Fault-Tolerant Energy Efficient Consensus on Ad Hoc Beeping Networks

Ny Aina ANDRIAMBOLAMALALA[1] and Vlady RAVELOMANANA[1]

[1]Université de Paris, IRIF, CNRS, F-75013 Paris, France

April 16, 2020

Abstract

With the emergence of Mobile technologies, Internet of Things and Sensor networks, ad hoc protocols have gained in importance during the last decade. As the devices of such technologies are battery-powered most of the time, conserving energy is a key problem on network's protocols design. In addition, randomness is a very important issue of any computer system. Thus, in this paper, we address the energy consumption of the decentralized global random bit (*resp.* number) generation problem and the binary (*resp.* multi-valued) consensus problem, two fundamental problems in decentralized networking. For cases when each of the n devices of the network may crash, we have designed a fault-tolerant Energy Conserving pseudo-random Bit Generator protocol or ECBG and a fault-tolerant Energy Conserving Binary Consensus protocol or ECBC having $\mathbf{O}(\log \mathbf{n})$ time complexity and a constant energy consumption per device. Such protocols have $O(n)$ bit complexity. We then adapted our protocol to design a fault-tolerant Energy Conserving pseudo-random Number Generator protocol or ECNGhaving $O(\log^2 n)$ time complexity, $O(\log n)$ energy consumption per device and conserving the $O(n)$ bits complexity.

1 Introduction

Designing ad hoc network's protocols became an important research field, especially after the growth of Mobile devices [1], Internet of Things devices (IoT) [2] and Sensors networks (as Body Area Networks or BANs [3]). All such devices are battery-powered and for the most part have limited battery life, thus, managing their energy consumption is of paramount importance [4]. As a consequence, in addition to time complexity[1], bit complexity[2], we focus on the energy complexity of the protocols. As transceiving

[1]It is measured by communication time's number instead of local computations. In this paper, we assume that internal computations are free [5].

[2]The total number of bits sent by all devices.

uses more energy than internal computations (in [6], each sensor device consumes respectively 1.8W, 0.6W and 0.05W when transmitting, receiving a message and having radio switched off), the energy consumption of a protocol is measured by the maximum over all devices of the number of time slots during which any device has its radio switched on (to listen or to transmit). Such energy consumption also depends on the size of exchanged messages [7], so, any reduction of the message's size is highly desirable. Thus in this paper, we consider the single-hop[3] Beeping model, introduced by Cornejo and Kuhn [8].

Beeping Model The Beeping Model is a harsh communication model in which the devices are allowed to transmit only one-bit messages by sending a jamming signal or a beep on a broadcast channel. Each device only needs to be able to do carrier-sensing for detecting such signals and can not distinguish between single and multiple transmitters. In [8], the authors noted that such carrier-sensing can typically be done much more reliably and requires significantly less energy and other resources than sending a message.

The Beeping Model can be simulated on existing Mobile, IoT and Sensor networks devices by making each device broadcast a single-bit message at each time slot of communication instead of sending a large sized message. It also could be implemented in a new generation of Mobile, IoT and Sensor devices [9–11] with specific sensing technologies that would allow them to communicate only with beeps. This leads us to ad hoc networks with a better energy conservation.

1.1 Addressed problems

Due to the limited battery life and many other possible causes, each device may fail by crashing, it can do any internal computations but can not communicate on the network when in a sleeping[4] state. In addition, it can choose to wake up (switch on its radio) or to sleep at any time slot. In order to use randomness, we assume that the devices are able to generate random discrete variables (see for instance Devroye [12]). With such assumption, we aim to design fault-tolerant protocols resolving the two following problems:

1.1.1 Fault-tolerant Energy Conserving pseudo-random Bit and pseudo-random Number Generators (ECBG and ECNG)

On one hand, pseudo-random bit generators are important routines of any computer system [13]. It consists of finding a protocol that allows all the devices of a given network to generate a common binary value $b \in \{0, 1\}$ such that $\mathbb{P}[b = 0] \sim \mathbb{P}[b = 1] \to 1/2$. On the other hand, given a common

[3]Each device is at communication range of each other.

[4]When the device has radio switched off.

value A to all devices, a pseudo-random number generator outputs a common number m in each device picked randomly from $[0, O(A)]$.

In a crash-prone ad hoc network, electing a leader to make a decision (generating a random bit or a random number) may not work as the elected device may crash at any time. As a consequence, a fault-tolerant protocol that generates a random bit in such a model without any central controller is very useful. Such a protocol can also be used to simulate more involved distributions [12] in a decentralized manner, then can become a basic brick of decentralized pseudo-random number generators [14] which are very useful for many computing applications (in casino industry [15] or in blockchain [16] for example).

1.1.2 Fault-tolerant Energy Conserving Binary Consensus or ECBC

The binary consensus problem introduced by Pease, Shostak and Lamport in 1980 [17] is one fundamental issue in decentralized networking. In a message passing system, given n devices denoted $s_1, s_2, \ldots, s_n$ at most f of which may fail, the problem consists in each device s_i beginning with an initial value $b_i \in \{0, 1\}$ and deciding on a common output $b \in \{0, 1\}$. b satisfies the 3 conditions:
(a) The agreement condition : all devices decide on the same output.
(b) The validity condition: if a device decides b, then b is the initial value of some device.
(c) The termination condition: every non-failing or correct device eventually decides, with probability almost 1.

The consensus problem has many application issues in the computer science area: we can cite the multiparty protocols, the reliable decentralized databases, the multicast protocols or the time stamping protocols [18]. Extensive researches has recently focused on such a problem using the new generation of Mobile [19], IoT [2] and Wireless devices [8].

Recently, with the growth of mobile robotics, multi-agent systems [11], sensors networks [20] and blockchain [16], the consensus problem was very well studied for the coordination of those decentralized systems.

1.2 Adversarial and failing scenarios

We consider a crash-prone network caused by a malicious adversary. If a device crashes at a time slot, then starting from such time slot, it can no longer send nor receive any messages. In this paper, we consider the *Non-adaptive* (*resp. Weakly-adaptive*) adversary which chooses the failing devices (*resp.* chooses the failing devices and the time when each device will fail) before the execution of the protocol [21, 22]. If a device never fails, it is said to be correct. The only restriction for those adversarial scenarios is that the adversary can fail up to f devices, where $0 \leq f < n$. Note that the presented protocols in this paper tolerate up to $f \leq n - n^{1/\gamma}$ failing devices for some constant $\gamma \geq 1$ not depending on n.

1.3 Related works

As a fundamental problem in decentralized system networks, there were many important researches on the consensus problem since the 80's. Such a problem was introduced by Pease, Shostak and Lamport in 1980 [17] with their binary consensus protocol terminating in polynomial time for systems with byzantine[5] devices.

The consensus problem was studied by many researchers but in this Section, we only present a small digest of works related to the settings considered in this paper. When up to f devices may fail and each device can send a message to each of its neighbors, Toueg, Perry and Srikanth [23] designed a protocol with $f + 2$ time complexity.

The first sub-linear time consensus protocol was designed in 1989 by Chor, Merritt and Shmoys [24] and terminates in $O(\log n)$ time slots on a synchronous crash-prone system. Such a protocol works on a network of processes where at each time slot, each device can broadcast a message, receive all incoming messages and perform some computations. Such a model is referred to as the LOCAL communication model. They proved a $\Omega(\log n/\log\log n)$ lower bound for the time complexity on this model. More recently, Amdur, Weber and Hadzilacos [25] proved a $\Omega(n)$ lower bound on the bit complexity for the consensus problem even if the system is failure-free. In 2010, Gilbert and Kowalski [21] achieved optimal bit complexity of $O(n)$ with almost optimal $O(\log n)$ time complexity for this system. Then in 2011, Ashrafi, Malmirchegini and Mostofi [26] presented a consensus protocol for CR (Cognitive Radio) networks. In 2013, Abdaoui and Elfouly [27] designed a protocol outputting binary consensus over a WSN (Wireless Sensor Network) containing some faulty devices. One of the last results on the problem was presented by Kowalski and Mirek in 2019 [22].

Results on the beeping model appeared in 2016, when Hounkanli, Miller and Pelc [28] presented a consensus protocol using beep in a fault-prone MAC (Multi Access Channel), terminating in logarithmic time. On their model, the devices are fault-free but the channel is faulty. As already mentioned, in order to optimize energy conservation on many networks (WSN, BANs, IoT, CR ...), it could be interesting to simulate the beeping communication model on such networks. In this sense, there are many beeping protocols addressing multiple decentralized problems: such as the leader election problem [29], the network's size approximation [30], the initialization and the counting problems [31, 32] or the maximal independent set problem [7]. Recently, many consensus protocols were designed to be subroutines of the

[5] A byzantine device can deviate from the protocol by doing arbitrary computations.

blockchain [33].

For the multi-valued consensus problem, Turpin and Coan [34] extended the binary consensus protocol designed in [35] in order to have a randomized multi-valued byzantine agreement which may output a value different from all inputs. It terminates in $2f + 5$ time slots using $O(n^3 \log n)$ bits of communication and does not respect the validity condition of the consensus problem. To respect the validity condition, Neiger [36] adapted the consensus protocol designed in [37] and obtained a multi-valued byzantine consensus terminating in exponential time. These protocols both work on the LOCAL model.

Considering that our consensus protocol uses a distributed pseudo random bit generator, designing such a protocol was studied in 1983 by Dwork, Shmoys and Stockmeyer [38]. Then, in the $90's$, Micali and Rabin [39], Beaver and So [18] addressed the problem for the LOCAL model. It recently gained in importance with the growth of Mobile networks [19], IoT devices [40], WSN [41] and the blockchain [15, 16]. As already remarked, in a crash-prone system, our pseudo-random number generator protocol can be more efficient than a leader election protocol for the generation of a random number. As it is one of the most fundamental problems in distributed computing, such leader election problem was extensively studied over the years under many models of communication and in many different network settings [42–48]. In particular, Ghaffari, Lynch and Sastry [49] proved that $\Omega(\log n)$ time slots are required to have a leader election.

1.4 Our main results

For n devices, at most $f = n - n^{1/\gamma}$ of which may crash ($\gamma \geq 1$), we design a fault-tolerant ECBG protocol outputting 0 or 1 with a probability close to 1/2. Such a protocol terminates in $O(\log n)$ time slots, using $O(n)$ bits of communication and more importantly has a constant energy complexity. Then, we used such protocol to design a distributed random number generator terminating in $O(\log^2 n)$ time slots with $O(n)$ bit complexity and $O(\log n)$ energy complexity. Note that as we can see in the simulation's results presented in Section 4, such a protocol works for small values of n (for $n \geq 50$). The random bit generator can be adapted to have a ECBC protocol keeping the same complexities and the random number generator can be used as a multi-valued consensus protocol not respecting the validity condition [35]. The following Table 1 compares the existing results for the consensus problem with ours.

1.5 Our new approach

The main idea of many distributed consensus protocols is to cause all the devices to agree on the value held by the largest number of devices. Our

Table 1: Showing exiting results and ours

Problem	Models	Failure	Time and Bit complexities	Energy complexity
Existing results				
Binary consensus	Local model [21]	crash $f \leq n/3$	$O(\log n)$ $O(n)$	-
Multi-values Consensus	Local model [35]	byzantine $f \leq n/3$	$2f + 5 = O(n)$ $O(n^3 \log n)$	-
Our results				
Binary Consensus	Beeping model	crash $f \leq n - n^{1/\gamma}$ $\gamma \geq 1$	$O(\log n)$ $O(n)$	$O(1)$
Multi-values Consensus	Beeping model [35]	crash $f \leq n - n^{1/\gamma}$	$O(\log^2 n)$ $O(n)$	$O(\log n)$

protocol design stands out from this approach and is based on the following property and protocols:

1.5.1 The parity of the maximum of some discrete random variable

There is a discrete random variable (*r.v.* for short) X distributed such that if $X_1, X_2, \ldots, X_n$ are n random copies of X,

$$\mathbb{P}\left[\max_{1\leq i\leq n}\{X_i\} \text{ is even}\right] \xrightarrow[n\to\infty]{} \frac{1}{2}.$$

Such a maximum is of order $O(\log n)$. Our main idea is then to make each device s_i generate a random copy X_i of the *r.v.* X and output b in function of the parity of $\max_{1\leq i\leq n}\{X_i\}$. The time complexity of our protocol then results from the communication time spent by all devices to find such a maximum. After generating X_i, each device s_i locally computes an interval I of integers containing the maximum. Let us note the number of integer values in I by $|I|$: $I = \{I_1, I_2, ..., I_{|I|}\}$ where I_j is an integer value, $I_{j+1} = I_j + 1$ such that $I_1 = 1$ and $I_{|I|} = O(\log n)$. After that, each device finds out if it holds $\max_{1 \leq i \leq n}\{X_i\}$ by browsing[6] through the interval I. At the end of such a browsing protocol, the devices holding the maximum transmit its parity to all the other devices. If the maximum is even, all devices output 0, otherwise, they output 1.

1.5.2 Browsing through the interval I

This uses a known procedure for finding the maximum value in a given interval. It consists of checking each value in such interval one by one from the last one. Each device s_i is initially in a sleeping state. The procedure starts at time slot $t_0 = 0$ when each device checks if the last value $I_{|I|} = O(\log n)$ of the interval I is equal to its random value X_i. Then, at each time slot $t_j = t_0 + j$, each device s_i compares X_i to both values $I_{|I|} - j$ and $I_{|I|} - j - 1$. If $X_i = I_{|I|} - j$, then s_i wakes up to beep at t_j. If $X_i = I_{|I|} - j - 1$, then s_i wakes up and listens to the network at t_j.

Remark 1 *The main idea is that if a device does not detect any beep when listening to the network, it has the maximum of all X_i. This can be incorrect as there may be some values in I not picked by any device: there may be some gaps in I. As a result, each device having X_i after one of these gaps can pretend to have the maximum.*

To avoid this problem, we introduce the following protocol.

1.5.3 Filling the possible gaps in I

Before executing the previous procedure, each device s_i locally and uniformly chooses at random one time slot t_j to witness for the presence of a beep during the browsing procedure. Then, at t_j, the devices that chose to witness at t_j wake up and listen to the network. Thus, at the next time slot $t_j + 1$, all devices hearing beep at t_j wake up and beep to notify the next listening devices that the maximum has already been found. As a result, if a device s_i never hears a beep when listening to the network, it knows that its X_i is the maximum of all random values.

2 Energy conserving random bit generator

Throughout the rest of the paper, we will use a *r.v.* X distributed as the geometric distribution with parameter 1/2 denoted $\textsc{Geom}(1/2)$. Let p_k be $\mathbb{P}[X = k]$ for all $k > 0$. For the sake of simplicity, we note the logarithm of n

[6] At each time slot $t_0, t_1, \ldots, t_j$, each device s_i checks if the corresponding value $I_{|I|} - j$ in the interval I is equal to its generated value X_i and do some computation at t_j.

in base 2 by $\lg n$ and we suppose that $\log n$ and $\lg n$ are integer values. The following technical result is important for our purpose:

Lemma 1 *Let X_1, X_2, $\cdots$, X_n be n independent copies of X distributed as* Geom(1/2). *We have*

$$\text{(a) } \mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \le 2\lg n\right] \ge 1 - O\left(\frac{1}{n}\right),$$

$$\text{(b) } \mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \text{ is even }\right] \ge \frac{1}{2} - O\left(\frac{1}{n}\right).$$

Proof. Let $X_1, X_2, \ldots, X_n$ be n random copies of Geom(1/2).
Proof of (a): Firstly, we have $p_k = 2^{-k-1}$.

$$\mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \le 2\lg n\right] = \mathbb{P}\left[\forall i \in \{1, 2, \ldots, n\}, X_i \le 2\lg n\right].$$

By definition,

$$\mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \le 2\lg n\right] = \left(\sum_{k=0}^{2\lg n} p_k\right)^n.$$

And by replacing p_k,

$$\mathbb{P}\left[\max_{1\le i\le n} X_i \le 2\lg n\right] = \left(1 - 2^{-2\log n - 1}\right)^n.$$

Then, as for all $x \in]0, \frac{1}{2}[$,

$$e^{-x-x^2} \le (1-x) \le e^{-x}, \tag{1}$$

$$\mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \le 2\lg n\right] \ge 1 - O\left(\frac{1}{n}\right).$$

Proof of (b): Let us set $P_{me} = \mathbb{P}\left[\max_{1\le i\le n}\{X_i\} \text{ is even }\right]$. By definition,

$$P_{me} = \sum_{k=0}^{\infty} \left(\sum_{i=0}^{2k+1} p_i\right)^n - \left(\sum_{i=0}^{2k} p_i\right)^n,$$

$$= \sum_{k=0}^{\infty} \left(1 - 2^{-2k-2}\right)^n - \left(1 - 2^{-2k-1}\right)^n.$$

Then, by using (1),

$$P_{me} \geq \sum_{k=0}^{\infty} \exp\left(-2^{-2k-2}n - 2^{-4k-4}n\right) - \exp\left(-2^{-2k-1}n\right).$$

For k large enough, $\exp\left(-2^{-4k-4}n\right) = 1 - \left(n/2^{4k}16\right) + O\left(n^2/2^{8k}\right)$. Thus,

$$P_{me} \geq \sum_{k=0}^{\infty} \exp\left(-2^{-2k-2}n\right)\left(1 - \left(\frac{n}{2^{4k}16}\right)\right)$$

$$- \exp\left(-2^{-2k-1}n\right).$$

Using the Euler-Maclaurin summation formula, we get

$$P_{me} \geq \int_{k=\frac{1}{3}\lg n}^{2\lg n} \exp\left(-2^{-2k-2}n\right)\left(1 - \frac{n}{2^{4k}16}\right)$$

$$- \exp\left(-2^{-2k-1}n\right) dk.$$

By noting $a = \frac{1}{3}\lg n$ and $b = 2\lg n$,

$$P_{me} \geq \underbrace{\int_{k=a}^{b} \exp\left(-2^{-2k-2}n\right) dk}_{F_1} - \underbrace{\int_{k=a}^{b} \exp\left(-2^{-2k-1}n\right) dk}_{F_2}$$

$$- \underbrace{\int_{k=a}^{b} \exp\left(-2^{-2k-2}n\right) \frac{n}{2^{4k}16} dk}_{F_3}.$$

We have

$$F_3 = \frac{1}{2n\log 2\exp\left(1/4n^3\right)} - \frac{1}{2n\log 2\exp\left(n^{1/12}\right)} -$$
$$\frac{1}{2n^{2/3}\log 2\exp\left(n^{1/12}\right)} + \frac{1}{2n^4\log 2\exp\left(1/4n\right)}.$$

After a bit algebra, we get

$$F_3 = O\left(\frac{1}{n}\right),$$

and

$$F_1 - F_2 = \frac{1}{2}\frac{-\gamma + 2\log 2 + 3\log n}{\log 2} - \frac{1}{2}\frac{-\gamma + \log 2 + 3\log n}{\log 2}$$

$$+ O\left(\frac{1}{\sqrt{\exp\left(n^{1/3}\right)}}\right) + O\left(\frac{1}{n}\right),$$

where γ is the Euler's constant [50]. Then, we obtain

$$F_1 - F_2 \geq \frac{1}{2},$$

and

$$P_{me} \geq \frac{1}{2} - O\left(\frac{1}{n}\right).$$

□

Having defined this probability distribution, we can now define our protocol.

2.1 ECBG protocol

We design a decentralized ECBG protocol using a random variable distributed as Geom(1/2). Such a protocol outputs a common binary value $b \in \{0, 1\}$ at each device with

$$\mathbb{P}[b = 1] \simeq \mathbb{P}[b = 0] \xrightarrow[n \to \infty]{} \frac{1}{2},$$

when the devices can only send $1-$bit messages. At the beginning of the protocol, all devices are in a sleeping state and the protocol is organized into 4 phases.

Phase 1: $\forall i = 1, 2, \ldots n$, each device s_i locally generates one random copy X_i of the *r.v.* X distributed as Geom(1/2) and computes a common interval $I = [1, 2\lg n]$.

Phase 2: Each device s_i sets $\delta = \text{UAR}(\{1, 2, \ldots, X_i - 1\} \cup \{X_i + 1, \ldots, 2\lg n\})$[7] in order to witness for a beep at t_δ during Phase 3. There are roughly $\Theta(n/\log n)$ devices witnessing for a beep at each time slot of Phase 3. Note that if $f < n$ devices crash, there may be some non-witnessed time slots when all devices browse through I and the protocol may output a biased bit. To circumvent such a problem, we make each device s_i set $\tau = \text{UAR}(\{1, 2, \ldots 2\lg n\} \setminus \{\delta, X_i\})$ and s_i will also witness for a beep at the time slot t_τ of Phase 3.

Phase 3: Remembering that the last integer value in the interval I is $I_{|I|} = 2\lg n$, if a device s_i has $X_i = 2\lg n$, it beeps at t_0. In parallel, the devices with $X_i = (2\lg n) - 1$ and those witnessing for a beep at t_0 wake up to listen to the network. If a device hears a beep at t_0, it beeps at t_1. The devices witnessing for a beep at t_1 and those having $X_i = (2\lg n) - 2$ listen to the network at t_1. We generalize such executions for each time

[7] UAR(A) picks one value uniformly at random from the set A.

slot $t_j, j = 0, 1, \ldots, 2\lg n$ by defining the TRANSMIT(j) and the RECEIVE(j) protocols. These protocols are executed by a single device s_i.

- RECEIVE(j): if the device s_i chose to witness for a beep at t_j (*i.e.* $j = \delta$ or $j = \tau$), or if $X_i = (2\lg n) - j - 1$, s_i listens to the network at t_j.

- TRANSMIT(j): If the device s_i heard a beep at $t_j - 1$ or $X_i = (2\lg n) - j$, it beeps at t_j.

During the browsing procedure, all devices run TRANSMIT(j) and RECEIVE(j) in a parallel manner for all $j = 0, 1, \ldots, 2\lg n$. If a device has to beep and listen to the network at the same time, it prioritizes the beeping computation.

Phase 4: After Phase 3, all devices wake up during two more time slots (let us say t_l and $t_l + 1$). If a device s_i does not have $X_i = \max_{1\leq j\leq n}\{X_j\}$ (it heard a beep at least once during Phase 3), then it listens to the network during those two time slots. In order to notify all devices that the maximum is even or odd, each device s_i holding $X_i = \max_{1\leq j\leq n}\{X_j\}$ encodes the parity of $\max_{1\leq j\leq n}\{X_j\}$ as follows: If X_i is even, it beeps at the first time slot t_l and remains silent at $t_l + 1$. Otherwise, it remains silent at t_l and beeps at $t_l + 1$. Thus, each device hearing a beep at t_l (*resp.* $t_l + 1$) knows that the maximum is even (*resp.* odd) and consequently outputs $b = 0$ (*resp.* $b = 1$).

Algorithm 1: ECBG( ) at a device s_i.

Phase 1: s_i locally generates one random copy X_i of GEOM$(1/2)$ and sets $I = [1, 2\lg n]$.
Phase 2: s_i sets $\delta = \mathrm{UAR}(\{1, 2, \ldots, 2\lg n\} \setminus \{X_i\})$ and $\tau = \mathrm{UAR}(\{1, 2, \ldots, 2\lg n\} \setminus \{X_i, \delta\})$.
Phase 3: **for** j *from* 1 *to* $2\lg n$ **do**
    s_i runs RECEIVE(j) and TRANSMIT(j) in parralel.
Phase 4: during the next time slots t_l and $t_l + 1$,
if *s_i beeped at least once and never heard beep during Phase 3* **then**
    if *X_i is even* **then**
        s_i beeps at t_l and outputs $b = 0$.
    else
        s_i beeps at $t_l + 1$ and outputs $b = 1$.
else
    s_i listens to the network.
    if *s_i hears beep at t_l* **then**
        s_i outputs $b = 0$.
    if *s_i hears beep at $t_l + 1$* **then**
        s_i outputs $b = 1$.

2.2 ECBG protocol's analysis

Lemma 2 *If $f \leq n - O(\lg^2 n)$ devices fail ($n - n^{1/\gamma} < n - O(\lg^2 n), \gamma \geq 1$), then each time slot of the browsing procedure is witnessed by at least one correct device.*

Proof. Each of the n devices chooses to witness for the presence of a beep at one of the $2 \lg n$ time slots of Phase 3. So, by means of a Chernoff bound, at least $n/4 \lg n$ devices witness at each time slot with a probability greater than $1 - \exp(-n/16 \lg n)$. Then, the devices rechoose uniformly at random to witness for a beep at one of these $2 \lg n$ time slots. In the scenario where $f = n - \lg^2 n$ devices fail, the $\lg^2 n$ correct devices may chose to witness at the same time slot. So, no correct device witnesses at the other $2 \lg n - 1$ time slots. When the devices rechoose to witness for a beep at on of the $2 \lg n$ time slots, the $\lg^2 n$ correct devices are redistributed into these $2 \lg n$ time slots. As a result, by means of a Chernoff bound, each time slot is then witnessed by at least $\Theta(\lg n)$ correct devices with a probability of almost $1 - O(n^{-1})$. □

Lemma 3 *During the execution of the protocol 1, each device wakes up during at most 7 time slots.*

Proof. Each device s_i may wake up to listen to the network during 3 time slots during the Phase 3 of the ECBG protocol: at t_j when it has $X_i = 2 \lg n - j - 1$, at t_δ and at t_τ. In the same way, s_i may wake up to beep twice: at t_j when $X_i = 2 \lg n - j$, at $t_\tau + 1$ or $t_\delta + 1$ if it heard a beep at t_τ or t_δ. In addition, each devices wakes up in a deterministic manner during the last 2 time slots of the protocol. □

Theorem 1 *In single hop beeping networks of size n, if up to $f \leq n - n^{1/\gamma}$ devices may fail by crashing, the* ECBG *procedure outputs* 0 *or* 1 *with a probability close* 1/2*. Such a protocol succeeds in $O(\log n)$ time slots using $O(n)$ bits of communication and $O(1)$ energy complexity.*

Proof. The maximum of all random values is in $I = [1, 2 \lg n]$. By Lemma 1, if up to $f \leq n - n^{1/\gamma}$ devices may fail, such a maximum is even with a probability greater than $1/2 - O\left((n-f)^{-1}\right) = 1/2 - O\left(n^{-1/\gamma}\right)$. Then, by browsing through the interval I, all the devices know if they hold the maximum after $O(\log n)$ time slots and by Lemma 3, the energy complexity is constant.

During each time slot of the Phase 3, at most $\Theta(n/\log n)$ devices may beep. Thus, the bit complexity of the protocol is at most $O(n/\log n) \times O(\log n) = O(n)$.

By Lemma 2, at any time slot of the browsing procedure, there is at least one correct device listening to the network. Thus, at the end of the execution of the ECBG procedure, all correct devices have the same output value with a probability greater than $1 - O\left(n^{-1/\gamma}\right)$ for some constant $\gamma \geq 1$ depending on the number of crashing devices f. $\square$

We simulate these results in Section 4.2 and note that such simulations succeed even if the network's size n is small ($n \geq 50$).

2.3 ECNG protocol

In this Section, given a value $A = O(n^\alpha), \alpha > 0$ to all devices, we design a protocol outputting a common number $m = \mathrm{UAR}(\{0, 1, \ldots O(A)\})$ in a distributed manner. To do so, we make all devices compute $\log A$ binary values $m_1, m_2, \ldots, m_{\log A}$ such that $m_i \in \{0, 1\}$ is computed by executing the ECBG protocol. Then, m is obtained by the decimal conversion of the binary value $m_1 m_2 \ldots m_{\log A}$. Such a protocol is subdivided into $\log A + 1$ steps.

Step 0: To save energy, we distribute all devices such that a device only participates to the computation of one binary value m_i. Each device then chooses to enter into a group G_d such that $d = \mathrm{UAR}(\{1, 2, \ldots, \log A\})$ and sets $m_i = 0, \forall i = 0, 1, \ldots \log A$.

Step 1: All devices in the group G_1 compute $b \leftarrow \mathrm{ECBG}()$ during $2 \lg n + 2$ time slots while all the devices in the other groups remain sleeping. All devices in G_1 then set $m_1 = b$. After these $2 \lg n + 2$ time slots, those devices in G_1 transmit $m = m_1 m_2 \ldots m_{\log A}$ bit by bit and all devices in G_2 wake up to listen to the network for $2 \log A$ time slots. During such transmission, one bit value m_i is sent during two time slots. $m_i = 0$ is encoded by one beep followed by a silent time slot. Similarly, $m_i = 1$ is encoded by one silent time slot followed by a beep.

Step 2: All devices in G_2 compute $m_2 \leftarrow \mathrm{ECBG}()$. Then, they send the new value of m bit by bit as in Step 1 while all devices in G_3 listen to the network.

Step 3 to Step $\log \mathbf{A} - 1$: The next $\log A - 3$ steps work exactly as Step 1 and Step 2.

Step $\log \mathbf{A}$: The devices in $G_{\log A}$ compute $m_{\log A} \leftarrow \mathrm{ECBG}()$. After that, they send m bit by bit and all the other devices listen to the network during $2 \log A$ time slots.

Algorithm 2: ECNG() at a device s_i.

1 **Step 0:** s_i sets $d = \mathrm{UAR}(\{1, 2, \ldots, \log A\})$ and enters the group G_d. s_i locally creates $m = m_1 m_2 \ldots m_{\log A}$ where $m_i = 0$ $\forall i \in \{1, 2, \ldots, \log A\}$.
2 **Step 1 to Step** $\log \mathbf{A} - \mathbf{1}$**: for** k *from* 1 *to* $\log A - 1$ **do**
3   **if** $s_i \in G_k$ **then**
4   s_i computes $b \leftarrow$ ECBG() during the first $2 \lg n + 2$ time slots of the Step k.
5   It sets $m_k = b$ and sends m bit by bit as follows.
6   **for** l *from* 1 *to* $\log A$ **do**
7   **if** $m_l = 0$ **then**
8   s_i beeps at t_{2l}
9   **else**
10   s_i beeps at t_{2l+1}
11   **if** $s_i \in G_{k+1}$ **then**
12   s_i remains sleeping during the first $2 \lg n + 2$ time slots of the Step k. Then, it wakes up and listens to the network during the next $2 \log A$ time slots as follows.
13   **for** l *from* 1 *to* $\log A$ **do**
14   **if** s_i *hears beep at* t_{2l} **then**
15   It sets $m_l = 0$
16   **if** s_i *hears beep at* t_{2l+1} **then**
17   It sets $m_l = 1$
18 **Step** $\log \mathbf{A}$**: if** $s_i \in G_{\log A}$ **then**
19   s_i computes $b \leftarrow$ ECBG() during the first $2 \lg n + 2$ time slots of the Step $\log A$.
20   It sets $m_{\log A} = b$ and sends m bit by bit as described before (lines $6 - 10$).
21 **else**
22   s_i listens to the network and updates m (as in lines $13 - 17$)

Theorem 2 *In single hop beeping networks of large size n, if a value A is given in advance to all devices and if up to $f \leq n - n^{1/\gamma}$ devices may crash, the* ECNG *protocol outputs a common value $m = \mathrm{UAR}(\{0, \ldots O(A)\})$ in $O(\log n \log A + \log^2 A)$ time slots. Such a protocol uses $O(n)$ bits of communication with $O(\log A)$ energy complexity.*

Proof. The time complexity of ECNG() comes from the $2 \log n + 2$ time slots per step taken to run ECBG() and the $2 \log A$ time slots per step used to transmit m bit by bit. As there are $\log A + 2$ steps, we obtain a $O(\log n \log A + \log^2 A)$ time complexity.

During the execution of the ECNG protocol, by Lemma 3, a device $s \in G_i$ wakes up during at most $O(1)$ time slots when running the ECBG() protocol at Step i. It wakes up during $O(\log A)$ time slots at the Step $\log A$. Thus, the ECNG protocol has $O(\log A)$ energy complexity.

By the proof of Theorem 1, one step of the protocol 2 uses $O(n/\log A)$ bits of communication. As a result, the protocol 2 has $O(n)$ bit complexity.

The ECNG() protocol succeeds if each call of ECBG() succeeds. Thus, it succeeds with a probability greater than

$$\left(1 - O\left(\frac{1}{n}\right)\right)^{O(\log n)} \geq 1 - O\left(\frac{1}{n^{9/10}}\right).$$

□

3 Energy Conserving Binary Consensus protocol

In this Section, each device s_i has an initial binary input value $b_i \in \{0, 1\}$. Having a decentralized protocol outputting 0 or 1 with a probability around 1/2, we design a binary consensus protocol as follows:

Each device s_i executes ECBG( ), then, wakes up during two more time slots t_z and $t_z + 1$ in order to communicate its input value b_i to all the other devices. If $b_i = 0$, then it beeps at t_z, otherwise, it beeps at $t_z + 1$. Then, in order to respect the validity condition of the consensus problem, if a device s_i has $b_i = 0$ (*resp.* $b_i = 1$) and does not hear a beep at $t_z + 1$ (*resp.* t_z), it outputs $b = 0$ (*resp.* $b = 1$). For the agreement condition, if s_i has $b_i = 0$ (*resp.* $b_i = 1$) and hears a beep at $t_z + 1$ (*resp.* t_z), it outputs the value b returned by ECBG( ).

These adaptations lead us to the following result:

Theorem 3 *In single hop beeping networks of size n, if up to $f \leq n - n^{1/\gamma}$ devices may fail by crashing,* ECBC( ) *terminates in $O(\log n)$ time slots, using $O(n)$ bits of communication and having $O(1)$ energy complexity.*

Proof. On one hand, by adding the previously described adaptations, all devices know all input values. The validity condition of the consensus problem is then respected. On the other hand, by Lemma 2, if up to $f \leq n - n^{1/\gamma}$ devices may fail, all correct devices have the same output. As a result, the ECBC protocol respects the agreement and the termination conditions of the consensus problem. □

Algorithm 3: ECBC( ) executed by a device s_i.

s_i sets $b \leftarrow$ ECBG( ). Noting that t_z is the next time slot after such execution,

if $b_i = 0$ **then**

  s_i beeps at t_z and listens to the network at $t_z + 1$. **if** s_i *hears beep at* $t_z + 1$ **then**

    s_i outputs b.

  else

    s_i outputs b_i.

else

  s_i listens at t_z and beeps at $t_z + 1$.

  if s_i *hears beep at* t_z **then**

    s_i outputs b.

  else

    s_i outputs b_i.

Remark 2 (The multi-valued consensus protocol) *As the extension of the binary consensus to a multi-valued consensus presented in [34], without any adaptation, our distributed random number generator can be used as a multi-valued consensus protocol that does not respect the validity condition of the consensus problem. Note that as in [34], all the correct nodes decide on the same number but it may be no node's input value. A common value* $A = O(n^\alpha), \alpha > 0$ *is given to each device and all nodes have a initial local value* $V_i = \mathrm{UAR}(\{0, 1, \ldots, A\})$. *By running the* ECNG() *protocol, each node has the same output value* $m = \mathrm{UAR}(\{0, 1, \ldots, O(A)\})$, *leading us to the following result.*

Theorem 4 *In single hop beeping networks of large size* n, *if each device has a local input value* $V_i \in \{0, 1, \ldots, A\}$ *and if up to* $f \leq n - n^{1/\gamma}$ *devices may fail by crashing, there is a multi-valued consensus protocol outputting the same value for all devices in* $O(\log n \log A + \log^2 A)$ *time slots. Such a protocol uses* $O(n)$ *bits of communication, has* $O(\log A)$ *energy complexity and does not respect the validity condition of the consensus problem.*

Proof. As all devices run the ECNG() protocol without any adaptation, the proof for the time complexity, the energy complexity and the bit complexity are given in the proof of Theorem 2.

As ECNG() outputs a common value for all correct devices, it respects the agreement and the termination conditions of the consensus problem.

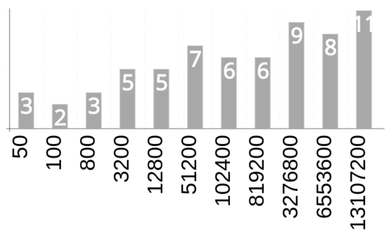
ECNG() does not respect the validity condition of the consensus problem since it can output a value not held by any correct device.

4 Simulation's results

In this Section, we show two simulation results for Lemma 2 and for Theorem 1.

4.1 Simulating the Lemma 2

The maple code of this first simulation is accessible on `http://www.irif.fr/~nixiton/witness.mw` or `http://www.irif.fr/~nixiton/witness.pdf`. We did a maple simulation for n devices, n varying from 10^2 to $O(10^7)$. For each value of n, we simulated each device choosing to witness at a time slot $t_\delta, \delta = \text{UAR}(\{1, 2, \ldots, \log n\})$. We then simulated that $n - 1.1 \log^2 n$ devices crash and that all correct devices witness at time slot t_4. After that, we simulated all devices re-choosing to witness at a time slot $t_\tau, \tau = \text{UAR}(\{1, 2, \ldots, \log n\})$. In the following Figure, for each value of n in the $x - axis$, we show the minimal number of correct devices witnessing at all time slots of the browsing procedure. We can see there that each time slot is witnessed by at least two correct device.

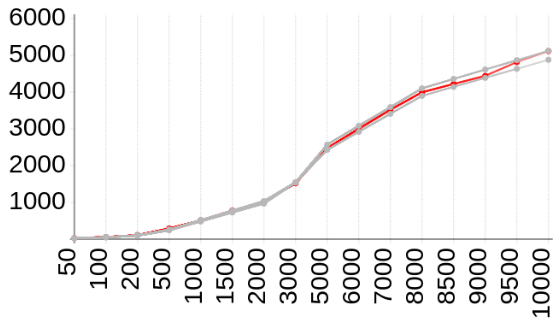


4.2 Simulating the Theorem 1

In this second simulation (`http://www.irif.fr/~nixiton/parity.mw` or `http://www.irif.fr/~nixiton/parity.pdf`), for each value of n from 50 to 10^5, we simulated each device generating one random value using the $\text{Geom}(1/2)$ distribution. Then we simulated the presence of $n - n^{1/\gamma}$ crashing devices, for a fixed $\gamma = 5$. In the following Table, we show the result of such simulations. For each value of n, we did the simulation n times, then, show how many time the result was even. If we denote this first variable by N, we can see that $N \sim n/2$ for all values of n. We then show its difference with $n/2$. We finally compare this latter value with a theoretical error rate $n^{1-1/\gamma}/13 = O(n^{1-1/\gamma})$.

n	N	$\|N - n/2\|$	$\frac{n^{1-1/\gamma}}{13}$
50	28	3	2
100	56	2	4
500	290	40	12
1000	513	13	20
3000	1521	21	47
6000	3004	4	82
10000	5115	115	122
100000	50186	186	770

In the following Figure, the $x-axis$ represents the values taken by n. The $y-axis$ represents how many times the result held by the correct devices was even when we did the simulation n times. The red line represents our simulation's result and the grey lines are the theoretical results.



5 Conclusion

In this paper, we have developed a decentralized random bit generator outputting a common binary value 0 or 1 with a probability of almost 1/2 in $O(\log n)$ time slots when up to $f \leq n - n^{1/\gamma}$ devices may crash. Our approach stands out from the commonly used approach in randomized decentralized protocols, consisting of the devices sending a message with a certain probability at each time slot. In contrast, it consists of the devices locally generating a random value and communicating in a deterministic manner at each time slot. It uses $O(n)$ bits of communication and has $O(1)$ energy complexity. We then used the latter protocol to design a random number generator outputting a value m picked uniformly at random from $\{0, 1, \ldots, A\}$ for some value $A = O(n^\alpha)$ in $O(\log n \log A + \log^2 A)$ time slots. Such a protocol has $O(n)$ bit complexity and $O(\log A)$ energy complexity. We finally developed a fault-tolerant binary consensus protocol succeeding with the same complexities. Note that all of our protocols tolerate up to $n - n^{1/\gamma}$ crashing devices for some constant $\gamma \geq 1$.

Our protocols are designed for the beeping networks but can be simulated on many other network such as the Wireless Sensor Networks, Body Area Networks or Internet of Things in order to optimize the energy conservation on these networks. Many open problems remain on this area, like finding a multi-valued consensus protocol with polynomial time complexity respecting the validity condition or optimizing the energy consumption.

References

[1] N. Agmon and D. Peleg, "Fault-tolerant gathering algorithms for autonomous mobile robots," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 56–82, 2006.

[2] K. Wallace, K. Moran, E. Novak, G. Zhou, and K. Sun, "Toward sensor-based random number generation for mobile and iot devices," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189–1201, 2016.

[3] R. Zhang, H. Moungla, and A. Mehaoua, "An energy-efficient leader election mechanism for wireless body area networks," in *2014 IEEE Global Communications Conference*, pp. 2411–2416, IEEE, 2014.

[4] Z. Li, Y. Liu, A. Liu, S. Wang, and H. Liu, "Minimizing convergecast time and energy consumption in green internet of things," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[5] Y.-J. Chang, V. Dani, T. P. Hayes, Q. He, W. Li, and S. Pettie, "The energy complexity of broadcast," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pp. 95–104, ACM, 2018.

[6] K. M. Sivalingam, M. B. Srivastava, and P. Agrawal, "Low power link and access protocols for wireless multimedia networks," in *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, vol. 3, pp. 1331–1335, IEEE, 1997.

[7] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn, "Beeping a maximal independent set," *Distributed computing*, vol. 26, no. 4, pp. 195–208, 2013.

[8] A. Cornejo and F. Kuhn, "Deploying wireless networks with beeps," in *International Symposium on Distributed Computing*, pp. 148–162, Springer, 2010.

[9] C. Scheideler, A. Richa, and P. Santi, "An o (log n) dominating set protocol for wireless ad-hoc networks under the physical interference model," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pp. 91–100, 2008.

[10] A. S. N. Khan, "Multifunctional all-in-one detachable wrist wireless mobile communication device," Nov. 12 2009. US Patent App. 12/151,607.

[11] S. Elouasbi and A. Pelc, "Deterministic rendezvous with detection using beeps," *International Journal of Foundations of Computer Science*, vol. 28, no. 01, pp. 77–97, 2017.

[12] L. Devroye, "The random bit model," in *Non-Uniform Random Variate Generation*, pp. 768–783, Springer, 1986.

[13] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation modeling and analysis*, vol. 3. McGraw-Hill New York, 2000.

[14] S. Popov, "On a decentralized trustless pseudo-random number generation algorithm," *Journal of Mathematical Cryptology*, vol. 11, no. 1, pp. 37–43, 2017.

[15] T. Nguyen-Van, T.-D. Le, T. Nguyen-Anh, M.-P. Nguyen-Ho, T. Nguyen-Van, M.-Q. Le-Tran, Q. N. Le, H. Pham, and K. Nguyen-An, "A system for scalable decentralized random number generation," in *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*, pp. 100–103, IEEE, 2019.

[16] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 403–412, IEEE, 2019.

[17] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[18] D. Beaver and N. So, "Global, unpredictable bit generation without broadcast," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 424–434, Springer, 1993.

[19] A. Suciu, D. Lebu, and K. Marton, "Unpredictable random number generator based on mobile sensors," in *2011 IEEE 7th international conference on intelligent computer communication and processing*, pp. 445–448, IEEE, 2011.

[20] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 63–70, IEEE, 2005.

[21] S. Gilbert and D. R. Kowalski, "Distributed agreement with optimal communication complexity," in *Proceedings of the twenty-first annual*

ACM-SIAM symposium on Discrete Algorithms, pp. 965–977, SIAM, 2010.

[22] D. R. Kowalski and J. Mirek, "On the complexity of fault-tolerant consensus," in *International Conference on Networked Systems*, pp. 19–31, Springer, 2019.

[23] S. Toueg, K. J. Perry, and T. Srikanth, "Fast distributed agreement," *SIAM Journal on Computing*, vol. 16, no. 3, pp. 445–457, 1987.

[24] B. Chor, M. Merritt, and D. B. Shmoys, "Simple constant-time consensus protocols in realistic failure models," *Journal of the ACM (JACM)*, vol. 36, no. 3, pp. 591–614, 1989.

[25] E. S. Amdur, S. M. Weber, and V. Hadzilacos, "On the message complexity of binary byzantine agreement under crash failures," *Distributed Computing*, vol. 5, no. 4, pp. 175–186, 1992.

[26] S. Ashrafi, M. Malmirchegini, and Y. Mostofi, "Binary consensus for cooperative spectrum sensing in cognitive radio networks," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pp. 1–6, IEEE, 2011.

[27] A. Abdaoui and T. M. Elfouly, "Distributed binary consensus algorithm in wireless sensor networks with faulty devices," in *2013 7th IEEE GCC Conference and Exhibition (GCC)*, pp. 495–500, IEEE, 2013.

[28] K. Hounkanli, A. Miller, and A. Pelc, "Global synchronization and consensus using beeps in a fault-prone mac," in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pp. 16–28, Springer, 2016.

[29] A. Czumaj and P. Davies, "Communicating with beeps," *Journal of Parallel and Distributed Computing*, vol. 130, pp. 98–109, 2019.

[30] P. Brandes, M. Kardas, M. Klonowski, D. Pająk, and R. Wattenhofer, "Fast size approximation of a radio network in beeping model," *Theoretical Computer Science*, 2017.

[31] B. S. Chlebus, G. De Marco, and M. Talo, "Naming a channel with beeps," *Fundamenta Informaticae*, vol. 153, no. 3, pp. 199–219, 2017.

[32] A. Casteigts, Y. Métivier, J. M. Robson, and A. Zemmari, "Counting in one-hop beeping networks," *Theoretical Computer Science*, vol. 780, pp. 20–28, 2019.

[33] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, 2020.

[34] R. Turpin and B. A. Coan, "Extending binary byzantine agreement to multivalued byzantine agreement," *Information Processing Letters*, vol. 18, no. 2, pp. 73–76, 1984.

[35] D. Dolev, M. J. Fischer, R. Fowler, N. A. Lynch, and H. R. Strong, "An efficient algorithm for byzantine agreement without authentication," *Information and Control*, vol. 52, no. 3, pp. 257–274, 1982.

[36] G. Neiger, "Distributed consensus revisited," *Information processing letters*, vol. 49, no. 4, pp. 195–201, 1994.

[37] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[38] C. Dwork, D. Shmoys, and L. Stockmeyer, "Flipping persuasively in constant expected time," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 222–232, IEEE, 1986.

[39] S. Micali and T. Rabin, "Collective coin tossing without assumptions nor broadcasting," in *Conference on the Theory and Application of Cryptography*, pp. 253–266, Springer, 1990.

[40] K. Wallace, K. Moran, E. Novak, G. Zhou, and K. Sun, "Toward sensor-based random number generation for mobile and iot devices," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189–1201, 2016.

[41] G. Lo Re, F. Milazzo, and M. Ortolani, "Secure random number generation in wireless sensor networks," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 15, pp. 3842–3862, 2015.

[42] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, vol. 19, no. 7, pp. 395–404, 1976.

[43] K. Nakano and S. Olariu, "A survey on leader election protocols for radio networks," in *Proceedings International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02*, pp. 71–76, IEEE, 2002.

[44] M. Ghaffari and B. Haeupler, "Near optimal leader election in multi-hop radio networks," in *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pp. 748–766, Society for Industrial and Applied Mathematics, 2013.

[45] D. E. Willard, "Log-logarithmic selection resolution protocols in a multiple access channel," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 468–477, 1986.

[46] R. Zhang and J. Yu, "Energy-efficient and reliable leader election mechanisms for wbsns," in *Energy-Efficient Algorithms and Protocols for Wireless Body Sensor Networks*, pp. 11–38, Springer, 2020.

[47] A. Czumaj and P. Davies, "Exploiting spontaneous transmissions for broadcasting and leader election in radio networks," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pp. 3–12, ACM, 2017.

[48] M. M. Halldórsson, S. Holzer, E. A. Markatou, and N. Lynch, "Leader election in sinr model with arbitrary power control," *Theoretical Computer Science*, 2019.

[49] M. Ghaffari, N. Lynch, and S. Sastry, "Leader election using loneliness detection," *Distributed Computing*, vol. 25, no. 6, pp. 427–450, 2012.

[50] J. Havil, "Gamma: exploring euler's constant," *The Australian Mathematical Society*, p. 250, 2003.