Global Synchronization and Consensus Using Beeps in a Fault-Prone MAC

Kokouvi Hounkanli[1], Avery Miller[2(✉)], and Andrzej Pelc[1]

[1] Université du Québec en Outaouais, Gatineau, Canada
[2] University of Manitoba, Winnipeg, Canada
avery@averymiller.ca

Abstract. Global synchronization is an important prerequisite to many distributed tasks. Communication between processors proceeds in synchronous rounds. Processors are woken up in possibly different rounds. The clock of each processor starts in its wakeup round showing local round 0, and ticks once per round, incrementing the value of the local clock by one. The global round 0, unknown to processors, is the wakeup round of the earliest processor. Global synchronization (or establishing a global clock) means that each processor chooses a local clock round such that their chosen rounds all correspond to the same global round t.

We study the task of global synchronization in a Multiple Access Channel (MAC) prone to faults, under a very weak communication model called the *beeping model*. Some processors wake up spontaneously, in possibly different rounds decided by an adversary. In each round, an awake processor can either listen, i.e., stay silent, or beep, i.e., emit a signal. In each round, a fault can occur in the channel independently with constant probability $0 < p < 1$. In a fault-free round, an awake processor hears a beep if it listens in this round and if one or more other processors beep in this round. A processor still dormant in a fault-free round in which some other processor beeps is woken up by this beep and hears it. In a faulty round nothing is heard, regardless of the behaviour of the processors. An algorithm working with error probability at most ϵ, for a given $\epsilon > 0$, is called ϵ-*safe*. Our main result is the design and analysis, for any constant $\epsilon > 0$, of a deterministic ϵ-safe global synchronization algorithm that works in constant time in any fault-prone MAC using beeps.

As an application, we solve the consensus problem in a fault-prone MAC using beeps. Processors have input values from some set V and they have to decide the same value from this set. If all processors have the same input value, then they must all decide this value. Using global synchronization, we give a deterministic ϵ-safe consensus algorithm that works in time $O(\log w)$ in a fault-prone MAC, where w is the smallest input value of all participating processors. We show that this time cannot be improved, even when the MAC is fault-free.

Keywords: Global synchronization · Consensus · Multiple access channel · Fault · Beep

Partially supported by NSERC discovery grant 8136 – 2013 and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

M. Chrobak et al. (Eds.): ALGOSENSORS 2016, LNCS 10050, pp. 16–28, 2017.
DOI: 10.1007/978-3-319-53058-1_2

1 Introduction

1.1 The Problem

Global synchronization is an important prerequisite to many distributed tasks. Communication between processors proceeds in synchronous rounds. Processors are woken up in possibly different rounds. The clock of each processor starts in its wakeup round showing local round 0, and ticks once per round, incrementing the value of the local clock by one. The global round 0, unknown to processors, is the wakeup round of the earliest processor. Global synchronization (or establishing a global clock) means that each processor chooses a local clock round such that their chosen rounds all correspond to the same global round t. Achieving global synchronization permits to assume that in a subsequent task all processors start in the same round. This assumption is often used in solving various distributed problems. Global synchronization was assumed, e.g., in [10] for broadcasting and gossiping, in [14] for leader election, in [30] for minimum connected dominating set construction, and in [20] for the conflict resolution and the membership problem.

1.2 Model Description

We study the task of global synchronization in a fault-prone Multiple Access Channel (MAC): all processors can communicate directly, i.e., the underlying communication graph is complete. We adopt a very weak communication model called the *beeping model*. We assume that processors are fault free, while the MAC is prone to random faults. Faults in the channel may be due to some random noise occurring in the background. We assume that processors in the channel do not have access to any random generator. Some processors wake up spontaneously, in possibly different rounds decided by an adversary. In each round, an awake processor can either *listen*, i.e., stay silent, or *beep*, i.e., emit a signal. In each round, a fault can occur in the channel independently with constant probability $0 < p < 1$. The value of p is known by all processors. In a fault-free round, an awake processor hears a beep if it listens in this round and if one or more other processors beep in this round. In a faulty round, nothing is heard regardless of the behaviour of the processors. A processor that is still dormant in a fault-free round in which some other processor beeps is woken up by this beep and hears it.

The beeping model was introduced in [9] for vertex coloring, used in [1] to solve the MIS problem, and later used in [14,16] to solve leader election. The beeping model is widely applicable, as it makes small demands on communicating devices by relying only on carrier sensing. In fact, as mentioned in [9], beeps are an even weaker way of communicating than using one-bit messages: one-bit messages allow three different states (0,1 and no message), while beeps permit to differentiate only between a signal and its absence.

It has been noted that communication by beeps is a good model for various kinds of biological networks that arise in nature [24]. Given this motivation, it is

especially important to understand how distributed coordination and communication can be carried out in a robust way to deal with unpredictable interference from the environment. We study deterministic global synchronization algorithms working in a probabilistic fault-prone MAC, which work with error probability at most ϵ, for a given $\epsilon > 0$. Such algorithms are called ϵ-*safe*. We assume that all processors know the value of ϵ.

1.3 Application

As an application of our global synchronization algorithm we solve the consensus problem in a fault-prone MAC, using beeps. Consensus is one of the fundamental tasks studied in distributed computing [21]. Processors have input values from some set V, and they have to decide the same value from this set. If all processors have the same input value, then they must all decide this value. Consensus has mostly been studied in the context of fault-tolerance. Either the communication between processors is assumed prone to faults [18,27,28], or processors themselves can be subject to crash [8,23] or Byzantine [25] faults. In the present paper, we study a scenario falling under the first of these variants.

We study the task of consensus defined precisely as follows [21]. Processors have input values from some set V of non-negative integers. The goal for all processors is to satisfy the following requirements.
Termination: all processors must output some value from V.
Agreement: all outputs must be equal.
Validity: if all input values are equal to v, then all output values must be equal to v.[1]

1.4 Our Results

Our main result is the design and analysis, for any constant $\epsilon > 0$, of a deterministic ϵ-safe global synchronization algorithm that works in constant time in any fault-prone MAC using beeps.

As an application we solve the consensus problem in a fault-prone MAC using beeps. Using global synchronization, we give a deterministic ϵ-safe consensus algorithm that works in time $O(\log w)$ in a fault-prone MAC, where w is the smallest input value of all participating processors. We show that this time cannot be improved, even when the MAC is fault-free. Moreover, we show how to reach consensus in the same round. Hence, as formulated in [23], we reach "double agreement, one on the decided value (data agreement) and one on the decision round (time agreement)".

Several proofs are omitted and will appear in the full version of the paper.

[1] Some authors use a stronger validity condition in which the output values must always be one of the input values, even if these are non-equal. In this paper we use the above formulation from [21].

1.5 Related Work

The Multiple Access Channel (MAC) is a popular and well-studied medium of communication. Most research concerning the MAC has been done under the radio communication model in which processors can send an entire message in a single round, and this message is heard by other processors if exactly one processor transmits, and all others listen in this round. This communication model is incomparable to the beeping model: on the one hand it is much stronger, as large messages (and not only beeps) can be sent in a single round, but on the other hand it is weaker, as it requires a unique transmitter in a round to make the transmission successful, while in the beeping model many beeps may be heard simultaneously. Broadcasting was studied in a MAC under the radio model, both in the deterministic [11,19] and in the randomized setting [5,29]. The throughput of a MAC under the radio model was studied in the situation where the channel could be jammed by an adversary, i.e., a collision is caused on the channel by the adversary at arbitrary times [2]. In [4], the authors give a randomized protocol for a MAC under the radio model that achieves constant throughput in the presence of an adversary that can arbitrarily jam the channel in a $(1-\epsilon)$-fraction of the time slots.

The differences between local and global clocks for the wake-up problem were first studied in [15] and then in [6,9,12]. The communication model used in these papers was that of radio networks in which the main challenge is the occurrence of collisions between simultaneously received messages. Global synchronization is often used in the study of broadcasting in radio networks (cf. [12]). The previously cited papers [10,14,30] used the assumption of global synchronization in the beeping model in multi-hop networks. In [22] the authors compared different variations of the beeping model, also assuming global synchronization. The authors of [20] used the assumption of global synchronization in the beeping model in a multiple access channel. All these papers assumed that communication is fault free.

Consensus is a classic problem in distributed computing, mostly studied assuming that processors communicate by shared variables or through message passing networks [3,21]. See the recent book [26] for a comprehensive survey of the literature on consensus, mostly concerning processor faults. In [17], the authors showed a randomized consensus for crash faults with optimal communication complexity. In [8], the feasibility and complexity of consensus in a multiple access channel (MAC) with simultaneous wake-up and crash failures were studied in the context of different collision detectors. Consensus (without faults) in a MAC with different wake-up times was studied in [13]. The authors also investigated the impact of a global clock on the time efficiency of consensus. Consensus in the quantum setting has been studied, e.g., in [7]. To the best of our knowledge, neither global synchronization nor consensus with faulty beeps have ever been studied before.

2 Global Synchronization

In this section, we provide an algorithm `GlobalSync` that establishes a global clock. Upon its wake-up, each processor in the channel executes `GlobalSync` with its local clock initialized to 0. The round in which the first wake-up occurs is defined as global round 0. Processors are not aware of the relationship between their local clock values and this global round. Establishing a global clock means that all processors in the channel exit `GlobalSync` in the same global round.

Fix any constant $\epsilon > 0$. Let γ be a constant such that $p^\gamma < \frac{\epsilon}{4}$. Hence, in a sequence of γ consecutive rounds of beeps, at least one of these beeps occurs in a fault-free round with probability at least $1 - \frac{\epsilon}{4}$.

We describe Algorithm `GlobalSync` whose aim is to ensure that all processors agree on a common global round, i.e. they establish a global clock. At a high level, the algorithm proceeds as follows. A processor that wakes up spontaneously beeps periodically trying to wake up all other processors that are still dormant. These beeps will be called *alarm beeps*. They are separated by time intervals of increasing size, which prevents an adversary from setting wake-up times so that all alarm beeps are aligned. In the intervals between alarm beeps, the processor is waiting for a response from other processors to indicate that they heard an alarm beep. If a large enough number of such intervals occur without any response, then the processor assumes that the entire channel was woken up at the same time, and a global round is chosen as the round in which the next alarm beep is scheduled. Otherwise, if a beep was heard in one of these intervals, the processor listens for 2γ consecutive rounds and then beeps for 2γ consecutive rounds. Similarly, a processor woken up by a beep listens for 2γ consecutive rounds and then beeps for 2γ consecutive rounds. The global round chosen by the algorithm is the round $r + 4\gamma + 1$, where r is the first round when an alarm beep was heard by some processor. The difficulty is for each processor to determine the round r. This is because, when a beep is heard, there are two possible cases: such a beep may be an alarm beep from another processor, or may be in response to an alarm beep. We overcome this difficulty as follows. Time is divided into blocks of 2γ consecutive rounds. If a single beep is heard in a block, the processor concludes that it was an alarm beep; if more than one beep is heard in a block, the processor concludes that these beeps were in response to an alarm beep. We will prove that such conclusions are correct with sufficiently high probability. Finally, each processor considers the first round s in which it heard a beep. If this beep was an alarm beep, the processor sets $r = s$. If this beep was in response to an alarm beep, the processor sets r to be the most recent round before s in which it beeped.

We now provide the details of Algorithm `GlobalSync`. The following procedure provides an aggregate count of the beeps recently heard by a processor. More specifically, for a given round t', the next 4γ rounds are treated as two blocks of 2γ rounds each, and for each block, the cases of 0, 1, or more beeps are distinguished.

Algorithm 1. `listenVector`(t')

$h_1 \leftarrow 0$
$h_2 \leftarrow 0$
$num_1 \leftarrow$ number of beeps heard in rounds $t', t'+1, \ldots, t'+2\gamma-1$
$num_2 \leftarrow$ number of beeps heard in rounds $t'+2\gamma, \ldots, t'+4\gamma-1$
if $num_1 = 1$, then $h_1 \leftarrow 1$
if $num_1 > 1$, then $h_1 \leftarrow *$
if $num_2 = 1$, then $h_2 \leftarrow 1$
if $num_2 > 1$, then $h_2 \leftarrow *$
return $[h_1\ h_2]$

Below we give the pseudocode of Algorithm `GlobalSync` using the above procedure.

Algorithm 2. `GlobalSync`

if woken up by a beep in some round *heard*: ▷ woken up by beep
    beep 2γ consecutive rounds starting at round $heard + 2\gamma + 1$
    $syncRound \leftarrow heard + 4\gamma + 1$
else: ▷ woken up spontaneously
    $i \leftarrow 0$
    $myNextBeep \leftarrow 0$
    repeat:
        $myCurrentBeep \leftarrow myNextBeep$
        beep in round $myCurrentBeep$
        $i \leftarrow i + 1$
        $myNextBeep \leftarrow myCurrentBeep + 4\gamma + i$
    until $(i = 3\gamma)$ or (a beep is heard in one of $\{myCurrentBeep + 1, \ldots, myNextBeep - 1\}$)
    if $i = 3\gamma$:
        $syncRound \leftarrow myNextBeep$
    else:
        $heard \leftarrow$ first round after $myCurrentBeep$ in which a beep was heard
        $[h_1\ h_2] = listenVector(myCurrentBeep + 1)$
        if $[h_1\ h_2] \in \{[0\ 0], [0\ 1], [1\ 0], [1\ 1], [1\ *]\}$:
            beep 2γ consecutive rounds starting at round $heard + 2\gamma + 1$
            $syncRound \leftarrow heard + 4\gamma + 1$
        if $[h_1\ h_2] \in \{[0\ *], [*\ 0], [*\ 1], [*\ *]\}$:
            $syncRound \leftarrow myCurrentBeep + 4\gamma + 1$
wait until round *syncRound* and **exit**

In the analysis of Algorithm `GlobalSync` we refer to global rounds, but it should be recalled that processors in the channel do not have access to the global clock values: all a processor sees is its local clock. The following fact follows from the algorithm description by induction on i.

Fact 1. *At the end of each loop iteration i, the variable myNextBeep is equal to $4\gamma i + \sum_{k=1}^{i} k$. Further, if a processor is woken up at time t, then, at the end of each loop iteration i, myNextBeep is equal to the local clock value corresponding to the global round $t + 4\gamma i + \sum_{k=1}^{i} k$.*

We say that a processor is *lonely in round t* if it has not heard a beep in any round up to and including round t. Using Fact 1, we can determine the number of rounds that elapse before a lonely processor beeps a given number of times.

Fact 2. *Suppose that a processor v wakes up spontaneously in round t_1. If v is lonely in round $t_1 + 4\gamma i + i(i+1)/2$, then v has beeped exactly i times before this round.*

In order to prove the correctness of the algorithm, we first consider the case when all processors wake up spontaneously in the same round.

Lemma 1. *Suppose that all processors wake up spontaneously in the same global round t_1. With probability 1, all processors terminate their execution of* `GlobalSync` *in global round $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$.*

Proof. Since every processor is woken up spontaneously in global round t_1, the **if** condition on line 1 evaluates to false at every processor. Therefore, all processors execute the loop at line 7. In particular, this means that all processors beep in their local round 0. By Fact 1, at the end of each loop iteration, the variable *myNextBeep* at every processor is equal to the local clock value corresponding to the global round $t_1 + 4\gamma i + \sum_{k=1}^{i} k$. It follows that all processors beep in the same rounds. In particular, this means that no processor ever hears a beep. Thus, at every processor, the loop exits with $i = 3\gamma$. So, the **if** condition on line 13 evaluates to true. By line 14, each processor sets *syncRound* to the value $4\gamma(3\gamma) + \sum_{k=1}^{3\gamma} k = 12\gamma^2 + (3\gamma)(3\gamma+1)/2$, which is their local clock value that corresponds to the global round $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$. Therefore, all processors terminate their execution of `GlobalSync` in global round $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$. □

Note that, when our algorithm is executed in the case where all processors wake up spontaneously in the same round, no processor ever hears a beep, and, after a fixed length of silence, all processors terminate their execution of `GlobalSync`. In the case where not all processors wake up spontaneously in the same round, if the same fixed length of silence is observed by all processors, then, again, all processors will terminate their execution of `GlobalSync`, but this time in different rounds. This would be a bad case for our algorithm. We now show that, with sufficiently high probability, such a bad case does not occur, i.e., that there exists some round t^* in which a beep is heard by some processor.

Lemma 2. *Suppose that not all processors wake up spontaneously in the same round, and suppose that the first spontaneous wake-up occurs in some round t_1. With probability at least $(1 - \frac{\epsilon}{2})$, there exists a global round $t^* \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$ in which all of the following hold: no processor has terminated its execution of* `GlobalSync`*, at least one processor beeps, at least one processor listens, and no fault occurs.*

We now proceed to prove the correctness of our algorithm for the case where not all processors wake up spontaneously in the same round. We will be able to do so when there exists a global round t^* satisfying the conditions specified in Lemma 2. The next lemma shows that all processors terminate their execution of `GlobalSync` in the same global round soon after t^*.

Lemma 3. *Suppose that not all processors wake up spontaneously in the same round. Let t^* be the first global round in which all of the following hold: no processor has terminated its execution of `GlobalSync`, at least one processor beeps, at least one processor listens, and no fault occurs. With probability at least $(1 - \frac{\epsilon}{2})$, all processors terminate their execution of `GlobalSync` in global round $t^* + 4\gamma + 1$.*

Finally, we show that Algorithm `GlobalSync` runs in constant time and fails with probability at most ϵ for any given constant $\epsilon > 0$.

Theorem 3. *Fix any constant $\epsilon > 0$. With probability at least $1 - \epsilon$, all processors terminate Algorithm `GlobalSync` in the same global round `sync`, which occurs $O(1)$ rounds after the first wake-up.*

Proof. Let t_1 be the first round in which a wake-up occurs. In the case where all processors wake up spontaneously in the same round, Lemma 1 implies that, with probability 1, all processors terminate Algorithm `GlobalSync` in global round $\texttt{sync} = t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$. In the case where not all processors wake up spontaneously in the same round, Lemmas 2 and 3 imply that all processors terminate Algorithm `GlobalSync` in global round $\texttt{sync} = t^* + 4\gamma + 1$, where $t^* \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$, with error probability at most $\frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$. □

3 Consensus

In this section, we provide a deterministic decision procedure that achieves consensus assuming that global synchronization has been done previously. It is performed after Algorithm `GlobalSync` and has the following property. Let `sync` be the global round in which all processors in the channel terminate their execution of Algorithm `GlobalSync`. Algorithm `Decision` achieves consensus with error probability at most ϵ in the global round $s = \texttt{sync} + O(\log w)$, where w is the smallest of all input values of processors in the channel.

Consider the input value *val* of a processor v and let $\mu = (a_1, \ldots, a_m)$ be its binary representation. We transform the sequence μ by replacing each bit 1 by (10), each bit 0 by (01) and appending (11) at the end. Hence the transformed sequence is $(c_1, \ldots, c_{2m+2})$, where
$c_i = 1$, for $i \in \{2m + 1, 2m + 2\}$, and,
for $j = 1, \ldots, m$:
$c_{2j-1} = 1$ and $c_{2j} = 0$, if $a_j = 1$,
$c_{2j-1} = 0$ and $c_{2j} = 1$, if $a_j = 0$.

The sequence $(c_1, \ldots, c_{2m+2})$ is called the *transformed input value* of processor v and is denoted by val^*. Notice that if the input values of two processors

are different, then there exists an index for which the corresponding bits of their transformed input values differ (this is not necessarily the case for the original input values, since one of the binary representations might be a prefix of the other).

The high-level idea of Algorithm `Decision` is the following. A processor beeps and listens in time intervals of prescribed length, starting in global round $\texttt{sync} + 1$, according to its transformed input value. If it does not hear any beep, it concludes that all input values are identical and outputs its input value. Otherwise, it concludes that there are different input values and then outputs a default value. We will prove that these conclusions are correct with probability at least $1 - \epsilon$, and that all processors make the decision in a common global round $s = \texttt{sync} + O(\log w)$.

We now give the pseudocode of the algorithm executed by a processor whose input value is *val*. We assume that the algorithm is started in global round $\texttt{sync} + 1$, and we let r be the processor's local clock value corresponding to the global round `sync`. Let x be the smallest positive integer such that $p^x < \epsilon/2$. Let val_0 be the smallest integer in V, which we will use as the default decision value.

Algorithm 3. `Decision`

```
1: (c_1, ..., c_k) ← val*
2: i ← 1
3: heard ← false
4: while (heard = false and i ≤ k) do
5:     if c_i = 1 then
6:         beep for x rounds and then listen for x rounds
7:     if c_i = 0 then
8:         listen for x rounds and then beep for x rounds
9:     if a beep was heard then heard ← true
10:    i ← i + 1
11: if heard = false then output val in round r + 2(i − 1)x + 1
12: else output val_0 in round r + 2(i − 1)x + 1
```

The following result shows that, with error probability at most ϵ, upon completion of Algorithm `Decision`, all processors in the channel correctly solve consensus in the same round, and this round occurs $O(\log w)$ rounds after global round `sync`, where w is the smallest of all input values of processors in the channel.

Theorem 4. *Let* `sync` *be the common global round in which all processors terminate their execution of Algorithm* `GlobalSync`*, and let w be the smallest of all input values of processors in the channel. There exists a global round $s = \texttt{sync} + O(\log w)$ such that, with probability at least $1 - \epsilon$, upon completion of Algorithm* `Decision`*, all processors in the channel output the same value in global round s, and this value is their common input value if all input values were identical.*

Proof. First, suppose that the input values of all processors in the channel are identical. Let $k \in O(\log w)$ be the length of their common transformed input value. Then each processor leaves the **while** loop with the value of the variable *heard* equal to false, and consequently, at line 11, it outputs the common input value in round $r + 2xk + 1$, which is its local clock value corresponding to global round $\texttt{sync} + 2xk + 1$. Since x is a constant, we have $2xk + 1 \in O(\log w)$, which concludes the proof in this case.

In the remainder of the proof, we suppose that there are at least two distinct input values. Let k_1 be the length of the transformed input value w^* corresponding to the input value w. Consider all transformed input values of processors in the channel, and let $j \leq k_1$ be the first index in which two of these transformed input values differ.

For any $t > 0$, let A_t be the global time interval $\{\texttt{sync} + 2x(t-1) + 1, \ldots, \texttt{sync} + 2x(t-1) + x\}$, and let B_t be the global time interval $\{\texttt{sync} + 2x(t-1) + x + 1, \ldots, \texttt{sync} + 2x(t-1) + 2x\}$. Let E be the event that at least one round in the time interval A_j is fault free and at least one round in the time interval B_j is fault free. By the definition of x, the probability of event E is at least $1 - \epsilon$. Suppose that event E holds. By the choice of j, no beep was heard in the channel in global rounds $\{\texttt{sync} + 1, \ldots, \texttt{sync} + 2x(j-1)\}$, hence all processors in the channel participate in the j^{th} iteration of the loop. Consider any processor v for which the j^{th} bit of its transformed input value is 0 and any processor v' for which the j^{th} bit of its transformed input value is 1. Processor v listens in all rounds of A_j and beeps in all rounds of B_j, whereas processor v' beeps in all rounds of A_j and listens in all rounds of B_j. Hence, v hears at least one beep in the time interval A_j, and v' hears at least one beep in the time interval B_j. Therefore, both v and v' set *heard* equal to true in iteration j of the **while** loop. Consequently, each processor outputs the default value val_0 at line 12 in round $r + 2xj + 1$, which is its local clock value corresponding to global round $\texttt{sync} + 2xj + 1$. Since x is constant and $j \leq k_1 \in O(\log w)$, we have $2xj + 1 \in O(\log w)$, which concludes the proof in the case where there are at least two distinct input values. □

Finally, given a bound $\epsilon > 0$ on error probability of consensus, we first run Algorithm `GlobalSync` and then Algorithm `Decision`, each with error probability bound $\frac{\epsilon}{2}$, to get the following corollary.

Corollary 1. *Fix any constant $\epsilon > 0$. With error probability at most ϵ, consensus can be solved deterministically using beeps in a fault-prone MAC in time $O(\log w)$, where w is the smallest of all input values of processors in the channel.*

We conclude this section by showing that, even in a model where every round in the MAC is fault-free and all processors are woken up spontaneously in the same round, deterministic consensus with m-bit inputs requires $\Omega(m)$ rounds, which implies that our consensus algorithm has optimal time complexity.

Theorem 5. *Consensus with m-bit inputs in a fault-free MAC with beeps requires $\Omega(m)$ rounds.*

Proof. Consider any consensus algorithm $\mathcal{A}$. Assume that, for every m-bit input value s, the execution of $\mathcal{A}$ with input s by a single processor on the channel uses $o(m)$ rounds. For any input s, let Pattern(s) be the beeping pattern of a processor that is alone on the channel and executes $\mathcal{A}$ with input s. By the Pigeonhole Principle, there exist distinct m-bit inputs a and b such that Pattern(a) = Pattern(b).

For each $s \in \{a, b\}$, let α_s be the execution of $\mathcal{A}$ in the case where a processor v_s is alone on the channel and is given input s. By Validity, for each $s \in \{a, b\}$, at the end of execution α_s, processor v_s must output s. Next, consider the execution $\alpha_{a,b}$ of $\mathcal{A}$ in the case where processors v_a and v_b are on the channel and are given inputs a and b, respectively. Since Pattern(a) = Pattern(b), it follows that executions α_a and $\alpha_{a,b}$ are indistinguishable to processor v_a, and that executions α_b and $\alpha_{a,b}$ are indistinguishable to processor v_b. Therefore, in execution $\alpha_{a,b}$, processor v_a outputs a and processor v_b outputs b, which contradicts Agreement. Therefore, we incorrectly assumed that, for every m-bit input s, the execution of $\mathcal{A}$ with input s by a single processor on the channel uses $o(m)$ rounds. It follows that there exists an execution of $\mathcal{A}$ that uses $\Omega(m)$ rounds, as claimed. □

4 Open Questions

Our work leaves open several interesting directions for future research. One question is how to achieve global synchronization in multi-hop networks. One could also ask how to design an efficient synchronization algorithm if the probability p of channel failures is not known by the processors. Probably the most intriguing direction is to think about different kinds of faults. Rather than jamming faults that affect the entire channel, we can think about faults that occur at individual processors, e.g., if a processor's beep fails and does not get transmitted on the channel, or if a listening processor fails to hear a beep that is transmitted by the channel.

References

1. Afek, Y., Alon, N., Bar-Joseph, Z., Cornejo, A., Haeupler, B., Kuhn, F.: Beeping a maximal independent set. In: Peleg, D. (ed.) DISC 2011. LNCS, vol. 6950, pp. 32–50. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24100-0_3
2. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Medium access control for adversarial channels with jamming. In: Kosowski, A., Yamashita, M. (eds.) SIROCCO 2011. LNCS, vol. 6796, pp. 89–100. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22212-2_9
3. Attiya, H., Welch, J.: Distributed Computing. John Wiley and Sons Inc., Chichester (2004)
4. Awerbuch, B., Richa, A.W., Scheideler, C., Schmid, S., Zhang, J.: Principles of robust medium access and an application to leader election. ACM Trans. Algorithms **10**(4), 24:1–24:26 (2014)
5. Bar-Yehuda, R., Goldreich, O., Itai, A.: On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization. J. Comput. Syst. Sci. **45**, 104–126 (1992)

6. Chlebus, B.S., Gąsieniec, L., Kowalski, D.R., Radzik, T.: On the wake-up problem in radio networks. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 347–359. Springer, Heidelberg (2005). doi:10.1007/11523468_29
7. Chlebus, B.S., Kowalski, D.R., Strojnowski, M.: Scalable quantum consensus for crash failures. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 236–250. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15763-9_24
8. Chockler, G., Demirbas, M., Gilbert, S., Lynch, N.A., Newport, C.C., Nolte, T.: Consensus and collision detectors in radio networks. Distrib. Comput. **21**, 55–84 (2008)
9. Cornejo, A., Kuhn, F.: Deploying wireless networks with beeps. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 148–162. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15763-9_15
10. Czumaj, A., Davis, P.: Communicating with beeps, arXiv:1505.06107 [cs.DC] (2015)
11. Clementi, A.E.F., Monti, A., Silvestri, R.: Selective families, superimposed codes, and broadcasting on unknown radio networks. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), pp. 709–718 (2001)
12. Czumaj, A., Rytter, W.: Broadcasting algorithms in radio networks with unknown topology. In: Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003), pp. 492–501 (2003)
13. Czyzowicz, J., Gąsieniec, L., Kowalski, D.R., Pelc, A.: Consensus and mutual exclusion in a multiple access channel. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 512–526. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04355-0_51
14. Förster, K.-T., Seidel, J., Wattenhofer, R.: Deterministic leader election in multi-hop beeping networks. In: Kuhn, F. (ed.) DISC 2014. LNCS, vol. 8784, pp. 212–226. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45174-8_15
15. Gasieniec, L., Pelc, A., Peleg, D.: The wakeup problem in synchronous broadcast systems. SIAM J. Discrete Math. **14**, 207–222 (2001)
16. Ghaffari, M., Haeupler, B.: Near optimal leader election in multi-hop radio networks. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013), pp. 748–766 (2013)
17. Gilbert, S., Kowalski, D.R.: Distributed agreement with optimal communication complexity. In: Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 965–977 (2010)
18. Gray, J.N.: Notes on data base operating systems. In: Bayer, R., Graham, R.M., Seegmüller, G. (eds.) Operating Systems an Advanced Course. LNCS, vol. 60, pp. 393–481. Springer, Heidelberg (1978). doi:10.1007/3-540-08755-9_9
19. Greenberg, A.G., Winograd, S.: A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. J. ACM **32**, 589–596 (1985)
20. Huang, B., Moscibroda, T.: Conflict resolution and membership problem in beeping channels. In: Afek, Y. (ed.) DISC 2013. LNCS, vol. 8205, pp. 314–328. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41527-2_22
21. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publ. Inc., San Francisco (1996)
22. Métivier, Y., Robson, J.M., Zemmari, A.: On distributed computing with beeps, arXiv:1507.02721 [cs.DC] (2015)
23. Moses, Y., Raynal, M.: Revisiting simultaneous consensus with crash failures. J. Parallel Distrib. Comput. **69**, 400–409 (2009)

24. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Commun. ACM **58**(1), 94–102 (2014)
25. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**, 228–234 (1980)
26. Raynal, M.: Fault-Tolerant Agreement in Synchronous Distributed Systems. Morgan & Claypool Publishers, San Francisco (2010)
27. Santoro, N., Widmayer, P.: Time is not a healer. In: Monien, B., Cori, R. (eds.) STACS 1989. LNCS, vol. 349, pp. 304–313. Springer, Heidelberg (1989). doi:10.1007/BFb0028994
28. Santoro, N., Widmayer, P.: Distributed function evaluation in presence of transmission faults. In: Proceedings of the International Symposium on Algorithms (SIGAL 1990), pp. 358–367 (1990)
29. Willard, D.E.: Log-logarithmic selection resolution protocols in a multiple access channel. SIAM J. Comput. **15**, 468–477 (1986)
30. Yu, Y., Jia, L., Yu, D., Li, G., Cheng, X.: Minimum connected dominating set construction in wireless networks under the beeping model. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2015), pp. 972–980 (2015)

Springer

http://www.springer.com/978-3-319-53057-4

Algorithms for Sensor Systems
12th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2016, Aarhus, Denmark, August 25-26, 2016, Revised Selected Papers
Chrobak, M.; Fernández Anta, A.; Gąsieniec, L.; Klasing, R. (Eds.)

2017, XI, 141 p. 15 illus., Softcover
ISBN: 978-3-319-53057-4