Gathering of Anonymous Agents

Arnhav Datar*
IIT Madras
Chennai, India
adatar@cmu.edu

Nischith Shadagopan M N
IIT Madras
Chennai, India
cs18b102@smail.iitm.ac.in

John Augustine†
IIT Madras
Chennai, India
augustine@cse.iitm.ac.in

ABSTRACT

Motivated by the increasing popularity of mobile agents and swarm robotics, we study the fundamental and widely studied problem of gathering k autonomous and anonymous agents placed in arbitrary vertices of a graph comprising n nodes. In this work, we present algorithms that, for the first time, ensure gathering of anonymous mobile agents in any arbitrary graph. Moreover, our algorithms are fast. The canonical case where the graph is complete and $k = n$ runs in expected time that is sublogarithmic in n.

Importantly, these robot swarms are often deployed in vulnerable contexts where security may be compromised. Thus, we consider the case where f of the agents are Byzantine (i.e., compromised and therefore malicious) and can deviate from the protocol in an adversarial manner. Our main result is a fast gathering algorithm when the Byzantine agents are controlled by a strongly adaptive adversary that – in each round – can view all the moves made by the good agents and then strategically make the moves of all Byzantine agents in a coordinated fashion. For the canonical case where the graph is complete and $k = n$, we provide a gathering algorithm that runs in time that is polylogarithmic in n provided $f \in O(k/\log k)$. This is the first known result on gathering anonymous agents with Byzantine failures.

Our results generalize to arbitrary graphs and hold with high probability. Moreover, we complement our upper bounds with lower bounds that are tight to within polylog(n) factors.

KEYWORDS

Gathering; Swarm Robotics; Byzantine Faults; Self-Organization

ACM Reference Format:
Arnhav Datar, Nischith Shadagopan M N, and John Augustine. 2023. Gathering of Anonymous Agents. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 15 pages.

1 INTRODUCTION

Swarm Robotics envisions groups of mobile robots (or agents as we call them) self-organizing and cooperating toward the resolution of common objectives. These smaller agents have various advantages such as efficiency, robustness, and scalability [22, 27]. There has been a lot of research in the areas of gathering [1, 28], scattering [40], exploring [13, 32], and flocking [29]. In many cases, the agents are deployed in adverse environments [15].

We study the *gathering problem* wherein the agents are required to gather at a single location in the absence of any prior knowledge or infrastructure (eg. no global positioning) and ensure coordination despite the presence of faulty/malicious robots. In particular, we assume that k *anonymous* agents are placed arbitrarily on a graph G of n nodes. The agents operate synchronously in rounds. Within each round, the agents co-located in the same node can communicate with each other via broadcasting messages and then choose to either stay or move to neighboring nodes. The goal is to ensure that the agents gather at some node in G. Importantly, the gathering protocol must be able to tolerate up to f Byzantine agents that can deviate arbitrarily from the protocol. This includes omitting to move, moving to arbitrary locations, and sending arbitrary messages to other agents to prevent *good agents* (i.e., non-Byzantine agents) from gathering. Furthermore, the Byzantine agents can collude with each other and coordinate their activities to foil gathering. To model this colluding behavior, we assume that a single Byzantine adversary controls all Byzantine agents simultaneously. Moreover, the adversary is adaptive and can adapt its behavior over time.

1.1 Related Works

Byzantine gathering of non-anonymous agents in networks was initially explored by Dieudonné *et al.* [16]. They proved fundamental bounds for the minimum number of good agents required for gathering in deterministic settings. They showed that gathering could be achieved when $k \geq 2f + 1$ even when Byzantine agents could lie about their identities. However, this bound worsened to $4f+1$ when the agents did not know n, the number of nodes in G. These bounds were tightened by Bouchard *et al.* [8]. They showed that $f + 1$ good agents suffice when n is known and the bound only worsens to $f + 2$ when n is not known. Unfortunately, the algorithms in [8, 16] require round complexities that are exponential in n, and therefore practically infeasible. Subsequently, Bouchard *et al.* [9] gave a polynomial-time deterministic algorithm using a global knowledge of size $O(\log\log\log n)$ that gathered agents when $k \geq 5f^2 + 6f + 2$. Hirose *et al.* [25] gave deterministic algorithms for agents that cannot lie about their IDs. Their algorithms run in $O((f + \Lambda_{all}) \cdot X(n))$ time where Λ_{all} represents the length of the maximum ID of agents and $X(n)$ is the number of rounds to explore a network of size n. Their algorithm ran when $k \geq 4f^2 + 9f + 4$.

Defago *et al.* [14] used scheduling strategies and visibility in networks (the ability to see agents in nodes less than a certain distance from the current node) to come up with probabilistic gathering algorithms. Miller *et al.* [36] were able to solve the Byzantine gathering problem when the visibility was at least the radius of the graph and provided an $O(mn^2)$ deterministic algorithm when $k \geq 2f + 1$, where m is the number of edges in the graph. Tsuchida *et al.* [42] came up with an $O(fm)$ deterministic algorithm, using authenticated whiteboards. Whiteboards are areas prepared on each node

*The first two authors are listed alphabetically.
†John Augustine is associated with and supported by the Cybersecurity Centre set up under the IIT Madras Institute of Eminence scheme.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

at which agents can leave information. Authenticated whiteboards take this a step further where each agent is dedicated a space to write information along with its digital signature. However, authenticated whiteboards and network visibility are significantly advanced features usually unavailable to mobile agents.

Many applications require agents to be anonymous owing to security issues, so it is natural to study the gathering of anonymous agents. Dieudonné and Pelc [17] gave a comprehensive study of all deterministically gatherable configurations and present deterministic algorithms that run in time polynomial in the size of the network. More recently, Bouchard *et al.* [10] showed that agents can gather in polynomial in n rounds even without any exchange of messages; in this work, agents cannot identify each other, but indeed possess unique IDs. See Pelc [38] for a current understanding of deterministic gathering protocols. Memory constraints are crucial in the context of mobile agents because they tend to be quite lightweight in terms of hardware. Thus, there has been some work on understanding the memory requirements of anonymous gathering in arbitrary graphs [12], in trees [24], and rings [33]. There have also been several works on anonymous gathering of agents in the asynchronous look-compute-move model on rings [18, 30, 31] and trees [21].

Randomized approaches for gathering and other related problems have been studied for many years; see the excellent book by Alpern and Gal [3]. Surprisingly, there has been very little work on randomized gathering in the anonymous Byzantine agents setting despite the somewhat obvious power of randomization to mitigate anonymity and Byzantine behavior. The notion of coalescing random walks [11] by Cooper *et al.* is quite relevant in our context. Here, a set of particles perform independent random walks on a graph, but whenever more than one of those particles meet at a node, they coalesce into a single particle. This notion can be readily implemented in our context because our model permits co-located agents to generate common random bits, thereby enabling co-located good agents to coalesce and continue their random walks in unison. The expected time for all particles in a graph of n nodes to coalesce is called the *coalescence time* and is denoted $C(n)$. The coalescence time is analogous in our context to the time required to gather. Cooper *et al.* [11] showed that $C(n) = O\left(\frac{1}{1-\lambda_2} \cdot \left(\log^4 n + \frac{n}{\nu}\right)\right)$. Here, λ_2 is the second eigenvalue of the transition matrix of the random walk and $\nu = \frac{n}{4m^2} \sum_{v \in V} d^2(v)$, where $d(.)$ represents the degree. Meanwhile, Aldous and Fill [2] showed that $C(n) \in O\left(\frac{(r-1)n}{r-2}\right)$ for r-regular graphs implying $C(n) \in O(n)$ for complete graphs. Eguchi *et al.* [19] worked on fast randomized algorithms for the rendezvous of two agents in a graph. However, they worked under the limiting assumptions that the agents were initially located in adjacent vertices and there were only two agents in the graph.

Our goal is to initiate work on fast randomized gathering of anonymous agents despite the presence of Byzantine agents. In fact, all our running times are polylog(n) when $k \in \Omega(n)$. We crucially rely on the ability of agents to toss coins and generate uniform and independent random bits. Combined with synchrony and broadcast based communication, we can also ensure that the good agents co-located at any node can also generate common random bits using the following primitive procedure. If all co-located agents broadcast a random string of (say) length s synchronously, then all agents will receive the same strings from each agent and a bitwise XOR of the strings will allow the agents to compute a common string that is s bits long and is entirely uniform and independent as long as at least one of the agents is good. We believe this is an appropriate abstraction for two complementary reasons. Firstly, this allows us to gain significant ground in producing results that are a lot more scalable, fast, and resilient to Byzantine failures. Secondly, we have several technological capabilities that can be leveraged to provide such a common coin. At a physical level, there are many algorithms designed to arrange the agents in the form of a circle [23, 43] or other patterns [41]. Once a suitable formation is reached, the agents can be forced to display their random bits collectively and simultaneously through physical mechanisms like lights. At an algorithmic level, there has been a rich body of work on collective coin tossing [6, 7, 20, 35, 39]. Regardless of the technology used, we believe common random bits will prove useful for a variety of robot coordination problems.

1.2 Our Model

We study the problem of gathering k anonymous mobile agents placed arbitrarily in the nodes of a graph $G = (V, E)$. The $n = |V|$ nodes of G are just containers for agents and cannot compute, communicate, or store any information. Edges are conduits for agents moving between neighboring nodes. Let δ_v denote the degree of a node $v \in V$. There are δ_v ports numbered 0 to $\delta_v - 1$ at each v (i.e., one per incident edge). An edge $e = (u, v) \in E$ bidirectionally connects a port from u to a port in v. The agents cannot identify/mark nodes, but they can see the port number from which they entered a node. Therefore, the sequence of port numbers from u to access node v – once learned – can be used to go back to u and return to v.

The agents operate synchronously. The k agents are initially (at round $r = 0$) placed in arbitrary nodes of the graph. The agents can compute, store information, and communicate with other agents in the same node using local broadcast. We focus on the case where k is at most n to illustrate our ideas. We briefly discuss how our techniques apply when $k \geq n$. Some f agents are Byzantine and can collude and deviate arbitrarily from the protocol. The rest are *good agents*. We assume that a malicious adversary (or just adversary) controls the behavior of all Byzantine agents. The agents are aware of an upper-bound on the number of Byzantine agents f and the number of agents k. The good agents do not know G, but the adversary knows G including all the port numbers.

Computation is defined by a finite state machine coupled with the ability to generate random bits and communicate messages. At each round $r > 0$, each good agent:

(1) broadcasts (to all other agents in their respective current node) a message that is a function of its state and includes public random bits,
(2) receives messages broadcasted by other co-located agents,
(3) transitions to a new state as a function of the current state, messages received, and random bits,
(4) chooses to either stay in the current location or move through one of the ports (based on the chosen state),

(5) and finally updates the state with port number through which it entered the new node (if it moved).

All agents must broadcast a message at the start of each round. Note that this is not a limitation, because agents can send an empty $\perp$ message if they have nothing to send. Once all agents broadcast their messages, the adversary learns all the messages (even if there are no Byzantine agents co-located with some good agents).

The Byzantine adversary is adaptive in the sense that it can change its behavior based on the behavior of the good agents as observed by the Byzantine agents. Under a weakly-adaptive Byzantine adversary, the good agents can use private random bits to compute their new state (and therefore their new location). This is disallowed when the Byzantine adversary is strongly-adaptive[1] and the implication is that – within each round – the strongly-adaptive Byzantine adversary can predict the new state and the move made by the good agents after broadcasts are received.

1.3 Our Techniques and Contributions

As is the case with all the foundational work on Byzantine fault tolerance in distributed settings [34], we focus our efforts on complete graphs. This serves us well in that it allows us to focus on essential gathering strategies without structural concerns. The symmetric nature of the complete graph however makes the solutions non-trivial. At a high level, our algorithms work by forming groups that are initially small. A group is essentially a set of good agents who have agreed to work collaboratively with each other. It is important to note that groups may be infiltrated by Byzantine agents, so good agents will not be able to distinguish between other good agents who are members of its group from the spurious Byzantine members. The groups act in a coordinated fashion to meet other groups and merge with them to form larger groups until all agents form one single group. Unlike the coalescing random walks approach where the groups mechanically perform random walks to find other groups, we employ careful strategies to ensure that groups find each other fast. For example, our groups are not averse to splitting and regrouping to maximize their chance of encountering other groups. In the context of coalescing random walks, all groups that land on a node can immediately combine to form a new group. In our context, when a group of agents split, the individual agents can potentially interact with multiple other groups. The agents must then decide which groups must coalesce and which ones don't. This situation requires strategizing and efficient symmetry breaking.

We provide a series of results for the complete graph comprising both algorithms with essentially optimal time complexities and complementary negative results indicating the need for randomization. Our algorithms are organized carefully so that the ideas build on each other.

No Byzantine Agents: We start in Section 2 with an algorithm that runs in $O\left(\frac{n}{k}\sqrt{\log k}\right)$ expected time. Our goal is to take full advantage of $f = 0$ to speed up gathering. Our algorithm is a sub-logarithmic expected-time algorithm when $k \in \omega(n/\sqrt{\log n})$; note that the canonical $k = n$ case only requires $O(\sqrt{\log n})$ time on expectation.

[1]In non-anonymous settings[8, 16, 25], the terms strong and weak Byzantine adversaries refer respectively to whether Byzantine agents can lie about their IDs or not. We use the terms exclusively in the context of their adaptive nature.

The algorithm works in three parts. In the first part, agents try to coalesce with each other to form groups of a reasonable size. In the second part, all groups try to merge with a single group which has been randomly chosen. The main idea is to introduce some asymmetry into the gathering process by isolating one group as a leader group and then growing that group to reach a size that is linear in k. However, this process can take too much time to gather all agents into the large group. So in the last part, the large group (that is now of size linear in k) deterministically sweeps through the entire graph in $O(n/k)$ rounds and locates the other agents.

This algorithm introduces several crucial ideas that prove useful subsequently (and quite likely to be useful in future works as well). Firstly, when two good agents become members of the same group, they remain together in the group. This seemingly simple invariant greatly facilitates our analysis because we can view the progression of our algorithms as groups coalescing with each other until a single group is formed. Secondly, unlike coalescing random walks, the groups are not always co-located in the same node; the agents in a group may disperse briefly and regroup and this greatly speeds up gathering. Thirdly, at each stage of our algorithm, the groups are partitioned into *seekers* and *settlers*. The seekers try to search for the settlers while the settlers branch out and try to get seekers to come to their node. The seekers then coalesce with their choice of settler groups. Finally, it introduces us to the paradigm that once there is a big enough group we can achieve the gathering quickly by deterministically scanning the graph.

Weakly-Adaptive Adversary: We then describe in Section 3 a simple and intuitive gathering algorithm against a weakly-adaptive adversary, i.e., the setting in which Byzantine agents will need to decide their move in each round without knowing the moves made by good agents in the current round. Note that the Byzantine agents are nevertheless adaptive in the sense that they know all the moves made by all the agents in prior rounds. This algorithm builds on ideas developed under the setting without Byzantine agents. It can tolerate any $f < k$ and achieve gathering in $O\left(\frac{n^2}{(k-f)(n-f)}\log n\right)$ whp[2].

Strongly-Adaptive Adversary: Our algorithm in Section 4 can tolerate $f \in O(k/\log^{1+\epsilon} k)$ Byzantine agents (for any fixed $\epsilon > 0$) and gather in $O\left(\frac{n}{k}\log n \log k\right)$ rounds whp. The strongly-adaptive adversary is significantly more powerful because it can predict all the moves of the good agents and rush in with carefully coordinated moves for the Byzantine agents. Our algorithm proceeds in two parts. In the first part, we gather majority of the agents using a slightly more involved seeker-settler algorithm. Among other adaptations to the previous seeker-settler approach, here we require both seeker and settler groups to disperse and regroup. Since there are strongly-adaptive Byzantine agents a simple deterministic sweep of the graph proves to be ineffective for the second part. To counter this, we provide an iterative algorithm where the large group successively doubles the numbers of agents it sends to gather the agents that did not reach the large group in the previous round.

Arbitrary Graphs: Fortunately, our approaches on the complete graph lend themselves to arbitrary graphs as well. In Section 4.1, we extend the complete graph algorithm for the strongly-adaptive

[2]whp refers to with high probability or with probability $\geq 1 - \frac{1}{n^c}$ for $c \geq 0$

Settings	Time Complexity	Max. Byz.
Coalescing rand. walks [2, 11]	$O(n)^*$	$f < k$
Weakly-Adaptive Adversary	$O\left(\frac{n^2 \log n}{(k-f)(n-f)}\right)$	$f < k$
Strongly-Adaptive Adversary	$O\left(\frac{n}{k} \log n \log k\right)$	$f \in O(\frac{k}{\log^{1+\epsilon} k})$
No Byzantine Agents	$O\left(\frac{n}{k}\sqrt{\log k}\right)^*$	$f = 0$
Lower bound	$\Omega\left(\frac{n}{k-f}\right)$	$f < k - 1$[3]

Table 1: Known results on randomized gathering protocols for anonymous agents on complete graphs. The results inferred from the literature on coalescing random walks [2, 11] were known previously; all other results are our contribution. Similar results hold for general graphs as well. The running times marked with an $*$ are on expectation while all others hold whp. The $\tilde{O}$ notation is used to hide polylog(n) **factors.**

adversary to a $O(\tau \cdot \frac{n}{k} \log n \log k)$ round algorithm for any general graph while being able to withstand $O(k/\log k)$ Byzantine agents. Here, τ is the mixing time of a lazy random walk on the graph. Similarly, we are able to extend our algorithm when there are no Byzantine agents to get an $O(\tau \cdot \frac{n}{k}\sqrt{\log k})$ algorithm.

Deterministic Impossibility: As we have mentioned before, randomization is crucial in the context of gathering anonymous agents (with Byzantine agents only making things worse). To formalize this, in Section 5.1 we show that for regular graphs, there exists a port-numbering such that no deterministic algorithm can successfully gather even two agents. Our construction crucially relies on a decomposition of the complete graph into cycles and at most one perfect matching by Alperns and Gavlas [3].

Lower Bound: In Section 5.2, we show that any gathering algorithm for complete graphs that succeeds with constant probability must take $\Omega\left(\frac{n}{k-f}\right)$ time implying that our algorithms are essentially optimal to within polylog$(n+k)$ factors.

The summary of our results can be found in Table 1. Due to space limitations, some proof details are deferred to the supplementary material.

2 NO BYZANTINE AGENTS

The fault-free setting with $f = 0$ is of fundamental importance. Algorithm 3 for the weakly-adaptive adversary already achieves $O\left(\frac{n}{k} \log n\right)$ time and the lower bound for gathering (Section 5.2) shows that any algorithm will require at least $\Omega\left(\frac{n}{k}\right)$ time. A natural question is to ask whether the overhead factor can be reduced to sub-logarithmic in n. We answer this affirmatively and provide a randomized algorithm that gathers in $O(\frac{n}{k}\sqrt{\log k})$ expected time. We have divided our algorithm into three parts:

(1) **Merging:** We use the mergeIteration (cf. Algorithm 3) for $\Theta(\sqrt{\log k})$ iterations followed by breaking up large groups to get $\Theta(k/(e/2)^{\sqrt{\log k}})$ groups with the maximum group size $\leq (e/2)^{\sqrt{\log k}}$ with probability at least $1/2$.
(2) **Gathering to half:** A leader group is elected randomly and uniquely with probability at least $1/e$. The groups then split and go to random nodes and try to find the leader group. We show that in $O\left(\frac{n}{k}\sqrt{\log k}\right)$ rounds with a constant probability the leader group will be of size $\geq k/2$.
(3) **Search and Rescue:** The leader group gets all agents to itself in $O(2n/k)$ time.

Algorithm 1 Merge Iteration

```
1:  function MERGEITERATION
2:      coin ← NODAL_RAND(2)
3:                  ▷ NODAL_RAND(x) ← random ∈ {0, ..., x − 1}
4:      if coin = 0 then                                  ▷ seeker group
5:          All agents traverse e ← NODAL_RAND(n)
6:          if the group finds at least one settler agent then
7:              Pick a random settler agent s
8:              Ask s for edge e′ to traverse to reach its group
9:              Traverse e′ and coalesce with agents
10:                 at that node to form a new group.
11:         end if
12:     else                                              ▷ settler group
13:         for each agent a in the (settler) group do
14:             Choose a uniformly random, independent edge e′
15:             Traverse e′ in the forward direction.
16:             Broadcast the port number to traverse e′ in reverse
17:                 ▷ In the round when seekers execute Line 8
18:             Traverse the edge(e′) in reverse to
19:                 return back to starting node
20:         end for
21:     end if
22: end function
```

Analysis. The detailed analysis including full proofs can be found in the Appendix Section A. We will show that every iteration of Algorithm 2 will succeed with constant probability by showing that each part will succeed with constant probability. For this purpose, we prove Lemmas 2 and 3 for parts 1 and 2. We initially state a useful lemma about MERGEITERATION. Here in this case $f = 0$.

Lemma 1. *After* $\lceil 24 \cdot \frac{n^2}{(k-f)(n-f)} \rceil$ *rounds of* MERGEITERATION *the number of required group merges in the graph shrinks to* $1/e$ *in expectancy.*

PROOF SKETCH. In a nutshell, the proof works by arguing that any group will coalesce with another group with probability $\geq \frac{(k-f)(n-f)}{24n^2}$. To prove this, we take advantage of the limitations of the weakly-adaptive Byzantine adversary in that when settler agents disperse to random locations, the Byzantine agents cannot guess their destinations. This ensures that a seeker will only meet settler agents and no Byzantine agents with a probability of at least $\frac{(k-f)(n-f)}{24n^2}$. We can then conclude that in $\alpha = \lceil \frac{24n^2}{(k-f)(n-f)} \rceil$ rounds any good group will merge with another group with a probability of at least $1 - 1/e$. □

Algorithm 2 Complete Graph Gathering - No Byzantine Agents

while not gathered **do** ▷ Part 1
  repeat $\lceil 48 \cdot \frac{n}{k} \rceil \sqrt{\log k}$ rounds **times**
    mergeIteration()
  end repeat
  if $size(group) > (\frac{e}{2})^{\sqrt{\log k}}$ **then**
    Split into $\left\lceil \frac{size(group)}{(\frac{e}{2})^{\sqrt{\log k}}} \right\rceil$ small groups of similar size
  end if
  leader ← True with probability $1/k$ else False ▷ Part 2
  leader_group ← True if leader is present in group else False
  repeat $T\lceil \frac{n}{k} \rceil \sqrt{\log k} + \left\lfloor 43 \log_{44/43} 2 \right\rfloor$ **times**
    All agents move through a random incident edge e
    All agents broadcasts whether leader or not
    Leaders send the edge e; others note e′ (if received)
    All agents return to their groups via their respective e
    Agents that found leader agents shares
      a pair of edges (e, e′) to other agents in group
    Moves to leader group if any (e, e′) pair was shared
  end repeat
  if $size(group) \geq k/2$ **then** ▷ Part 3
    Scan the entire graph and ask agents to come to you
  else
    Wait $\lceil 2n/k \rceil$ rounds for the $\geq k/2$-size group
    If it arrives at any point merge with the big group
  end if
end while

Lemma 2. *With probability at least* $1/2$ *at the end of part 1 there will be at most* $3k/(e/2)^{\sqrt{\log k}}$ *groups with each group of size at most* $(e/2)^{\sqrt{\log k}}$.

Proof Sketch. We prove this lemma by using Markov's inequality on Equation 4 to show that X decreases by a factor of $2/e$ with a probability of at least $1/2$ after α rounds. Therefore in some $C \cdot \alpha$ rounds (i.e., C phases), X will decrease by a factor of $\leq 2/e$ in at least $C/2$ phases with probability $1/2$.

Following this we show that when the larger groups split, asymptotically we will not have a lot of new groups. We do this by leveraging the fact that there are at most k agents which will go to groups of size $\geq 0.5 \cdot (e/2)^{\sqrt{\log k}}$. □

Lemma 3. *At the end of Part 2, there will be a unique leader group with size* $\geq k/2$ *with probability at least* $1/40e$.

Proof Sketch. We can see that with probability $k \cdot \left(1 - \frac{1}{k}\right)^{k-1} \cdot \frac{1}{k} \geq 1/e$ there will be a unique leader group. We consider Y_i as the size of the leader group and g_i as the number of groups after $i \cdot \lceil \frac{n}{k} \rceil$ iterations of the algorithm respectively. We show that:

$$\mathbb{E}\left[Y_{i+1} | Y_i\right] \geq Y_i + \frac{k}{11} \left[\frac{\frac{Y_i}{4g_i}}{1 + \frac{Y_i}{4g_i}} \right]$$

From this point we are primarily focused on the case where $\frac{Y_i}{4g_i} \leq 1$. While we are able to show that Y increases by a large enough rate in expectancy it does not imply anything about how frequently it increases. To prove that it frequently rises by a significant margin we bound the standard deviation of $(Y_{i+1} - Y_i | Y_i)$ in Lemma 9. Following which we show that in $T\sqrt{\log k}$ phases the leader group will either have $\frac{Y_i}{4g_i} \geq 1$ or $Y_i \geq k/2$.

After $\frac{Y_i}{4g_i} \geq 1$, we can show that in a constant number of rounds we will have $Y_i > k/2$ with constant probability. We use Markov's inequality to show that $k - Y$ shrinks by a constant fraction in each phase with constant probability. □

Conclusion. Finally, we have a dominant group of size $> k/2$. We can see that it can scan the entire graph of size n in $\lceil \frac{2n}{k} \rceil$ rounds and get all agents to come to its location. Therefore we conclude that each iteration accomplishes gathering with constant probability. Since there are no Byzantine agents we can also argue that the agents will know when they have gathered as they can simply count the number of agents in its node to affirm that gathering has been completed. Thus,

Theorem 2.1. *Given* k *agents placed arbitrarily on a complete graph* G *of* n *nodes, there exists a randomized gathering protocol that can gather all agents in* $O(\frac{n}{k}\sqrt{\log k})$ *rounds.*

3 WEAKLY-ADAPTIVE ADVERSARY

In this section, we present an algorithm that is resilient against weakly-adaptive Byzantine agents that can tolerate up to any $f < k$ Byzantine agents. This problem is particularly challenging in regular graphs as it's difficult to tell nodes apart from each other as the agents cannot place any markers behind for other agents. Simple methods such as choosing a leader and the leader agent gathering all non-leader agents will fail because the Byzantine agents can pretend to be leaders and distract non-leader agents away from the leader agent.

Recall that agents can start at arbitrary nodes of the graph. If a node is occupied in the beginning, the occupant agents form a group. We present a protocol by which, over time, the groups can merge into larger groups until there is one group with all good agents. We maintain the invariant that once two agents become part of a group, they will continue to be a part of the same group. Note however that the agents in a group can disperse briefly and then recombine.

The algorithm (Algorithm 3) works iteratively as follows. In each iteration, each group A (currently at a vertex v) chooses to be a seeker or a settler uniformly at random. If A is a seeker then it traverses to a random node. Otherwise, A is a settler and each agent $a \in A$ disperses to a random node u_a and, upon reaching u_a, broadcasts the port number p at u_a to return to v. Subsequently a returns to v and is reunited with other members of A. If A is a seeker group and it encounters at least one or more settler agents, it will (as a group) choose one such settler agent s (chosen randomly and belonging to a settler group B) and follow s as it reunites with other members of B; A will then merge into B. We defer the analysis to Appendix Section B.

Theorem 3.1. *Given* k *agents placed arbitrarily on a graph* G *of* n *nodes, there exists a randomized gathering protocol that is resilient to*

[3] $f = k - 1$ achieves instant gathering since there is only one good agent

Algorithm 3 Complete Graph Gathering - Weakly-Adaptive Adversary

All agents form groups with other co-located agents.
repeat $C \cdot \lceil 24 \cdot \frac{n^2}{(k-f)(n-f)} \cdot \log n \rceil$ **times**
    Call MERGEITERATION()
end repeat ▷ By Theorem 3.1, good agents gather whp.

a weakly-adaptive Byzantine adversary that can whp gather all good agents in $C \cdot \lceil 24 \cdot \frac{n^2}{(k-f)(n-f)} \cdot \log n \rceil$ *rounds as long as the number of Byzantine agents* f *is fewer than* k.

Note. In the analysis of the algorithm, we have assumed that f and $k - f$ are less than n. This assumption is not limiting. In case $k - f \geq n$, at the start of the algorithm we can have all good agents go to random nodes and form super-agents with other agents co-located in the same node. If we execute this maneuver and run Algorithm 3 on the super-agents the number of good agents now becomes less than n. However, the number of Byzantine agents remains as f. But we see that when $f \geq n$, the Byzantine agents can potentially position themselves on every node in the graph and thus spread a lot of misinformation making the problem challenging. Similar techniques for larger values of f and k are also applicable for other algorithms discussed subsequently.

4 STRONGLY-ADAPTIVE ADVERSARY

We move on to the strongly-adaptive adversary. Therefore we consider the case where in any round, the Byzantine agents can see exactly where all agents are heading in the next round and can then make their moves accordingly.

Why Algorithm 3 won't work: Suppose at some stage there are only two groups. Furthermore, let's consider the optimistic case where one group is a seeker and another is a settler. When a seeker group tries to find a settler agent, the Byzantine agents can overwhelm the seeker group by going to the same random node and posing as f different settler agents. Therefore, just for two groups to gather the previous algorithm will take at least $\Omega(f)$ time. To counter this we require both seekers and settlers to split. This will increase the number of meetings between seekers and settlers in a given iteration. Furthermore, we also make the seeker agents monitor a settler agent it finds for $\lceil \frac{n}{k} \rceil$ time. After a seeker agent successfully *meets* a settler agent it returns with a proposal path and a path is randomly chosen from these proposed paths by broadcasting all proposed paths, storing them in a sorted list(ls), and choosing one of those paths uniformly at random. The gathering protocol is described in detail in Algorithms 4 and 5.

Unlike the previous algorithm, it is difficult to gather all agents (even if there are just two groups) primarily because all f Byzantine agents can pose as good seekers of the same group to a good seeker agent. Therefore, $\tilde{O}(f)$ time may be required to meet any other group as there will be $f + 1$ entries in ls, thereby isolating some good agents for $\Omega(f)$ rounds. However, we show that the Byzantine agents cannot isolate more than a fraction of the agents whp, thereby implying that a large group of good agents must form. After such a sufficiently large group is formed, it performs the search and rescue method (cf. Algorithm 5) where it gathers all agents in

Algorithm 4 Complete Graph Gathering - Strongly-Adaptive Adversary : Executed by agent a

repeat $5c \cdot \log n \cdot \log k$ **times** ▷ c is a constant that determines the success probability
    coin ← NODAL_RAND(2)
    All agents traverse e ← NODAL_RAND(n)
    edge_list ← [e]
    repeat $2 \cdot \lceil \frac{n}{k} \rceil$ **times**
        if coin = 0 **then** ▷ Agent a is part of seeker group
            if settler agent found and stays for $\lceil \frac{n}{k} \rceil$ rounds **then**
                Ask settler for edge to traverse(e′)
                ▷ If there are multiple choose arbitrarily
            else
                Agent a goes to random node through e
                Agent a appends e to edge_list
            end if
        else if coin = 1 and finds seeker **then**
            tells the seeker the edge to traverse(e)
        end if
    end repeat
    Return to starting node by reverse traversing edge_list
    if coin = 0 **then** ▷ seeker
        Seeker agents broadcast path edge_list + e′ ▷ If found
        Seeker agents read all broadcasted paths
        Seeker agents store paths in a sorted list ls
        Seeker agents traverse a random path from ls
    end if
end repeat
searchAndRescue() ▷ Algorithm 5

$O\left(\left(\frac{n}{k} + \log k\right) \log n\right)$ time. The search and rescue is also somewhat non-trivial and is divided into two parts. In the first part, the large group randomly searches the graph to find out agents that are not a part of the large group. In the second part, the large group splits itself into sub-groups and goes to the remaining agents and brings them back to the large group's location while doubling the size of the sub-groups in each iteration to thwart the Byzantine agents from pooling their resources to target a few agents. The detailed algorithm (Algorithm 5) can be found in the Appendix Section C.1.

Analysis. We focus on the seeker's probability of meeting an agent. At any round if there are h groups then let the groups be numbered $c_1, \ldots c_h$. For any seeker group c_i let us consider the following variables: (1) g_i: The number of good agents in the group (2) b_i: The number of Byzantine agents in the group (3) b'_i: The number of good agents that encountered Byzantine agents that posed as settlers.

We now bound the probability that a group c_i merges with a settler. We show that

Lemma 4. *For any group* c_i*:*

$$\Pr(c_i \text{ merges with a settler} \mid c_i \text{ is a seeker}) \geq \left(1 - e^{\frac{-|Settlers|}{k}}\right) \cdot \frac{1 - f'_i}{1 + f_i}$$

where $f'_i := b'_i/g_i, f_i := b_i/g_i$ *and* $|Settlers|$ *denotes the number of good settler agents in a round.*

Proof Sketch. The detailed proof of Lemma 4 can be found in the Appendix C.2. We compute the probability by finding the number of entries in ls that will lead to a group merge and the number of maximum entries. It is easy to see that the total number of entries is upper-bounded by $g_i + b_i$. Meanwhile, we are able to show that the number of entries are at most $p \cdot g_i - b'_i$ in expectancy, where $p = \left(1 - e^{\frac{-|Settlers|}{k}}\right)$. We show this by analyzing case-by-case as to what may happen to a seeker agent: it may be stopped from exploring for a sufficient time, a Byzantine agent may pretend to be a settler, or it actually explored sufficiently. □

From this point onward, we consider phases of $5c \log n$ iterations. Formally, we show that:

Lemma 5. *When the Byzantine agents are limited to $O\left(\frac{k}{\log^{1+\epsilon} k}\right)$ for any fixed $\epsilon > 0$, in each phase there can be at most $k/(3 \log k)$ agents that did not merge with a group (abandoned agents).*

Proof Sketch. The detailed proof of Lemma 4 can be found in the Appendix C.2. We show that in a phase, for a group to be stopped from merging with another group whp, it either needs to have a high $\sum f_i$ or $\sum f'_i$ in that phase. However, the sums of f_i and f'_i are bounded when we sum them over all good agents. We prove this using Lemma 10. We are able to leverage the fact that since there are few Byzantine agents, they can only fool $O(k/\log k)$ good seeker agents in an iteration(i.e. only $O(\log^{\epsilon} k)$ agents fooled per Byzantine agent). Therefore, we conclude that at most $k/(3 \log k)$ agents can be stopped from merging with another group. □

The non-abandoned groups would have merged with a group in all phases. Therefore, the number of non-abandoned groups would have gone down by at least half in any phase. We can also conclude that in $\log k$ such phases, there will at most be $k/3$ agents that have not merged with a group in some phase because of Lemma 5. Therefore, the non-abandoned agents would have formed a large group of size $\geq 2k/3 - f \geq (k+f)/2$. This threshold of $(k+f)/2$ is necessary because now every good agent in the large group will definitely know it is a part of a group that has more than half of the total good agents. Furthermore, every agent not in the large group will know that it not a part of the large group. Following this we simply use the searchAndRescue method to gather all agents to the large node of size $\geq k/2$ in $O\left(\left(\frac{n}{k} + \log k\right) \log n\right)$ rounds. Formally, we show that

Lemma 6. *At the i^{th} iteration of $O\left(\frac{n}{k} + \log k\right)$ rounds of the* searchAndRescue *method the expected number of abandoned agents(Z_i) follow:*

$$\mathbb{E}[Z_i] \leq 0.8 \cdot \mathbb{E}[Z_{i-1}]$$

Considering $i^* = C'' \log n$ and applying Markov's inequality, we can see that $\Pr(Z_{i^*} \geq 1) \leq \frac{1}{n^{C''-1}}$. Thus we conclude that:

Theorem 4.1. *Given k agents placed arbitrarily on a graph G of n nodes, there exists a randomized gathering protocol that is resilient to a strongly-adaptive Byzantine adversary that can whp gather all good agents in $O\left(\frac{n}{k} \log n \log k\right)$ rounds as long as the number of Byzantine agents $f \in O\left(\frac{k}{\log^{1+\epsilon} k}\right)$ for any fixed $\epsilon > 0$.*

4.1 Extension to general graphs

We extend our algorithms for the complete graphs to general (connected and undirected) graphs. We note that we cannot use our complete graph algorithm against a weakly-adaptive adversary. This is because when an agent moves to a random location in the complete graph the Byzantine agents did not have a better alternative than to randomly guess. However, for a graph $G = (V, E)$ with lower degrees, the Byzantine adversary can accurately predict the location of an agent to one of $(\delta_u + 1)$ nodes. However, we can use the algorithms we developed for the strongly-adaptive adversary. We note that Algorithm 4 can directly be applied to a general graph if we can go to a random node in bounded time. We assume that all agents know an upper bound Δ on the maximum-degree of G. We can then virtually add $\Delta - \delta_v$ self-loops for all nodes $v \in V$. This will facilitate lazy random walks on G with a stationary distribution of $1/n$ for all $v \in V$. We assume that the agents know the mixing time τ of this lazy random walk. Therefore we can conclude that gathering can be achieved in $O\left(\tau \cdot \frac{n}{k} \log n \log k\right)$ time. For the case when there are no Byzantine nodes, we can gather faster. We can naturally extend Algorithm 2 for when there are no Byzantine agents in the complete graph to get an algorithm for gathering on G that runs in $O\left(\tau \cdot \frac{n}{k} \sqrt{\log k}\right)$ expected time.

5 NEGATIVE RESULTS

As we mentioned before, despite its importance, there has been very little work on gathering anonymous agents in graphs. We now formally present our negative results that shed light on this paucity of work in this area. Specifically, we begin by showing that deterministic gathering is impossible. Subsequently, we show a lower bound on randomized gathering that essentially implies that all our algorithms are optimal to within polylog$(n+k)$ factors.

5.1 Impossibility of Deterministic Algorithms

The algorithms presented in this work are all randomized for good reason. We now show that there exists a port-numbering such that it is impossible for any deterministic algorithm to gather for any $2 \leq k \leq n$ agents in a complete graph; we extend it subsequently to regular graphs of arbitrary degree. Formally, we show that:

Theorem 5.1. *For any complete graph, there exists a port-numbering, such that no deterministic algorithm can gather any $2 \leq k \leq n$ agents that are initially located on different vertices of the graph.*

Before we prove Theorem 5.1, we provide a useful lemma (Lemma 7). The proof can be found in the Appendix Section D. Given a graph, a port numbering takes each edge (u, v) and assigns a unique port number from 0 to $\delta_u - 1$ (where δ_u is the degree of u) at u and a unique port number from 0 to $\delta_v - 1$ at v for the edge (u, v), i.e., no two distinct edges incident to the same vertex u can share the same port number at u.

Lemma 7. *For any graph G that is the union of c mutually edge-wise disjoint Hamiltonian cycles and at most one perfect-matching, there exists a port-numbering, such that there is no deterministic algorithm that can gather any $2 \leq k \leq n$ agents that are located on different vertices of the graph.*

Proof of Theorem 5.1. There are two cases to consider:
(1) n is odd: We consider G as the union of $(n-1)/2$ mutually edge-wise disjoint Hamiltonian cycles. We know such a decomposition exists because of Theorem 1.2[4] in Alspach and Gavlas [4].
(2) n is even: In this case, we assign the port numbers in P^* by considering $(n-2)/2$ Hamiltonian cycles of length n and 1 perfect matching. We know that such a decomposition exists because of Theorem 1.1[5] in Alspach and Gavlas [4].

Since G is now the union of mutually edge-wise disjoint cycles and at most one matching, we can use Lemma 7 to conclude that no deterministic algorithm can successfully gather the k agents. □

The same idea can be extended to r-regular graphs as well.

Theorem 5.2. *For any $r \in [2, n-1]$ (where $n \cdot r \equiv 0 \pmod 2$), there exists an r-regular graph and a port-numbering for this r-regular graph, such that there is no deterministic algorithm that can gather k agents, $2 \leq k \leq n$, that are located on different vertices of the graph.*

Proof. We again provide a constructive proof that leverages Lemma 7. We consider the construction of the complete graph G described in the proof of Theorem 5.1. We again consider two cases:

(1) n **is odd.** We can see that when n is odd, r must be even. Let the cycle decomposition of K_n be $C = [C_1, \ldots C_{(n-1)/2}]$. Since the K_n has degree $n-1$, we can simply arbitrarily delete the edges of any of the $(n-1-r)/2$ cycles in C. This will ensure that the graph is connected and is made out of mutually edge-wise disjoint cycles
(2) n **is even.** Let the cycle decomposition of $K_n - I$ be $C = [C_1, \ldots C_{(n-2)/2}]$, where I is the perfect matching. We consider two subcases here:
   (a) r is odd. We arbitrarily delete the edges of any of the $(n-1-r)/2$ cycles in C.
   (b) r is even. We delete the edges of any of the $(n-2-r)/2$ cycles in C and delete I.

We once again see that the resulting graph is a union of mutually edge-wise disjoint cycles and at most one perfect matching. We can again apply lemma 7 to see that it is impossible for k agents to gather under a deterministic algorithm. □

5.2 Lower bound for Complete Graphs

We now show that $\Omega(n/(k-f))$ is a lower bound for gathering k agents in a complete graph of n nodes even when the f Byzantine agents, $0 \leq f < k-1$, just crash at the beginning. Formally:

Theorem 5.3. *Consider a complete graph G on n vertices on which k agents are placed arbitrarily with $0 \leq f \leq k-2$ agents being faulty (i.e., they just don't move or communicate). Consider any protocol $\Pi(n, k, f)$ that runs for $r^* = \lfloor n/(2k-2f) \rfloor$ rounds and is designed for gathering the $k-f$ good agents. With probability at least 1/2, Π will fail to gather the $k-f$ good agents even when $k-f-1$ agents are centrally controlled by a single entity C that can coordinate their movements.*

[4]Theorem 1.2 in [4] states that for positive odd integers m and n with $3 \leq m \leq n$, the graph K_n can be decomposed into cycles of length m if and only if the number of edges in K_n is a multiple of m.
[5]Theorem 1.1 in [4] states that for positive even integers m and n with $4 \leq m \leq n$, the graph $K_n - I$ can be decomposed into cycles of length m if and only if the number of edges in $K_n - I$ is a multiple of m. See [5] for a minor correction.

Proof. Let $k' = k - f$ denote the number of non-faulty agents. Consider a complete graph on n vertices with port numbering P^* as described in the proof of Lemma 7 based on the mutually edge-wise disjoint cycle decomposition with at most one matching described in the proof of Theorem 5.1. C controls $k'-1$ agents, which leaves us with one non-faulty agent ℓ that is not controlled by C; we call ℓ the lost agent. For this lower bound, ℓ is initially placed in u^* chosen uniformly at random; note that C is unaware of u^*. The f faulty agents can be colocated in a node chosen uniformly at random; they will neither help nor deter the efforts of C.

Since the lost agent also executes Π, which is known to C, we assume that C knows the sequence S of port numbers taken by ℓ. Note that Π may be randomized with decisions based on random bits generated by ℓ. By revealing S to C, we have essentially revealed those private random bits, which is acceptable in the context of a lower bound proof. The position of ℓ in each round r is a function $p_r(u^*)$ that maps each possible u^* to the set of vertices in G.

We claim that for a fixed $r \geq 0$ and fixed S, p_r is a bijection on the vertices of G. Clearly, the bijection holds when $r = 0$. To complete the inductive proof, we assume that the bijection has held up to round $r-1$ (for $r \geq 1$). At round r, the agent can either (i) take a step in one of the cycles and in one of the two directions, or (ii) it can take a step along the matching (if it exists). Regardless, the bijection continues to hold.

The above claim implies that the position of ℓ at round r can be deduced from u^* and vice versa and this bijection is known to C. Thus, the task at hand for C can be viewed as finding the exact starting point u^* of ℓ. There are n choices for u^* and in $\lfloor n/2k' \rfloor$ rounds, C can eliminate fewer than $n/2$ of the n choices. Since u^* is chosen uniformly at random, the correct starting node will be missed with probability at least a 1/2. □

6 CONCLUSION

We studied the role played by randomization in foiling the adversary to give fast gathering algorithms that are resilient to harsh forms of failures. Mirroring research in other fundamental global symmetry breaking problems [26, 34], our primary focus was on complete graphs. Our algorithms and analysis techniques are quite non-trivial and indicate to us the need for developing this theory further. Our work raises many questions. For example, we believe that the canonical and elementary setting on complete graphs with $k = n$ and $f = 0$ is of fundamental interest. Our algorithm takes $O(\sqrt{\log n})$ rounds on expectation. Can this bound be optimized further? Can we get sub-logarithmic bounds with (perhaps weaker forms of) Byzantine adversaries or crash failures?

Another important question is whether we can tolerate larger bounds on f against strongly-adaptive adversaries? Or, is $f \in O(k/\log k)$ a fundamental limit? Furthermore, when we consider values of $k \gg n$, even a small f (as a function of k) can be quite challenging because the Byzantine agents can essentially spread throughout the graph. While our work deals with anonymous agents, it is conceivable that there is interest in non-anonymous versions as well, which is in fact, the subject of study in earlier works like [16]. The natural question is whether randomization can provide us with more efficient bounds.

REFERENCES

[1] Noa Agmon and David Peleg. 2006. Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. *SIAM J. Comput.* 36, 1 (July 2006), 56–82. https://doi.org/10.1137/050645221

[2] David Aldous and James Allen Fill. 2002. Reversible Markov Chains and Random Walks on Graphs. Unpublished monograph, recompiled 2014, available at http://www.stat.berkeley.edu/~aldous/RWG/book.html.

[3] Steve Alpern and Shmuel Gal. 2006. *The theory of search games and rendezvous*. Vol. 55. Springer Science & Business Media.

[4] Brian Alspach and Heather Gavlas. 2001. Cycle Decompositions of K_n and $K_n - I$. *Journal of Combinatorial Theory, Series B* 81, 1 (2001), 77–99. https://doi.org/10.1006/jctb.2000.1996

[5] Brian Alspach and Heather Jordon. 2021. Corrigendum to "Cycle decompositions of K_n and $K_n - I$" [J. Combin. Theory Ser. B 81 (1) (2001) 77—99]. *Journal of Combinatorial Theory, Series B* 146 (2021), 532–533. https://doi.org/10.1016/j.jctb.2020.09.002

[6] James Aspnes. 1998. Lower bounds for distributed coin-flipping and randomized consensus. *Journal of the ACM (JACM)* 45, 3 (1998), 415–450.

[7] Michael Ben-Or and Nathan Linial. 1985. Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *26th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 408–416.

[8] Sébastien Bouchard, Yoann Dieudonné, and Bertrand Ducourthial. 2016. Byzantine Gathering in Networks. *Distrib. Comput.* 29, 6 (Nov. 2016), 435–457. https://doi.org/10.1007/s00446-016-0276-9

[9] Sébastien Bouchard, Yoann Dieudonné, and Anissa Lamani. 2018. Byzantine Gathering in Polynomial Time. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 107)*, Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 147:1–147:15. https://doi.org/10.4230/LIPIcs.ICALP.2018.147

[10] Sébastien Bouchard, Yoann Dieudonné, and Andrzej Pelc. 2020. Want to Gather? No Need to Chatter!. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 253–262. https://doi.org/10.1145/3382734.3405693

[11] Colin Cooper, Robert Elsässer, Hirotaka Ono, and Tomasz Radzik. 2012. Coalescing Random Walks and Voting on Graphs. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing* (Madeira, Portugal) *(PODC '12)*. Association for Computing Machinery, New York, NY, USA, 47–56. https://doi.org/10.1145/2332432.2332440

[12] Jurek Czyzowicz, Adrian Kosowski, and Andrzej Pelc. 2012. How to meet when you forget: log-space rendezvous in arbitrary graphs. *Distributed Computing* 25, 2 (2012), 165–178.

[13] Shantanu Das et al. 2013. Mobile agents in distributed computing: Network exploration. *Bulletin of EATCS* 1, 109 (2013).

[14] Xavier Défago, Maria Gradinariu, Stéphane Messika, and Philippe Raipin-Parvédy. 2006. Fault-Tolerant and Self-stabilizing Mobile Robots Gathering. In *Distributed Computing*, Shlomi Dolev (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 46–60.

[15] Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin Parvédy. 2020. Self-stabilizing gathering of mobile robots under crash or Byzantine faults. *Distributed Comput.* 33, 5 (2020), 393–421. https://doi.org/10.1007/s00446-019-00359-x

[16] Yoann Dieudonné, Andrzej Pelc, and David Peleg. 2014. Gathering Despite Mischief. *ACM Trans. Algorithms* 11, 1, Article 1 (Aug. 2014), 28 pages. https://doi.org/10.1145/2629656

[17] Yoann Dieudonné and Andrzej Pelc. [n.d.]. *Anonymous Meeting in Networks*. 737–747. https://doi.org/10.1137/1.9781611973105.53 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611973105.53

[18] Gianlorenzo D'Angelo, Gabriele Di Stefano, and Alfredo Navarra. 2014. Gathering on rings under the look–compute–move model. *Distributed Computing* 27, 4 (2014), 255–285.

[19] R. Eguchi, N. Kitamura, and T. Izumi. 2020. Fast Neighborhood Rendezvous. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE Computer Society, Los Alamitos, CA, USA, 168–178. https://doi.org/10.1109/ICDCS47774.2020.00030

[20] Uriel Feige. 1999. Noncryptographic selection protocols. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 142–152.

[21] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. 2010. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theoretical Computer Science* 411, 14-15 (2010), 1583–1598.

[22] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. 2019. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*. Lecture Notes in Computer Science, Vol. 11340. Springer. https://doi.org/10.1007/978-3-030-11072-7

[23] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. 2017. Distributed computing by mobile robots: uniform circle formation. *Distributed Comput.* 30, 6 (2017), 413–457. https://doi.org/10.1007/s00446-016-0291-x

[24] Pierre Fraigniaud and Andrzej Pelc. 2013. Delays induce an exponential memory gap for rendezvous in trees. *ACM Transactions on Algorithms (TALG)* 9, 2 (2013), 1–24.

[25] Jion Hirose, Junya Nakamura, Fukuhito Ooshita, and Michiko Inoue. 2021. Gathering with a Strong Team in Weakly Byzantine Environments. In *International Conference on Distributed Computing and Networking 2021* (Nara, Japan) *(ICDCN '21)*. Association for Computing Machinery, New York, NY, USA, 26–35. https://doi.org/10.1145/3427796.3427799

[26] Daniel S. Hirschberg and James B Sinclair. 1980. Decentralized extrema-finding in circular configurations of processors. *Commun. ACM* 23, 11 (1980), 627–628.

[27] Zool Hilmi Ismail and Nohaidda Sariff. 2019. A Survey and Analysis of Cooperative Multi-Agent Robot Systems: Challenges and Directions. In *Applications of Mobile Robots*, Efren Gorrostieta Hurtado (Ed.). IntechOpen, Rijeka, Chapter 1. https://doi.org/10.5772/intechopen.79337

[28] Taisuke Izumi, Tomoko Izumi, Sayaka Kamei, and Fukuhito Ooshita. 2013. Feasibility of Polynomial-Time Randomized Gathering for Oblivious Mobile Robots. *IEEE Transactions on Parallel and Distributed Systems* 24, 4 (2013), 716–723. https://doi.org/10.1109/TPDS.2012.212

[29] Gangshan Jing, Yuanshi Zheng, and Long Wang. 2016. Consensus of Multiagent Systems With Distance-Dependent Communication Networks. *IEEE Transactions on Neural Networks and Learning Systems* PP (08 2016). https://doi.org/10.1109/TNNLS.2016.2598355

[30] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. 2010. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science* 411, 34-36 (2010), 3235–3246.

[31] Ralf Klasing, Euripides Markou, and Andrzej Pelc. 2008. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science* 390, 1 (2008), 27–39.

[32] Evangelos Kranakis and Danny Krizanc. 2016. *Mobile Agents and Exploration*. Springer New York, New York, NY, 1338–1341. https://doi.org/10.1007/978-1-4939-2864-4_242

[33] Evangelos Kranakis, Danny Krizanc, and Pat Morin. 2011. Randomized rendezvous with limited memory. *ACM Transactions on Algorithms (TALG)* 7, 3 (2011), 1–12.

[34] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (jul 1982), 382–401. https://doi.org/10.1145/357172.357176

[35] Silvio Micali and Tal Rabin. 1990. Collective coin tossing without assumptions nor broadcasting. In *Conference on the Theory and Application of Cryptography*. Springer, 253–266.

[36] Avery Miller and Ullash Saha. 2020. Fast Byzantine Gathering with Visibility in Graphs. In *Algorithms for Sensor Systems: 16th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2020, Pisa, Italy, September 9–10, 2020, Revised Selected Papers* (Pisa, Italy). Springer-Verlag, Berlin, Heidelberg, 140–153. https://doi.org/10.1007/978-3-030-62401-9_10

[37] Roger B. Nelsen. 2020. Titu's Lemma. *Mathematics Magazine* 93, 1 (2020), 70–70. https://doi.org/10.1080/0025570X.2020.1682745 arXiv:https://doi.org/10.1080/0025570X.2020.1682745

[38] Andrzej Pelc. 2019. Deterministic Rendezvous Algorithms. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro (Eds.). Lecture Notes in Computer Science, Vol. 11340. Springer, 423–454. https://doi.org/10.1007/978-3-030-11072-7_17

[39] Michael Saks. 1989. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics* 2, 2 (1989), 240–244.

[40] Masahiro Shibata, Toshiya Mega, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. 2016. Uniform Deployment of Mobile Agents in Asynchronous Rings. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing* (Chicago, Illinois, USA) *(PODC '16)*. Association for Computing Machinery, New York, NY, USA, 415–424. https://doi.org/10.1145/2933057.2933093

[41] Ichiro Suzuki and Masafumi Yamashita. 1999. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM J. Comput.* 28, 4 (1999), 1347–1363. https://doi.org/10.1137/S009753979628292X arXiv:https://doi.org/10.1137/S009753979628292X

[42] Masashi Tsuchida, Fukuhito Ooshita, and Michiko Inoue. 2017. Byzantine Gathering in Networks with Authenticated Whiteboards. In *WALCOM: Algorithms and Computation*, Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen (Eds.). Springer International Publishing, Cham, 106–118.

[43] Xiaoping Yun, Gokhan Alptekin, and Okay Albayrak. 1997. Line and circle formation of distributed physical mobile robots. *J. Field Robotics* 14, 2 (1997), 63–76.