Chapter 13

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 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.

13.1 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.
13.2. INITIALIZATION

13.2.1 Non-Uniform Initialization

Theorem 13.2. If the nodes know \( n \), we can initialize them in \( O(n) \) time slots.

Proof. We repeatedly elect a leader using e.g., Algorithm 50. 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, \ldots, \( n \). For a more realistic scenario we need a uniform algorithm, i.e, the nodes do not know \( n \).

13.2.2 Uniform Initialization with CD

Definition 13.3 (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.

Algorithm 51 RandomizedSplit(b)

1: Every node \( v \) executes the following code:
2: repeat
3: if \( b_v = b \) then
4: choose \( r \) uniformly at random from \( \{0, 1\} \)
5: in the next two time slots:
6: transmit in slot \( r \), and listen in other slot
7: end if
8: until there was at least 1 transmission in both slots
9: if \( b_v = b \) then
10: \( b_v := b_v + r \) \{append bit \( r \) to bitstring \( b_v \)\}
11: end if
12: if some node \( u \) transmitted alone in slot \( r \in \{0, 1\} \) then
13: node \( u \) gets ID \( m \) \{and becomes passive\}
14: \( m := m + 1 \)
15: else
16: RandomizedSplit(b + r)
17: end if

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.

Algorithm 52 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 13.4. Algorithm 52 correctly initializes the set of nodes in \( O(n) \).

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 \( O(n) \) splits.

Remarks:
- What if we do not have collision detection?

13.2.3 Uniform Initialization without CD

Let us assume that we have a special node \( \ell \) (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 noise and nothing. 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 13.1).

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

Table 13.1: Using a leader to distinguish between noise and silence: \( \checkmark \) represents noise/silence, \( \checkmark \) represents a successful transmission.

Remarks:
- As such, Algorithm 52 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.
13.3 LEADER ELECTION

- 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.

13.3 Leader Election

13.3.1 With High Probability

Definition 13.5 (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 13.6. Algorithm 50 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

\[
\left(1 - \frac{1}{n^c}\right)^{\log n} = \left(1 - \frac{1}{n^c}\right)^{c \log n} \leq \frac{1}{e^{c \cdot \log n}} = \frac{1}{n^c}.
\]

Remarks:
- What about uniform algorithms, i.e. the number of nodes \( n \) is not known?

13.3.2 Uniform Leader Election

Algorithm 53 Uniform leader election

1. Every node \( v \) executes the following code:
2. for \( k = 1, 2, 3, \ldots \) do
3. for \( i = 1 \) to \( c \cdot k \) 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 13.7. By using Algorithm 53 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 \( k \) 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 \( k = \log n \) phases, we have \( k \approx \log n \) and thus the nodes transmit with probability \( \approx \frac{1}{2} \). 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 \( p = \frac{1}{2} \). Theorem 13.6 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 \( O(\log^2 n) \).

Remarks:
- Can we be even faster?

13.3.3 Fast Leader Election with CD

Algorithm 54 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 13.8. With collision detection we can elect a leader using Algorithm 54 w.h.p. in \( O(\log n) \) 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 \( 2 \cdot \log n \) time slots are sufficient to elect a leader. Now let \( Y \) be a random variable which counts the number of successful time slots after \( 2 \cdot \log n \) time slots. The expected value is

\[
E[Y] \approx 8 \cdot 2 \cdot \log n \cdot \frac{1}{2} = 8 \cdot 2 \cdot \log n.
\]

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 \( 2 \cdot 2 \cdot \log n \) time slots. The expected value is

\[
E[Y] \geq 8 \cdot 2 \cdot \log n \cdot \frac{1}{2} = 2 \cdot \log n.
\]

Since all those time slots are independent from each other, we can apply a Chernoff bound (see Theorem 13.22) with \( \delta = \frac{1}{2} \) which states

\[
Pr[Y \leq (1 - \delta)E[Y]] \leq e^{-\frac{\delta^2}{2}E[Y]} = e^{-\frac{1}{4} \cdot 2 \cdot \log n} \leq n^{-\alpha}
\]

for any constant \( \alpha \).

Remarks:
- Can we be even faster?
13.3. LEADER ELECTION

13.3.4 Even Faster Leader Election with CD

Let us first briefly describe an algorithm for this. In the first phase the nodes transmit with probability \(1/2^i\), \(1/2^j\), \(1/2^k\), \(...\), 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 55 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: \(l := 2^{i-1}\)
7: \(u := 2^{i+1}\)
8: while \(l+1 < u\) do
9: \(j := \left\lfloor \frac{l+u}{2} \right\rfloor\)
10: transmit with probability \(1/2^l\)
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 13.9. If \(j > \log n + \log \log n\), then \(\Pr[X > 1] \leq \frac{1}{n^j}\).

Proof. The nodes transmit with probability \(1/2^i < 1/2^{\log n + \log \log n} = \frac{1}{n^j}\).

The expected number of nodes transmitting is \(E[X] = \frac{1}{n^j}\). Using Markov’s inequality (see Theorem 13.21) yields \(\Pr[X > 1] \leq \frac{E[X]}{\log \log n} \leq \frac{1}{n^j}\).

Lemma 13.10. If \(j < \log n - \log \log n\), then \(\Pr[X = 0] \leq \frac{1}{4}\).

Proof. The nodes transmit with probability \(1/2^i < 1/2^{\log n - \log \log n} = \frac{1}{2^n}\).

Hence, the probability for a silent time slot is \((1 - 1/2^n)^n = e^{-1/2^n} = \frac{1}{4}\).

Corollary 13.11. If \(i > 2 \log n\), then \(\Pr[X > 1] \leq \frac{1}{n^i}\).

Proof. This follows from Lemma 13.9 since the deviation in this corollary is even larger.

Corollary 13.12. If \(i < \frac{1}{2} \log n\), then \(\Pr[X = 0] \leq \frac{1}{4}\).

Proof. This follows from Lemma 13.10 since the deviation in this corollary is even larger.

Lemma 13.13. 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

\[\Pr[X > 1] = \Pr[X > \frac{2k}{n} E[X]] \leq \frac{E[X]}{2} \leq \frac{1}{4}\]

Lemma 13.14. If \(k < v - 2\), then \(\Pr[X = 0] \leq \frac{1}{4}\).

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

\[\Pr[X = 0] = \left(1 - \frac{1}{2^n}\right)^n < e^{-\frac{n}{2^n}} < e^{-\frac{1}{4^n}} < \frac{1}{4}\]

Lemma 13.15. If \(v - 2 \leq k \leq v + 2\), then the probability that exactly one node transmits is constant.

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

Lemma 13.16. With probability \(1 - \frac{1}{n^2}\) we find a leader 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 these transmissions elect a leader with probability \(1 - \frac{1}{n^2}\).
13.3. LEADER ELECTION

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 13.20), we can upper bound the probability that no error occurred by $\frac{\log n}{n}$. Thus, we know that after phase 2 our estimate is at most $\log n$ away from $\log n$ with probability of at least $1 - \frac{\log n}{n}$. Hence, we can apply Lemma 13.16 and thus successfully elect a leader with probability of at least $1 - \frac{\log n}{n}^2$ (again using a union bound) in time $O(\log \log n)$. \hfill \Box

Remarks:

- Tightening this analysis a bit more, one can elect a leader with probability $1 - \frac{1}{\log n}^{1/2}$ in time $\log \log n + o(\log n)$.
- Can we be even faster?

13.3.5 Lower Bound

Theorem 13.18. 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

$$1 - \frac{1}{2 \log \log n}.$$ \hfill \Box

13.3.6 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 a uniform and anonymous (nodes do not have an identifier and thus cannot base their decision on it) algorithm?

Theorem 13.19. 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 $w = \frac{2}{3} \ln n$ 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 $E$ that exactly one of them transmits in that first time slot. Using the inequality $(1 + t/n)^n \leq e^t$ from Lemma 13.23 we get

$$Pr[E] = w \cdot p \cdot (1 - p)^{w-1} = c \cdot n \cdot (1 - p)^{c - 1} \cdot e^p \leq c \cdot n \cdot e^{-c/n} \cdot e^p = n^{-c} \cdot \Theta(\log n) \leq \frac{1}{e^c}.$$ 

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 $E_n$ 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 13.24 we get

$$Pr[E_n] = (1 - Pr(E))^{n/k} > \left(1 - \frac{1}{e^c}\right)^{\Theta(n/\log n)} > 1 - \frac{1}{e^c}.$$ 

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

13.4 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 13.20. 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 13.21. Markov’s inequality: If $X$ is any random variable and $a > 0$, then

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

Theorem 13.22. Chernoff bound: Let $Y_1, \ldots, Y_n$ be independent Bernoulli random variables let $Y := \sum Y_i$. For any $0 \leq \delta \leq 1$ it holds

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

and for $\delta > 0$

$$Pr[Y \geq (1 + \delta) \cdot E[Y]] \leq e^{-(1/2/\delta^2) \cdot E[Y]}.$$
Theorem 13.23. 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 13.24. For all \( p, k \) such that \( 0 < p < 1 \) and \( k \geq 1 \) we have
\[ 1 - p \leq (1 - p/k)^k. \]

Chapter Notes

The Aloha protocol is presented and analyzed in [Abr70, BAK+75, Abr85]; the basic technique that unslotted protocols are twice as bad a slotted protocols is from [Rob75]. The idea to broadcast in a packet radio network by building a tree was first presented in [TM78, Cap79]. This idea is also used in [JNO98] to initialize the nodes. Willard [Wil86] was the first that managed to elect a leader in \( O(\log \log n) \) time in expectation. Looking more carefully at the success rate, it was shown that one can elect a leader with probability \( 1 - \frac{1}{\log n} \) in time \( \log \log n + o(\log \log n) \) [NO98]. Finally, approximating the number of nodes in the network is analyzed in [JKZ02, CGK05]. The lower bound for probabilistic wake-up is published in [JS02]. In addition to single-hop networks, multi-hop networks have been analyzed, e.g. broadcast [BYGI92, KM98, CR06], or deployment [MvRW06].

This chapter was written in collaboration with Philipp Brandes.

Bibliography


