Chapter 10

Wireless Protocols

Wireless communication was one of the major success stories of the last decades. Today, different wireless standards such as wireless local area networks (WLAN) are omnipresent. In some sense, from a distributed computing viewpoint wireless networks are quite simple, as they cannot form arbitrary network topologies. Simplistic models of wireless networks include geometric graph models such as the so-called unit disk graph. Modern models are more robust: The network graph is restricted, e.g., the total number of neighbors of a node which are not adjacent is likely to be small. This observation is hard to capture with purely geometric models, and motivates more advanced network connectivity models such as bounded growth or bounded independence.

However, on the other hand, wireless communication is also more difficult than standard message passing, as for instance nodes are not able to transmit a different message to each neighbor at the same time. And if two neighbors are transmitting at the same time, they interfere, and a node may not be able to decipher anything.

In this chapter we deal with the distributed computing principles of wireless communication: We make the simplifying assumption that all n nodes are in the communication range of each other, i.e., the network graph is a clique. Nodes share a synchronous time, in each time slot a node can decide to either transmit or receive (or sleep). However, two or more nodes transmitting in a time slot will cause interference. Transmitting nodes are never aware if there is interference because they cannot simultaneously transmit and receive.

10.1 Basics

The basic communication protocol in wireless networks is the medium access control (MAC) protocol. Unfortunately it is difficult to claim that one MAC protocol is better than another, because it all depends on the parameters, such as the network topology, the channel characteristics, or the traffic pattern. When it comes to the principles of wireless protocols, we usually want to achieve much simpler goals. One basic and important question is the following: How long does it take until one node can transmit successfully, without interference? This question is often called the wireless leader election problem (Chapter 2), with the node transmitting alone being the leader.

Clearly, we can use node IDs to solve leader election, e.g., a node with ID i transmits in time slot i. However, this may be incredibly slow. There are better deterministic solutions, but by and large the best and simplest algorithms are randomized.

Throughout this chapter, we use a random variable X to denote the number of nodes transmitting in a given slot.

Algorithm 38 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 10.1. *Using Algorithm 38 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

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

where the last approximation is a result from Theorem 10.23 for sufficiently large n. Hence, if we repeat this process e times, we can expect one success. □

Remarks:

- The origin of the name is the ALOHAnet which was developed at the University of Hawaii.
- How does the leader know that it is the leader? One simple solution is a "distributed acknowledgment". The nodes just continue Algorithm 38, including the ID of the the leader in their transmission. So the leader learns that is the leader.
- One more problem?! Indeed, node v which managed to transmit the acknowledgment (alone) is the only remaining node which does not know that the leader knows that it is the leader. We can fix this by having the leader acknowledge v's successful acknowledgment.
- One can also imagine an unslotted time model. In this model two messages which overlap partially will interfere and no message is received. As everything in this chapter, Algorithm 38 also works in an unslotted time model, with a factor 2 penalty, i.e., the probability for a successful transmission will drop from $\frac{1}{e}$ to $\frac{1}{2e}$. Essentially, each slot is divided into t small time slots with $t \to \infty$ and the nodes start a new t-slot long transmission with probability $\frac{1}{2nt}$.

10.2 Initialization

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

10.2.1 Non-Uniform Initialization

Theorem 10.2. *If the nodes know n, we can initialize them in $\mathcal{O}(n)$ time slots.*

Proof. We repeatedly elect a leader using e.g., Algorithm 38. The leader gets the next free number and afterwards leaves the process. We know that this works with probability $1/e$. The expected time to finish is hence $e \cdot n$. □

Remarks:

- But this algorithm requires that the nodes know n in order to give them IDs from $1, \dots, n$! For a more realistic scenario we need a uniform algorithm, i.e, the nodes do not know n.

10.2.2 Uniform Initialization with CD

Definition 10.3 (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 cases.*

Let us first present a high-level idea. The set of nodes is recursively partitioned into two non-empty sets, similarly to a binary tree. This is repeated recursively until a set contains only one node which gets the next free ID. Afterwards, the algorithm continues with the next set.

Algorithm 39 RandomizedSplit(b)

```
Every node $v$ executes the following code:
repeat
    if $b_v = b$ then
        choose $r$ uniformly at random from $\{0, 1\}$
        in the next two time slots:
        transmit in slot $r$, and listen in other slot
    end if
until there was at least 1 transmission in both slots
if $b_v = b$ then
    $b_v := b_v + r$ {append bit $r$ to bitstring $b_v$}
end if
for $r \in \{0, 1\}$ do
    if some node $u$ transmitted alone in slot $r$ then
        node $u$ gets ID $m$ {and becomes passive}
        $m := m + 1$
    else
        RandomizedSplit($b + r$)
    end if
end for
```

Remarks:

- In line 8 the transmitting nodes need to know if they were the only one transmitting. Since we have enough time, we can do a leader election first and use a similar trick as before to ensure this. Or we can add a round in which nodes transmit a message after they hear a node transmitting alone.

Algorithm 40 Initialization with Collision Detection

```
1: **Every node** $v$ executes the following code:
2: global variable $m := 0$ {number of already identified nodes}
3: local variable $b_v :=$ ‘’ {current bitstring of node $v$, initially empty}
4: RandomizedSplit(‘’)
```

Theorem 10.4. *Algorithm 40 correctly initializes the set of nodes in $\mathcal{O}(n)$.*

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-empty 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 $\mathcal{O}(n)$ splits. □

Remarks:

- What if we do not have collision detection?

10.2.3 Uniform Initialization without 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 before 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 $S \cup \{\ell\}$ transmit. This gives the nodes sufficient information to distinguish the different cases (see Table 10.1).

	nodes in S transmit	nodes in $S \cup \{\ell\}$ transmit
$\lvert S\rvert = 0$	X	✓
$\lvert S\rvert = 1, S = \{\ell\}$	✓	✓
$\lvert S\rvert = 1, S \neq \{\ell\}$	✓	X
$\lvert S\rvert \geq 2$	X	X

Table 10.1: Using a leader to distinguish between noise and silence: X represents noise/silence, ✓ represents a successful transmission.

Remarks:

- As such, Algorithm 40 works also without CD, with only a factor 2 overhead.
- More generally, a leader immediately brings CD to any protocol.
- This protocol has an important real life application, for instance when checking out a shopping cart with items which have RFID tags.
- But how do we determine such a leader? And how long does it take until we are "sure" that we have one? Let us repeat the notion of *with high probability.*

10.3 Leader Election

10.3.1 With High Probability

Definition 10.5 (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 may be chosen arbitrarily, but it is considered constant with respect to Big-O notation.*

Theorem 10.6. *Algorithm 38 elects a leader w.h.p. in* $\mathcal{O}(\log n)$ *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

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

Remarks:

- What about uniform algorithms, i.e. the number of nodes n is not known?

10.3.2 Uniform Leader Election

Algorithm 41 Uniform leader election

```
1: Every node v executes the following code:
2: for $k = 1, 2, 3, \ldots$ do
3:   for $i = 1$ to $ck$ do
4:     transmit with probability $p := 1/2^k$
5:     if node $v$ was the only node which transmitted then
6:       $v$ becomes the leader
7:       break
8:     end if
9:   end for
10: end for
```

Theorem 10.7. *By using Algorithm 41 it is possible to elect a leader w.h.p. in* $\mathcal{O}(\log^2 n)$ *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 $k \approx \log n$ 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 know that after $\log n$ iterations, we have $p = \frac{1}{n}$. Theorem 10.6 yields that we can elect a leader w.h.p. in $\mathcal{O}(\log n)$ slots. Since we have to try $\log n$ estimates until $k \approx n$, the total runtime is $\mathcal{O}(\log^2 n)$. □

Remarks:

- Note that our proposed algorithm has not used collision detection. Can we solve leader election faster in a uniform setting with collision detection?

10.3.3 Fast Leader Election with CD

Algorithm 42 Uniform leader election with CD

```
1: Every node v executes the following code:
2: repeat
3:   transmit with probability $\frac{1}{2}$
4:   if at least one node transmitted then
5:     all nodes that did not transmit quit the protocol
6:   end if
7: until one node transmits alone
```

Theorem 10.8. *With collision detection we can elect a leader using Algorithm 42 w.h.p. in $\mathcal{O}(\log n)$ time slots.*

Proof. The number of active nodes k is monotonically decreasing and always greater than 1 which yields 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 slot is successful as

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

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 slots. The expected value is $E[Y] \geq 8 \cdot c \cdot \log n \cdot \frac{1}{4} \geq 2 \cdot \log n$. Since all those time slots are independent from each other, we can apply a Chernoff bound (see Theorem 10.22) with $\delta = \frac{1}{2}$ which states

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

for any constant α. □

Remarks:

- Can we be even faster?

10.3.4 Even Faster Leader Election with CD

Let us first briefly describe an algorithm for this. In the first phase the nodes transmit with probability $1/2^{2^0}, 1/2^{2^1}, 1/2^{2^2}, \ldots$ until no node transmits. This yields a first approximation on the number of nodes. Afterwards, a binary search is performed to determine an even better approximation of n. Finally, the third phase finds a constant approximation of n using a biased random walk. The algorithm stops in any case as soon as only one node is transmitting which will become the leader.

Algorithm 43 Fast uniform leader election

```
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}
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}
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
```

Lemma 10.9. *If $j > \log n + \log\log n$, then $Pr[X > 1] \le \frac{1}{\log n}$.*

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 10.21) yields $Pr[X > 1] \le Pr[X > E[X] \cdot \log n] \le \frac{1}{\log n}$. □

Lemma 10.10. *If $j < \log n - \log\log n$, then $P[X = 0] \le \frac{1}{n}$.*

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 $(1 - \frac{\log n}{n})^n = e^{-\log n} = \frac{1}{n}$. □

Corollary 10.11. *If $i > 2\log n$, then $Pr[X > 1] \leq \frac{1}{\log n}$.*

Proof. This follows from Lemma 10.9 since the deviation in this corollary is even larger. □

Corollary 10.12. *If $i < \frac{1}{2}\log n$, then $P[X = 0] \leq \frac{1}{n}$.*

Proof. This follows from Lemma 10.10 since the deviation in this corollary is even larger. □

Lemma 10.13. *Let v be such that $2^{v-1} < n \leq 2^v$, i.e., $v \approx \log n$. If $k > v+2$, then $Pr[X > 1] \leq \frac{1}{4}$.*

Proof. Markov's inequality yields

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

□

Lemma 10.14. *If $k < v - 2$, then $P[X = 0] \leq \frac{1}{4}$.*

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 10.15. *If $v - 2 \leq k \leq v + 2$, then the probability that exactly one node transmits is constant.*

Proof. The transmission probability is $p = \frac{1}{2^{v \pm \Theta(1)}} = \Theta(1/n)$, and the lemma follows with a slightly adapted version of Theorem 10.1.

□

Lemma 10.16. *With probability $1 - \frac{1}{\log n}$ we find a leader in phase 3 in $O(\log\log n)$ time.*

Proof. For any k, because of Lemmas 10.13 and 10.14, 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 10.15 we obtain $E[Y] = \Omega(\log\log n)$. Now a direct application of a Chernoff bound (see Theorem 10.22) yields that these transmissions elect a leader with probability $1 - \frac{1}{\log n}$. □

Theorem 10.17. *The Algorithm 43 elects a leader with probability of at least $1 - \frac{\log\log n}{\log n}$ in time $\mathcal{O}(\log\log n)$.*

Proof. ¿From Corollary 10.11 we know that after $\mathcal{O}(\log\log n)$ time slots, the first phase terminates. Since we perform a binary search on an interval of size $\mathcal{O}(\log n)$, the second phase also takes at most $\mathcal{O}(\log\log n)$ time slots. For the third phase we know that $\mathcal{O}(\log\log n)$ slots are sufficient to elect a leader with probability $1 - \frac{1}{\log n}$ by Lemma 10.16. Thus, the total runtime is $\mathcal{O}(\log\log n)$.

Now we can combine the results. We know that the error probability for every time slot in the first two phases is at most $\frac{1}{\log n}$. Using a union bound (see Theorem 10.20), we can upper bound the probability that no error occurred by $\frac{\log\log n}{\log n}$. Thus, we know that after phase 2 our estimate is at most $\log\log n$ away from $\log n$ with probability of at least $1 - \frac{\log\log n}{\log n}$. Hence, we can apply Lemma 10.16 and thus successfully elect a leader with probability of at least $1 - \frac{\log\log n}{\log n}$ (again using a union bound) in time $\mathcal{O}(\log\log n)$.

□

Remarks:

- Tightening this analysis a bit more, one can elect a leader with probability $1 - \frac{1}{\log n}$ in time $\log\log n + o(\log\log n)$.
- Can we be even faster?

10.3.5 Lower Bound

Theorem 10.18. *Any uniform protocol that elects a leader with probability of at least* $1 - \frac{1}{\log n}$ *must run for at least* $\log\log n$ *time slots.*

Proof. The probability that exactly one node transmits is

$$Pr[X = 1] = n \cdot p \cdot (1-p)^{n-1}.$$

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

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

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

10.3.6 Uniform Asynchronous Wakeup without CD

Until now we have assumed that all nodes start the algorithm in the same time slot. But what happens if this is not the case? How long does it take to elect a leader if we want an uniform and anonymous (nodes do not have an identifier and thus cannot base their decision on it) algorithm?

Theorem 10.19. *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 successfully transmit.*

Proof. Nodes must transmit at some point, or they will surely never successfully transmit. With a uniform protocol, every node executes the same code. We focus on the first slot where nodes may transmit. No matter what the protocol is, this happens with probability p. Since the protocol is uniform, p must be a constant, independent of n.

The adversary wakes up $w = \frac{c}{p}\ln n$ nodes in each time slot with some constant c. All nodes woken up in the first time slot will transmit with probability p. We study the event E_1 that exactly one of them transmits in that first time slot. Using the inequality $(1+t/n)^n \leq e^t$ from Lemma 10.23 we get

$$\begin{aligned} Pr[E_1] &= w \cdot p \cdot (1-p)^{w-1} \\ &= c\ln n\,(1-p)^{\frac{1}{p}(c\ln n - p)} \\ &\leq c\ln n \cdot e^{-c\ln + p} \\ &= c\ln n \cdot n^{-c}e^{p} \\ &= n^{-c}\cdot \mathcal{O}(\log n) \\ &< \frac{1}{n^{c-1}} = \frac{1}{n^{c'}}. \end{aligned}$$

In other words, w.h.p. that time slot will not be successful. Since the nodes cannot distinguish noise from silence, the same argument applies to every set of nodes which wakes up. Let E_α be the event that all n/w time slots will not be successful. Using the inequality $1-p \leq (1-p/k)^k$ from Lemma 10.24 we get

$$Pr[E_\alpha] = (1 - Pr(E_1))^{n/w} > \left(1 - \frac{1}{n^{c'}}\right)^{\Theta(n/\log n)} > 1 - \frac{1}{n^{c''}}.$$

In other words, w.h.p. it takes more than n/w time slots until some node can transmit alone.

□

10.4 Useful Formulas

In this chapter we have used several inequalities in our proofs. For simplicity's sake we list all of them in this section.

Theorem 10.20. *Boole's inequality or union bound: For a countable set of events $E_1, E_2, E_3, \ldots$, we have*

$$Pr[\bigcup_i E_i] \leq \sum_i Pr[E_i].$$

Theorem 10.21. *Markov's inequality: If X is any random variable and $a > 0$, then*

$$Pr[|X| \geq a] \leq \frac{E[X]}{a}.$$

Theorem 10.22. *Chernoff bound: Let $Y_1, \ldots, Y_n$ be a independent Bernoulli random variables let $Y := \sum_i Y_i$. For any $0 \leq \delta \leq 1$ it holds*

$$Pr[Y < (1-\delta)E[Y]] \leq e^{-\frac{\delta^2}{2}E[Y]}$$

and for $\delta > 0$

$$Pr[Y \geq (1+\delta) \cdot E[Y]] \leq e^{-\frac{\min\{\delta,\delta^2\}}{3} \cdot E[Y]}$$

Theorem 10.23. *We have*

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

for all $n \in \mathbb{N}, |t| \leq n$. *Note that*

$$\lim_{n\to\infty}\left(1+\frac{t}{n}\right)^n = e^t.$$

Theorem 10.24. *For all* p, k *such that* $0 < p < 1$ *and* $k \geq 1$ *we have*

$$1 - p \leq (1 - p/k)^k.$$