# Beeping a Random Bit

Ny Aina Andriambolamalala, Vlady Ravelomanana

*IRIF UMR CNRS 8243 — University of Paris — France*

*Email addresses:* Ny-Aina.Andriambolamalala@irif.fr (Ny Aina Andriambolamalala), vlad@irif.fr (Vlady Ravelomanana)

*Preprint submitted to Elsevier* *September 19, 2018*

# Beeping a Random Bit

Ny Aina Andriambolamalala, Vlady Ravelomanana
*IRIF UMR CNRS 8243 — University of Paris — France*

**Abstract**

We consider networks of processors which interact with beeps. In a single-hop network, nodes can communicate synchronously with each other by broadcasting one bit through a shared channel. In each time slot, all nodes receive feedback from the channel and in the Beeping Model, each node learns whether zero (Null) or at least one (Beep) node has transmitted. In such model, a procedure generating uniformly at random a common bit $b \in \{0, 1\}$ can be used as a subroutine to generate random numbers according to various distributions. In this work, we design a randomized distributed random bit generator working with high probability in $O(\log \log n)$ time slots and satisfying $\lim_{n\to+\infty} \mathbb{P}[\text{bit} = 0] = \lim_{n\to+\infty} \mathbb{P}[\text{bit} = 1] = \frac{1}{2}$.

*Keywords:* Distributed Computing, Beep Model, Randomized Algorithms

## 1. Introduction

Pseudo random number generators are important routines of any computer system [7, 8]. In distributed computing [9], the problem consists in finding a protocol that allows all the participants of a given network to generate a common value picked from some set according to a given distribution.

Recently, Cornejo and Kuhn [3] have introduced the *beeping* communication model or the *beeping model* where the processors have very limited communication capabilities. In this paper, we study a *single-hop* network of nodes with the same communication model : the transmission of each node reaches all other nodes (the underlying graph is complete). We also assume that nodes are anonymous (identical and indistinguishable) but each node

*Email addresses:* Ny-Aina.Andriambolamalala@irif.fr (Ny Aina Andriambolamalala), vlad@irif.fr (Vlady Ravelomanana)

has the ability to generate locally a random number. The system under consideration is synchronous and in each discrete time slot or round, each node independently decides whether to transmit to the shared channel or not. In every time slot the channel can be in one of the two following states :
- NULL when no node is transmitting and
- BEEP if at least one node is transmitting.
As a common assumption in this model [1, 3], we also assume that all nodes receive the state of the channel immediately after each communication round.

In this paper, we are interested in designing a distributed protocol in a single-hop beeping network of unknown size that outputs 0 or 1 with probability $\frac{1}{2}$. Such a question

(i) is not trivial due to the fact that beeping networks have an extremely harsh model of communication

(ii) and quite natural since the uniform generation of a random bit can be used to simulate much more involved distributions (see Devroye [4]).

## 2. Related works

In single-hop beeping networks, two important problems include the approximation of the size of the network or the design of leader election algorithms.

Under the same settings as ours, Brandes *et al.* [1] designed an efficient probabilistic counting algortithm working with probability greater than $1 - O(n^{-1})$ in $O(\log n)$ time slots. Their protocol outputs a linear approximation of the number $n$ of the nodes whose precision can be tuned. They have also shown that $\Omega(\log n)$ time slots are required for any randomized protocol able to give linear approximation of the size of the network with high probability.

Leader election consists in the task of agreeing on the election of a single node in a network. Many distributed routines require a distinguished node to organize the tasks in the network and leader election is the subroutine that provides such organizer. As it is one of the most fundamental problems in distributed computing, leader election has been extensively studied over the years under many models of communication and in many different network settings [10, 11, 5, 13]. In particular, [6] proved that $\Omega(\log n)$ time slots is required to have a leader election protocol succeeding with probability higher than $1 - O(n^{-c})$ (for some $c > 0$) in radio networks with collision detection (RNCD). Note that RNCD is a distributed model close to the beeping model.

The RNCD model has been introduced in [2] and under this setting, nodes can send message other a shared channel and in each time slot the status of the common channel is NULL (no message), SINGLE (exactly one sender) or COLLISION (at least two senders). It is attempting to use a leader election algorithm to furnish a solution to the current problem since the elected node can dictate to the others the value of its own bit. We remark that due to the logarithmic lower-bound in [6], leader election algorithms can provide correct solution to our problem but they required at least $\Omega(\log n)$ rounds to work with high probability before generating a single random bit.

## 3. Result and main ideas of the proof

Throughout this paper, $n$ designs the number of nodes of the network and in the scenario we are considering, $n$ needs not to be known beforehand by the processors. Under these setting, we have the following result.

**Theorem 1.** *In a single-hop beeping network of size $n$, there exists a randomized algorithm that with probability at least $1 - O\left(\frac{1}{n^{\varepsilon}}\right)$ (for any $\varepsilon > 0$) outputs 0 or 1 with probability tending to $1/2$ within $O(\log \log n)$ time slots.*

The main ideas behind are the following. *a)* Since $n$ is not known by the nodes, an upper bound of $\log n$ is computed in $O(\log \log n)$ rounds. *b)* Each node $v$ generates locally a discrete random variable $X_v$ according to the same distribution $X$. Our choice of this distribution is such that the maximum of the $n$ random variables grows as a polylogarithm in $n$ so that this maximum value can be found by means of a simple binary search giving raise to a protocol with log-logarithmic time complexity. The parity of the maximum can be then used by all the nodes of the whole network so that all the participants agree on the value of a single random bit. We prove that our algorithm outputs 0 or 1 with probability $1/2 \pm o(1)$ but to output 0 or 1 with probability $1/2$, the well-known von Neumann procedure can be used to obtain a fair flip [12].

## 4. Algorithms

Throughout the rest of the paper, we will use the following notations. Let

$$p_k = e^{-\sqrt{k}} - e^{-\sqrt{k+1}},\ k \geq 0 \tag{1}$$

and let $X$ be a discrete random variables such that

$$\mathbb{P}[X = k] = p_k \,. \tag{2}$$

We have the following technical result.

**Lemma 1.** *Let $X_1, X_2, \cdots, X_n$ be $n$ independent copies of $X$. Then for any fixed $\varepsilon > 0$ with probability greater than $1 - O(n^{-\varepsilon})$ all the $X_i$s are less than $(1+\varepsilon)^2 \log^2 n$.*

*Proof.* By independence and for $k$ sufficiently large, we have

$$\mathbb{P}[\forall i,\, X_i < k] = \mathbb{P}[X_1 < k]^n = \left(\sum_{i=0}^{k-1} p_k\right)^n = \left(1 - e^{-\sqrt{k}}\right)^n \geq e^{-2ne^{-\sqrt{k}}} \,.$$

The proof is completed by letting $k = (1+\varepsilon)^2 \log^2 n$.
Suppose now that each node $w$ of the network generates a (local) random variable $X_w$ such that $\mathbb{P}[X_w = k] = p_k$. We can then use the lemma above to design a distributed algorithm (working under the beep model) that allows the nodes to agree on a common value of order $O(\log \log n)$ which reflects the logarithm of the maximum of the $X_i$s. The deterministic protocol MaxRV given below computes distributively such a common value $j$ s.t. $2^j \geq \max_i X_i$.

**Algorithm 1:** MaxRVat a node $w$.

**Input :** Each node $w$ holding a local random variable $X_w$ s.t. $\mathbb{P}[X_w = k] = p_k$.

**Output:** A common upper bound of $\max(\lceil \log_2 X_w \rceil)$.

```
for i ∈ {0, 1, ··· , ⌈log2 X_w⌉} do
    w beeps.                          /⋆ w beeps until 2^i ≥ X_w ⋆/
w stores the last round i when w beep or heard a Beep.
if w does not beep and learns that zero node has transmitted (Null)
  then
    w quits the protocol.
```

Observe that all the nodes quit the protocol as soon as all the participants remain silent. Thus, using Lemma 1 we have the following.

**Lemma 2.** *For any fixed $\varepsilon > 0$, with probability at least $1 - O(n^{-\varepsilon})$ after $2\log_2 \log n + O(1)$ time slots, the protocol* MaxRV *terminates with each node being aware of a common value $M$ such that $\forall w$, $2^M \geq X_w$.*

We are now ready to write the protocol to localize the node(s) $w$ such that $X_w = \max_i X_i$. Its principle is based on a simple binary search. In what follows, Test($a$, $b$) is a local operation that is called at some round $t$. It works exactly in one round. During such a round, a node $s$ beeps if its random variable $X_s$ verifies $a \leq X_s \leq b$, otherwise $s$ listens. In any round, Test($a$, $b$) outputs Beep if and only if at least one node of the network beeps during this round (that is at least one node $v$ has $a \leq X_v \leq b$). Observe that during such a round, all beeping nodes know they just beeped and all listening nodes can check the status of the common shared channel : all the nodes are aware of the result of the subroutine Test($a$, $b$).

**Algorithm 2:** FindMax: a binary search based distributed protocol to find the maximum of bounded discrete random variables.

**Input :** $B$ a positive integer (upper bound of all the involved random variables).

**Output:** Each node $v$ with a status($v$) $\in$ {Maximum, Eliminated}.

All nodes set INF $= 0$ and SUP $= B$.
**while** INF $\neq$ SUP **do**
  **if** Test($\lceil$(INF + SUP)/2$\rceil$, SUP) = Null **then**
    SUP $= \lceil \frac{\text{INF}+\text{SUP}}{2} \rceil - 1$
  **else**
    INF $= \lceil \frac{\text{INF}+\text{SUP}}{2} \rceil$
**For each node s if** $X_s =$ SUP **then**
  status($s$) = Maximum
**else**
  status($s$) = Eliminated

We can now design the main subroutine that can output a random bit in $O(\log \log n)$ time slots.

**Algorithm 3:** MAINPROTOCOL at node $w$.

**Output:** 0 or 1

16 Each node $w$ generates $X_w$ such that $\mathbb{P}[X_w = k] = p_k$ as defined by (1).

17 Each node computes locally the same value $B = 2^{\text{MAXRV()}}$. /$\star$ in $O(\log\log n)$ rounds $\star$/

18 Each node invokes FINDMAX($B$). /$\star$ in $O(\log\log n)$ rounds $\star$/

19 The nodes having the common maximum value $M$ know their status and send the parity of $M$ to all the others.

The MAINPROTOCOL algorithm has the following properties which prove Theorem 1.

**Lemma 3.** *(i) For any $\varepsilon > 0$ with probability at least $1 - O(n^{-\varepsilon})$* MAINPROTOCOL *terminates in $O(\log\log n)$ time slots and (ii) it returns 0 with probability at least $\frac{1}{2} - O\left(\frac{1}{\log n}\right)$.*

*Proof.* The part (i) is a consequence of Lemma 1. For part (ii), let $P_0$ be the probability that MAINPROTOCOL returns 0. As the probability that the maximum of the $X_i$s is even satisfies

$$\mathbb{P}[\max_{1\le i\le n} X_i \text{ is even}] \ge \mathbb{P}[\max_{1\le i\le n} X_i \text{ is even and is unique}] \tag{3}$$

we have

$$P_0 \ge \sum_{k=1}^{\infty} \binom{n}{1} p_{2k} \left(\sum_{i=0}^{2k-1} p_i\right)^{n-1} \ge \sum_{k=1}^{\infty} \left(1 - e^{-\sqrt{2k}}\right)^n . \tag{4}$$

As $k$ is large, we have

$$p_{2k} = \frac{e^{-\sqrt{2k}}}{2\sqrt{2k}} \left(1 - O\left(\frac{1}{\sqrt{k}}\right)\right) \text{ and } \left(1 - e^{-\sqrt{2k}}\right)^n \ge \exp\left(-ne^{-\sqrt{2k}} - 2ne^{-2\sqrt{2k}}\right). \tag{5}$$

By choosing $L$ such that $ne^{-\sqrt{8L}} \ll \frac{1}{\sqrt{L}}$, we then get

$$P_0 \ge \sum_{k=L}^{\infty} n \frac{e^{-\sqrt{2k}}}{2\sqrt{2k}} e^{-ne^{-\sqrt{2k}}} \left(1 - O\left(\frac{1}{\sqrt{L}}\right)\right) . \tag{6}$$

Using the Euler-Maclaurin summation formula, after some algebra we get

$$\sum_{k=L}^{\infty} n\frac{e^{-\sqrt{2k}}}{2\sqrt{2k}}e^{-ne^{-\sqrt{2k}}} = \frac{1}{2} - \frac{1}{2}e^{-ne^{-\sqrt{2L}}} + O\left(\frac{ne^{-\sqrt{2L}}}{\sqrt{L}}\, e^{-ne^{-\sqrt{2L}}}\right) \qquad (7)$$

To complete the proof, it suffices to choose $L = O(\log^2 n)$ such that $ne^{-\sqrt{2L}} = n^c$ for some $c > 0$ but $ne^{-\sqrt{8L}} \ll \frac{1}{\sqrt{L}}$.

## 5. Conclusion

In the Beeping Model [3] which is typically an extremely harsh broadcast model relying only on carrier sensing, we design a randomized algorithm that is able to produce a random bit within $O(\log \log n)$ time slots. The work presented in this paper shows that beeping networks is powerful and efficient enough to generate random numbers as our algorithm can be used as a way to simulate/generate random real numbers represented in floating-point with arbitrary precision [4] and random number generators are fundamental tools for various simulations and applications of Monte Carlo methods [7, 8].


[1] P. Brandes, M. Kardas, M. Klonowski, D. Pająk and R. Wattenhofer (2016). "Approximating the size of a radio network in beeping model", In *Proc. Int. Coll. on Structural Information and Communication Complexity – SIROCCO*, 358 – 373.

[2] I. Chlamtac and S. Kutten (1985). On broadcasting in radio networks–problem analysis and protocol design, *IEEE Transactions on Communications*, 33(12), 1240 – 1246.

[3] A. Cornejo and F. Kuhn (2010). "Deploying wireless networks with beeps", In *Proc. of the 24th Int. Symp. on Distr. Computing – DISC*, 148 -âĂŞ 162.

[4] L. Devroye (1986). "Non-Uniform Random Variate Generation", `http://www.nrbook.com/devroye/`

[5] M. Ghaffari and B. Haeupler (2013). "Near optimal leader election in multi-hop radio networks", In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, 748 – 766.

[6] M. Ghaffari, N. Lynch and S. Sastry (2012). "Leader election using loneliness detection", *Distributed Computing*, 25(6), 427 – 450.

[7] D.E. Knuth (1998). "The Art of Computer Programming, Volume 2: Seminumerical Algorithms", Addison-Wesley, Reading, MA

[8] A.M. Law (2014). "Simulation Modeling and Analysis", McGraw-Hill, New York

[9] N. Lynch (1996). "Distributed Algorithms", Morgan Kaufmann Publishers.

[10] R. Metcalfe and D. Boggs (1976). "Ethernet: Distributed packet switching for local computer networks", *Communications of the ACM*, 19(7), 395 – 404.

[11] K. Nakano and S. Olariu (2002). "A survey on leader election protocols for radio networks", In *International Symposium on Parallel Architectures, Algorithms and Networks. IEEE I-SPAN*, 71 – 76.

[12] J. von Neumann (1951). "Various techniques used in connection with random digits", *National Bureau of Standards Applied Math Series*, 12, 36 – 38.

[13] D. E. Willard (1986). "Log-logarithmic selection resolution protocols in a multiple access channel". *SIAM J. on Comp., 15(2),* 468-477.