Byzantine Gathering in Networks*

Sébastien Bouchard[1], Yoann Dieudonné[1], and Bertrand Ducourthial[2]

[1] Laboratoire MIS & Université de Picardie Jules Verne Amiens, France
[2] Heudiasyc, CNRS & Université de Technologie de Compiègne, Compiègne, France

Abstract. This paper investigates an open problem introduced in [14]. Two or more mobile agents start from different nodes of a network and have to accomplish the task of gathering which consists in getting all together at the same node at the same time. An adversary chooses the initial nodes of the agents and assigns a different positive integer (called label) to each of them. Initially, each agent knows its label but does not know the labels of the other agents or their positions relative to its own. Agents move in synchronous rounds and can communicate with each other only when located at the same node. Up to f of the agents are Byzantine. A Byzantine agent 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. *What is the minimum number $\mathcal{M}$ of good agents that guarantees deterministic gathering of all of them, with termination?* We provide exact answers to this open problem by considering the case when the agents initially know the size of the network and the case when they do not. In the former case, we prove $\mathcal{M} = f + 1$ while in the latter, we prove $\mathcal{M} = f + 2$. More precisely, for networks of known size, we design a deterministic algorithm gathering all good agents in any network provided that the number of good agents is at least $f + 1$. For networks of unknown size, we also design a deterministic algorithm ensuring the gathering of all good agents in any network but provided that the number of good agents is at least $f + 2$. Both of our algorithms are optimal in terms of required number of good agents, as each of them perfectly matches the respective lower bound on $\mathcal{M}$ shown in [14], which is of $f + 1$ when the size of the network is known and of $f + 2$ when it is unknown. Perhaps surprisingly, our results highlight an interesting feature when put in perspective with known results concerning a relaxed variant of this problem in which the Byzantine agents cannot change their initial labels. Indeed under this variant $\mathcal{M} = 1$ for networks of known size and $\mathcal{M} = f + 2$ for networks of unknown size. Following this perspective, it turns out that when the size of the network is known, the ability for the Byzantine agents to change their labels significantly impacts the value of $\mathcal{M}$. However, the relevance for $\mathcal{M}$ of such an ability completely disappears in the most general case where the size of the network is unknown, as $\mathcal{M} = f + 2$ regardless of whether Byzantine agents can change their labels or not.

Keywords: deterministic gathering, mobile agent, Byzantine fault.

* Partially supported by the European Regional Development Fund (ERDF) and the Picardy region under Project TOREDY.

© Springer International Publishing Switzerland 2015
C. Scheideler (Ed.): SIROCCO 2015, LNCS 9439, pp. 179–193, 2015.
DOI: 10.1007/978-3-319-25258-2_13

1 Introduction

1.1 Context

Gathering is one of the most fundamental tasks in the field of distributed and mobile systems in the sense that, the ability to gather is in fact a building block to achieve more complex cooperative works. Loosely speaking, the task of gathering consists in ensuring that a group of mobile entities, initially located in different places, ends up meeting at the same place at the same time. These mobile entities, hereinafter called agents, can vary considerably in nature ranging from human beings and robots to animals and software agents. The environment in which the agents are supposed to evolve can vary considerably as well: it may be a terrain, a network modeled as a graph, a three-dimensional space, etc. We can also consider that the sequences of instructions followed by the agents in order to ensure their gathering are either deterministic or randomized.

In this paper, we consider the problem of gathering in a deterministic way in a network modeled as a graph. Thus, the agents initially start from different nodes of the graph and have to meet at the same node by applying deterministic rules. We assume that among the agents, some are Byzantine. A Byzantine agent is an agent subject to unpredictable and arbitrary faults. For instance such an agent may choose to never stop or to never move. It may also convey arbitrary information to the other agents, etc. The case of Byzantine fault is very interesting because it is the worst fault that can occur to agents. As a consequence, gathering in such a context is challenging.

1.2 Model and Problem

The distributed system considered in this paper consists of a group of mobile agents that are initially placed by an adversary at arbitrary but distinct nodes of a network modeled as a finite, connected, undirected graph $G = (V, E)$. We assume that $|V| = n$. In the sequel n is also called the size of the network. 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.

An adversary chooses the starting nodes of the agents. The starting nodes are chosen so that there are not two agents sharing initially the same node. At the beginning, an agent has a little knowledge about its surroundings: it does not know either the graph topology, or the number of other agents, or the positions of the others relative to its own. Still regarding agents' knowledge, we will study two scenarios: one in which the agents initially know the parameter n and one in which the agents do not initially know this parameter or even any upper bound on it.

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 another agent enters the starting node of the dormant agent. We assume that the adversary wakes up at least one agent. 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 ask the agent, located at node u, to traverse the edge having port number i, where $0 \leq i < deg(u) - 1$. Let us denote by $(u, v) \in E$ this traversed edge. In round $r' + 1$, the agents 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 the labels of the other agents. When several agents are at the same node in the same round, they see the labels of the other agents and can exchange all the information they currently have. This exchange is done in a "shouting" mode in one round: all the exchanged information becomes common knowledge for agents that are currently at the node. 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 and the up-to-date memory of what agent a has seen and learnt since its waking up. Note that in the absence of a way of distinguishing the agents, the gathering problem would have no deterministic solution in some graphs. 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 up to f of the agents are Byzantine. The parameter f is known to all agents. 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, provided that there are at most 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.

What is the minimum number $\mathcal{M}$ of good agents that guarantees f-Byzantine gathering?

At first glance, the question might appear as not being really interesting since, after all, the good agents might always be able to gather in some node, regardless of the number of Byzantine agents evolving in the graph. However, this is not the case as pointed out by the study that introduced this question in [14]. More specifically, when this size is initially known to the agents, the authors of this study described a deterministic algorithm gathering all good agents in any network provided that there are at $2f + 1$ of them, and gave a lower bound of $f + 1$ on $\mathcal{M}$ by showing that if the number of good agents is not larger than f, then there are some graphs in which the good agents are not able to gather deterministically with termination. When the size of the network is unknown, they did a similar thing but with different bounds: they gave an algorithm working for a team including at least $4f + 2$ good agents, and showed a lower bound of $f + 2$ on $\mathcal{M}$. However, the question of what the tight bounds are was left as an open problem.

1.3 Our Results

In this paper, we solve this open problem by proving that the lower bounds of $f+1$ and $f+2$ on $\mathcal{M}$, shown in [14], are actually also upper bounds respectively when the size of the network is known and when it is unknown. More precisely, we design deterministic algorithms allowing to gather all good agents provided that the number of good agents is at least $f + 1$ when the size of the network is initially known to agents, and at least $f + 2$ when this size is initially unknown.

Perhaps surprisingly, our results highlight an interesting feature when put in perspective with results concerning a relaxed variant of this problem (also

introduced in [14]) in which the Byzantine agents cannot change their initial labels. Indeed under this variant $\mathcal{M} = 1$ for networks of known size and $\mathcal{M} = f+2$ for networks of unknown size[1]. Following this perspective, it turns out that when the size of the network is known, the ability for the Byzantine agents to change their labels significantly impacts the value of $\mathcal{M}$. However, the relevance for $\mathcal{M}$ of such an ability completely disappears in the most general case where the size of the network is unknown, as $\mathcal{M} = f+2$ regardless of whether Byzantine agents can change their labels or not.

1.4 Related Works

Historically, the first mention of the gathering problem appeared in [28] under the appellation of rendezvous problem. Rendezvous is the term which is usually used when the studied task of gathering is restricted to a team of exactly two agents. From this publication until now, the problem has been extensively studied so that there is henceforth a huge literature about this subject. This is mainly due to the fact that there is a lot of alternatives for the combinations we can make when approaching 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. Naturally, in this paper we are more interested in the research works that are related to deterministic gathering in networks modeled as graphs. This is why 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 [7,1,19] 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 [2,3,21].

Concerning the context of this paper, the closest work to ours is obviously [14]. Nonetheless, in similar settings but without Byzantine agents, there are some papers that should be cited here. This is in particular the case of [13] in which the author presented a deterministic protocol for solving the rendezvous problem, which guarantees a meeting of the two involved agents after a number of rounds that is polynomial in the size n of the graph, the length l of the shorter of the two labels and the time interval τ between their wake-up times. As an open problem, the authors ask whether it is possible to obtain a polynomial solution to this problem which would be independent of τ. A positive answer to this question was given, independently of each other, in [20] and [29]. While these algorithms ensure rendezvous in polynomial time (i.e., a polynomial number of rounds),

[1] The proof that both of these values are enough, under their respective assumptions regarding the knowledge of the network size, relies on algorithms using a mechanism of blacklists that are, informally speaking, lists of labels corresponding to agents having exhibited an "inconsistent" behavior. Of course, in the context of our paper, we cannot use such blacklists as the Byzantine agents can change their labels and in particular steal the identities of good agents.

they also ensure it at polynomial cost since the cost of a rendezvous protocol is the number of edge traversals that are made by the agents until meeting and since each agent can make at most one edge traversal per round. However, it should be noted that despite the fact a polynomial time implies a polynomial cost, the reciprocal is not always true as the agents can have very long waiting periods sometimes interrupted by a movement. Thus these parameters of cost and time are not always linked to each other. This was highlighted in [25] where the authors studied the tradeoffs between cost and time for the deterministic rendezvous problem. More recently, some efforts have been 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., [11,24]. 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 [16,17] for tree networks and in [9] for general networks.

All the aforementioned studies that are related to gathering in graphs take place in a synchronous scenario i.e., a scenario in which the agents traverse the edges in synchronous rounds. Some efforts have been also dedicated to the scenario in which the agents move asynchronously: 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 [23,10,15,18] for rendezvous in finite graphs and [4,8] for rendezvous in infinite grids.

Aside from the gathering problem, our work is also in conjunction with the field of fault tolerance via the assumption of Byzantine faults to which some agents are subjected. First introduced in [26], 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 [22] or the survey [5]). To a lesser extend, the occurrence of Byzantine faults has been also studied in the context of mobile entities evolving in the plane, cf. [1,12]. Prior to our work, gathering in arbitrary graphs in presence of Byzantine agents was considered only in [14]. As mentioned in the previous section, it is proven in [14] that the minimum number $\mathcal{M}$ of good agents that guarantees f-Byzantine gathering is precisely 1 for networks of known size and $f+2$ for networks of unknown size, provided that the Byzantine agents cannot lie about their labels. The proof that both of these values are enough, under their respective assumptions regarding the knowledge of the network size, relies on algorithms using a mechanism of blacklists that are, informally speaking, lists of labels corresponding to agents having exhibited an "inconsistent" behavior. Of course, in the context of our paper, we cannot use such blacklists as the Byzantine agents can change their labels and in particular steal the identities of good agents.

2 Preliminaries

Throughout the paper, the number of nodes of a graph is called its size. In this section we present two procedures, that will be used as building blocks in our algorithms. The aim of both of them is graph exploration, i.e., visiting all nodes of the graph by a single agent. The first procedure, based on universal exploration sequences (UXS), is a corollary of the result of Reingold [27]. Given any positive integer N, this procedure allows the agent to traverse all nodes of any graph of size at most N, starting from any node of this graph, using $P(N)$ edge traversals, where P is some polynomial. After entering a node of degree d by some port p, the agent can compute the port q by which it has to exit; more precisely $q = (p + x_i) \mod d$, where x_i is the corresponding term of the UXS of length $P(N)$.

The second procedure [6] needs no assumption on the size of the network but it is performed by an agent using a fixed token placed at a node of the graph. It works in time polynomial in the size of the graph. (It is well known that a terminating exploration even of all anonymous rings of unknown size by a single agent without a token is impossible.) In our applications the roles of the token and of the exploring agent will be played by agents or by groups of agents. At the end of this second procedure, the agent has visited all nodes and determined a BFS tree of the underlying graph.

We call the first procedure $EXPLO(N)$ and the second procedure EST, for *exploration with a stationary token*. We denote by $T(EXPLO(n))$ the execution time of procedure $EXPLO$ with parameter n (note that $T(EXPLO(n)) = P(n) + 1$). We denote by $T(EST(N))$ the maximum time of execution of the procedure EST in a graph of size at most N.

3 Known Graph Size

This section aims at proving the following theorem

Theorem 1. *Deterministic f-Byzantine gathering of k good agents is possible in any graph of known size if, and only if $k \geq f + 1$.*

As mentioned in Subsection 1.2, we know from [14] that:

Theorem 2 ([14]). *Deterministic f-Byzantine gathering of k good agents is not possible in some graph of known size if $k \leq f$.*

Thus, to prove Theorem 1, it is enough to show the following theorem.

Theorem 3. *Deterministic f-Byzantine gathering of k good agents is possible in any graph of known size if $k \geq f + 1$.*

Hence, the rest of this section is devoted to proving Theorem 3. To do so, we show a deterministic algorithm that gathers all good agents in an arbitrary network of known size, provided there are at least $f + 1$ of them.

Before presenting the algorithm, we first give the high level idea which is behind it. Let us assume an ideal situation in which each agent would have as input, besides its label and the network size n, a parameter $\rho = (G^*, L^*)$ corresponding to the initial configuration of the agents in the graph such that:

- G^* represents the n-node graph with all port numbers, in which each node are assigned an identifier belonging to $\{1, \cdots, n\}$. The node identifiers are pairwise distinct. Note that the representation G^* contains more information than there is in the actual graph G as it also includes node identifiers which do not exist in G.
- $L^* = \{(v_1, l_1), (v_2, l_2), \cdots, (v_k, l_k)\}$ where $(v_i, l_i) \in L^*$ iff there is a good agent having label l_i which is initially placed in G at the node having identifier v_i in G^*. Remark that $k \geq f + 1$.

Let us also assume that all the agents in the graph are woken up at the same time by the adversary. In such ideal situation, gathering all good agents can be easily achieved by ensuring that each agent moves towards the node v where the agent having the smallest label is located. Each agent can indeed do that by using the knowledge of $\rho = (G^*, L^*)$ and its own label. Of course, all the good agents do not necessarily reach node v at the same time. However, each agent can compute the remaining time which is required to wait at node v in order to be sure that all good agents are at node v: again this time can be computed using $\rho = (G^*, L^*)$ and the fact that all agents are woken up in the same round. Unfortunately, the agents are not in such ideal situation. First, every agent is not necessarily woken up by the adversary, and for those that are woken by the adversary, this is not necessarily in the same round. Second, the agents do not have configuration ρ as input of the algorithm. In our algorithm we cope with the first constraint by requiring the first action to be a traversal of the entire graph (using procedure $EXPLO(n)$) which allows to wake up all encountered agents that are still dormant. In this way, the agents are "almost synchronized" as the delay between the starting times of any two agents is at most $T(EXPLO(n))$: the waiting time periods can be adjusted regarding this maximum delay. The second constraint i.e., the non-knowledge of ρ, is more complicated to deal with. To handle the lack of information about ρ, agents make successive assumptions about it that are "tested" one by one. More precisely, let $\mathcal{P}$ be the recursively enumerable set of all the configurations $\rho_i = (G_i^*, L_i^*)$ such that G_i^* is a connected n-node graph and $|L_i^*| \geq f + 1$. Let $\Theta = (\rho_1, \rho_2, \rho_3, \cdots)$ be a fixed enumeration of $\mathcal{P}$ (all good agents agree on this enumeration). Each agent proceeds in phases numbered $1, 2, 3, \cdots$. In each phase i, an agent supposes that $\rho = \rho_i$ and, similarly as in the ideal situation, tries to go to the node which is supposed to correspond to node v, where v is the node where the agent having the smallest label is initially located (according to ρ_i). For some reasons detailed in the algorithm (refer to the description of state `setup`), when $\rho_i \neq \rho$ some agents may be unable to make such a motion. As a consequence, these agents will consider that, rightly, $\rho_i \neq \rho$. On the other hand, whether $\rho_i \neq \rho$ or not, some other good agents may reach a node for which they had no reason to think it is not v (and thus $\rho_i \neq \rho$). The danger here is that when reaching

the supposed node v these successful agents could see all the $|L_i^*|$ labels of ρ_i (with the possible "help" of some Byzantine agents). At this point, it may be tempting to consider that gathering is over but this could be wrong especially in the case where $\rho_i \neq \rho$ and some good agents did not reach a supposed node v in phase i. To circumvent this problem, the idea is to get the good agents thinking that $\rho_i = \rho$ to fetch the (possible) others for which $\rho_i \neq \rho$ via a traversal of the entire graph using procedure $EXPLO(n)$ (refer to the description of state `tower`). To allow this, an agent for which $\rho_i \neq \rho$ will wait a prescribed amount of rounds in order to leave enough time for possible good agents to fetch it (refer to the description of state `wait-for-a-tower`). For our purposes, it is important to prevent the agents from being fetched any old how by any group, especially those containing only Byzantine agents. Hence our algorithm is designed in such a way that within each phase at most one group, called a *tower* and made up of at least $f+1$ agents, will be unambiguously recognized as such and be allowed to fetch the other agents via an entire traversal of the graph (this guarantee principally results from the rules that are prescribed in the description of state `tower builder`). When a tower has finished the execution of procedure $EXPLO(n)$ in some phase i, our algorithm guarantees that all good agents are together and declare gathering is over at the same time (whether the assumed configuration ρ_i corresponds to the real initial configuration or not). On the other hand, in every phase i, if a tower is not created or "vanishes" (because there at not at least $f+1$ agents inside of it anymore) before the completion of its traversal, no good agent will declare that gathering is over in phase i. In the worst case, the good agents will have to wait until assuming a good hypothesis about the real initial configuration, in order to witness the creation of a tower which will proceed to an entire traversal of the network (and thus declare gathering is over). We now give a detailed description of the algorithm. (Due to the lack of space, the proof of correctness of the algorithm is omitted but will appear in the journal version of the paper).

Algorithm Byz-Known-Size with parameter n (known size of the graph)

The algorithm is made up of two parts. The first part aims at ensuring that all agents are woken up before proceeding to the second part which is actually the heart of the algorithm.

Part 1. As soon as an agent is woken up by the adversary or another agent, it starts proceeding to a traversal of the entire graph and wakes up all encountered agents that are still dormant. This is done using procedure $EXPLO(n)$ where n is the size of the network which is initially known to all agents. Once the execution of $EXPLO(n)$ is accomplished, the agent backtracks to its starting node by traversing all edges traversed in $EXPLO(n)$ in the reverse order and the reverse direction.

Part 2. In this part, the agent works in phases numbered $1, 2, 3, \cdots$. During the execution of each phase, the agent can be in one of the following five states:

`setup`, `tower builder`, `tower`, `wait-for-a-tower`, `failure`. Below we describe the actions of an agent A in each of the states as well as the transitions between these states within phase i. We assume that in every round agent A tells the others (sharing the same node as agent A) in which state it is. In some states, the agent will be required to tell more than just its current state: we will mention it in the description of these states. Moreover, in the description of every state X, when we say "agent A transits to state Y", we exactly mean agent A remains in state X until the end of the current round and is in state Y in the following round. Thus, in each round of this part, agent A is always exactly in one state.

At the beginning of phase i, agent A enters state `setup`.

State `setup`.

Let ρ_i be the i-th configuration of enumeration Θ (refer to above). If the label l of agent A is not in ρ_i, then it transits to state `wait-for-a-tower`. Otherwise, let X be the set of the shortest paths in ρ_i leading from the node containing the agent having label l, to the node containing the smallest label of the supposed configuration. Each path belonging to X is represented as the corresponding sequence of port numbers. Let π be the lexicographically smallest path in X (the lexicographic order can be defined using the total order on the port numbers). Agent A follows path π in the real network. If , following path π, agent A has to leave by a port number that does not exist in the node where it currently resides, then it transits to state `wait-for-a-tower`. In the same way, it also transits to state `wait-for-a-tower` if, following path π, agent A enters at some point a node by a port number which is not the same as that of path π. Once path π is entirely followed by agent A, it transits to state `tower builder`.

State `tower builder`.

When in state `tower builder`, agent A can be in one of the following three substates: `yellow`, `orange`, `red`. In all of these substates the agent does not make any move: it stays at the same node denoted by v. At the beginning, agent A enters substate `yellow`. By misuse of language, in the rest of this paper we will sometimes say that an agent "is `yellow`" instead of "is in substate `yellow`". We will also use the same kind of shortcut for the two other colors. In addition to its state, we also assume that in every round agent A tells the others in which substate it is.

Substate `yellow`

Let k be the number of labels in configuration ρ_i. Agent A waits $T(EXPLO(n)) + n$ rounds. If during this waiting period, there are at some point at least k `orange` agents at node v then agent A transits to substate `red`. Otherwise, if at the end of this waiting period there are not at least k agents residing at node v such that each of them is either `yellow` or `orange`, then agent A transits to state `wait-for-a-tower`, else it transits to substate `orange`.

Substate `orange`

Agent A waits at most $T(EXPLO(n))+n$ rounds to see the occurrence of one of the following two events. The first event is that there are not at least k agents residing at node v such that each of them is either `yellow` or `orange`. The second

event is that there are at least k `orange` agents residing at node v. Note that the two events cannot occur in the same round. If during this waiting period, the first (resp. second) event occurs, then agent A transits to state `wait-for-a-tower` (resp. substate `red`). If at the end of the waiting period, none of these events has occurred, then agent A transits to substate `wait-for-a-tower`.

Substate `red`

Agent A waits $T(EXPLO(n)) + n$ rounds. If at each round of this waiting period there are at least k `red` agents at node v, then at the end of the waiting period, agent A transits to state `tower`. Otherwise, there is a round during the waiting period in which there are not at least k `red` agents at node v: agent A then transits to state `wait-for-a-tower` as soon as it notices this fact.

State `tower`.

Agent A can enter state `tower` either from state `tower builder` or state `wait-for-a-tower`. While in this state, agent A will execute all or part of procedure $EXPLO(n)$. In both cases we assume that, in every round, agent A tells the others the edge traversal number of $EXPLO(n)$ it has just made (in addition to its state). We call this number the index of the agent. Below, we distinguish and detail the two cases.

When agent A enters state `tower` from state `tower builder`, it starts executing procedure $EXPLO(n)$. In the first round, its index is 0. Just after making the j-th edge traversal of $EXPLO(n)$, its index is j. Agent A carries out the execution of $EXPLO(n)$ until its term, except if at some round of the execution the following condition is not satisfied, in which case agent A transits to state `failure`. Here is the condition: the node where agent A is currently located contains a group $\mathcal{S}$ of at least $f+1$ agents in state `tower` having the same index as agent A. $\mathcal{S}$ includes agent A but every agent that is in the same node as agent A is not necessarily in $\mathcal{S}$. If at some point this condition is satisfied and the index of agent A is equal to $P(n)$, which is the total number of edge traversals in $EXPLO(n)$ (refer to Section 2), then agent A declares that gathering is over.

When agent A enters state `tower` from state `wait-for-a-tower`, it has just made the s-th edge traversal of $EXPLO(n)$ for some s
(cf. state `wait-for-a-tower`) and thus, its index is s. Agent A executes the next edge traversals i.e., the $s+1$-th, $s+2$-th, $\cdots$, and then its index is successively $s+1$, $s+2$, etc. Agent A carries out this execution until the end of procedure $EXPLO(n)$, except if the same condition as above is not fulfilled at some round of the execution of the procedure, in which case agent A also transits to state `failure`. As in the first case, if at some point the node where agent A is currently located contains a group $\mathcal{S}$ of at least $f+1$ agents in state `tower` having an index equal to $P(n)$, then agent A declares that gathering is over.

State `wait-for-a-tower`.

Agent A waits at most $5T(EXPLO(n)) + 4n$ rounds to see the occurrence of the following event: the node where it is currently located contains a group of at least $f+1$ agents in state `tower` having the same index t. If during this waiting period, agent A sees such an event, we distinguish two cases. If $t < P(n)$, then it

makes the $t+1$-th edge traversal of procedure $EXPLO(n)$ and transits to state `tower`. If $t = P(n)$, then it declares that gathering is over.

Otherwise, at the end of the waiting period, agent A has not seen such an event, and thus it transits to state `failure`.

State `failure`. Agent A backtracks to the node where it was located at the beginning of phase i. To do this, agent A traverses in the reverse order and the reverse direction all edges it has traversed in phase i before entering state `failure`. Once at its starting node, agent A waits $10T(EXPLO(n)) + 9n - p$ rounds where p is the number of elapsed rounds between the beginning of phase i and the end of the backtrack it has just made. At the end of the waiting period, phase i is over. In the next round, agent A will start phase $i+1$.

4 Unknown Graph Size

In this section, we consider the same problem, except we assume that the agents are not initially given the size of the graph. Under this harder scenario, we aim at proving the following theorem.

Theorem 4. *Deterministic f-Byzantine gathering of k good agents is possible in any graph of unknown size if, and only if $k \geq f + 2$.*

As mentioned in Subsection 1.2, we know from [14] that:

Theorem 5 ([14]). *Deterministic f-Byzantine gathering of k good agents is not possible in some graphs of unknown size if $k \leq f + 1$.*

In view of Theorem 5, it is then enough to show the following theorem in order to prove Theorem 4.

Theorem 6. *Deterministic f-Byzantine gathering of k good agents is possible in any graph of unknown size if $k \geq f + 2$.*

Hence, similarly as in Section 3, the rest of this section is devoted to showing a deterministic algorithm that gathers all good agents, but this time in an arbitrary network of unknown size and provided there are at least $f + 2$ good agents.

Before giving the algorithm, which we call *Algorithm Byz-Unknown-Size*, let us provide some intuitive ingredients on which our solution is based.

The algorithm of this section displays a number of similarities with the algorithm of the previous section, but there are also a number of changes to tackle the non-knowledge of the network size. Among the most notable changes, there is firstly the way of enumerating the configurations. Previously, the agents were considering the enumeration $\Theta = (\rho_1, \rho_2, \rho_3, \cdots)$ of $\mathcal{P}$ where $\mathcal{P}$ is the set of every configuration corresponding to a n-node graph in which there are at least $f + 1$ robots with pairwise distinct labels. Now, instead of considering Θ, the agents will consider the enumeration $\Omega = (\phi_1, \phi_2, \phi_3, \cdots)$ of $\mathcal{Q}$ where $\mathcal{Q}$ is the set of all configurations corresponding to a graph of any size (instead of size n only) in which there are at least $f + 2$ agents (instead of at least $f + 1$) with pairwise

distinct labels. Note that, as for set $\mathcal{P}$, set $\mathcal{Q}$ is also recursively enumerable. Another change stems from the function performed by a tower, which we also find here. In Algorithm Byz-Known-Size, the role of a tower was to fetch all awaiting good agents (which know that the tested configuration is not good) via procedure $EXPLO(n)$: in the new algorithm, we keep the exact same strategy. However, to be able to use procedure $EXPLO$ with a parameter corresponding to the size of the network, it is necessary, for the good agents that are members of a tower, to know this size. Hence, in our solution, before being considered as a tower and then authorized to make a traversal of the graph, a group of agents will have to learn the size of the graph. To do this, at least each good agent of the group will be required to make a simulation of procedure EST by playing the role of an explorer and using the others as its token. To carry out these simulations, it is also required for the group of agents to contain initially at least $f + 2$ members (explorer + token), even if subsequently it is required for a group of agents forming a tower to contain at least $f + 1$ members. Our algorithm is designed in such a way that if during the simulation of procedure EST by an agent playing the role of an explorer, we have the guarantee there are always at least $f + 1$ agents playing the role of its token, then the explorer will be able to recognize its own token without any ambiguity (and thus will act as if it performed procedure EST with a "genuine" token). Of course, the agents will not always have such a guarantee (especially due to the possible bad behavior of Byzantine agents when testing a wrong configuration) and will not be able to detect in advance whether they will have it or not. Besides, some other problems can arise including, for example, some Byzantine explorer which takes too much time to explore the graph (or worse still, "never finishes" the exploration). However we will show that in all cases, the good agents can never learn an erroneous size of the graph (even with the duplicity of Byzantine agents when testing a wrong configuration). We also show that good agents are assured of learning the size of the network when testing a good configuration at the latest (in particular as the creation of a group of at least $f + 2$ agents and the aforementioned guarantee are ensured when testing a good configuration). As for Algorithm Byz-known-Size, in the worst case the good agents will have to wait until assuming a good hypothesis about the real initial configuration, in order to declare gathering is over. The details of Algorithm Byz-Unknown-Size (sketched above) and its analysis will appear in the journal version of the paper.

5 Conclusion

We provided a deterministic f-Byzantine gathering algorithm for arbitrary connected graphs of known size (resp. unknown size) provided that the number of good agents is at least $f+1$ (resp. $f+2$). By providing these algorithms, we closed the open question of what minimum number of good agents $\mathcal{M}$ is required to solve the problem, as each of our algorithms perfectly matches the corresponding lower bound on $\mathcal{M}$ stated in [14], which is of $f + 1$ when the size of the network is known and of $f + 2$ when it is unknown. Our work also highlighted the fact

that the ability for the Byzantine agents to change their labels has no impact in terms of feasibility when the size of the network is initially unknown, since it was proven in [14] that $\mathcal{M}$ is also equal to $f + 2$ when the Byzantine agents do not have this ability.

While we gave algorithms that are optimal in terms of required number of good agents, we did not try to optimize their time complexity. Actually, the time complexity of both our solutions depends on the enumerations of the initial configurations, which clearly makes them exponential in n and the labels of the good agents in the worst case. Hence, the question of whether there is a way to obtain algorithms that are polynomial in n and in the labels of the good agents (with the same bounds on $\mathcal{M}$) remains an open problem.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM J. Comput. 36(1), 56–82 (2006)
2. Alpern, S.: Rendezvous search: A personal perspective. Operations Research 50(5), 772–795 (2002)
3. Alpern, S.: The theory of search games and rendezvous. International Series in Operations Research and Management Science. Kluwer Academic Publishers (2003)
4. Bampas, E., Czyzowicz, J., Gąsieniec, L., Ilcinkas, D., Labourel, A.: Almost optimal asynchronous rendezvous in infinite multidimensional grids. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 297–311. Springer, Heidelberg (2010)
5. Barborak, M., Malek, M.: The consensus problem in fault-tolerant computing. ACM Comput. Surv. 25(2), 171–220 (1993)
6. Chalopin, J., Das, S., Kosowski, A.: Constructing a map of an anonymous graph: Applications of universal sequences. In: Lu, C., Masuzawa, T., Mosbah, M. (eds.) OPODIS 2010. LNCS, vol. 6490, pp. 119–134. Springer, Heidelberg (2010)
7. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: Gathering. SIAM J. Comput. 41(4), 829–879 (2012)
8. Collins, A., Czyzowicz, J., Gąsieniec, L., Labourel, A.: Tell me where I am so I can meet you sooner. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 502–514. Springer, Heidelberg (2010)
9. Czyzowicz, J., Kosowski, A., Pelc, A.: How to meet when you forget: log-space rendezvous in arbitrary graphs. Distributed Computing 25(2), 165–178 (2012)
10. Czyzowicz, J., Pelc, A., Labourel, A.: How to meet asynchronously (almost) everywhere. ACM Transactions on Algorithms 8(4), 37 (2012)
11. Das, S., Dereniowski, D., Kosowski, A., Uznański, P.: Rendezvous of distance-aware mobile agents in unknown graphs. In: Halldórsson, M.M. (ed.) SIROCCO 2014. LNCS, vol. 8576, pp. 295–310. Springer, Heidelberg (2014)
12. Défago, X., Gradinariu, M., Messika, S., Raipin-Parvédy, P.: Fault-tolerant and self-stabilizing mobile robots gathering. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 46–60. Springer, Heidelberg (2006)
13. Dessmark, A., Fraigniaud, P., Kowalski, D.R., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica 46(1), 69–96 (2006)
14. Dieudonné, Y., Pelc, A., Peleg, D.: Gathering despite mischief. ACM Transactions on Algorithms 11(1), 1 (2014)

15. Dieudonné, Y., Pelc, A., Villain, V.: How to meet asynchronously at polynomial cost. In: ACM Symposium on Principles of Distributed Computing, PODC 2013, Montreal, QC, Canada, July 22-24, pp. 92–99 (2013)
16. Fraigniaud, P., Pelc, A.: Deterministic rendezvous in trees with little memory. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 242–256. Springer, Heidelberg (2008)
17. Fraigniaud, P., Pelc, A.: Delays induce an exponential memory gap for rendezvous in trees. ACM Transactions on Algorithms 9(2), 17 (2013)
18. Guilbault, S., Pelc, A.: Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. Theor. Comput. Sci. 509, 86–96 (2013)
19. Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., Yamashita, M.: The gathering problem for two oblivious robots with unreliable compasses. SIAM J. Comput. 41(1), 26–46 (2012)
20. Kowalski, D.R., Malinowski, A.: How to meet in anonymous network. Theor. Comput. Sci. 399(1-2), 141–156 (2008)
21. An, H.-C., Krizanc, D., Rajsbaum, S.: Mobile agent rendezvous: A survey. In: Flocchini, P., Gąsieniec, L. (eds.) SIROCCO 2006. LNCS, vol. 4056, pp. 1–9. Springer, Heidelberg (2006)
22. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann (1996)
23. De Marco, G., Gargano, L., Kranakis, E., Krizanc, D., Pelc, A., Vaccaro, U.: Asynchronous deterministic rendezvous in graphs. Theor. Comput. Sci. 355(3), 315–326 (2006)
24. Miller, A., Pelc, A.: Fast rendezvous with advice. In: Algorithms for Sensor Systems - 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2014, Wroclaw, Poland, September 12, pp. 75–87 (2014); Revised Selected Papers
25. Miller, A., Pelc, A.: Time versus cost tradeoffs for deterministic rendezvous in networks. In: ACM Symposium on Principles of Distributed Computing, PODC 2014, Paris, France, July 15-18, pp. 282–290 (2014)
26. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM 27(2), 228–234 (1980)
27. Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4) (2008)
28. Schelling, T.: The Strategy of Conflict. Oxford University Press, Oxford (1960)
29. Ta-Shma, A., Zwick, U.: Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences. ACM Transactions on Algorithms 10(3), 12 (2014)