
$\lvert S\rvert = 0$	$\times$	$\checkmark$
$\lvert S\rvert = 1, S = \{l\}$	$\checkmark$	$\checkmark$
$\lvert S\rvert = 1, S \neq \{l\}$	$\checkmark$	$\times$
$\lvert S\rvert \geq 2$	$\times$	$\times$

Table: Using a leader to distinguish between noise and silence: $\times$ represents noise/silence, $\checkmark$ represents a successful transmission.

Uniform Initialization with CD

Let us assume that we have a special node l (leader) and let S denote the set of nodes which want to transmit. We now split every time slot from Algorithm 12.5 into two time slots and use the leader to help us distinguish between silence and noise. In the first slot every node from the set S transmits, in the second slot the nodes in $S \cup \{l\}$ transmit. This gives the nodes sufficient information to distinguish the different cases(see Table below).

	nodes in S transmit	nodes in $S \cup \{l\}$ transmit
$\lvert S\rvert = 0$	$\times$	$\checkmark$
$\lvert S\rvert = 1, S = \{l\}$	$\checkmark$	$\checkmark$
$\lvert S\rvert = 1, S \neq \{l\}$	$\checkmark$	$\times$
$\lvert S\rvert \geq 2$	$\times$	$\times$

Table: Using a leader to distinguish between noise and silence: $\times$ represents noise/silence, $\checkmark$ represents a successful transmission.

A leader immediately brings CD to any protocol

Definition 12.8 (With High Probability).

Some probabilistic event is said to occur with high probability ($w.h.p.$), if it happens with a probability $p \geq 1 - 1/n^c$, where c is a constant. The constant c mat be chosen arbitrarily, but it is considered constant with respect to Big-O notation.

Leader Election

Definition 12.8 (With High Probability).

Some probabilistic event is said to occur with high probability ($w.h.p.$), if it happens with a probability $p \geq 1 - 1/n^c$, where c is a constant. The constant c mat be chosen arbitrarily, but it is considered constant with respect to Big-O notation.

Theorem 12.9

Algorithm 12.1 elects a leader $w.h.p.$ in $O(\log n)$ time slots.

Definition 12.8 (With High Probability).

Some probabilistic event is said to occur with high probability ($w.h.p.$), if it happens with a probability $p \geq 1 - 1/n^c$, where c is a constant. The constant c mat be chosen arbitrarily, but it is considered constant with respect to Big-O notation.

Theorem 12.9

Algorithm 12.1 elects a leader $w.h.p.$ in $O(\log n)$ time slots.

Proof. The probability for not electing a leader after $c \cdot \log n$ time slots, $i.e.$, $c \log n$ slots without a successful transmission is

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

Leader Election

Definition 12.8 (With High Probability).

Some probabilistic event is said to occur with high probability ($w.h.p.$), if it happens with a probability $p \geq 1 - 1/n^c$, where c is a constant. The constant c mat be chosen arbitrarily, but it is considered constant with respect to Big-O notation.

Theorem 12.9

Algorithm 12.1 elects a leader $w.h.p.$ in $O(\log n)$ time slots.

Proof. The probability for not electing a leader after $c \cdot \log n$ time slots, $i.e.$, $c \log n$ slots without a successful transmission is

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

What about uniform algorithms?

Uniform Leader Election

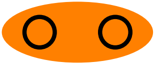
ALGORITHM 12.10 UNIFORM LEADER ELECTION()

```
Every node v executes the following code:
for k ← 1, 2, 3, ... do
    for i ← 1 to c · k do
        transmit with probability p ← 1/2^k
        if node v was the only node which transmitted then
            v becomes the leader
            break
```

Uniform Leader Election

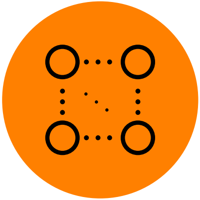
Theorem 12.11

By using Algorithm 12.10 it is possible to elect a leader *w.h.p.* in $O(\log^2 n)$ time slots if n is not known.



Round 1.
Each node transmit at probability $1/2$ for c times.
Estimate n is 2.

⋮



Round k.
each node transmit at probability $1/2$ for c times
Estimate n is 2^k.

⋮

Uniform Leader Election

Theorem 12.11

By using Algorithm 12.10 it is possible to elect a leader $w.h.p.$ in $O(\log^2 n)$ time slots if n is not known.

Proof. Let us briefly describe the algorithm. The nodes transmit with probability $p = 2^{-k}$ for ck time slots for $k = 1, 2, \ldots$. At first p will be too high and hence there will be a lot of interference. But after $\log n$ phases, we have $k \approx logn$ and thus the nodes transmit with probability $\approx \frac{1}{n}$. For simplicity's sake, let us assume that n is a power of 2. Using the approach outlined above, we konw that after $\log n$ iterations, we have $p = \frac{1}{n}$. Theorem 12.9 yields that we can elect a leader $w.h.p.$ in $O(\log n)$ slots. Since we have to try $\log n$ estimates until $k \approx n$, the total runtime is $O(\log^2 n)$.

Uniform Leader Election

Theorem 12.11

By using Algorithm 12.10 it is possible to elect a leader $w.h.p.$ in $O(\log^2 n)$ time slots if n is not known.

Proof. Let us briefly describe the algorithm. The nodes transmit with probability $p = 2^{-k}$ for ck time slots for $k = 1, 2, \ldots$. At first p will be too high and hence there will be a lot of interference. But after $\log n$ phases, we have $k \approx logn$ and thus the nodes transmit with probability $\approx \frac{1}{n}$. For simplicity's sake, let us assume that n is a power of 2. Using the approach outlined above, we konw that after $\log n$ iterations, we have $p = \frac{1}{n}$. Theorem 12.9 yields that we can elect a leader $w.h.p.$ in $O(\log n)$ slots. Since we have to try $\log n$ estimates until $k \approx n$, the total runtime is $O(\log^2 n)$.

- Algorithm 12.10 has not used collision detection. Can we solve leader election faster in a uniform setting with collision detection?

Fast Leader Election with CD

Algorithm 12.12 Uniform leader election with CD()

```
Every node v executes the following code:
repeat
    transmit with probability 1/2
    if at least one node transmitted then
        all nodes that did not transmit quit the protocal
until one node transmits alone
```

Fast Leader Election with CD

Algorithm 12.12 Uniform leader election with CD()

```
Every node v executes the following code:
repeat
    transmit with probability 1/2
    if at least one node transmitted then
        all nodes that did not transmit quit the protocal
until one node transmits alone
```

Theorem 12.13

With collision detection we can elect a leader using Algorithm 12.12 *w.h.p.* in $O(\log n)$ time slots.

Proof of Theorem 12.13

Proof. The number of active nodes k is monotonically decreasing and always greater than 1 which yeilds the correctness. A slot is called successful if at most half the active nodes transmit. We can assume that $k \geq 2$ since otherwise we would have already elected a leader. We can calculate the probability that a time silt is successful as

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

Since the number of active nodes at least halves in every successful time slot, $\log n$ successful time slots are sufficient to elect a leader. Now let Y be a random variable which counts the number of successful time slots after $8 \cdot c \cdot \log n$ time solts. The expected value is $E[Y] \geq 8 \cdot c \cdot \log n \cdot \frac{1}{4} \geq 2 \cdot c \cdot \log n$. Since all those time slots are independent form each other, we can apply a Chernoff bound (see Theorem 12.27) with $\delta = \frac{1}{2}$ which states

$$Pr[Y < (1-\delta)E[Y]] \leq e^{-\frac{\delta^2}{2}E[Y]} \leq e^{-\frac{1}{8}\cdot 2c\log n} \leq n^{-\alpha}$$

for any constant α

Even Faster Leader Election with CD

ALGORITHM 12.14 EVEN FASTER LEADER ELECTION WITH CD()

$i \leftarrow 1$
repeat
    $i \leftarrow 2 \cdot i$
    transmit with probability $1/2^i$
until no node transmitted ▷ End of Phase 1
$l \leftarrow 2^{i/2}$, $u \leftarrow 2^i$
while $l+1 < u$ **do**
    $j \leftarrow \lceil \frac{l+u}{2} \rceil$
    transmit with probability $1/2^j$
    if no node transmitted **then**
        $u \leftarrow j$
    else
        $l \leftarrow j$ ▷ End of Phase 2
$k \leftarrow u$
repeat
    transmit with probability $1/2^k$
    if no node transmitted **then**
        $k \leftarrow k-1$
    else
        $k \leftarrow k+1$
until exactly one node transmitted

Even Faster Leader Election with CD

ALGORITHM 12.14 EVEN FASTER LEADER ELECTION WITH CD()

```
i ← 1
repeat
    i ← 2·i
    transmit with probability 1/2^i
until no node transmitted                          ▷ End of Phase 1
l ← 2^(i/2),    u ← 2^i
while l+1 < u do
    j ← ⌈(l+u)/2⌉
    transmit with probability 1/2^j
    if no node transmitted then
        u ← j
    else
        l ← j                                      ▷ End of Phase 2
k ← u
repeat
    transmit with probability 1/2^k
    if no node transmitted then
        k ← k−1
    else
        k ← k+1
until exactly one node transmitted
```

Theorem 12.23.

The Algorithm 12.14 elects a leader with probability of at least $1-\frac{\log\log n}{\log n}$ in time $O(\log\log n)$.

Lower Bound

Theorem 12.24.

Any uniform protocal that elects a leader with probability of at least $1 - \frac{1}{2^t}$ must run for at least t time slots.

Lower Bound

Theorem 12.24.

Any uniform protocal that elects a leader with probability of at least $1 - \frac{1}{2^t}$ must run for at least t time slots.

Proof. Consider a system with only 2 nodes. The probability that exactly one transmits is at most

$$Pr[X = 1] = 2p \cdot (1 - p) \leq \frac{1}{2}.$$

Thus, after t time slots the probability that a leader was elected is at most $1 - \frac{1}{2^t}$.

Lower Bound

Theorem 12.24.

Any uniform protocal that elects a leader with probability of at least $1 - \frac{1}{2^t}$ must run for at least t time slots.

Proof. Consider a system with only 2 nodes. The probability that exactly one transmits is at most

$$Pr[X = 1] = 2p \cdot (1 - p) \leq \frac{1}{2}.$$

Thus, after t time slots the probability that a leader was elected is at most $1 - \frac{1}{2^t}$.

- Setting $t = \log \log n$ shows that Algorithm 12.14 is almost tight.

Uniform Asynchronous Wakeup without CD

Theorem 12.25

If nodes wake up in an arbitrary (worst-case) way, any algorithm may take $\Omega(n/\log n)$ time slots until a single node can successfuylly transmit

Uniform Asynchronous Wakeup without CD

Theorem 12.25

If nodes wake up in an arbitrary (worst-case) way, any algorithm may take $\Omega(n/\log n)$ time slots until a single node can successfuylly transmit

Uniform $\Rightarrow$ all nodes executed the same code
At some point the nodes must transmit.

First transmission at time t, with probability p independent of n
Adversary wakes up $w = \frac{c}{p}\ln n$ nodes in each slot

$Pr[E_1] = P[X = 1 \text{ at time } t] < \frac{1}{n^{c-1}} = \frac{1}{n^{c'}}$.
$P[X \neq 1 \text{ at time } t \text{ and the following } n/w \text{ time slots}]$
$= (1 - Pr(E_1))^{n/w} > (1 - \frac{1}{n^{c'}})^{\Theta(n/\log n)} > 1 - \frac{1}{n^{c''}}$.