Byzantine Gathering in Polynomial Time

Sébastien Bouchard
Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, LIP6 UMR 7606, Paris, France
sebastien.bouchard@lip6.fr

Yoann Dieudonné
Laboratoire MIS & Université de Picardie Jules Verne, Amiens, France
yoann.dieudonne@u-picardie.fr

Anissa Lamani
Laboratoire MIS & Université de Picardie Jules Verne, Amiens, France
anissa.lamani@u-picardie.fr

Abstract

Gathering a group of mobile agents is a fundamental task in the field of distributed and mobile systems. This can be made drastically more difficult to achieve when some agents are subject to faults, especially the Byzantine ones that are known as being the worst faults to handle. In this paper we study, from a deterministic point of view, the task of Byzantine gathering in a network modeled as a graph. In other words, despite the presence of Byzantine agents, all the other (good) agents, starting from possibly different nodes and applying the same deterministic algorithm, have to meet at the same node in finite time and stop moving. An adversary chooses the initial nodes of the agents (the number of agents may be larger than the number of nodes) and assigns a different positive integer (called label) to each of them. Initially, each agent knows its label. The agents move in synchronous rounds and can communicate with each other only when located at the same node. Within the team, f of the agents are Byzantine. A Byzantine agent acts in an unpredictable and arbitrary way. For example, it can choose an arbitrary port when it moves, can convey arbitrary information to other agents and can change its label in every round, in particular by forging the label of another agent or by creating a completely new one.

Besides its label, which corresponds to a local knowledge, an agent is assigned some global knowledge denoted by $\mathcal{GK}$ that is common to all agents. In literature, the Byzantine gathering problem has been analyzed in arbitrary n-node graphs by considering the scenario when $\mathcal{GK} = (n, f)$ and the scenario when $\mathcal{GK} = f$. In the first (resp. second) scenario, it has been shown that the minimum number of good agents guaranteeing deterministic gathering of all of them is $f+1$ (resp. $f+2$). However, for both these scenarios, all the existing deterministic algorithms, whether or not they are optimal in terms of required number of good agents, have the major disadvantage of having a time complexity that is exponential in n and L, where L is the value of the largest label belonging to a good agent. In this paper, we seek to design a deterministic solution for Byzantine gathering that makes a concession on the proportion of Byzantine agents within the team, but that offers a significantly lower complexity. We also seek to use a global knowledge whose the length of the binary representation (that we also call size) is small. In this respect, assuming that the agents are in a *strong team* i.e., a team in which the number of good agents is at least some prescribed value that is quadratic in f, we give positive and negative results. On the positive side, we show an algorithm that solves Byzantine gathering with all strong teams in all graphs of size at most n, for any integers n and f, in a time polynomial in n and the length $|l_{min}|$ of the binary representation of the smallest label of a good agent. The algorithm works using a global knowledge of size $\mathcal{O}(\log\log\log n)$, which is of optimal order of magnitude in our context to reach a time complexity that is polynomial in n and $|l_{min}|$. Indeed, on the negative side, we show that there is no deterministic algorithm solving Byzantine gathering with all strong teams, in all graphs of size at most n, in a time polynomial in n and $|l_{min}|$ and using a global knowledge of size $o(\log\log\log n)$.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 147; pp. 147:1–147:15


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Distributed algorithms, Computing methodologies → Mobile agents

Keywords and phrases gathering, deterministic algorithm, mobile agent, Byzantine fault, polynomial time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.147

Related Version A full version of the paper is available at `https://arxiv.org/abs/1801.07656`.

Funding This work was performed within Project ESTATE (Ref. ANR-16-CE25-0009-03) and Project TOREDY. The first project is supported by French state funds managed by the ANR (Agence Nationale de la Recherche), while the second project is supported by the European Regional Development Fund (ERDF) and the Hauts-de-France region.

1 Introduction

1.1 Context

Gathering a group of mobile agents is a basic problem that has been widely studied in literature dedicated to mobile and distributed systems. One of the main reasons for this popularity stems from the fact that this task turns out to be an essential prerequisite to achieve more complex cooperative works. In other words, getting fundamental results on the problem of gathering implies *de facto* getting fundamental results on a large set of problems whose resolution needs to use gathering as a building block.

The scale-up when considering numerous agents is inevitably tied to the occurrence of faults among them, the most emblematic of which is the Byzantine one. Byzantine faults are very interesting under multiple aspects, especially because the Byzantine case is the most general one, as it subsumes all the others kind of faults. In fact, in the field of fault tolerance they are considered as the worst faults that can occur.

In this paper, we consider the problem of gathering in a deterministic way in a network modeled as a graph, wherein some agents are Byzantine. A Byzantine agent acts in an unpredictable and arbitrary manner. For instance it may choose to never stop or to never move. It may also convey arbitrary information to the other agents, impersonate the identity of another agent, and so on. In such a context, gathering is very challenging, and so far the power of such Byzantine agents has been offset by a huge complexity when solving this problem. In what follows, we seek a solution allowing to withstand Byzantine agents while keeping a "reasonable" complexity.

1.2 Model and problem

A team of mobile agents are initially placed by an adversary at arbitrary nodes of a network modeled as a finite, connected, undirected graph $G = (V, E)$. We assume that $|V| \leq n$. Several agents may initially share the same node and the size of the team may be larger than n. Two assumptions are made about the labelling of the two main components of the graph that are nodes and edges. The first assumption is that nodes are anonymous i.e., they do not have any kind of labels or identifiers allowing them to be distinguished from one another. The second assumption is that edges incident to a node v are locally ordered with a fixed port numbering ranging from 0 to $deg(v) - 1$ where $deg(v)$ is the degree of v. Therefore, each edge

has exactly two port numbers, one for each of both nodes it links. The port numbering is not supposed to be consistent: a given edge $(u, v) \in E$ may be the i-th edge of u but the j-th edge of v, where $i \neq j$. These two assumptions are not fortuitous. The primary motivation of the first one is that if each node could be identified by a label, gathering would become quite easy to solve as it would be tantamount to explore the graph (via e.g. a breadth-first search) and then meet in the node having the smallest label. While the first assumption is made so as to avoid making the problem trivial, the second assumption is made in order to avoid making the problem impossible to solve. Indeed, in the absence of a way allowing an agent to distinguish locally the edges incident to a node, gathering could be proven as impossible to solve deterministically in view of the fact that some agents could be precluded from traversing some edges and visit some parts of the graph.

Time is discretized into an infinite sequence of rounds. In each round, every agent, which has been previously woken up (this notion is detailed in the next paragraph), is allowed to stay in place at its current node or to traverse an edge according to a deterministic algorithm. The algorithm is the same for all agents: only the input, whose nature is specified further in the subsection, varies among agents.

Before being woken up, an agent is said to be dormant. A dormant agent may be woken up only in two different ways: either by the adversary that wakes some of the agents at possibly different rounds, or as soon as a non-dormant agent is at the starting node of the dormant agent. We assume that the adversary wakes up at least one agent. Note that, when the adversary chooses to wake up in round r a dormant agent located at a node v, all the dormant agents that are at node v wake up in round r.

When an agent is woken up in a round r, it is told the degree of its starting node. As mentioned above, in each round $r' \geq r$, the executed algorithm can ask the agent to stay idle or to traverse an edge. In the latter case, this takes the following form: the algorithm asks the agent, located at node u, to traverse the edge having port number i, where $0 \leq i \leq deg(u) - 1$. Let us denote by $(u, v) \in E$ this traversed edge. In round $r' + 1$, the agent enters node v: it then learns the degree $deg(v)$ as well as the local port number j of (u, v) at node v (recall that in general $i \neq j$). An agent cannot leave any kind of tokens or markers at the nodes it visits or the edges it traverses.

In the beginning, the adversary also assigns a different positive integer (called label) to each agent. Each agent knows its label but does not know *a priori* the labels of the other agents (except if some or all of them are inserted in the global knowledge $\mathcal{GK}$ that is introduced below). When several agents are at the same node v in the same round t, they see, for each agent x at node v, the label of agent x and all information it wants to share with the others in round t. This transmission of information is done in a "shouting" mode in one round: all the transmitted information by all agents at node v in round t becomes common knowledge for agents that are currently at node v in round t. On the other hand when two agents are not at the same node in the same round they cannot see or talk to each other: in particular, two agents traversing simultaneously the same edge but in opposite directions, and thus crossing each other on the same edge, do not notice this fact. In every round, the input of the algorithm executed by an agent a is made up of the label of agent a, the up-to-date memory of what agent a has seen and learnt since its waking up and some global knowledge denoted by $\mathcal{GK}$. Parameter $\mathcal{GK}$ is a piece of information that is initially given to all agents and common to all of them (i.e., $\mathcal{GK}$ is the same for all agents): its nature is precised at the end of this subsection. Note that in the absence of a way of distinguishing the agents, the gathering problem would have no deterministic solution in some graphs, regardless of the nature of $\mathcal{GK}$. This is especially the case in a ring in which at each node the

edge going clockwise has port number 0 and the edge going anti-clockwise has port 1: if all agents are woken up in the same round and start from different nodes, they will always have the same input and will always follow the same deterministic rules leading to a situation where the agents will always be at distinct nodes no matter what they do.

Within the team, it is assumed that f of the agents are Byzantine. A Byzantine agent has a high capacity of nuisance: it can choose an arbitrary port when it moves, can convey arbitrary information to other agents and can change its label in every round, in particular by forging the label of another agent or by creating a completely new one. All the agents that are not Byzantine are called good. We consider the task of f-Byzantine gathering which is stated as follows. The adversary wakes up at least one good agent and all good agents must eventually be in the same node in the same round, simultaneously declare termination and stop, despite the fact there are f Byzantine agents. Regarding this task, it is worth mentioning that we cannot require the Byzantine agents to cooperate as they may always refuse to be with some agents. Thus, gathering all good agents with termination is the strongest requirement we can make in such a context. The time complexity of an algorithm solving f-Byzantine gathering is the number of rounds counted from the start of the earliest good agent until the task is accomplished.

We end this subsection by explaining what we mean by global knowledge, that can be viewed as a kind of advice given to all agents. Following the paradigm of algorithms with advice [1, 25, 38, 9, 20, 19, 33], $\mathcal{GK}$ is actually a piece of information that is initially provided to the agents at the start, by an oracle knowing the initial instance of the problem. By instance, we precisely mean: the entire graph with its port numbering, the initial positions of the agents with their labels, the f agents that are Byzantine, and for each agent the round, if any, when the adversary wakes it up in case it has not been woken up before by another agent. So, for example, $\mathcal{GK}$ might correspond to the size of the network, the number of Byzantine agents, or a complete map of the network, etc. As mentionned earlier, we assume that $\mathcal{GK}$ is the same for all agents. The size of $\mathcal{GK}$ is the length of its binary representation.

1.3 Related works

When reviewing the chronology of the works that are related to the gathering problem, it can be seen that this problem has been first studied in the particular case in which the team is made of exactly two agents. Under such a limitation, gathering is generally referred to as *rendezvous*. From the first mention of the rendezvous problem in [36], this problem and its generalization, gathering, have been extensively studied in a great variety of ways. Indeed, there is a lot of alternatives for the combinations we can make when addressing the problem, e.g., by playing on the environment in which the agents are supposed to evolve, the way of applying the sequences of instructions (i.e., deterministic or randomized) or the ability to leave some traces in the visited locations, etc. In this paper, we are naturally closer to the research works that are related to deterministic gathering in networks modeled as graphs. Hence, we will mostly dwell on this scenario in the rest of this subsection. However, for the curious reader wishing to consider the matter in greater depth, we invite him to consult [8, 2, 24] that address the problem in the plane via various scenarios, especially in a system affected by the occurrence of faults or inaccuracies for the last two references. Regarding randomized rendezvous, a good starting point is to go through [3, 4, 28].

Now, let us focus on the area that concerns the present paper most directly, namely deterministic rendezvous and/or gathering in graphs. In most papers on rendezvous in networks, a synchronous scenario was assumed, in which agents navigate in the network in synchronous rounds. Under this context, a lot of effort has been dedicated to the study of

the feasibility and to the time (i.e., number of rounds) required to achieve the task, when feasible. For instance, in [16] the authors show a rendezvous algorithm polynomial in the size of the graph, in the length of the shorter label and in the delay between the starting time of the agents. In [26] and [37] solutions are given for rendezvous, which are polynomial in the first two of these parameters and independent of the delay. While these algorithms ensure rendezvous in polynomial time (i.e., a polynomial number of rounds), they also ensure it at polynomial cost where the cost corresponds here to the total number of edge traversals made by both agents until meeting. Indeed, since each agent can make at most one edge traversal per round, a polynomial time always implies a polynomial cost. However, the reciprocal may be not true, for instance when using an algorithm relying on a technique similar to "coding by silence" in the time-slice algorithm for leader election [29]: "most of the time" both agents stay idle, in order to guarantee that agents rarely move simultaneously. Thus these parameters of cost and time are not always linked to each other. This was recently highlighted in [32] where the authors studied the tradeoffs between cost and time for the deterministic rendezvous problem. Some other efforts have been also dedicated to analyse the impact on time complexity of rendezvous when in every round the agents are brought with some pieces of information by making a query to some device or some oracle, see, e.g., [14, 31]. Along with the works aiming at optimizing the parameters of time and/or cost of rendezvous, some other works have examined the amount of memory that is required to achieve deterministic rendezvous e.g., in [21, 22] for tree networks and in [12] for general networks.

Apart from the synchronous scenario, the academic literature also contains several studies focusing on a scenario in which the agents move at constant speed, which are different from each other, or even move asynchronously: in this latter case the speed of agents may then vary and is controlled by the adversary. For more details about rendezvous under such a context, the reader is referred to [30, 13, 23, 18, 27] for rendezvous in finite graphs and [5, 10] for rendezvous in infinite grids.

As stated in the previous subsection, our paper is also related to the field of fault tolerance since some agents may be prone to Byzantine faults. First introduced in [34], a Byzantine fault is an arbitrary fault occurring in an unpredictable way during the execution of a protocol. Due to its arbitrary nature, such a fault is considered as the worst fault that can occur. Byzantine faults have been extensively studied for "classical" networks i.e., in which the entities are fixed nodes of the graph (cf., e.g., the book [29] or the survey [6]). To a lesser extend, the occurrence of Byzantine faults has been also studied in the context of mobile entities evolving on a one-dimensional or two-dimensional space, cf. [2, 15, 11].

Gathering in arbitrary graphs in presence of many Byzantine agents was considered in [17, 7]. Actually, our model is borrowed from both these papers, and thus they are naturally the closest works to ours. In [17], the problem is introduced via the following question: *what is the minimum number $\mathcal{M}$ of good agents that guarantees f-Byzantine gathering in all graphs of size n?* In [17], the authors provided several answers to this problem by firstly considering a relaxed variant, in which the Byzantine agents cannot lie about their labels, and then by considering a harsher form (the same as in our present paper) in which Byzantine agents can lie about their identities. For the relaxed variant, it has been proven that the minimum number $\mathcal{M}$ of good agents that guarantees f-Byzantine gathering is precisely 1 when $\mathcal{GK} = (n, f)$ and $f + 2$ when $\mathcal{GK}$ is reduced to f only. The proof that both these values are enough, relies on polynomial algorithms using a mechanism of blacklists that are, informally speaking, lists of labels corresponding to agents having exhibited an "inconsistent" behavior. Of course, such blacklists cannot be used when the Byzantine agents can change

their labels and in particular steal the identities of good agents. Still in [17], the authors give for the harsher form of f-byzantine gathering a lower bound of $f+1$ (resp. $f+2$) on $\mathcal{M}$ and a deterministic gathering algorithm requiring at least $2f+1$ (resp. $4f+2$) good agents, when $\mathcal{GK}=(n,f)$ (resp. $\mathcal{GK}=f$). Both these algorithms have a huge complexity as they are exponential in n and L, where L is the largest label of a good agent evolving in the graph. Some advances are made in [7], via the design of an algorithm for the case $\mathcal{GK}=(n,f)$ (resp. $\mathcal{GK}=f$) that works with a number of good agents that perfectly matches the lower bound of $f+1$ (resp. $f+2$) shown in [17]. However, these algorithms also suffer from a complexity that is exponential in n and L.

1.4 Our results

As mentioned just above, the existing deterministic algorithms dedicated to f-Byzantine gathering all have the major disadvantage of having a time complexity that is exponential in n and L, when Byzantine agents are allowed to change their labels. Actually, these solutions are all based on a common strategy that consists in enumerating the possible initial configurations, and successively testing them one by one. Once the testing reaches the correct initial configuration, the gathering can be achieved. However, in order to get a significantly more efficient algorithm, such a costly strategy must be abandoned in favor of a completely new one.

In this paper, we seek to design a deterministic solution for Byzantine gathering that makes a concession on the proportion of Byzantine agents within the team, but that offers a significantly lower complexity. We also seek to use a global knowledge whose the length of the binary representation (that we also call size) is small. In this respect, assuming that the agents are in a *strong team* i.e., a team in which the number of good agents is at least the quadratic value $5f^2+6f+2$, we give positive and negative results. On the positive side, we show an algorithm that solves f-Byzantine gathering with all strong teams in all graphs of size at most n, for any integers n and f, in a time polynomial in n and $|l_{min}|$. The algorithm works using a global knowledge of size $\mathcal{O}(\log\log\log n)$, which is of optimal order of magnitude in our context to reach a time complexity that is polynomial in n and $|l_{min}|$. Indeed, on the negative side, we show that there is no deterministic algorithm solving f-Byzantine gathering with all strong teams, in all graphs of size at most n, in a time polynomial in n and $|l_{min}|$ and using a global knowledge of size $o(\log\log\log n)$.

1.5 Roadmap

The next section is dedicated to the presentation of some definitions and routines that we need in the rest of this paper. In Section 3, we give a description of our main algorithm corresponding to our positive result. In Section 4, we prove our negative result. Finally we make some concluding remarks in Section 5. Note that due to the lack of space, several details and proofs are omitted but will appear in the journal version of the paper.

2 Preliminaries

Throughout the paper, log denotes the binary logarithm. The length of the binary representation of the smallest label of a good agent in a given team will be denoted by $|l_{min}|$.

In this paper, we will use a procedure whose aim is graph exploration, i.e., visiting all nodes of the graph. This procedure, based on universal exploration sequences (UXS), is a corollary of the result of Reingold [35]. Given any positive integer n, this procedure, called

EXPLO(n), allows the executing agent to traverse all nodes of any graph of size at most n, starting from any node of this graph, using a number of edge traversals that is polynomial in n. We denote by X_n the execution time of procedure EXPLO with parameter n.

Besides this exploration procedure, we will use a label transformation derived from [16]. Let ℓ_B be the label of an agent B and $b_1 \dots b_c$ its binary representation with c its length. The binary representation of the corresponding transformed label ℓ_B^* is $10b_1b_1 \dots b_cb_c0110b_1b_1 \dots b_cb_c01$. This transformation is made to ensure the following property on which the algorithm of Section 3 partly relies.

▶ **Proposition 1.** *Let ℓ_B and ℓ_X be two labels such that $\ell_B < \ell_X$. Let $b_1^*b_2^* \dots b_{4c+8}^*$ and $x_1^*x_2^* \dots x_{4y+8}^*$ be the respective binary representations of ℓ_B^* and ℓ_X^*, with c and y the lengths of the binary representations of ℓ_B and ℓ_X respectively. There exist two positive integers $i \leq 2c+4$ and $2c+4 < j \leq 4c+8$ such that $b_i^* \neq x_i^*$ and $b_j^* \neq x_j^*$.*

To establish our main result that is presented in Section 3, we needed to design two building blocks, namely procedures GROUP and MERGE. These procedures are not trivial, but due to space restrictions we only give here the two associated theorems.

The first building block called GROUP takes as input three integers $\mathcal{T}$, n and *bin* such that $bin \in \{0; 1\}$. Let x be an integer that is at least $f+2$. Roughly speaking, subroutine GROUP($\mathcal{T}, n, bin$) ensures that $(x-f)$ good agents finish the execution of the subroutine at the same round and in the same node in a graph of size at most n provided some conditions given in the next theorem are satisfied.

▶ **Theorem 1.** *Consider a team made of at least $(x-1)(f+1)+1$ good agents in a graph of size at most n, where $x \geq f+2$. Let Δ be the first round when a good agent starts executing* GROUP($\mathcal{T}, n, bin$). *If all good agents start executing* GROUP($\mathcal{T}, n, bin$) *by round $\Delta + \mathcal{T} - 1$, and parameter bin is 0 (resp. 1) for at least one good agent, then we have the following property. After at most a time polynomial in n and $\mathcal{T}$ from Δ, at least $(x-f)$ good agents finish the execution of* GROUP *at the same round and in the same node.*

The second building block called MERGE takes as input two integers n and $\mathcal{T}$. Subroutine MERGE($\mathcal{T}, n$) allows all the good agents to finish their executions of the subroutine in the same node and at the same round, provided some conditions given in the next theorem are satisfied.

▶ **Theorem 2.** *Consider a team of agents in a graph of size at most n. Let Δ be the first round when a good agent starts executing* MERGE($\mathcal{T}, n$). *If every good agent starts executing* MERGE($\mathcal{T}, n$) *by round $\Delta + \mathcal{T} - 1$ and among them at least $4f+2$ start the execution in the same node and at the same round, then all good agents finish their executions of procedure* MERGE *in the same node and at the same round $r < \Delta + 4\mathcal{T} + 6X_n - 1$.*

3 The positive result

In this section we present an algorithm, called GATHER, that solves f-Byzantine gathering with strong teams in all graphs of size at most n, assuming that $\mathcal{GK} = \lceil \log \log n \rceil$: note that such a global knowledge can be coded using $\mathcal{O}(\log \log \log n)$ bits. The algorithm works in a time polynomial in n and $|l_{min}|$, and it makes use of the building blocks introduced in the section Preliminaries.

In the sequel, we denote by G_n the maximal time complexity of procedure $\texttt{GROUP}(X_n, n, \rho)$ with $\rho \in \{0;1\}$ in all graphs of size at most n. We also denote by M_n the maximal time complexity of procedure $\texttt{MERGE}(X_n + G_n, n)$ in all graphs of size at most n. Note that according to Theorems 1 and 2, G_n and M_n exist and are polynomials in n.

In order to better describe the high level idea of our solution, let us first consider a situation that would be ideal to solve Byzantine gathering with a strong team and that would be as follows. Instead of assigning distinct labels to all agents, the adversary assigns to each of them just one bit $\rho \in \{0;1\}$, so that there are at least one good agent for which $\rho = 0$ and at least one good agent for which $\rho = 1$. Such a situation would clearly constitute an infringement of our model, but would allow the simple protocol described in Algorithm 1 to solve the problem in a time that is polynomial in n when $\mathcal{GK} = \lceil \log \log n \rceil$. Let us briefly explain why.

Algorithm 1 Algorithm executed by every good agent in the ideal situation.

1: Let ρ be the bit assigned to me by the adversary
2: Execute $\mathcal{A}(\rho)$
3: Declare that gathering is achieved

Algorithm 2 $\mathcal{A}(\rho)$ executed by a good agent.

1: $N \leftarrow 2^{(2^{\mathcal{GK}})}$
2: Execute $\texttt{EXPLO}(N)$
3: Execute $\texttt{GROUP}(X_N, N, \rho)$
4: Execute $\texttt{MERGE}(X_N + G_N, N)$

Algorithm 1 consists mainly of a call to $\mathcal{A}(\rho)$ that is given by Algorithm 2. Since $\mathcal{GK} = \lceil \log \log n \rceil$, we know that at line 1 of Algorithm 2, N is a polynomial upperbound on n, and the execution of $\texttt{EXPLO}(N)$ in a call to $\mathcal{A}(\rho)$ by the first woken-up good agent permits to visit every node of the graph and to wake up all dormant agents. As a result, the delay between the starting times of $\texttt{GROUP}(X_N, N, \rho)$ by any two good agents of the strong team is at most X_N. According to the properties of procedure GROUP (cf. Theorem 1) and the fact that the number of good agents is at least $5f^2 + 6f + 2 = ((5f + 2) - 1)(f + 1) + 1$, this guarantees in turn that the delay between the starting times of $\texttt{MERGE}(X_N + G_N, N)$ by any two good agents is at most $X_N + G_N$, and at least $4f + 2$ good agents start this procedure at the same time in the same node. Hence, in view of the properties of procedure MERGE (cf. Theorem 2), all good agents declare gathering is achieved at the same time in the same node after a polynomial number of rounds (w.r.t n) since the wake-up time of the earliest good agent.

Unfortunately, we are not in such an ideal situation. At first glance, one might argue that it is not really a problem because all agents are assigned distinct labels that are, after all, distinct binary strings. Thus, by ensuring that each good agent applies on its label the transformation given in Section 2, and then processes one by one each bit b_i of its transformed label by executing $\mathcal{A}(b_i)$, we can guarantee (with some minor technical adjustments) that the gathering of all good agents is done in time polynomial in n and $|l_{min}|$. Indeed, in view of Proposition 1 the conditions of the ideal situation are recreated when the agents process their j-th bits for some $j \leq 2|l_{min}| + 4$. Unfortunately it is not enough for our purpose. In fact, in the ideal situation, there is just one bit to process: thus, *de facto* every good agent knows that every good agent knows that gathering will be done at the end of this single process. However, it is no longer the case when the agents have to deal with sequences of bit processes: the good agents have a priori no mean to detect collectively and simultaneously

when they are gathered. It should be noted that if the agents knew f, we could use an existing algorithmic component (cf. [17]) allowing to solve f-Byzantine gathering if at some point some good agents detect the presence of a group of at least $2f+1$ agents in the network. Such a group is necessarily constructed during the sequence of bit processes given above, but again, it cannot be a priori detected as the agents do not know f or an upperbound on it. Hence, in our goal to optimize the amount of global knowledge, we need to implement a new strategy to allow the good agents to declare gathering achieved jointly and simultaneously. It is the purpose of the rest of this subsection.

To get all good agents declare simultaneously the gathering achieved, we want to reach a round in which every good agent knows that every good agent knows that gathering is done. So, let us return to our sequence of bit processes. As mentioned above, when a good agent has finished to read the first half of its transformed label – call such an agent *experienced* – it has the guarantee that the gathering of all good agents has been done at least once. Hence, when an experienced agent starts to process the second half of its transformed label, it actually knows an approximation of the number of good agents with a margin of error of f at the most. For the sake of convenience, let us consider that an experienced agent knows the exact number μ of good agents: the general case adds a slight level of complexity that is unnecessary to understand the intuition. So, each time an experienced agent completes the process of a bit in the second half of its transformed label, it is in a node containing less than μ agents or at least μ agents. In the first case, the experienced agent is sure that the gathering is not achieved. In the second case, the experienced agent is in doubt. In our solution, we build on this doubt. How do we do that? So far, each bit process was just made of one call to procedure $\mathcal{A}$: now at the end of each bit process, we add a waiting period of some prescribed length, followed by an extra step that consists in applying $\mathcal{A}$ again, but this time according to the following rule. If during the waiting period it has just done, an agent X was in a node containing, for a sufficiently long period, an agent pretending to be experienced and in doubt (this agent may be X itself), then agent X is said to be *optimistic* and the second step corresponds to the execution of $\mathcal{A}(0)$. Otherwise, agent X is said to be *pessimistic* and the second step corresponds to the execution of $\mathcal{A}(1)$.

If at least one good agent is optimistic within a given second step, then the gathering of all good agents is done at the end of this step. Indeed, through similar arguments of partition to those used for the ideal situation, we can show it is the case when at least another agent is pessimistic. However, it is also, more curiously, the case when there is no pessimistic agents at all. This is due in part to the fact that two good experienced agents cannot have been in doubt in two distinct nodes during the previous waiting period (otherwise, we would get a contradiction with the definition of μ). Thus, all good agents start $\mathcal{A}(0)$ from at most $f+1$ distinct nodes (as the Byzantine agents can mislead the good agents in at most f distinct nodes during the waiting period), which implies by the pigeonhole principle that at least $4f+2$ good agents start it from the same node. Combined with some other technical arguments, we can show that the conditions of Theorem 2 are fulfilled when the agents execute `MERGE` at the end of $\mathcal{A}(0)$, thereby guaranteeing again gathering of all good agents.

As a result, the addition of an extra step to each bit process gives us the following interesting property: when a good agent is optimistic at the beginning of a second step, at its end the gathering is done and, more importantly, the optimistic agent knows it because its existence ensures it. Note that, it is a great progress, but unfortunately it is not yet sufficient, particularly because the pessimistic agents do not have the same kind of guarantee. The way of remedying this is to repeat once more the same kind of algorithmic ingredient as above. More precisely, at the end of each second step, we add again a waiting period of

some prescribed length, followed by a third step that consists in applying $\mathcal{A}$ in the following manner. If during the waiting period it has just done, an agent X was in a node containing, for a sufficiently long period, an agent pretending to be optimistic, then the third step of agent X corresponds to the execution of $\mathcal{A}(0)$ and it becomes optimistic if it was not. Otherwise, the third step of agent X corresponds to the execution of $\mathcal{A}(1)$ and the agent stays pessimistic.

By doing so, we made a significant move forward. To understand why, we want to invite the reader to reconsider the case when there is at least one good agent that is optimistic at the beginning of a second step. As we have seen earlier, at the end of this second step, all good agents are necessarily gathered and every optimistic agent knows it. In view of the last changes made to our solution, when starting the third step, every good agent is then optimistic. As explained above the absence of pessimistic good agent is very helpful, and using here the same arguments, we are sure that when finishing the third step, all good agents are gathered and every good agent knows it because all of them are optimistic. Actually, it is even a little more subtle: the optimistic agents of the first generation (i.e., those that were already optimistic when starting the second step) know that the gathering is done and know that every good agent knows it. Concerning the optimistic agents of the second generation (i.e., those that became optimistic only when starting the third step), they just know that the gathering is done, but do not know whether the other agents know it or not. Recall that to get all good agents declare simultaneously the gathering achieved, we want to reach a round in which every good agent knows that every good agent knows that gathering is done. We are very close to such a consensus. To reach it, at the end of a third step, the optimistic agents of the first generation make themselves known to all agents. Note that if there were at least $f+1$ agents declaring to be optimistic agents of the first generation and if f was part of $\mathcal{GK}$, the consensus would be reached. Indeed, among the agents declaring to be optimistic of the first generation, at least one is necessarily good and every agent can notice it: at this point we can show that every good agent knows that every good agent knows that gathering is done.

However, the agents do not know f. That being said, at the end of a third step, note that an optimistic agent knowing that the gathering is done can compute an approximation $\tilde{f}$ of the number of Byzantine agents. More precisely, if the number of agents gathered in its node is p, the optimistic agent knows than the number of Byzantine agents cannot exceed $\tilde{f} = max\{y|(5y+1)(y+1)+1 \leq p\}$ according to the definition of a strong team. Based on this fact, we are saved. Indeed, our algorithm is designed in such a way that all good agents correctly declare the gathering is achieved in the same round after having computed the same approximation $\tilde{f}$ and noticed at least $\tilde{f}+1$ agents that claim being optimistic of the first generation during a third step. We show that such an event necessarily occurs before any agent finishes the $(4|l_{min}|+8)$-th bit process of its transformed label, which permits to obtain the promised polynomial complexity. This is where our feat of strength is: obtaining such a complexity with a small amount of global knowledge, while ensuring that the Byzantine agents cannot confuse the good agents in any way. Actually, our algorithm is judiciously orchestrated so that the only thing Byzantine agents can really do is just to accelerate the resolution of the problem.

▶ **Theorem 3.** *Assuming that* $\mathcal{GK} = \lceil \log \log n \rceil$*, Algorithm* `GATHER` *solves* f*-Byzantine gathering with every strong team in all graph of size at most* n*, and has a time complexity that is polynomial in* n *and* $|l_{min}|$*.*

4 The negative result

Algorithm GATHER introduced in the previous section uses the value $\lceil \log \log n \rceil$ as global knowledge, which can be coded with a binary string of size $\mathcal{O}(\log \log \log n)$. In this section, we show that, to solve Byzantine gathering with all strong teams, in all graph of size at most n, in a time polynomial in n and $|l_{min}|$, the order of magnitude of the size of knowledge used by our algorithm GATHER is optimal. More precisely, we have the following theorem.

▶ **Theorem 4.** *There is no algorithm solving f-Byzantine gathering with all strong teams for all f and in all graphs of size at most n, which is polynomial in n and $|l_{min}|$ and which uses a global knowledge of size $o(\log \log \log n)$.*

Proof. Suppose by contradiction that the theorem is false. Hence, there exists an algorithm Alg that solves f-Byzantine gathering with all strong teams for all f in all graphs of size at most n, which is polynomial in n and $|l_{min}|$ and which uses a global knowledge of size $o(\log \log \log n)$. The proof relies on the construction of a family $\mathcal{F}_n$ (for any $n \geq 4$) of initial instances with strong teams such that for each of them the graph size is at most n. Our goal is to prove that there is an instance from $\mathcal{F}_n$ for which algorithm Alg needs a global knowledge whose size does not belong to $o(\log \log \log n)$, which would be a contradiction with the definition of Alg. Let us first present the construction of an infinite sequence of instances $\mathcal{I} = I_0, I_1, I_2, \ldots, I_i, \ldots$ by induction on i. Instance I_0 consists of an oriented ring of 4 nodes (i.e., a ring in which at each node the edge going clockwise has port number 0 and the edge going anti-clockwise has port 1). In this ring, there is no Byzantine agent but there are two good agents labeled 0 and 1 that are placed in diametrically opposed nodes. All the agents in I_0 wake up at the same time.

Now let us describe the construction of instance I_i with $i \geq 1$ using some features of instance I_{i-1}. Let c be the smallest constant integer such that the time complexity of algorithm Alg is at most n^c from every instance made of a graph of size at most n with a strong team in which $|l_{min}| = 1$. Let μ_{i-1} and n_{i-1} be respectively the total number of agents in I_{i-1} and the number of nodes in the graph of I_{i-1}. Instance I_i consists of an oriented ring of $(n_{i-1})^{4c}$ nodes. In this ring an agent labeled 0 is placed on a node denoted by v_0. In each of both nodes that are adjacent to v_0, $(n_{i-1})^c \cdot \mu_{i-1}$ Byzantine agents are placed (which gives a total of $2(n_{i-1})^c \cdot \mu_{i-1}$ Byzantine agents). On the node that is diametrically opposed to v_0, enough good agents are placed in order to have a strong team. The way of assigning labels to all agents that are not at v_0 is arbitrary but respects the condition that initially no two agents share the same label. Finally, all the agents in I_i wake up at the same time. This closes the description of the construction of $\mathcal{I}$, for which we have the following claim.

Claim 1. For any two instances I_j and $I_{j'}$ of $\mathcal{I}$, algorithm Alg requires a distinct global knowledge.

Proof of Claim 1. Assume by contradiction that the claim does not hold for two instances I_j and $I_{j'}$ such that $j < j'$. Consider any execution EX_j of algorithm Alg from I_j. According to the construction of $\mathcal{I}$, we know that every agent is woken up at the first round of EX_j. We denote by $r_1, r_2, \ldots, r_k$ the sequence of consecutive rounds from the first round of EX_j to the round when all good agents declare that gathering is done. We also denote by G_i the group of agents (possibly empty) that are with the good agent labeled 0 at round r_i of EX_j. Now, using execution EX_j, let us describe a possible execution $EX_{j'}$ of algorithm Alg

from $I_{j'}$: this execution is designed in such a way that it will fool the good agent labeled 0 and will induce it into premature termination. According to the construction of $\mathcal{I}$, all the agents of $I_{j'}$ are woken up in the first round of $I_{j'}$ and all the good ones are executing algorithm `Alg`. In the first round of $EX_{j'}$ the agent labeled 0 is alone (as in the first round of EX_j). Then, for each $i \in 2, \ldots, k$, the good agent labeled 0 in $EX_{j'}$ meets a group of $|G_i|$ Byzantine agents whose the multiset of labels is exactly the same as the multiset of labels belonging to the agents of G_i in the ith round of EX_j. This is always possible in view of the fact that for each $i \in 1, \ldots, k$, $|G_i| \leq \mu_j$ and the Byzantine agents of $I_{j'}$ can choose to move by ensuring that in the ith round of $EX_{j'}$ it remains at least $(k-i) \cdot \mu_j$ Byzantine agents in the node adjacent to the one occupied by the agent labeled 0 in the clockwise direction (resp. anti-clockwise direction): indeed according to the construction of $I_{j'}$, in each of both nodes adjacent to the starting node of the good agent labeled 0, there are initially $(n_{j'-1})^c \cdot \mu_{j'-1} \geq k \cdot \mu_{j'-1}$ Byzantine agents, as $k \leq (n_j)^c \leq (n_{j'-1})^c$. Finally, if algorithm `Alg` prescribes some message exchange between agents during their meetings, then the Byzantine agents in execution $EX_{j'}$ give exactly the same information to 0, as the agents with respective labels in execution EX_j. Hence, from the point of view of agent 0, the first k rounds of EX_j look exactly identical to the first k rounds of $EX_{j'}$. This is due to the actions of Byzantine agents, the fact that all nodes in I_j and $I_{j'}$ look identical, and also because $k \leq (n_j)^c$ which implies that, regardless of the algorithm `Alg`, the agent labeled 0 cannot meet any good agent in the first k rounds of $EX_{j'}$ as the distance between agent 0 and any other good agent is initially at least $\frac{(n_{j'-1})^{4c}}{2} \geq \frac{(n_j)^{4c}}{2}$. Therefore, in the kth round of execution $EX_{j'}$, the good agent labeled 0 declares having met all good agents and stops, which is incorrect, since it has not met any good agent. This contradicts the definition of algorithm `Alg` and closes the proof of this claim.

Now, consider the largest x such that in each of the $x+1$ first instances $I_0, I_1, \ldots, I_x$ of $\mathcal{I}$, the graph size is at most n: these $x+1$ instances constitute family $\mathcal{F}_n$. In view of the construction of sequence $\mathcal{I}$ and the definition of x, we have $4^{((4c)^x)} \leq n < 4^{((4c)^{x+1})}$. Hence, x belongs to $\Omega(\log\log n)$. However, according to Claim 1, the global knowledge given to distinct instances in this family must be different. Hence, there is at least one instance of $\mathcal{F}_n$ for which algorithm `Alg` uses a global knowledge of size $\Omega(\log x)$: since $x \in \Omega(\log\log n)$, we have $\Omega(\log x) \in \Omega(\log\log\log n)$. This contradicts the fact that `Alg` uses a global knowledge of size $o(\log\log\log n)$ and proves the theorem. ◀

5 Conclusion

In this paper, we designed the first polynomial algorithm w.r.t n and $|l_{min}|$ allowing to gather all good agents in presence of Byzantine ones that can act in an unpredictable way and lie about their labels. Our algorithm works under the assumption that the team evolving in the network is strong i.e., the number of good agents is roughly at least quadratic in the number f of Byzantine agents. The required global knowledge $\mathcal{GK}$ is of size $\mathcal{O}(\log\log\log n)$, which is of optimal order of magnitude to get a time complexity that is polynomial in n and $|l_{min}|$ even with strong teams.

A natural open question that immediately comes to mind is to ask if we can do the same by reducing the ratio between the good agents and the Byzantine agents. For example, could it be still possible to solve the problem in polynomial time with a global knowledge of size $\mathcal{O}(\log\log\log n)$ if the number of good agents is at most $o(f^2)$? Note that the answer to this question may be negative but then may become positive with a little bit more global knowledge. Actually, we can even easily show that the answer is true if the agents are initially

given a complete map of the graph with all port numbers, and in which each node v is associated to the list of all labels of the good agents initially occupying node v. However, the size of $\mathcal{GK}$ is then huge as it belongs to $\Omega(n^2)$. In fact, in this case what is really interesting is to find the optimal size for $\mathcal{GK}$. This observation allows us to conclude with the following open problem that is more general and appealing.

What are the trade-offs among the ratio good/Byzantine agents, the time complexity and the amount of global knowledge to solve f-Byzantine gathering?

Bringing an exhaustive and complete answer to this question appears to be really challenging but would turn out to be a major step in our understanding of the problem.

References

1. Serge Abiteboul, Haim Kaplan, and Tova Milo. Compact labeling schemes for ancestor queries. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 547–556, 2001.
2. Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, 2006.
3. Steve Alpern. Rendezvous search: A personal perspective. *Operations Research*, 50(5):772–795, 2002.
4. Steve Alpern. *The theory of search games and rendezvous.* International Series in Operations Research and Management Science, Kluwer Academic Publishers, 2003.
5. Evangelos Bampas, Jurek Czyzowicz, Leszek Gasieniec, David Ilcinkas, and Arnaud Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, pages 297–311, 2010.
6. Michael Barborak and Miroslaw Malek. The consensus problem in fault-tolerant computing. *ACM Comput. Surv.*, 25(2):171–220, 1993.
7. Sébastien Bouchard, Yoann Dieudonné, and Bertrand Ducourthial. Byzantine gathering in networks. *Distributed Computing*, 29(6):435–457, 2016.
8. Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. Comput.*, 41(4):829–879, 2012.
9. Reuven Cohen, Pierre Fraigniaud, David Ilcinkas, Amos Korman, and David Peleg. Label-guided graph exploration by a finite automaton. *ACM Trans. Algorithms*, 4(4):42:1–42:18, 2008.
10. Andrew Collins, Jurek Czyzowicz, Leszek Gasieniec, and Arnaud Labourel. Tell me where I am so I can meet you sooner. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, pages 502–514, 2010.
11. Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil M. Shende. Search on a line by byzantine robots. In *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, pages 27:1–27:12, 2016.
12. Jurek Czyzowicz, Adrian Kosowski, and Andrzej Pelc. How to meet when you forget: log-space rendezvous in arbitrary graphs. *Distributed Computing*, 25(2):165–178, 2012.
13. Jurek Czyzowicz, Andrzej Pelc, and Arnaud Labourel. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms*, 8(4):37, 2012.
14. Shantanu Das, Dariusz Dereniowski, Adrian Kosowski, and Przemyslaw Uznanski. Rendezvous of distance-aware mobile agents in unknown graphs. In *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, pages 295–310, 2014.

15 Xavier Défago, Maria Gradinariu, Stéphane Messika, and Philippe Raipin Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Distributed Computing, 20th International Symposium, DISC 2006, Stockholm, Sweden, September 18-20, 2006, Proceedings*, pages 46–60, 2006.

16 Anders Dessmark, Pierre Fraigniaud, Dariusz R. Kowalski, and Andrzej Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006.

17 Yoann Dieudonné, Andrzej Pelc, and David Peleg. Gathering despite mischief. *ACM Transactions on Algorithms*, 11(1):1, 2014.

18 Yoann Dieudonné, Andrzej Pelc, and Vincent Villain. How to meet asynchronously at polynomial cost. *SIAM J. Comput.*, 44(3):844–867, 2015.

19 Pierre Fraigniaud, Cyril Gavoille, David Ilcinkas, and Andrzej Pelc. Distributed computing with advice: information sensitivity of graph coloring. *Distributed Computing*, 21(6):395–403, 2009.

20 Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Tree exploration with advice. *Inf. Comput.*, 206(11):1276–1287, 2008.

21 Pierre Fraigniaud and Andrzej Pelc. Deterministic rendezvous in trees with little memory. In *Distributed Computing, 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings*, pages 242–256, 2008.

22 Pierre Fraigniaud and Andrzej Pelc. Delays induce an exponential memory gap for rendezvous in trees. *ACM Transactions on Algorithms*, 9(2):17, 2013.

23 Samuel Guilbault and Andrzej Pelc. Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. *Theor. Comput. Sci.*, 509:86–96, 2013.

24 Taisuke Izumi, Samia Souissi, Yoshiaki Katayama, Nobuhiro Inuzuka, Xavier Défago, Koichi Wada, and Masafumi Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM J. Comput.*, 41(1):26–46, 2012.

25 Michal Katz, Nir A. Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004.

26 Dariusz R. Kowalski and Adam Malinowski. How to meet in anonymous network. *Theor. Comput. Sci.*, 399(1-2):141–156, 2008.

27 Evangelos Kranakis, Danny Krizanc, Euripides Markou, Aris Pagourtzis, and Felipe Ramírez. Different speeds suffice for rendezvous of two agents on arbitrary graphs. In *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, pages 79–90, 2017.

28 Evangelos Kranakis, Danny Krizanc, and Sergio Rajsbaum. Mobile agent rendezvous: A survey. In *Structural Information and Communication Complexity, 13th International Colloquium, SIROCCO 2006, Chester, UK, July 2-5, 2006, Proceedings*, pages 1–9, 2006.

29 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

30 Gianluca De Marco, Luisa Gargano, Evangelos Kranakis, Danny Krizanc, Andrzej Pelc, and Ugo Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci.*, 355(3):315–326, 2006.

31 Avery Miller and Andrzej Pelc. Fast rendezvous with advice. *Theor. Comput. Sci.*, 608:190–198, 2015.

32 Avery Miller and Andrzej Pelc. Time versus cost tradeoffs for deterministic rendezvous in networks. *Distributed Computing*, 29(1):51–64, 2016.

33 Nicolas Nisse and David Soguet. Graph searching with advice. *Theor. Comput. Sci.*, 410(14):1307–1318, 2009.

34 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

35 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

36 Thomas Schelling. *The Strategy of Conflict*. Oxford University Press, Oxford, 1960.

37 Amnon Ta-Shma and Uri Zwick. Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences. *ACM Transactions on Algorithms*, 10(3):12, 2014.

38 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.