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 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 the same 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, 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 13.1 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 13.2. *Using Algorithm 13.1 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

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

where the last approximation is a result from Theorem 13.29 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 13.1, including the ID of the the leader in their transmission. So the leader learns that it 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 13.1 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 $\frac{1}{2nt}$.

13.2   Non-Uniform 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.

Theorem 13.3. *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 13.1. 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 $e \cdot n$.

$\square$

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.

13.3   Uniform Initialization with CD

Definition 13.4 (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 cases.*

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 next 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 thus has n leaves.

Algorithm 13.5 Initialization with Collision Detection

```
 1: Every node v executes the following code:
 2: nextId := 0
 3: myBitstring := ''                    ◁ initialize to empty string
 4: bitstringsToSplit := ['']            ◁ a queue with sets to split

 5: while bitstringsToSplit is not empty do
 6:   b := bitstringsToSplit.pop()

 7:   repeat
 8:     if b = myBitstring then
 9:       choose r uniformly at random from {0, 1}
10:       in the next two time slots:
11:         transmit in slot r, and listen in other slot
12:     else
13:       it is not my bitstring, just listen in both slots
14:     end if
15:   until there was at least 1 transmission in both slots
16:   if b = myBitstring then
17:     myBitstring := myBitstring + r        ◁ append bit r
18:   end if

19:   for r ∈ {0, 1} do
20:     if some node u transmitted alone in slot r then
21:       node u becomes ID nextId and becomes passive
22:       nextId := nextId + 1
23:     else
24:       bitstringsToSplit.push(b + r)
25:     end if
26:   end for
27: end while
```

Remarks:

- In Line 20 a transmitting node needs to know whether it was the only one transmitting. This is achievable in several ways, for instance by adding an acknowledgement round. To notify a node v that it has transmitted alone in round r, every node that was silent in round r sends an acknowledgement in round $r + 1$, while v is silent. If v hears a message or interference in $r + 1$, it knows that it transmitted alone in round r.

Theorem 13.6. *Algorithm 13.5 correctly initializes n nodes in expected time $\mathcal{O}(n)$.*

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?

13.4　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 Algorithm 13.5 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 $S \cup \{\ell\}$ transmit. This gives the nodes sufficient information to distinguish the different cases (see Table 13.7).

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

Table 13.7: Using a leader to distinguish between noise and silence: ✗ represents noise/silence, ✔ represents a successful transmission.

Remarks:

- As such, Algorithm 13.5 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*.

13.5　Leader Election

Definition 13.8 (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 may be chosen arbitrarily, but it is considered constant with respect to Big-O notation.*

Theorem 13.9. *Algorithm 13.1 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

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

□

Remarks:

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

Algorithm 13.10 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 13.11. *By using Algorithm 13.10 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 13.9 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 $2^k \approx n$, 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?

13.6　Fast Leader Election with CD

Algorithm 13.12 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 13.13. *With collision detection we can elect a leader using Algorithm 13.12 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

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

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 slots. 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 from each other, we can apply a Chernoff bound (see Theorem 13.28) 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.

□

Remarks:

- Can we be even faster?

- Let us first briefly describe an algorithm for this.

- In the first phase the nodes transmit with probability $1/2^{2^0}$, $1/2^{2^1}$, $1/2^{2^2}$, ... 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.

Lemma 13.15. *If $j > \log n + \log \log n$, then $Pr[X > 1] \leq \frac{1}{\log n}$.*

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 13.27) yields $Pr[X > 1] \leq Pr[X > E[X] \cdot \log n] \leq \frac{1}{\log n}$.

□

Lemma 13.16. *If $j < \log n - \log \log n$, then $P[X = 0] \leq \frac{1}{n}$.*

Proof. The nodes transmit with probability $1/2^j > 1/2^{\log n - \log \log n} = \frac{\log n}{n}$. Thus, the probability that a node is silent is at most $1 - \frac{\log n}{n}$. Hence, the probability for a silent time slot, i.e., $Pr[X = 0]$, is at most $(1 - \frac{\log n}{n})^n = e^{-\log n} = \frac{1}{n}$.

□

Algorithm 13.14 Fast uniform leader election
```
 1: i := 1
 2: repeat
 3:     i := 2 · i
 4:     transmit with probability 1/2^i
 5: until no node transmitted
    {End of Phase 1}
 6: l := i/2
 7: u := i
 8: while l + 1 < u do
 9:     j := ⌈(l+u)/2⌉
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
```

Corollary 13.17. *If $i > 2 \log n$, then $Pr[X > 1] \leq \frac{1}{\log n}$.*

Proof. This follows from Lemma 13.15 since the deviation in this corollary is even larger.

□

Corollary 13.18. *If $i < \frac{1}{2} \log n$, then $P[X = 0] \leq \frac{1}{n}$.*

Proof. This follows from Lemma 13.16 since the deviation in this corollary is even larger.

□

Lemma 13.19. *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

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

□

Lemma 13.20. *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

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

$\square$

Lemma 13.21. *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 13.2. $\square$

Lemma 13.22. *With probability $1 - \frac{1}{\log n}$ we find a leader in Phase 3 in $\mathcal{O}(\log \log n)$ time.*

Proof. For any k, because of Lemmas 13.19 and 13.20, the random walk of the third phase is biased towards the good area. One can show that in $\mathcal{O}(\log \log n)$ steps one gets $\Omega(\log \log n)$ good transmissions. Let Y denote the number of times exactly one node transmitted. With Lemma 13.21 we obtain $E[Y] = \Omega(\log \log n)$. Now a direct application of a Chernoff bound (see Theorem 13.28) yields that these transmissions elect a leader with probability $1 - \frac{1}{\log n}$. $\square$

Theorem 13.23. *The Algorithm 13.14 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 13.17 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 13.22. 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 13.26), we can upper bound the probability that an error occurred during these phases 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 13.22 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 $\mathcal{O}(\log \log n)$. $\square$

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?

13.7   Lower Bound

Theorem 13.24. *Any uniform protocol 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 $1 - \frac{1}{2^t}$. $\square$

Remarks:

- Setting $t = \log \log n$ shows that Algorithm 13.14 is almost tight.

13.8   Uniform Asynchronous Wakeup

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 13.25. *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 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 13.29 we get

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

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 $1 - p \le (1 - p/k)^k$ from Lemma 13.30 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.

$\square$

13.9 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 13.26. *Boole's inequality or union bound: For a countable set of events $E_1, E_2, E_3, \ldots$, we have*

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

Theorem 13.27. *Markov's inequality: If X is any random variable and $a > 0$, then*

$$Pr[|X| \ge a] \le \frac{E[X]}{a}.$$

Theorem 13.28. *Chernoff bound: Let $Y_1, \ldots, Y_n$ be a independent Bernoulli random variables let $Y := \sum_i Y_i$. For any $0 \le \delta \le 1$ it holds*

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

and for $\delta > 0$

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

Theorem 13.29. *We have*

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

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

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

Theorem 13.30. *For all p, k such that $0 < p < 1$ and $k \ge 1$ we have*

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

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 [HNO99] to initialize the nodes. Willard [Wil86] was the first that managed to elect a leader in $\mathcal{O}(\log \log n)$ 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, BKK+16]. 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

[Abr70] Norman Abramson. THE ALOHA SYSTEM: another alternative for computer communications. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 281–285, 1970.

[Abr85] Norman M. Abramson. Development of the ALOHANET. *IEEE Transactions on Information Theory*, 31(2):119–123, 1985.

[BAK+75] R. Binder, Norman M. Abramson, Franklin Kuo, A. Okinaka, and D. Wax. ALOHA packet broadcasting: a retrospect. In *American Federation of Information Processing Societies National Computer Conference (AFIPS NCC)*, 1975.

[BKK+16] Philipp Brandes, Marcin Kardas, Marek Klonowski, Dominik Pajk, and Roger Wattenhofer. Approximating the Size of a Radio Network in Beeping Model. In *23rd International Colloquium on Structural Information and Communication Complexity, Helsinki, Finland*, July 2016.

[BYGI92] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the Time-Complexity of Broadcast in Multi-hop Radio Networks: An Exponential Gap Between Determinism and Randomization. *J. Comput. Syst. Sci.*, 45(1):104–126, 1992.

[Cap79] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Trans. Inform. Theory*, 25(5):505–515, 1979.

[CGK05] Ioannis Caragiannis, Clemente Galdi, and Christos Kaklamanis. Basic Computations in Wireless Networks. In *International Symposium on Algorithms and Computation (ISAAC)*, 2005.

[CR06] Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. *J. Algorithms*, 60(2):115–143, 2006.

[HNO99] Tatsuya Hayashi, Koji Nakano, and Stephan Olariu. Randomized Initialization Protocols for Packet Radio Networks. In *13th International Parallel Processing Symposium & 10th Symposium on Parallel and Distributed Processing (IPPS/SPDP)*, 1999.

[JKZ02] Tomasz Jurdzinski, Miroslaw Kutylowski, and Jan Zatopianski. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Computing and Combinatorics (COCOON)*, 2002.

[JS02] Tomasz Jurdzinski and Grzegorz Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *International Symposium on Algorithms and Computation (ISAAC)*, 2002.

[KM98] Eyal Kushilevitz and Yishay Mansour. An Omega(D log (N/D)) Lower Bound for Broadcast in Radio Networks. *SIAM J. Comput.*, 27(3):702–712, 1998.

[MvRW06] Thomas Moscibroda, Pascal von Rickenbach, and Roger Wattenhofer. Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks. In *25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Barcelona, Spain*, April 2006.

[NO98] Koji Nakano and Stephan Olariu. Randomized O (log log n)-Round Leader Election Protocols in Packet Radio Networks. In *International Symposium on Algorithms and Computation (ISAAC)*, 1998.

[Rob75] Lawrence G. Roberts. ALOHA packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, April 1975.

[TM78] B. S. Tsybakov and V. A. Mikhailov. Slotted multiaccess packet broadcasting feedback channel. *Problemy Peredachi Informatsii*, 14:32–59, October - December 1978.

[Wil86] Dan E. Willard. Log-Logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM J. Comput.*, 15(2):468–477, 1986.