Appeared in ALGOSENSORS-2009.

Near-Optimal Radio Use For Wireless Network Synchronization*

Milan Bradonjić[1], Eddie Kohler[2], and Rafail Ostrovsky[3]**

[1] Theoretical Division, and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. Email: milan@lanl.gov
[2] Computer Science Department, University of California Los Angeles, CA 90095, USA. Email: kohler@cs.ucla.edu
[3] Computer Science Department and Department of Mathematics, University of California Los Angeles, CA 90095, USA. Email: rafail@cs.ucla.edu

Abstract In this paper we consider the model of communication where wireless devices can either switch their radios off to save energy (and hence, can neither send nor receive messages), or switch their radios on and engage in communication. The problem has been extensively studied in practice, in the setting such as deployment and clock synchronization of wireless sensor networks – see, for example, [31,41,33,29,40]. The goal in these papers is different from the classic problem of radio broadcast, i.e. avoiding interference. Here, the goal is instead to minimize the use of the radio for both transmitting and receiving, and for most of the time to shut the radio down completely, as the radio even in listening mode consumes a lot of energy.

We distill a clean theoretical formulation of minimizing radio use and present near-optimal solutions. Our base model ignores issues of communication interference, although we also extend the model to handle this requirement. We assume that nodes intend to communicate periodically, or according to some time-based schedule. Clearly, perfectly synchronized devices could switch their radios on for exactly the minimum periods required by their joint schedules. The main challenge in the deployment of wireless networks is to *synchronize* the devices' schedules, given that their initial schedules may be offset relative to one another (even if their clocks run at the same speed). In this paper we study how frequently the devices must switch on their radios in order to both synchronize their clocks and communicate. In this setting, we significantly improve previous results, and show optimal use of the radio for two processors and near-optimal use of the radio for synchronization of an arbitrary number of processors. In particular, for two processors we prove **deterministic** matching $\Theta(\sqrt{n})$ upper and lower bounds on the number of times the radio has to be on, where n is the discretized uncertainty period of the

* Full version of the paper is available on-line [8].

** The third author was supported in part by IBM Faculty Award, Xerox Innovation Group Award, NSF grants 0430254, 0716835, 0716389, 0830803 and U.C. MICRO grant.

clock shift between the two processors. (In contrast, all previous results for two processors are randomized, e.g. [33], [29]).
For $m = n^\beta$ processors (for any positive $\beta < 1$) we prove $\Omega(n^{(1-\beta)/2})$ is the lower bound on the number of times the radio has to be switched on (per processor), and show a nearly matching (in terms of the radio use) $\tilde{O}(n^{(1-\beta)/2})$ randomized upper bound per processor, (where $\tilde{O}$ notation hides *poly-log*(n) multiplicative term) with failure probability exponentially close to 0. For $\beta \geqslant 1$ our algorithm runs with at most *poly-log*(n) radio invocations per processor. Our bounds also hold in a radio-broadcast model where interference must be taken into account.

1 Introduction

MOTIVATION: Radios are inherently power-hungry. As the power costs of processing, memory, and other computing components drop, the lifetime of a battery-operated wireless network deployment comes to depend largely on how often a node's radio is left on. System designers therefore try to power down those radios as much as possible. This requires some form of *synchronization*, since successful communication requires that the sending and receiving nodes have their radios on at the same time. Synchronization is relatively easy to achieve in a wired, powered, and well-administered network, whose nodes can constantly listen for periodic heartbeats from a well-known server. In an ad hoc wireless network or wireless sensor network deployment, the problem becomes much more difficult. Nodes may be far away from any wired infrastructure; deployments are expected to run and even to initialize themselves autonomously (imagine sensors dropped over an area by plane); and environmental factors make sensors prone to failure and clock drift. Indeed there has been a lot of work in this area, see for example: [4,5,6,9,14,15,16,24,27,28,29,31,32,33,34,35,36,37,38,39,40]. Many distinct problems are considered in these papers, and it is beyond the scope of this paper to survey all these works, however most if these papers (among other issues) consider the following problem of radio-use consumption:

INFORMAL PROBLEM DESCRIPTION: Consider two (or more) processors that can switch their radios on or off. The processors' clocks are not synchronized. That is, when a processor wakes up, each clock begins to count up from 0; however, processors may awake at different times. The maximum difference between the time when processors wake up is bounded by some positive integer parameter $n \in \mathbb{N}$. If processors within radio range have their radios on in the same step, they can hear each other and can synchronize their clocks. When a processor's radio is off, it saves energy, but can neither receive nor transmit. Initially, processors awaken with clock shifts that differ by at most n time units. The objective for all the processors is to synchronize their clocks while minimizing the use of radio (both transmitting and receiving). We count the maximum number of times any processor's radio has to be on in order to guarantee synchronization. Indeed, as argued in many papers referenced above, the total time duration during which the radio is on is one of the critical parameters of energy consumption, and

operating the radio for considerable time is far costlier than switching radio off and switching it back on. We assume that all the processors that have their radios on at the same time can communicate with each other. The goal of all processors is to synchronize their clocks, i.e. to figure out how much to add to their offset so that all processors wake up at the same time. (We also consider an extension that models radio *interference*, where if more then one processor is broadcasting at the same time, all receiving processors that have their radio switched on hear only noise.)

For multiple processors, we assume that all processors know the maximum drift n, otherwise the adversary can make the delay unbounded, It is also assumed that all processors know the total number of processors m, although, we also consider a more general setting where n is known for all processors, but m is not. In this setting, we relax the problem, and instead of requiring synchronization of all m processors, we instead require synchronization of an arbitrarily close to 1 constant fraction of all processors. In this relaxation of our model, we require that the radio usage guarantee holds only for those processors that eventually synchronize.

Furthermore, our model assumes that all processors are within radio range of each other, so that the link graph is complete. Our techniques can be thought of as establishing synchronization within completely connected single-hop regions. Clearly, single-hop synchronization is necessary for multi-hop synchronization. Our single-hop synchronization protocol, with simple changes, can synchronize a connected multi-hop network in the sense that (1) two directly connected nodes know one another's clock offsets, and (2) given any two nodes in the network v and w, there exists a path $v_0 = v, v_1, \ldots, v_n = w$ where each adjacent pair of nodes is connected and synchronized. Thus our central concern in this paper is on establishing lower bounds and constructing nearly optimal solutions for the single-hop case.

Towards Formalizing the Abstract Model: To simplify our setting we wish to minimize both transmit and receive cost (i.e., all the times when the radio must be "on" either transmitting or receiving). We discretize time to units of the smallest possible interval that allows a processor to send a message to or receive a message from another processor within radio range. We normalize the cost of transmitting and receiving to 1 unit of energy per time step. (In practice, transmission can be about twice as expensive as receiving. We can easily rescale our algorithms to accommodate this as well, but for clarity of exposition we make these costs equal.) We ignore the energy consumption needed to power the radio on and to power it off, which is at most comparable but in many cases insignificant compared to the energy consumption of having the radio active. This is the model considered, for example, in [31,29,5,6,37,42,12,34].

Informal Model Description: For the purposes of analysis only, we assume that there is global time (mapped to positive integers). All clocks can run at different speeds, but we assume that clock drifts are bounded; i.e., there exists a global constant c, such that for any two clocks their relative speed ratio is bounded by c. Now, we define as a time "unit" the number of steps of the slowest

clock, such that if two of the fastest processors' consecutive awake times overlap by at least a half of their length according to global time, then the number of steps of the slowest clock is sufficient time for any two processors to communicate with each other. This issue is elaborated in Section 7. In our report [8], that is, the full version of the paper, we give the Synchronization Algorithm. We now formalize the informal model description into the precise definition of our model.

Our Formal Model and Problem Statement: Global time is expressed as a positive integer. m processors start at an arbitrary global time between 1 and n, where each processor starts with a local "clock" counter set to 0. The parameter n refers to the discretized uncertainty period, or equivalently, to the possible maximal clock difference, i.e., to the maximal offset between clocks; hence, we will use these terms interchangeably. Both global time and each started processor's clock counter increments by 1 each time unit. The global clock is for analysis only and is not accessible to any of the processors, but an upper bound on n is known to all processors. Each processor algorithm is synchronous, and can specify, at each time unit, if the processor is "awake" or "sleeping." (The "awake" period is assumed to be sufficiently long to ensure that the energy consumption of powering the radio on and then shutting it off at each time unit is far less than the energy expenditure to operate the radio even for a single time unit). All processors that are awake at the same time unit can communicate with each other. (Our interference model changes this so that exactly two awake processors can communicate with each other, but if three or more processors are simultaneously awake, none of them can communicate.) The algorithm can specify what information they exchange. The goal is for all m processors to adjust their local clocks to be equal to each other, at which point they should all terminate. The protocol is correct if this happens either always or if the protocol is randomized with probability of error that is negligible. The objective is to minimize, per processor, the total number of times its radio is awake.

We remark that the above model is sufficiently expressive to capture a more general case where clocks at different nodes run at somewhat different speeds, as long as the ratio of different speeds is bounded by a constant, which is formally proven in Section 7.

Our Results: We develop algorithms for clock synchronization in radio networks that minimize radio use, both with and without modeling of interference. In particular, our results are the following.

1. For two processors we show a $\Omega(\sqrt{n})$ deterministic lower bound and a matching deterministic $O(\sqrt{n})$ upper bound for the number of time intervals a processor must switch its radio on to obtain one-hop synchronization.
2. For arbitrary $m = n^{\beta}$ processors, we prove $\Omega\left(n^{\frac{1-\beta}{2}}\right)$ is the lower bound on the number of time intervals the processor must switch its radio for any deterministic protocol and show a nearly-matching (in terms of the number of times the radio is in use) $O\big(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\big)$ randomized protocol, which fails to synchronize with probability of failure exponentially (in n) close to

zero. Furthermore, our upper bound holds even if there is interference, i.e., if more than one processor is broadcasting, listening processors hear noise.

3. It is easy to see that processors cannot perform synchronization if n is unknown and unbounded, using a standard evasive argument. However, if n is known, we show that 8/9 (or any other constant fraction) of the processors can synchronize without knowing m, yet still using $O(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n))$ radio send/receive steps, with probability of failure exponentially close to zero.

We stress that while the upper bound for two processors is simple, the matching lower bound is nontrivial. This (with some additional machinery) holds true for the multi-processor case as well.

Comparison with Previous (Systems) Work: Tiny, inexpensive embedded computers are now powerful enough to run complex software, store significant amounts of information in stable memory, sense wide varieties of environmental phenomena, and communicate with one another over wireless channels. Widespread deployments of such nodes promise to reveal previously unobservable phenomena with significant scientific and technological impact. Energy is a fundamental roadblock to the long-lived deployment of these nodes, however. The size and weight of energy sources like batteries and solar panels have not kept pace with comparable improvements to processors, and long-lived deployments must shepherd their energy resources carefully.

Wireless radio communication is a particularly important energy consumer. Already, communication is expensive in terms of energy usage, and this will only become worse in relative terms: the power cost of radio communication is fundamentally far higher than that of computation. In one example coming from sensor networks, a Mica2 sensor node's CC1000 radio consumes almost as much current while listening for messages as the node's CPU consumes in its most active state, and transmitting a message consumes up to 2.5 times more current than active CPU computation [36]. In typical wireless sensor networks, transmitting is about two times more expensive than listening, and about 1.5 times more expensive than receiving, but listening or transmitting is about 100 times more expensive as keeping the CPU idle and the radio switched off[4] (i.e., in a "sleep" state).

Network researchers have designed various techniques for minimizing power consumption [5,6,37]. For example, Low-Power Listening [34] trades more expensive transmission cost for lower listening cost. Every node turns on its radio for listening for a short interval τ once every interval $n > \tau$. If the channel is quiet, the node returns to sleep for another n; otherwise it receives whatever message is being transmitted. To transmit, a node sends a *preamble* of at least n time units long before the actual message. This ensures that no matter how clocks are offset, any node within range will hear some part of the preamble and stay

[4] Example consumption costs: CPU idle with clock running and radio off ("standby mode"), 0.1–0.2 mA (milliamps); CPU on and radio listening, 10 mA; CPU on and radio receiving, 15 mA; CPU on and radio transmitting, 20–25 mA.

awake for the message. A longer n means a lower relative receive cost (as τ/n is smaller), but also longer preambles, and therefore higher transmission cost.

A more efficient solution in terms of radio use was proposed by PalChaudhuri and Johnson [33], and further by Moscibroda, Von Rickenbach and Wattenhofer [29]. The idea is as follows. Notice that in the proposal of [34], the proposal was for a transmitting processor to broadcast continuously for n time units, while receiving processors switch their radios on once every n time units to listen. Even for two processors, this implies that total use of the radio is $n + 1$ time units (i.e., it is linear in n). The observation of [33,29] is that we can do substantially better by using randomization: if both processors wake their radios $O(\sqrt{n})$ time units at random (say both sending and receiving), then by birthday paradox with constant probability they will be awake at the same time and will be able to synchronize their clocks. As indicated before, we show instead a deterministic solution to this problem, its practical importance, and a matching lower bound.

Our results strengthen and generalize previous works that appeared in the literature [31,42,41,12]. See further comparisons in the relevant sections.

Comparison with Radio Broadcast:

Usually, in a broadcast setup, a node is able to receive a message if and only if it does not transmit, and there is one and only one of its neighbors that transmits, at that time. In the case when nodes are not able to detect collision, [3,2] showed randomized protocols. A deterministic broadcast algorithm, with work time $O(n^{11/6})$, has been given in [11]. The improvements of these algorithms have followed [25]: for undirected radio network graphs, with diameter $diam$, for randomized broadcast the expected work time has been $O(diam \cdot \log(n/diam) + \log^2 n)$, while for deterministic broadcast the expected work time has been $\Omega(n \log n / \log(n/diam))$. In [26], a faster algorithm for directed radio network graphs has provided running time $O(n \log n \log(diam))$. Additionally, other algorithms for broadcast [13,17,19,22,24] as well as for clock synchronization [32,35,14,4] have been proposed. The work of radio broadcast is different from the problem we address at this paper. However, as we mention in the technical description, once we resolve the problem of meeting times, we can easily combine our solutions with radio broadcast goal to avoid interference.

High-level ideas of our constructions and proofs

- For the two processor upper bound, we prove that two carefully chosen affine functions will overlap no matter what the initial shift is. The only technically delicate part is that the shift is over the reals, and thus the proof must take this into account.
- For the two processor lower bound, we show that for any two strings with sufficiently low density (of 1's) there always exists a small shift such that none of the 1's overlap. This is done by a combinatorial counting argument.
- For multiple processors, the idea of the lower bound is to extend the previous combinatorial argument, while for the upper bound, the idea is to establish a "connected" graph of pairwise processor synchronization, and then show that this graph is an expander. The next idea is that instead of running global

synchronization, we can repeat the same partial synchronization a logarithmic number of times (using the same randomness) to yield a communication graph which is an expander. We then use standard synchronization protocol over this "virtual" expander to reach global synchronization.

- For handling interference, we observe that standard "back-off" protocols [1,10] can be combined with previous machinery to achieve non-interference, costing only a poly-logarithmic multiplicative term.
- For the protocol that does not need to know m (recall that m is the total number of processors within radio-reach), we first observe that if $m > n$, by setting $m = n$ our protocol already achieves synchronization with near-optimal radio use. The technical challenge is thus to handle the case where $m < n$ but the value of m is unknown to the protocol. Our first observation is to show that processors can overestimate m, in which case the amount of energy needed is much smaller (per processor) than for smaller m, and then "check" if the synchronized component of nodes has reached current estimate on m. If it did not, than our current estimate of m can be reduced (by a constant factor) by all the processors. To assure that estimates are lowered by all the processors at about the same time, we divide the protocol into "epochs" which are big enough not to overlap even with a maximal clock drift (of n). Summing, the energy consumption is essentially dominated by the smallest estimate of m, which is within a constant factor of correct value of m, and all processors that detect it stop running subsequent (more expensive) "epochs".

2 Mathematical Preliminaries

Lemma 1 (Two-Color Birthday Problem). *For any absolute constant* $C > \sqrt{1 - \ln 0.1} \approx 1.8173$ *and any positive* $s, t \in (0, 1)$*, where* $s + t = 1$*, the following holds. Suppose* $r = Cn^s$ *identical red balls and* $b = Cn^t$ *identical blue balls are thrown independently and uniformly at random into* n *bins. Then, for sufficiently big* n*, the probability that there is a bin containing both red and blue balls is* $\geqslant 0.8$*.*

Proof. See the full version [8] [5]. □

3 Lower Bounds

The problem of asynchronous wakeup, i.e., low-power asynchronous neighbors discovery, has already been known in the literature [31,42,41,12]. Its goal is to design an optimal wake-up schedule, i.e., to minimize the radio use for both transmitting and receiving. The techniques used, e.g., in the previously cited papers, vary from the birthday paradox in [31], block-design in [42], the quorum based protocol in [41] to an adaption of Chinese reminder theorem [20] in [12]. In our work, we firstly generalize the birthday paradox, obtaining the

[5] For the proofs of all subsequent theorems, lemmas, and claims, see the full version [8].

Two Color Birthday Problem (see Lemma 1). Next, we build the tools for our main analysis on the upper and lower bounds on the optimal radio use for wireless network synchronization. In particular, we start with Lemma 2, which is a stronger combinatorial bound compared to [42], and then generalize results in Lemma 3.

Recall that n is the maximum offset between processor starting times and $m = n^\beta$ is the number of processors. Assume that each processor runs for some time L. Its radio schedule can then be represented as a bit string of length L, where the ith bit is 1 if and only if the processor turned its radio on during that time unit. We first consider the two-processor case. Recall that in our model maximal assumed offset is at most n. If we take 2 bit strings corresponding to the two processors, the initial clock offset corresponds to a *shift* of one string against the other by at most n positions. Note that if we set $L \geqslant 4n$, the maximal shift is at most $n \leqslant L/4$.

Note. *In the next sections, without loss of generality we apply the ceiling function to any real number, e.g., L^α, L/C^2 are treated as $\lceil L^\alpha \rceil$, $\lceil L/C^2 \rceil$, respectively.*

To prove our lower bound, we need to prove the following: for any two L-bit strings with at most $\sqrt{L}/C$ ones in each string (for some constant $C > 1/\sqrt{2}$), there always exists a shift $< L/4$ of one string against another such that none of the ones after the shift in the first string align with any of the ones in the second string. In this case we say that the strings do not *overlap*. W.l.o.g., we make both strings (before the shift) identical. To see that this does not limit the generality, we note that if the two strings are not identical, we can make a new string by taking their bitwise OR, what we call the *"union"* of strings. If the distinct strings overlap at a given offset, then the "union" string will overlap with itself at the same offset.

Lemma 2 (Two Non-Colliding Strings). *For any absolute constant $C \geqslant 1/\sqrt{2}$, and for every L-bit string with $\ell \leqslant \frac{\sqrt{L}}{C}$ ones, there is at least one shift within $L/(2C^2)$ such that the string and its shifted copy do not overlap.*

Next, we want to prove a general lower bound for multiple strings. The high-level approach of our proof is as follows. We pick one string, and then upper bound the total number of ones possible in the "union" of all the remaining (potentially shifted) strings. If we can prove that assuming the density of all the strings is sufficiently small, and there always exists a shift of the first string that does not overlap the "union" of all the remaining strings, we are done. The "union" string is simply a new string with a higher density.

Lemma 3 (General Two Non-Colliding Strings with Different Densities). *Let $s, t > 0$ such that $s + t < 1$, and let $C > 1$. For two L-bit strings such that the number of ones in the first string is $a = L^s/C$, and the number of ones in the second string is $b = L^t/C$, there is a shift within $L/C^2 + 1$ such that the first string and the shifted second string do not overlap.*

Here, wlog, we considered only "left" shift. If we needed both left and right shift, then we would have an additional factor of 2. Using Lemma 3, the lower bounds immediately follow.

Theorem 1. *There exists an absolute constant $C > 1$, such that for any n^β strings of length L with at most $n^{(1-\beta)/2}$ ones in each string, there always exists a set of shifts for each string by at most $L/4$ such that no string's ones overlap any of the ones in all the other strings.*

4 Matching Upper Bound for Two Processors

We now show the upper bound. That is, we give the deterministic algorithm for two devices. In particular, for any initial offset of at most n, we show a schedule where two processors meet with probability equal to one inside a "time-window" of length $W = 2n + 4\sqrt{n} + 2$.

Theorem 2. *For any n, there exists a string of length $W = 2n + 4\sqrt{n} + 2$ with at most $4\sqrt{n} + 4$ ones such that this string will overlap itself for all shifts from 1 to n.*

We remark that the bound that we prove in the above section is in fact more general than the subsequent independent work of [12], which appeared after our report [8]. Note that our bound holds for all values of n, and in fact the two strings could be made identical by doubling the cost.

5 Upper Bound for m Processors

In this setting we have $m = n^\beta$ processors (and as before the maximum shift is at most n). We first state our theorem:

Theorem 3. *There exists a randomized protocol for n^β processors (which fails with probability at most $1/2^{O(n)}$) such that: (i) if $\beta < 1$ the protocol is using at most $O\big(n^{\frac{1-\beta}{2}} \cdot \text{poly-log}(n)\big)$ radio steps per processor, and (ii) if $\beta \geqslant 1$ using at most $O\left(\text{poly-log}(n)\right)$ radio steps per processor. Furthermore, the same bounds hold for the synchronization in the radio communication model, where a processor can hear a message if one (and strictly one) message is broadcasted.*

Next we give a high-level outline of the construction of our algorithm for $\beta \in [0, 1)$. For the case of $\beta \geqslant 1$ we only need Steps 4 and 5, see below. The formal analysis and proofs of the Main Algorithm are given in [8].

Outline of the Main Algorithm:

Step 1. We let each processor run for $L = 4n$ steps, waking up during this time $O\left(n^{\frac{1-\beta}{2}}\right)$ times uniformly at random. It is important to point out that each processor uses independent randomness. We view it as m-row and L ($L \geqslant W + n$) column matrix A (taking into account all the shifts), where $W = 2n + 4\sqrt{n} + 2$ is defined in Theorem 2. Fix any row of this matrix (say the first one). We say that this row "meets" some other row, if 1 in the first row also appears (after the shifts) in some other row. If this happens, the first processor can "communicate" with another processor. We show that for a fixed row, this happens with a constant probability.

Step 2. Each processor repeats Step 1 (using independent randomness) $O(\log m)$ times. Here, we show that a fixed row has at least $O(\log m)$ connections to other rows (not necessarily distinct) with probability greater than $1 - 1/\text{poly}(m)$.

Step 3. From Step 2. we conclude that the first row meets at least a constant number of *distinct* other rows with probability greater than $1 - 1/(2m)$.

Step 4. We use the union bound to conclude that *every* row meets at least constant number of distinct other rows with probability greater than $1/2$. If we repeat this process a logarithmic number of times, we show that we get an expander graph with overwhelming probability (for the definition of an expander see [30]). Thus, considering every row (i.e., every processor) as a node, this represents a random graph with degree of at least a constant number for each node, which is an expander with high probability.

Step 5. During the synchronization period, a particular processor will synchronize with some other processor, without collision, by attempting to communicate whenever it has a 1 in its row. (In the case of interference, the processor can communicate if only one other processor is up at this column, which we can achieve as well, using standard "back-off" protocols [1,10], costing only poly-logarithmic multiplicative term.)

Step 6. The processors can now communicate along the edges of the formed expander (which has logarithmic diameter) as follows. The main insight that we prove below is that if processors repeat *the same random choices* of Step 1 through Step 5, the communication pattern of the expander graph is preserved. Hence, the structure developed in Step 2 can be reused to establish a logarithmic-diameter (in m) spanning tree and synchronize nodes with poly-logarithmetic overhead (using known machinery over this "virtual" graph). We show in [8], by using standard methods, that communicating over the implicit expander graph to synchronize all nodes can be done in $diam + 2$ steps, where $diam$ is the diameter of the expander.

6 Protocol That Does Not Need to Know the Number of Processors

Suppose our processors know the offset n but not the number of all processors in the system, that is, m. The main observation here is that once we make a spanning tree of the graph, each node can also compute the number of nodes in its spanning tree. Hence, we can make an estimate of m and then check to see if this estimate is too big. Thus, until the right (within a constant factor) estimate is reached, all nodes will reject the estimate and continue. Adjusting constants appropriately, we can guarantee that an arbitrary constant fraction of the processors will terminate with the right estimate of m (within some fixed constant fraction). The algorithm for the estimation of m is as follows.

Algorithm: Estimation of m

E1. Set $i = 0$.

E2. Build a spanning tree using the Main Algorithm (from the previous section) for $m_i = n/2^i$ and count the number of nodes in the tree. If the number of the nodes in the tree is less than m_i then set $i := i + 1$ and go to step E2.
E3. Output m_i.

End of Algorithm: Estimation of m

Theorem 4. *Any constant fraction of the processors can synchronize without knowing m, yet still use $O(n^{\frac{1-\beta}{2}}$ poly-log$(n))$ radio send/receive steps (with probability of failure exponentially close to zero). The bound on the radio use holds only for processors that synchronize.*

7 Our Model Can Handle Different Clocks' Speeds with Bounded Ratio

Here are the technical details that explain why our model is realistic even if processors have somewhat different clock speeds. For m processors, let their clock ticks be $\{\tau_1, \tau_2, \ldots, \tau_m\}$. Let τ_{min}, τ_{max} be minimum, maximum of the set $\{\tau_1, \tau_2, \ldots, \tau_m\}$, respectively. The clock ticks are in general different, but the ratio $\tau_{max}/\tau_{min} \leqslant c$ is bounded by some constant c, and each processor knows that upper bound c. Let τ_{trans} be the lower bound on the time necessary for the transmission, i.e., on the time necessary for communication and synchronization between two processors.It is also assumed that the lower bound on τ_{trans} is known to all processors. Now, knowing c and τ_{trans}, each processor i counts k_i clock ticks as a single *time step* s_i such that k_i is defined by $s_i = k_i\tau_i \geqslant 2\tau_{trans}$. In other words, each processor enables the condition necessary for the communication, making its time step $s_i \geqslant 2\tau_{trans}$. It follows that if two processors i and j overlap for a period of time $\geqslant \frac{1}{2}\min\{s_i, s_j\}$, then they can communicate. For further details see the full version.

Claim. If two processors i, j work within the same global time unit, then they can communicate and can synchronize.

8 Conclusions and Open Problems

In this paper, we have studied an important problem of power consumption in radio networks and completely resolved the deterministic case for two processors, showing matching upper and lower bound. For multiple processors, we were able to show a poly-logarithmic gap between our randomized protocol and our deterministic lower bound. However, this is not completely satisfactory. Our lower bound holds only for deterministic protocols, while our upper bound in multi-processor case is probabilistic (unlike the two-processor case, where our upper bound is deterministic as well). Closing this gap remains an interesting open problem.

Another interesting question is the following. It is important to note that in radio communication, conservation of power can be achieved in two different

ways: one approach is to always broadcast the signal with the same intensity (or to power down radios completely in order to save energy); this is what we explored in this paper. The second approach is the ability for a radio to broadcast and receive signals at different intensity, the stronger the signal the further it reaches. In the case where all processors are at the same distance from each other, this is a non-issue (i.e., our single-hop networks, the main focus of this paper). However, for multi-hop networks the question of optimal power-consumption strategies with varying signal strength is still completely open.

References

1. Aldous, D. Ultimate instability of exponential back-off protocol for acknowledgment-based transmission control of random access communication channels. Information Theory, IEEE Transactions on Information Theory, vol.33, no.2, pp. 219-223, Mar 1987.
2. N. Alon, A. Bar-Noy, N. Linial and D. Peleg A lower bound for radio broadcast. Journal of Computer and System Sciences 43 (1991), 290–298.
3. R. Bar-Yehuda, O. Goldreich and A. Itai On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization Journal of Computer and System Sciences 45 (1992), 104–126.
4. P. Blum, L. Meier and L. Thiele Improved interval-based clock synchronization in sensor networks IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks, pp. 349-358, 2004.
5. A. Boulis, M. Srivastava Node-Level Energy Management for Sensor Networks in the Presence of Multiple Applications. Wireless Networks 10(6): 737-746 (2004)
6. A. Boulis, S. Ganeriwal, M. Srivastava Aggregation in sensor networks: an energy-accuracy trade-off. Ad Hoc Networks 1(2-3): 317-331 (2003)
7. B. Bollobas, W.F. de la Vega The diameter of random graphs. In Combinatorica 2, 1982.
8. M. Bradonjić and E. Kohler and R. Ostrovsky Near-Optimal Radio Use For Wireless Network Synchronization. http://arxiv.org/abs/0810.1756 (2008.)
9. Bush, S.F. Low-energy sensor network time synchronization as an emergent property. In Proc. Proceedings. 14th International Conference on Communications and Networks, (ICCCN 2005.) pp. 93-98, 17-19 Oct. 2005.
10. Cali, F., Conti, M., Gregori, E. IEEE 802.11 protocol: design and performance evaluation of an adaptive backoff mechanism. Selected Areas in Communications, IEEE Journal on , vol.18, no.9, pp.1774-1786, Sep 2000.
11. B. Chlebus and L. Gasieniec and A. Gibbons and A. Pelc and W. Rytter Deterministic broadcasting in ad hoc radio networks. Distributed Computing, Volume 15, Number 1, January 2002 Pages: 27–38.
12. P. Dutta and D. Culler Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08), pp. 71–84, 2008.
13. M. L. Elkin, G. Kortsarz Polylogarithmic Inapproximability of the Radio Broadcast Problem. Proc. of 7th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, pp. 105-114, Cambridge, MA, 2004.

14. J. Elson and K. Römer Wireless sensor networks: a new regime for time synchronization SIGCOMM Comput. Commun. Rev., vol. 33. no. 1. pp. 149–154, 2003.
15. J. Elson, L. Girod, and D. Estrin Fine-Grained Network Time Synchronization using Reference Broadcasts. Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Vol 36, pp. 147–163, 2002.
16. R. Fan, I. Chakraborty, N. Lynch Clock Synchronization for Wireless Networks. OPODIS 2004: pp. 400-414.
17. I. Gaber and Y. Mansour Centralized broadcast in multihop radio networks. Journal of Algorithms 46(1) (2003), 1–20.
18. N. Honda and Y. Nishitani The Firing Squad Synchronization Problem for Graphs. Theoretical Computer Sciences 14(1):39-61, April 1981.
19. A. Kesselman and D. Kowalski Fast distributed algorithm for convergecast in ad hoc geometric radio networks. Conference onWireless On demand Network Systems and Services, 2005.
20. Donald Knuth The Art of Computer Programming. vol. 2: Seminumerical Algorithms, Third Edition. Addison-Wesley, 1997.
21. K. Kobayashi The Firing squad synchronization problem for a class of polyautomata networks. Journal of Computer and System Science 17:300-318, 1978.
22. C. Koo Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In Proceedings of 23rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), pages 275-282, 2004.
23. A. P. Kamath and R. Motwani and K. Palem and P. Spirakis Tail bounds for occupancy and the satisfiability threshold conjecture. Random Structures and Algorithms, vol. 7, pp. 59-80, 1995.
24. K. Kothapalli, M. Onus, A. Richa and C. Scheideler Efficient Broadcasting and Gathering in Wireless Ad Hoc Networks in IEEE International Symposium on Parallel Architectures, Algorithms and Networks(ISPAN), 2005.
25. D. Kowalski and A. Pelc Broadcasting in undirected ad hoc radio networks. In Proceedings of the twenty-second annual symposium on Principles of distributed computing, pages 73-82. ACM Press, 2003.
26. D. Kowalski and A. Pelc Faster deterministic broadcasting in ad hoc radio networks. Proc. 20th Ann. Symp. on Theor. Aspects of Comp. Sci. (STACS'2003), LNCS 2607, 109-120, 2003.
27. Kopetz, H., wilhelm Ochsenreiter Global time in distributed real-time systems. Technical Report 15/89, Technische Universitat Wien, Wien Austria (1989).
28. Mills, D.L. Internet time synchronization: the network time protocol. Communications, IEEE Transactions on , vol.39, no.10, pp.1482-1493, Oct 1991.
29. Moscibroda, T., von Rickenbach, P., Wattenhofer, R. Analyzing the Energy-Latency Trade-Off During the Deployment of Sensor Networks. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp.1-13, April 2006.
30. R. Motwani and P. Raghavan Randomized algorithms. Cambridge University Press, New York, NY, USA, 1995.
31. M. McGlynn and S. Borbash Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pp. 137-145, 2001.
32. V. Park, M. Corson A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks INFOCOM '97. Sixteenth Annual Joint Conference

of the IEEE Computer and Communications Societies. Driving the Information Revolution, 1997.
33. S. PalChaudhuri and D. Johnson Birthday paradox for energy conservation in sensor networks. Proceedings of the 5th Symposium of Operating Systems Design and Implementation, 2002.
34. Polastre, J., Hill, J., and Culler, D. Versatile low power media access for wireless sensor networks. In Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (Baltimore, MD, USA, November 03 - 05, 2004). SenSys '04. ACM Press, New York, NY, 95-107.
35. Sichitiu, M.L., Veerarittiphan, C. Simple, accurate time synchronization for wireless sensor networks Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, vol.2, no.pp. 1266-1273 vol.2, 16-20 March 2003.
36. V. Shnayder, M. Hempstead, B. Chen, G. Allen and M. Welsh Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. 2004, pp 188-200, ACM Press.
37. C. Schurgers, V. Raghunathan, M. Srivastava Power management for energy-aware communication systems. ACM Trans. Embedded Comput. Syst. 2(3): 431-447 (2003)
38. Sivrikaya, F., Yener, B. Time synchronization in sensor networks: a survey. Network, IEEE , vol.18, no.4, pp. 45-50, July-Aug. 2004.
39. M. L. Sichitiu and C. Veerarittiphan Simple, Accurate Time Synchronization for Wireless Sensor Networks. Proc. IEEE Wireless Communications and Networking Conference (WCNC 2003), pp. 1266-1273, 2003.
40. B. Sundararaman, U. Buy and A. D. Kshemkalyani Clock synchronization for wireless sensor networks: a survey. Ad-hoc Networks, 3(3): 281-323, May 2005.
41. Y.-C. Tseng, C.-S. Hsu, Chih-Shun and T.-Y. Hsieh Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. Comput. Netw. vol. 43(3), pp 317-337, 2003.
42. R. Zheng, J. Hou, and L. Sha Asynchronous wakeup for ad hoc networks. Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc '03), pp 35-45, 2003.