Computing Time Complexity of Population Protocols with Cover Times - The ZebraNet Example

Joffroy Beauquier, Peva Blanchard, Janna Burman, Sylvie Delaët

Computing Time Complexity of Population Protocols with Cover Times - the ZebraNet Example

Joffroy Beauquier[1,3], Peva Blanchard[1] *, Janna Burman[2] **, and Sylvie Delaët[1]

[1] LRI, Univ. Paris-Sud 11, Orsay, France, {jb, blanchard, delaet}@lri.fr
[2] MASCOTTE, INRIA, I3S (CNRS/University of Nice Sophia-Antipolis), France
janna.burman@inria.fr
[3] Grand Large project, INRIA Saclay, France

Abstract. *Population protocols* are a communication model for large sensor networks with resource-limited mobile agents. The agents move asynchronously and communicate via pair-wise interactions. The original *fairness* assumption of this model involves a high level of asynchrony and prevents an evaluation of the convergence time of a protocol (via deterministic means). The introduction of some "partial synchrony" in the model, under the form of *cover times*, is an extension that allows evaluating the time complexities.
In this paper, we take advantage of this extension and study a *data collection* protocol used in the ZebraNet project for the wild-life tracking of zebras in a reserve in central Kenya. In ZebraNet, sensors are attached to zebras and the sensed data is collected regularly by a mobile *base station* crossing the area. The data collection protocol of ZebraNet has been analyzed through simulations, but to our knowledge, this is the first time, that a purely analytical study is presented. Our first result is that, in the original protocol, some data may never be delivered to the base station. We then propose two slightly modified and correct protocols and we compute their worst case time complexities. Still, in both cases, the result is far from the optimal.

1 Introduction

Population Protocols (PP) have been introduced [1] as a model of sensor networks consisting of very simple mobile agents. In this model, anonymous mobile agents move asynchronously and any two of them can exchange information and change their states whenever they are chosen by a *scheduler*. When this happens, we say that an *event*, or a *meeting* between two moving agents, happens. Initially, one of the goals of PP was to determine what can be computed in such a model with a minimal hypothesis. That is why agents are anonymous, move

* The work of this author was supported by grants from INRIA Saclay.
** The work of this author was supported by the Chateaubriand grant from the French Government.

asynchronously and have a small memory. No specific assumption is made on the scheduler, except for a fairness condition that states that an infinitely often reachable configuration is reached infinitely often. It was shown in [3] that the computational power of the model is rather limited. Hence, various extensions were suggested (e.g., [13,7,12,2,5]).

In this paper, we assume a version of the PP model, where an indicator of "speed", a *cover time*, is associated to each agent [5]. A cover time is the minimum number of global events happening in the system for being certain that an agent has met every other agent. A scheduler schedules global events according to the cover times. The assumption that an agent communicates with all other agents periodically, within a finite period, has been experimentally justified for some types of mobility. Indeed, in the case of human or animal mobility within a bounded area or with a "home coming" tendency (the tendency to return to some specific places periodically), the statistical analysis of experimental data sets confirms this assumption (e.g., [14,16,8]). These data sets concern students on a campus [10], participants to a network conference [9] or visitors at Disneyland. All exhibit the fact that the *inter-contact time* (ICT) between two agents, considered as a random variable, follows a truncated Pareto distribution. In particular, this involves that the ICTs, measured in terms of real time, are finite in practice. Thus, they are also finite when measured in events. So is the cover time of an agent, which is the maximum of its ICTs measured in events.

The notion of cover times may be viewed as an introduction of "partial synchrony" assumptions [11] in the original PP model (partial - because the cover times are not assumed to be known by the agents). This extension allows to compute deterministic time complexities expressed in the number of events (also called *event complexities*). This is impossible in the original PP model.

This paper presents, on an example, some techniques for computing the event complexity of population protocols. The example is a slight modification of an existing *data collection* protocol, used by the ZebraNet project [15]. ZebraNet is a project conducted by the Princeton University and deployed in central Kenya. It aims at studying populations of zebras using sensors attached to the animals. This project uses a history-based protocol to deliver the sensed values to a base station. When an agent x has the possibility to relay its data to other agents, it may select the one, y, that has recently met the base station more frequently. The protocol assumes that y will continue meeting the base station frequently in the near future and will deliver data sooner.

The first result in this paper theoretically shows that the original ZebraNet protocol does not ensure the delivery of all the values to the base station. There are infinite executions in which some values cycle between some mobile agents. The fact that about 10% of the sensed values are lost, as exhibited by the simulations in [15], is supported here by a formal explanation. To ensure the delivery without modifying the main structure of the executions, we propose two slightly modified versions respectively called Modified ZebraNet Protocols 1 and 2 (MZP1 and MZP2). We then provide an analysis of their event complexities thanks to the notion of cover times. In both cases, the worst case complexity is

worse than for the algorithm presented in [5] (this algorithm reaches the optimal worst case complexity in general cases).

2 Model and Notations

The model is as in [5]. Let $\mathbf{A}$ be the set of all the agents in the system where $|\mathbf{A}| = \mathbf{n}$ and $\mathbf{n}$ is unknown to the agents. The Base Station (BS) is a distinguishable agent with extended resources and which may be also non-mobile.[1] In contrast with BS, all the other agents are finite-state, anonymous and are referred in the paper as *mobile*. We denote by $\mathbf{A}^*$ the set of mobile agents. Mobile agents are enumerated from 1 to $\mathbf{n} - 1$.

Population protocols can be modeled as transition systems. We adopt the following common definitions (for formal definitions, refer, e.g., to [18]) : *state* of an agent (vector of the values of its variables), *configuration* (a vector of states of all the agents), *transition* (atomic step of two communicating agents and their associated state changes), *execution* (a possibly infinite sequence of configurations related by transitions).

An *event* $(x\ y)$ is a pairwise communication (meeting) of two agents x and y. An event corresponds to a transition. Without loss of generality, we assume that no two events happen simultaneously. A *schedule* is an infinite sequence of events. A schedule, together with an initial configuration, uniquely determines an execution[2]. By abusing the notation, we often write a sequence of events to represent both a schedule and the corresponding execution. Intuitively, it is convenient to see executions as if a scheduler (adversary) "chooses" which two agents participate in the next event. Formally, a scheduler $\mathcal{D}$ is a predicate on schedules. A schedule of $\mathcal{D}$ is a schedule that satisfies the predicate $\mathcal{D}$. For the sake of simplicity, we assume that all agents start an execution *simultaneously* (e.g., on sunrise, according to a clock, or on receipt of a global signal from BS). The *non-simultaneous* start is treated, e.g., in [5,6].

Cover Time Property. In the model, each agent x is associated with a positive integer $\mathbf{cv}_x$, called the *cover time* of x. Agents are not assumed to know the cover times. We denote by $\overline{\mathbf{cv}}$ the vector of agents' cover times and by $\mathbf{cv_{min}}$ (resp. $\mathbf{cv_{max}}$) the minimum (resp. maximum) cover time in $\overline{\mathbf{cv}}$.

Definition (Cover Time Property). *Given a population* $\mathbf{A}$ *of* $\mathbf{n}$ *agents and a vector* $\overline{\mathbf{cv}}$ *of positive integers, a scheduler* $\mathcal{D}$ *(and any of its schedules) is said to satisfy the* cover time property*, if and only if, for every* $x \in \mathbf{A}$*, in any* $\mathbf{cv}_x$ *consecutive events of any schedule of* $\mathcal{D}$*, agent* x *meets every other agent at least once.*

In the paper, we consider only the schedulers that satisfy the cover time property. We say that the cover time vector $\overline{\mathbf{cv}}$ is *uniform* if all its entries are equal, i.e., $\mathbf{cv_{min}} = \mathbf{cv_{max}}$. In this case, we denote by $\mathbf{cv}$ the common value of the agents' cover times.

[1] BS is required here only by the nature of the data collection problem.
[2] We only consider deterministic systems.

Data Collection and Convergence. In the context of *data collection*, an *initial configuration* is a configuration in which each mobile agent owns an *input value*. Each input value has to be *delivered* to BS exactly once. When this happens, we say that a *legal* configuration is reached. An execution is said to *converge* if it reaches a legal configuration. The *length* of an execution that converges is the minimum number of events until convergence. The *worst case event complexity* of an algorithm is the maximum length of its executions. A protocol (or an algorithm) is said to converge, if all its executions converge.

When describing an execution, we may annotate each event as follows. The notation $(x\ \underline{y})$ indicates that there is a transfer from x to y. To specify one of the values being transferred, v for example, we note $(x\ \underline{y})^{(v)}$. Note that after $(x\ \underline{y})$, agent x *does not keep any copy* of the transferred values. Also, the notation $(x\ \overline{y})$ does not imply that there is no transfer.

For some finite sequences $S_1, S_2, \ldots, S_k$, their concatenation in the given order is denoted by $S_1 \cdot S_2 \cdots S_k$ (or just $S_1 S_2 \ldots S_k$). For any finite sequence S and any positive integer l, the sequence S^l is the sequence obtained by repeating l times the sequence S. In addition, the infinite sequence S^ω denotes the infinite repetition of S.

3 Non Convergence of the Original Protocol

In the original ZebraNet data collection protocol [15] that we consider, an agent chooses, among the agents in its range, the one which is the most likely to meet BS in a near future, and transfers its values to it. In this paper, we chose to use the model with pairwise communications, in contrast to the multi-wise communications possible in ZebraNet. Hence, the ZebraNet Protocol (ZP), Algorithm 1 presented below, is a restricted version of the original ZebraNet protocol. However, as any execution of ZP is also an execution of the original protocol, the non convergence of ZP involves the non convergence of the latter.

In ZP, the state of an agent x is defined by integer variables $accumulation_x$ and $distance_x$, an array of data values $values_x$[1] and an integer constant **decay** that is the same for every agent. The integer variables are initially set to 0. The array $values_x$ holds initially the value provided by the sensor (e.g., temperature or heart-rate). For the sake of simplicity, we assume first that the memory available for each agent is large enough, so that it can store the values of all the others. This assumption prevents memory overflows during transfers[2].

In Algorithm 1, when an agent x meets BS, its variable $accumulation_x$ is incremented and $distance_x$ is reset to 0. When an agent x meets another mobile agent, its variable $distance_x$ is incremented. If $distance_x$ becomes larger than **decay**, $accumulation_x$ is decremented and $distance_x$ is reset to 0.[3] When an

[1] We do not define the type of these arrays explicitly.

[2] In other words, we assume that agents have an unbounded $O(\mathbf{n})$ memory. The case of bounded memory is discussed in Sec. 6

[3] For avoiding overflow problems, we assume that the *accumulation* variables are periodically reset to 0.

agent x holds some values in $values_x$ and meets another mobile agent y, if $accumulation_y$ is strictly greater than $accumulation_x$, then agent x transfers all its values to agent y. An agent always transfers all its values when it meets BS.

Algorithm 1 ZebraNet Protocol

```
when x meets BS do
  <x transfers values_x to BS>
  accumulation_x := accumulation_x + 1
  distance_x := 0
end when
when x meets y ≠ BS do
  if accumulation_x < accumulation_y ∧ <values_x is not empty> then
    <x transfers values_x to y>
  end if
  distance_x := distance_x + 1
  if distance_x > decay then
    if accumulation_x ≠ 0 then
      accumulation_x := accumulation_x − 1
    end if
    distance_x := 0
  end if
end when
```

It appears that not all executions of ZP converge. Indeed, a value can circulate between mobile agents without ever being delivered to BS.

Theorem 1 (Non Convergence of ZP). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 4$ *agents, for any* **decay** ≥ 1*, there exist a uniform cover time vector* $\overline{\mathbf{cv}}$ *and an execution of* ZP *that does not converge.*

Proof. Consider a population $\mathbf{A}$ of $\mathbf{n} \geq 4$ agents and a constant **decay** ≥ 1. We first define specific sequences of events :

- $U_1 = (1\ BS)(2\ 1)$
- $V = [(2\ 3)\ldots(2\ \mathbf{n}-1)] \cdot [(3\ 4)\ldots(3\ \mathbf{n}-1)] \cdot \ldots \cdot (\mathbf{n}-2\ \mathbf{n}-1)$
  All mobile agents, except for agent 1, meet each other once.
- $W_1 = (1\ 2)\ldots(1\ \mathbf{n}-1)$
  Agent 1 meets every other mobile agent once.
- $U_2 = (2\ BS)(1\ 2)$
- $W_2 = (2\ 1)(2\ 3)\ldots(2\ \mathbf{n}-1)$
  Agent 2 meets every other mobile agent once.
- $Z = (3\ BS)\ldots(\mathbf{n}-1\ BS)$
  All mobile agents, except for agents 1 and 2, meet BS.

We choose an integer $\mathbf{g}$ such that $\mathbf{g} \cdot (\mathbf{n} - 3) \geq \mathbf{decay} + 1$. Now we build a schedule $\mathcal{S}$ as follows :

$$X = U_1 \; V^{\mathbf{g}} \; W_1^{\mathbf{g}} \; U_2 \; W_2^{\mathbf{g}} \; Z$$
$$\mathcal{S} = X^{\omega}$$

By construction, in X, all the agents meet each other at least once. For any mobile agent x, we choose $\mathbf{cv}_x = \mathbf{cv} = |X|$. That implies that $\mathcal{S}$ satisfies the cover time property. Precisely, $\mathbf{cv} = \mathbf{g} \cdot \frac{(\mathbf{n}-3)(\mathbf{n}-2)}{2} + (2\mathbf{g}+1)(\mathbf{n}-2) + 3$.

We claim that the initial value v of agent 2 is never delivered to BS. To see that, consider what happens when the sequence X is applied to an initial configuration $\mathcal{C}_0$. During $U_1 = (1\ BS)(1\ 2)$, agent 1 receives the initial value v of agent 2. During the sequence $V^{\mathbf{g}}$, only agents 2 to $\mathbf{n}-1$ are involved, thus, at the end, agent 1 still holds v. Then comes the sequence $W_1^{\mathbf{g}}$: agent 1 meets every other mobile agent $\mathbf{g}$ times. Since agents 2 to $\mathbf{n}-1$ have not met BS yet, their variables *accumulation* equal 0 and agent 1 cannot transfer v to any of them. In addition, since agent 1 is involved in $\mathbf{g} \cdot (\mathbf{n}-2) \geq \mathbf{decay}+1$ (thanks to the choice of $\mathbf{g}$) meetings, the decay mechanism of ZP implies that at the end of $W_1^{\mathbf{g}}$, the variable $accumulation_1$ of agent 1 equals 0.

Therefore, during $U_2 = (2\ BS)(2\ 1)$, agent 1 transfers v to agent 2. In $W_2^{\mathbf{g}}$, agent 2 is involved in $\mathbf{g} \cdot (\mathbf{n}-2) \geq \mathbf{decay}+1$ meetings with other mobile agents. But all their variables *accumulation* equal 0, hence agent 2 keeps v. Note that the decay mechanism implies that at the end of $W_2^{\mathbf{g}}$, the variable $accumulation_2$ of agent 2 equals 0. Finally, during Z, all mobile agents $x \notin \{1, 2\}$ meet BS and increment their variable $accumulation_x$ accordingly. Therefore, the application of the sequence X to an initial configuration $\mathcal{C}_0$ leads to a configuration $\mathcal{C}_1$ that satisfies the property $\mathcal{P}$ defined as follows :

- agent 2 holds its initial value v
- $accumulation_1 = accumulation_2 = 0$
- $\forall x \in \mathbf{A}^* - \{1, 2\}, accumulation_x = 1$

Now, apply X to $\mathcal{C}_1$. At the end of U_1, agent 1 has received v from agent 2 and satisfies $accumulation_1 = 1$. During $V^{\mathbf{g}}$, each mobile agent $x \neq 1$ is involved in $\mathbf{g} \cdot (\mathbf{n}-3) \geq \mathbf{decay}+1$ meetings. Therefore, thanks to the decay mechanism, at the end of $V^{\mathbf{g}}$, all the agents, except for agent 1, have their variable *accumulation* equal to 0. Hence during $W_1^{\mathbf{g}}$, agent 1 cannot transfer v to any other mobile agents. In addition, the decay mechanism implies that at the end of $W_1^{\mathbf{g}}$, the variable $accumulation_1$ of agent 1 equals 0. Hence, we see that the same arguments as in the previous paragraph can be applied to the sequence $U_2\ W_2^{\mathbf{g}}\ Z$ that follows. Thus, the application of the sequence X to $\mathcal{C}_1$ leads to a configuration $\mathcal{C}_2$ that also satisfies the property $\mathcal{P}$.

Hence, no matter how many sequences X are applied, the initial value v of agent 2 is never delivered to BS. □

4 Modified ZebraNet Protocol 1

To ensure the convergence, we modify the algorithm by ensuring that a mobile agent that transfers data to another mobile agent can no longer accept data. For this purpose, we add a boolean variable $active_x$, initially set to **true**, that indicates whether agent x is *active* or not, and we impose that only active agents can receive values. Once an active agent has transferred its values to another *mobile* agent, it becomes *inactive*. A formal description of MZP1 is given in Algorithm 2.

Algorithm 2 Modified ZebraNet Protocol 1

when x meets BS **do**
  $<x$ transfers $values_x$ to $BS>$
  $accumulation_x := accumulation_x + 1$
  $distance_x := 0$
end when
when x meets $y \neq BS$ **do**
  if $accumulation_x < accumulation_y \wedge active_y \wedge <values_x$ is not empty$>$ **then**
    $<x$ transfers $values_x$ to $y>$
    $active_x :=$ **false**
  end if
  $distance_x := distance_x + 1$
  if $distance_x >$ **decay then**
    if $accumulation_x \neq 0$ **then**
      $accumulation_x := accumulation_x - 1$
    end if
    $distance_x := 0$
  end if
end when

4.1 Convergence of MZP1

We now show that any execution of MZP1 converges. The proof relies on the fact that the set of active agents cannot increase, so that at some point of any execution, it remains constant. From that point, there is no transfer between two mobile agents, and since all mobile agents eventually meet BS (due to the cover time property), all values are eventually delivered.

Theorem 2 (Convergence of MZP1). MZP1 *converges.*

Proof. Let $\mathcal{E}$ be an execution. We note $ACT(k)$ the set of active agents in the k-th configuration in $\mathcal{E}$. The sequence $(ACT(1), ACT(2), \dots)$ is non-increasing, thus it is eventually constant : $\exists k_0 \in \mathbb{N}, \forall k \geq k_0, ACT(k) = ACT(k_0)$. Starting from the k_0-th configuration, there cannot be any further transfer between two active agents. Otherwise, the set of active agents would decrease. Also, according

to Algorithm 2, there cannot be any transfer from an active agent to another inactive agent, nor from an inactive agent to an inactive agent. In other words, once the set of active agents remains constant, there cannot be any transfer between two mobile agents. Since all mobile agents meet BS in the next $\mathbf{cv_{max}}$ events, all the values are eventually delivered. □

4.2 Upper Bound to the MZP1 Complexity

We compute an upper bound to the number of events needed to collect all the values at the base station. First we define the notion of path.

Definition (Path followed by a value). *Let $\mathcal{E}$ be an execution and v be a value in the system. The* path followed by v in $\mathcal{E}$ *is the sequence (possibly infinite) of mobile agents that successively carry v.*

For example, let x_1 be an agent whose initial value is v. It is possible that x_1 transfers v to some agent x_2, then agent x_2 transfers v to some agent x_3 which finally delivers v to BS. In this case, the path followed by v is $x_1x_2x_3$. Note that, without the *active* variable (e.g. in ZP), agent x_1 and agent x_3 could be the same.

Theorem 3 (Upper Bound - MZP1). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 3$ *agents, for any cover time vector* $\overline{\mathbf{cv}}$*, and for any* $\mathbf{decay} \geq 1$*, any execution of* MZP1 *converges in no more than* $\sum_{x \in A^*} \mathbf{cv}_x$ *events.*

Proof. Let $\mathcal{E}$ be an execution of MZP1. By Theorem 2, $\mathcal{E}$ converges, i.e., all the values are eventually delivered. Let v be an initial value of some agent x_1 such that v is the last delivered value in $\mathcal{E}$. Consider the path π followed by v in $\mathcal{E}$. It is of the form $x_1x_2 \dots x_k$ for some $k \geq 1$, x_k being the agent that delivers v to BS. Since a mobile agent becomes inactive as soon as it transfers some values, all the agents appearing in π are different. Hence, we have $1 \leq k \leq \mathbf{n} - 1$. Then the execution $\mathcal{E}$ can be written as the following sequence of events[1] :

$$\mathcal{E} = \underbrace{\left[\dots (x_1\ \underline{x_2})^{(v)}\right]}_{e_1}\ \underbrace{\left[\dots (x_2\ \underline{x_3})^{(v)}\right]}_{e_2}\ \dots\ \underbrace{\left[\dots (x_{k-1}\ \underline{x_k})^{(v)}\right]}_{e_{k-1}}\ \underbrace{\left[\dots (x_k\ \underline{BS})^{(v)}\right]}_{e_k}\ \dots$$

The subsequence e_i starts after the transfer of v from x_{i-1} to x_i and ends with the transfer of v from x_i to x_{i+1}. At the end of e_k, v is delivered to the base station. For all $1 \leq i \leq k-1$, the length of e_i is upper bounded by $\mathbf{cv}_{x_i}$, because x_i does not meet BS in e_i (at the beginning of e_i, x_i has received v and transfers it to x_{i+1} at the very end of e_i). In addition, the length of e_k is upper bounded by $\mathbf{cv}_{x_k}$, because there the first meeting of x_k with BS necessarily occurs in the first $\mathbf{cv}_{x_k}$ events that follow the reception of v. As a consequence, the value v is delivered to BS in less than $\sum_{x \in \pi} \mathbf{cv}_x \leq \sum_{x \in \mathbf{A}^*} \mathbf{cv}_x$. Since all other values are delivered before v, $\mathcal{E}$ converges in $\sum_{x \in \mathbf{A}^*} \mathbf{cv}_x$ events. □

[1] We remind the reader that this is an abuse of notation, refer to Section 2.

4.3 Lower Bound to MZP1 Complexity

Now we present a lower bound that almost matches the upper bound of the previous section. For the sake of clarity, we assume a uniform cover time vector $\overline{\mathbf{cv}}$. Hence, the upper bound stated in Theorem 3 becomes $(\mathbf{n}-1)\cdot\mathbf{cv}$. In the sequel, we build an execution that converges in $(\mathbf{n}-2)\cdot\mathbf{cv}$, which is close to this upper bound.

Theorem 4 (Lower Bound - MZP1). *For any population* $\mathbf{A}$ *of* $\mathbf{n}\geq 4$ *agents, for any* $\mathbf{decay}\geq 1$*, there exist a uniform cover time vector* $\overline{\mathbf{cv}}$ *and an execution of* MZP1 *that does not converge in strictly less than* $(\mathbf{n}-2)\cdot\mathbf{cv}$ *events.*

Proof. We consider a population $\mathbf{A}$ of $\mathbf{n}\geq 4$ agents and a constant $\mathbf{decay}\geq 1$. Let $\mathbf{g}$ be an integer such that $\mathbf{g}\cdot(\mathbf{n}-3)\geq\mathbf{decay}+1$. We consider a uniform cover time vector $\overline{\mathbf{cv}}$, the value of which is defined later.

We build an execution in which the initial value of agent 1 is successively carried by every other agent. For each $1\leq k\leq\mathbf{n}-2$, we consider a sequence E_k of length $\mathbf{cv}$ in which the value v is transferred from agent k to $k+1$, and another sequence Δ in which agent $\mathbf{n}-1$ delivers v to BS. Since a schedule is an infinite sequence, we also consider a repeating pattern Ω and we define a schedule $\mathcal{S}=E_1E_2\cdots E_{\mathbf{n}-2}\Delta\Omega^\omega$. The difficulty lies in the definition of the sequences E_k,Δ and Ω so that the schedule $\mathcal{S}$ satisfies the cover time property and the value v is delivered at the end of Δ.

For this purpose, we define specific sequences as follows :

- For $1\leq k\leq\mathbf{n}-1$, $U(k)$ is a sequence of events in which all the mobile agents, except for agent k, meet each other once. Hence, each mobile agent (except for agent k) is involved in $\mathbf{n}-3$ meetings. We have $|U(k)|=\frac{(\mathbf{n}-3)(\mathbf{n}-2)}{2}$.
- For $1\leq k\leq\mathbf{n}-1$, $V(k)$ is a sequence in which agent k meets every other mobile agent once. We have $|V(k)|=\mathbf{n}-2$.
- For $1\leq p\leq q\leq\mathbf{n}-1$, $B_q^p=(q\ BS)(q-1\ BS)\ldots(p\ BS)$ is a sequence in which each agent x, from q to p, successively meets BS in this order. We have $|B_q^p|=q-p+1$.
- For $1\leq p\leq q\leq\mathbf{n}-1$, $C_q^p=[(q\ q+1)(q\ BS)]\ldots[(p\ p+1)(p\ BS)]$ is a sequence in which each agent x, from q to p, meets its successor $x+1$ then BS. We have $|C_q^p|=2\cdot(q-p+1)$.

First, we look at what happens when sequences such as $U(k)$ or $V(k)$ are repeatedly applied. In $U(k)^{\mathbf{g}}$, each mobile agent $x\neq k$ is involved in $\mathbf{g}\cdot(\mathbf{n}-3)\geq\mathbf{decay}+1$ meetings. Thus, thanks to the decay mechanism, applying $U(k)^{\mathbf{g}}$ to any configuration of the system makes each non-zero $accumulation_x$, with $x\neq k$, decrease at least by one. The same argument shows that applying $V(k)^{\mathbf{g}}$ to any configuration makes $accumulation_k$ decrease at least by one, unless $accumulation_k$ already equals 0. In other words, the sequences $U(k)^{\mathbf{g}}$ and $V(k)^{\mathbf{g}}$ help resetting the variables *accumulation*.

Now, consider a configuration in which for all $x\in\mathbf{A}^*$, $accumulation_x=0$. In addition, assume that some mobile agent k, such that $1\leq k\leq\mathbf{n}-2$, holds a value

w and that agent $k+1$ is active (i.e., it can receive values). Then it is easy to see that during the sequence $B_{\mathbf{n}-1}^{k+1} \cdot C_k^1 = B_{\mathbf{n}-1}^{k+2}(k+1\ BS)(k\ k+1)(k\ BS)C_{k-1}^1$, agent k transfers w to $k+1$. Moreover, at the end, every $accumulation_x$ (with x a mobile agent) equals 1. In other words, applying $B_{\mathbf{n}-1}^{k+1} \cdot C_k^1$ to the appropriate configuration results in a transfer from agent k to agent $k+1$.

We also define, for each $1 \leq k \leq \mathbf{n}-2$, a "filling" sequence F_k of meetings between mobile agents. We only require that $|F_k| = \mathbf{n}-2-k$ (which implies that $F_{\mathbf{n}-2} = \emptyset$). The purpose of the sequence F_k is to ensure that the length of E_k is constant (independent of k). Now we are ready to define the sequences E_k $(1 \leq k \leq \mathbf{n}-2)$, Δ and Ω :

$$E_k = \underbrace{U(k)^{\mathbf{g}}(k\ k+1)F_k}_{prologue} \cdot \underbrace{B_{\mathbf{n}-1}^{k+1}C_k^1}_{center} \cdot \underbrace{U(k)^{\mathbf{g}}V(k)^{\mathbf{g}}}_{epilogue}$$

$$\Delta = U(\mathbf{n}-1)^{\mathbf{g}} \cdot (\mathbf{n}-1\ BS)$$

$$\Omega = B_{\mathbf{n}-1}^{\mathbf{n}-1}C_{\mathbf{n}-2}^1 \cdot U(\mathbf{n}-1)^{\mathbf{g}}V(\mathbf{n}-1)^{\mathbf{g}} \cdot \Delta$$

Then we set $\mathbf{cv} = |E_k|$. Precisely, we have $\mathbf{cv} = \mathbf{g}\cdot(\mathbf{n}-3)(\mathbf{n}-2)+(\mathbf{g}+2)(\mathbf{n}-2)+2$. Proving that the schedule $\mathcal{S}$ satisfies the cover time property is not difficult but tedious. This proof can be found in the appendix of [4]. Instead, we focus on the circulation of the initial value v of agent 1. Let $\mathcal{C}_1$ be an initial configuration. The *prologue* of E_1 only involves meetings between mobile agents, and, since each mobile agent has its variable *accumulation* equal to 0, there is no transfer. At the end of the *center* of E_1, the previous remarks show that agent 1 has transferred v to agent 2 and each mobile agent x satisfies $accumulation_x = 1$. The *epilogue* of E_1 first begins by $U(2)^{\mathbf{g}}$ at the end of which, each mobile agent x, except for agent 2, has its variable $accumulation_x$ equal to 0. The *epilogue* ends with $V(2)^{\mathbf{g}}$ during which there is no transfer from agent 2 to any other mobile agents (their *accumulation* being equal to 0). Moreover, at the end of E_1, all mobile agents (including agent 2), have their variable *accumulation* equal to 0 and agent 2 holds the initial value v of agent 1. Also, only agent 1 has become inactive. We denote by $\mathcal{C}_2$ the configuration at the end of E_1.

If we focus on the variables *accumulation*, we see that the configuration $\mathcal{C}_2$ is similar to the configuration $\mathcal{C}_1$. Hence, the same arguments show that during E_2, agent 2 transfers v to agent 3. In the resulting configuration $\mathcal{C}_3$, all the agents have their variables *accumulation* equal to 0 again, and the process can be iterated. At the end of $E_{\mathbf{n}-2}$, agent $\mathbf{n}-1$ holds the value v. Therefore, the value v is delivered to BS exactly at the end of $\Delta = U(\mathbf{n}-1)^{\mathbf{g}}(\mathbf{n}-1\ BS)$. In summary, with the schedule $\mathcal{S}$, the algorithm does not converge before the first $(\mathbf{n}-2)\cdot\mathbf{cv}$ events. □

5 Modified ZebraNet Protocol 2

As already explained, the non convergence of ZP is due to the fact that a value can circulate between two or more mobile agents, without ever being delivered to the base station. To prevent that, in MZP1, we imposed that a mobile agent that

transfers some values cannot receive the values later. Another way to prevent cycling of values is to impose that a mobile agent receiving some values cannot transfer them to any other mobile agent later. For this purpose, an *active* bit is also introduced, but with different functionality than in MZP1. The resulting protocol, called MZP2, is given in Algorithm 3.

Algorithm 3 Modified ZebraNet Protocol 2

```
when x meets BS do
  <x transfers its values to BS>
  accumulation_x := accumulation_x + 1
  distance_x := 0
end when
when x meets y ≠ BS do
  if accumulation_x < accumulation_y ∧ active_x ∧ <values_x is not empty> then
    <x transfers its values to y>
    active_y := false  // agent y becomes inactive
  end if
  distance_x := distance_x + 1
  if distance_x > decay then
    if accumulation_x ≠ 0 then
      accumulation_x := accumulation_x − 1
    end if
    distance_x := 0
  end if
end when
```

5.1 Upper Bound to MZP2 Complexity

Theorem 5 (Upper Bound - MZP2). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 1$ *agents, for any cover time vector* $\overline{\mathbf{cv}}$ *and for any* $\mathbf{decay} \geq 1$*, any execution of* MZP2 *converges in less than* $2 \cdot \mathbf{cv_{max}}$ *events.*

Proof. Consider an execution of MZP2 and an agent x with initial value v. During the first $\mathbf{cv_{max}}$ events, there are two possibilities. Either agent x does not transfer v to any other mobile agent then, meeting BS, it delivers v. Or, some mobile agent y has received v from agent x and has become inactive. Hence, agent y cannot transfer v to any other mobile agent, which implies that agent y will transfer v to BS during the next $\mathbf{cv_{max}}$ events. In all cases, v is delivered to the base station in less than $2 \cdot \mathbf{cv_{max}}$ events. Since v can be any value, we see that all values are delivered to the base station in less than $2 \cdot \mathbf{cv_{max}}$ events. □

5.2 Lower Bound to MZP2 Complexity

Theorem 6 (Lower Bound - MZP2). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 4$ *agents and any* $\mathbf{decay} \geq 1$*, there exist a uniform cover time vector* $\overline{\mathbf{cv}}$ *and an execution of* MZP2 *that does not converge in strictly less than* $2 \cdot \mathbf{cv} - 2$ *events.*

Proof. We consider an integer $\mathbf{g}$ such that $\mathbf{g} \cdot (\mathbf{n}-3) \geq \mathbf{decay}+1$, and we define specific sequences as follows :

- $U = (3\ BS) \dots (\mathbf{n}-1\ BS)$.
  Agents 3 to $\mathbf{n}-1$ meet the base station once.
- $V = [(2\ 3) \dots (2\ \mathbf{n}-1)] \cdot [(3\ 4) \dots (3\ \mathbf{n}-1)] \cdot \dots \cdot (\mathbf{n}-2\ \mathbf{n}-1)$
  In V, all mobile agents, except for agent 1, meet each other once.
- $W = (1\ 3) \dots (1\ \mathbf{n}-1)$.
  Agent 1 meets every other mobile agent, except for agent 2, exactly once.
- $X = U \cdot V^{\mathbf{g}} \cdot W \cdot (2\ BS)(1\ 2)(1\ BS)$

We build a schedule $\mathcal{S}$ by repeating X infinitely many times : $\mathcal{S} = X^{\omega}$. We choose the same cover time, $\mathbf{cv} = |X|$, for all the agents. A simple calculation shows that $\mathbf{cv} = 2\mathbf{n} - 3 + \mathbf{g} \cdot \frac{(\mathbf{n}-3)(\mathbf{n}-2)}{2}$. It is easy to see that $\mathcal{S}$ satisfies the cover time property.

Now we prove that the execution of MZP2 induced by $\mathcal{S}$ does not converge before the first $2 \cdot \mathbf{cv} - 2$ events. At the end of the first U in $\mathcal{S}$, agents 3 to $\mathbf{n}-1$ have successively met BS and transferred their values to it. Thus, all the variables $accumulation_x$ for $3 \leq x \leq \mathbf{n}-1$ equal 1. Then comes the sequence $V^{\mathbf{g}}$ in which each agent $x \neq 1$ is involved in $\mathbf{g} \cdot (\mathbf{n}-3) \geq \mathbf{decay}+1$ meetings. Hence, thanks to the decay mechanism, at the end of the first $V^{\mathbf{g}}$, every agent x, from 2 to $\mathbf{n}-1$, has its variable $accumulation_x$ reset to 0. As a consequence, there is no transfer from agent 1 to any other mobile agent during the sequence W that follows $V^{\mathbf{g}}$. Then during the sequence $(2\ BS)(1\ 2)(1\ BS)$, agent 2 receives the initial value v of 1. From this point, agent 2 cannot transfer v to any other agent but BS, which is done precisely $\mathbf{cv}$ events later (during the event $(2\ BS)$ in the second X of $\mathcal{S}$). Therefore, the value v is delivered to BS exactly after the $(2 \cdot \mathbf{cv} - 2)$-th events of the schedule. □

6 Bounded Memory

Up to now, we have assumed that mobile agents have an unbounded ($O(\mathbf{n})$) memory. In this section, we discuss the case of bounded memory, i.e., a memory size independent of the number of agents. We assume now that the memory of an agent can hold at most $\mathbf{k}$ values, with $\mathbf{k} \geq 1$. Both MZP1 and MZP2 can be adapted to this case. Indeed, any transfer of values is limited by the available memory and the transfer may be *partial*. During an event, as much as possible values are transferred. Note that all values are equivalent for the data collection problem, thus it is unnecessary to precise which values are actually transferred. In an adapted MZP1, once an agent has transferred some values, even if the transfer is only partial, it becomes inactive and cannot receive other values. For every agent x, the values held by x are stored in a dynamic array $values_x$, whose size is denoted by $size(values_x)$. By definition, we have $size(values_x) \leq \mathbf{k}$. Algorithm 4 presents an adaptation of MZP1, but the same idea can be applied to MZP2. For the sake of clarity, we do not present in the code the management

Algorithm 4 Modified ZebraNet Protocol 1 - Bounded memory

when x meets BS **do**
  $<x$ transfers its values to $BS>$
  $accumulation_x := accumulation_x + 1$
  $distance_x := 0$
end when
when x meets $y \neq BS$ **do**
  $count := min(size(values_x), \mathbf{k} - size(values_y))$
  if $accumulation_x < accumulation_y \wedge active_y \wedge count > 0$ **then**
    $<x$ transfers $count$ values to $y>$
    $active_x :=$ **false**
  end if
  $distance_x := distance_x + 1$
  if $distance_x >$ **decay then**
    if $accumulation_x \neq 0$ **then**
      $accumulation_x := accumulation_x - 1$
    end if
    $distance_x := 0$
  end if
end when

of the dynamic array $values_x$. We denote by MZP1-BM (resp. MZP2-BM) the bounded-memory version of MZP1 (resp. MZP2).

It appears that, for both MZP1 and MZP2, the proofs given in the previous sections (Sections 4.1, 4.2, 4.3, 5.1 and 5.2) are still applicable. Indeed, the memory size tightens the constraints on transfers, but do not fundamentally affect the structures of both MZP1 and MZP2. Still, we sketch the proofs for MZP1-BM and MZP2-BM.

Theorem 7 (Bounds to MZP1-BM complexity). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 1$ *agents, for any cover time vector* $\overline{\mathbf{cv}}$*, for any* **decay** ≥ 1*, any execution of* MZP1-BM *converges in less than* $\sum_{x \in \mathbf{A}^*} \mathbf{cv}_x$ *events.*

For any population $\mathbf{A}$ *of* $\mathbf{n} \geq 4$ *agents, for any* **decay** ≥ 1*, there exist a uniform cover time vector* $\overline{\mathbf{cv}}$ *and an execution of* MZP1-BM *that does not converge in strictly less than* $(\mathbf{n} - 2) \cdot \mathbf{cv}$ *events.*

Proof. The fact that MZP1-BM converges is due to the fact that the set of active agents cannot increase. As in MZP1, once the set of active agents remains constant, there cannot be any transfer between any two mobile agents. Since all mobile agents meet BS in the next $\mathbf{cv_{max}}$ events, the protocol converges.

The upper bound to the complexity of MZP1-BM is computed by looking at the path followed by the last delivered value v, i.e., the mobile agents that successively carry v. The memory size does not affect the fact that a mobile agent in this path cannot appear twice, thanks to the bit *active*, nor the fact that a mobile agent x in this path holds v for at most $\mathbf{cv}_x$ consecutive events. Thus any execution of MZP1-BM converges in less than $\sum_{x \in \mathbf{A}^*} \mathbf{cv}_x$ events.

The lower bound to MZP1-BM complexity is obtained thanks to the same schedule described in Section 4.3. Indeed, applying this schedule to an initial configuration gives an execution in which each agent holds at most one value, which is compatible with the assumption $\mathbf{k} \geq 1$. □

Theorem 8 (Bounds to MZP2-BM complexity). *For any population* $\mathbf{A}$ *of* $\mathbf{n} \geq 1$ *agents, for any cover time vector* $\overline{\mathbf{cv}}$*, for any* $\mathbf{decay} \geq 1$*, any execution of* MZP2-BM *converges in less than* $2 \cdot \mathbf{cv_{max}}$ *events.*

For any population $\mathbf{A}$ *of* $\mathbf{n} \geq 4$ *agents, for any* $\mathbf{decay} \geq 1$*, there exist a uniform cover time vector* $\overline{\mathbf{cv}}$ *and an execution of* MZP2-BM *that does not converge in strictly less than resp.* $2 \cdot \mathbf{cv} - 2$ *events.*

Proof. During the first $\mathbf{cv_{max}}$ events, an agent x either transfers its initial value v to BS or to another mobile agent y. In the second case, agent y is then inactive and cannot transfer v to any other agent, but BS, which is done in the next $\mathbf{cv_{max}}$ events. Thus MZP2-BM also converges in less than $2 \cdot \mathbf{cv_{max}}$ events.

The lower bound to MZP2-BM is obtained thanks to the same schedule described in Section 5.2. Indeed, applying this schedule to an initial configuration gives an execution in which each agent holds at most one value, which is compatible with the assumption $\mathbf{k} \geq 1$. □

7 Conclusion

In this paper, we study the ZebraNet data collection protocol in the context of Population Protocols. We show that the original version does not converge in all cases, the problem being the possibility for a value to cycle among the mobile agents without reaching the base station.

To ensure convergence, we propose slightly modified versions of the original protocol, MZP1 and MZP2. Notice that MZP1 is a multi-hop protocol. In contrast, MZP2 is a two-hop one. Hence, MZP1 approximates better the original ZebraNet protocol than MZP2. For both modified versions, the worst case complexity is much worse than for the near optimal data collection protocol presented in [5] (its complexity is less than $2 \cdot \mathbf{cv_{min}}$). However, this protocol assumes that, when two agents meet, both know which of them has a smaller cover time. We do not make such an assumption here, but one could consider that the ZebraNet Protocol is an approximation of the near optimal protocol in the following sense. An agent that has met BS many times in the past, has intuitively to be fast and thus, must have a small cover time. Comparing the values of the accumulation variables, when two agents meet, can be viewed as an approximation of comparing their cover times. This papers shows that this approximation is bad when the worst case complexity is considered. Note that optimal bounds to the worst case complexity can be found in [4]; precisely $\sum_{x \in \mathbf{A}^*} \mathbf{cv}_x - 2 \cdot (\mathbf{n} - 2)$ for MZP1/MZP1-BM and $2 \cdot \mathbf{cv_{max}} - 2$ for MZP2/MZP2-BM. A possible, but surely difficult extension to this work would be to compute the average complexity of the protocols. Perhaps the gap between the protocol in [5] and the

protocols MZP1 and MZP2 is not so large when considering average complexity. Such an analysis would also highlight the role of the memory size.

Another perspective would be to apply our purely analytical methodology to more intricate data collection protocols, as for instance PROPHET [17], for which only simulation results are available. For this protocol, as well as for others, the analytical approach is not supposed to replace simulations, but allows to obtain some information quickly and with less investment.

References

1. D. Angluin, J. Aspnes, Z. Diamadi, M. Fisher, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, pages 290–299, 2004.
2. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *DC*, 21(3):183–199, 2008.
3. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, Nov. 2007.
4. J. Beauquier, P. Blanchard, J. Burman, and S. Delaet. Exact time complexity of zebranet with cover times. *LRI Internal Report*, 2011.
5. J. Beauquier, J. Burman, J. Clément, and S. Kutten. On utilizing speed in networks of mobile agents. In *PODC*, pages 305–314, 2010.
6. J. Beauquier, J. Burman, and S. Kutten. A self-stabilizing transformer for population protocols with covering. *Theor. Comput. Sci.*, 412(33):4247–4259, 2011.
7. J. Beauquier, J. Clément, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in mobile sensor networks with a base station. In *DISC*, pages 63–76, 2007.
8. H. Cai and D. Y. Eun. Crossing over the bounded domain: from exponential to power-law inter-meeting time in MANET. In *MOBICOM*, pages 159–170, 2007.
9. A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, June 2007.
10. D. College. The dartmouth wireless trace archive, 2007.
11. C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.
12. M. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *OPODIS*, pages 395–409, 2006.
13. R. Guerraoui and E. Ruppert. Even small birds are unique: Population protocols with identifiers. In *Technical Report CSE-2007-04*. York University, 2007.
14. S. Hong, I. Rhee, S. J. Kim, K. Lee, and S. Chong. Routing performance analysis of human-driven delay tolerant networks using the truncated levy walk model. In *MobilityModels*, pages 25–32, 2008.
15. P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS*, pages 96–107, 2002.
16. T. Karagiannis, J. L. Boudec, and M. Vojnovic. Power law and exponential decay of inter contact times between mobile devices. In *MOBICOM*, pages 183–194, 2007.
17. A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7:19–20, July 2003.
18. G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2001.