Brief Announcement: Gathering with a strong team in weakly Byzantine environments

Jion Hirose[1], Masashi Tsuchida[1], Junya Nakamura[2], Fukuhito Ooshita[1], and Michiko Inoue[1]

[1] Nara Institute of Science and Technology, Japan
[2] Toyohashi University of Technology, Japan

Abstract. We study the gathering problem requiring a team of synchronous mobile agents in arbitrary networks. The team consists of k agents with unique identifiers (IDs), and f of them are weakly Byzantine agents, which behave arbitrarily except falsifying their IDs. In this paper, we focus on a strong team, i.e., a team with a few Byzantine agents, and propose two gathering algorithms, one with non-simultaneous termination and one with simultaneous termination, in the case of $4f^2+9f+4 \leq k$. The second algorithm is faster than the existing algorithm if the largest IDs among non-Byzantine agents and among all agents are comparable and all agents awake at the same time.

Keywords: Mobile agents · Gathering · Byzantine faults.

1 Introduction

Mobile agents (in short, agents) are software programs that move autonomously and perform various tasks in a distributed system. A task that collects multiple agents on the same node is called a *gathering*. This task has been widely studied because, by accomplishing this task, agents can exchange information among them and easily carry out future cooperative behaviors.

In operations of large-scale distributed systems, we cannot avoid facing faults of agents. Byzantine faults are known to be the worst faults because Byzantine faults do not make any assumption about the behavior of faulty agents (called *Byzantine agents*).

In the literature, gathering algorithms have been considered for environments with *weakly Byzantine agents* [4] and *strongly Byzantine agents* [1, 2]. Weakly Byzantine agents perform arbitrary behaviors except falsifying their own IDs, and strongly Byzantine agents perform arbitrary behaviors, including falsifying their own IDs.

We seek an algorithm that makes a team of agents gather with small time complexity in synchronous environments with weakly Byzantine agents. We follow the model in [4] in principle. The team consists of k agents with unique identifiers (IDs), and f of them are weakly Byzantine agents. Every agent moves in synchronous rounds and cannot leave any information on nodes. The agents know the upper bound N of the number of nodes and awake at the same time.

Table 1. Gathering algorithms with weakly Byzantine agents in synchronous environments. Here, n is the number of nodes, N is the upper bound of n, $|\Lambda_{good}|$ (resp., $|\Lambda_{all}|$) is the length of the largest ID among good (resp., all) agents, $X(n)$ is the number of rounds required to explore any network composed of at most n nodes.

	Input	Startup delay	Condition of #Byzantine agents	Simultaneous termination	Time complexity
[4]	n	Possible	$f+1 \leq k$	Possible	$O(n^4 \cdot \|\Lambda_{good}\| \cdot X(n))$
Algorithm 1	N	Impossible	$4f^2+9f+4 \leq k$	Impossible	$O((f+\|\Lambda_{good}\|) \cdot X(N))$
Algorithm 2	N	Impossible	$4f^2+9f+4 \leq k$	Possible	$O((f+\|\Lambda_{all}\|) \cdot X(N))$

Our contributions: Table 1 shows a comparison between the two proposed algorithms and the existing fastest algorithm [4]. The algorithm [4] tolerates any number of weakly Byzantine agents, however not so many agents are subject to faults in practice. Hence, in this paper, we reduce the time complexity by taking advantage of a *strong team*, that is, a team with a few Byzantine agents. We propose two algorithms, one with non-simultaneous termination and one with simultaneous termination, that tolerate f weakly Byzantine agents with a strong team composed of at least $4f^2+9f+4$ agents. The second algorithm significantly reduces the time complexity compared to the algorithm [4] if $|\Lambda_{all}| = O(|\Lambda_{good}|)$ holds and all agents awake at the same time.

2 Algorithms

Before explaining the proposed algorithms, we introduce the graph exploration procedure used in them. The exploration procedure, called $EXPLO(N)$, allows an agent to traverse all nodes of any graph composed of at most N nodes, starting from any node of the graph. An implementation of this procedure is based on universal exploration sequences and is a corollary of the result by Reingold [5]. The number of moves of $EXPLO(N)$ is denoted by X_N.

Non-simultaneous termination: First we explain the algorithm *with non-simultaneous termination*, which requires the following conditions: (1) every good agent terminates the algorithm, and (2) when all the good agents terminate an algorithm, they are on the same node.

The algorithm achieves the gathering by three stages: CollectID, MakeGroup, and Gather stages. In the CollectID stage, agents collect IDs of all good agents. In the MakeGroup stage, agents make a *reliable group*, which is composed of at least $4f+4$ agents. In the Gather stage, all good agents gather on a single node and achieve the gathering. Due to space limitation, we explain the overview under the assumption that agents know f.

In the CollectID stage, agents collect IDs of all good agents. To do this, each agent a_i executes a rendezvous algorithm, which repeats waiting (waiting for X_N rounds) and exploration (executing $EXPLO(N)$) depending on the extended label [3]. This guarantees that a_i meets all good agents in $O((\log a_i.ID) \cdot X_N)$ rounds, where $a_i.ID$ is the ID of a_i. Hence, by recording $a_j.ID$ in variable $a_i.L$ when a_i meets a_j, a_i can eventually know IDs of all good agents.

In the MakeGroup stage, agents make a reliable group composed of at least $4f+4$ agents. To do this, agents with small IDs keep waiting, and the other agents search for the agents with small IDs. More concretely, if $f+1$ smallest IDs in $a_i.L$ contains $a_i.ID$, a_i keeps waiting during this stage. Otherwise, a_i assigns $\min(a_i.L)$ to variable $a_i.target$, and searches for the agent with ID $a_i.target$, say a_{target}, by executing $EXPLO(N)$. If a_i finds a_{target}, it ends the search and waits there. If a_i does not find a_{target} even after completing $EXPLO(N)$, it regards a_{target} as a Byzantine agent. In this case, a_i assigns the second smallest ID in $a_i.L$ to $a_i.target$, and searches again. Agent a_i continues this behavior until it finds the target agent. This behavior guarantees that at least $4f+4$ agents gather in one node before each agent completes the $(f+1)$-th exploration. In other words, agents can make a reliable group. The ID of the target agent in a reliable group is used as the group ID. For the Gather stage, a reliable group is divided into two groups, an exploring group and a waiting group, so that each of which contains at least $2f+2$ agents.

In the Gather stage, agents achieve the gathering after at least one reliable group is created. To do this, agents first collect group IDs of all reliable groups. More concretely, while agents in a waiting group keep waiting, the other agents (in an exploring group or not in a reliable group) execute $EXPLO(N)$. When agent finds a reliable group, it records the group ID. After collecting group IDs, agents move to the node where the waiting group of the smallest group ID stays.

However, there are two issues to implement the above behavior. The first issue is that agents not in a reliable group cannot instantly know the fact that a reliable group has been created, and so they do not know when to transition to the Gather stage. The second issue is that agents do not know the exact value of f. Our proposed algorithm is designed to solve these problems.

Simultaneous termination: We briefly explain the algorithm *with simultaneous termination*, which requires all the good agents to terminate the algorithm at the same round. Although the first algorithm allows agents to gather at a single node, agents may terminate the algorithm at different times. Hence, agents wait without terminating and estimate the time when all good agents arrive at the gathering node. Since this time depends on the largest ID among agents (because of the CollectID stage), the agents make a consensus on the largest ID and estimate the termination time. At the estimated time, all the agents terminate.

References

1. Bouchard, S., Dieudonné, Y., Ducourthial, B.: Byzantine gathering in networks. Distributed Computing **29**(6), 435–457 (2016)
2. Bouchard, S., Dieudonné, Y., Lamani, A.: Byzantine gathering in polynomial time. In: ICALP. pp. 147:1–147:15 (2018)
3. Dessmark, A., Fraigniaud, P., Kowalski, D.R., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica **46**(1), 69–96 (2006)
4. Dieudonné, Y., Pelc, A., Peleg, D.: Gathering Despite Mischief. ACM Transactions on Algorithms **11**(1), 1–28 (2014)
5. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 17:1–17:24 (2008)