




Article

Asynchronous Gathering in a Dangerous Ring [†]

Stefan Dobrev ^{1,‡}, Paola Flocchini ^{2,‡} , Giuseppe Prencipe ^{3,‡}  and Nicola Santoro ^{4,*,‡} ¹ Informatics Department, Slovak Academy of Science, 841 04 Bratislava, Slovakia² School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada³ Dipartimento di Informatica, Università di Pisa, 56127 Pisa, Italy⁴ School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada

* Correspondence: santoro@scs.carleton.ca

[†] This paper is an extended version of our paper published in the Proceedings of the 7th International Conference on Principles of Distributed Systems (OPODIS), La Martinique, France, 10–13 December 2003.[‡] These authors contributed equally to this work.

Abstract: Consider a set of k identical asynchronous mobile agents located in an anonymous ring of n nodes. The classical GATHER (or RENDEZVOUS) problem requires all agents to meet at the same node, not a priori decided, within a finite amount of time. This problem has been studied assuming that the network is safe for the agents. In this paper, we consider the presence in the ring of a stationary process located at a node that disables any incoming agent without leaving any trace. Such a dangerous node is known in the literature as a black hole, and the determination of its location has been extensively investigated. The presence of the black hole makes it deterministically unfeasible for all agents to gather. So, the research concern is to determine how many agents can gather and under what conditions. In this paper we establish a complete characterization of the conditions under which the problem can be solved. In particular, we determine the maximum number of agents that can be guaranteed to gather in the same location depending on whether k or n is unknown (at least one must be known). These results are tight: in each case, gathering with one more agent is deterministically unfeasible. All our possibility proofs are constructive: we provide mobile agent algorithms that allow the agents to gather within a predefined distance under the specified conditions. The analysis of the time costs of these algorithms show that they are optimal. Our gathering algorithm for the case of unknown k is also a solution for the black hole location problem. Interestingly, its bounded time complexity is $\Theta(n)$; this is a significant improvement over the existing $O(n \log n)$ bounded time complexity.

Keywords: mobile agents; rendezvous; gathering; black hole; harmful host; ring network; asynchronous; anonymous; distributed computing



Citation: Dobrev, S.; Flocchini, P.; Prencipe, G.; Santoro, N. Asynchronous Gathering in a Dangerous Ring. *Algorithms* **2023**, *16*, 222. <https://doi.org/10.3390/a16050222>

Academic Editors: Charalampos Konstantopoulos, Grammati Pantziou and Frank Werner

Received: 14 February 2023

Revised: 7 April 2023

Accepted: 21 April 2023

Published: 26 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Premise

In networked systems that support autonomous mobile agents, a main concern is how to develop efficient agent-based *system algorithms*; that is, to design algorithms that will allow a team of rather “simple” agents to cooperatively perform complex system tasks. A main approach to reach this goal is to break a complex task down into more elementary operations. Example of these primitive operations are *traversal*, *search*, *gathering*, and *election*. The coordination of the agents necessary to perform these operations is not necessarily simple or easy to achieve [1].

In fact, the computational problems related to these operations are definitely non-trivial, and a great deal of theoretical research is devoted to the study of the conditions for the solvability of these problems and to the discovery of efficient algorithmic solutions. At an abstract level, these environments, which we shall call *distributed mobile systems*, can be described as a collection \mathcal{E} of mobile computational entities, called *agents*, operating

in a graph G . The agents have local computing capabilities and bounded storage, have limited means of communication, execute the same algorithm, and can move from node to neighboring node. The research concern is on determining what problems can be solved by such entities, under what conditions, and at what cost [2].

We consider the computationally difficult environment where the agents are *anonymous*, in the sense that they have no identifiers, and they are *asynchronous*, in the sense that every action they perform takes a finite, but otherwise unpredictable, amount of time (decided by a worst-case adversary). Each node of the network, also called a *host*, provides a storage area called a *whiteboard* for incoming agents to communicate and compute; the whiteboards are accessed using fair mutual exclusion.

For such environments, we focus in this paper on a fundamental problem in distributed mobile computing, *gathering*, in the (simplest) fully symmetric topology, the *ring* network. We consider its solution in the presence of a severe security threat, a *black hole*: a network site where a harmful process destroys all incoming agents without leaving a trace. A black hole models a computer that is accidentally off-line or a network site in which a resident process (e.g., an unknowingly-installed virus) deletes any visiting agents or incoming data upon their arrival, without leaving any observable trace; similarly, any undetectable crash failure of a site in a network transforms that site into a black hole (e.g., see [3]).

1.2. Background

Gathering. The *gathering* problem consists in having the agents meet at the same node and enter a terminal state within a finite amount of time; there is no a priori restriction on which node will become the gathering point. The gathering problem has been extensively investigated under a large variety of settings and communication mechanisms. In anonymous and symmetric settings, like the one considered here, the possibility (and difficulty) of a solution is related to the possibility (and difficulty) to deterministically find or create an asymmetry in the system, enabling the agents to agree on a meeting point.

The majority of investigations have focused on the case of *two* agents, very few results are known for anonymous ring networks (see [4] for a recent survey). Almost all these investigations have assumed *synchronous* agents: the agents are synchronized and the amount of time required by an agent for processing and for movement along a link is one unit; this assumption is crucial for the correctness of their solutions.

In the *asynchronous* setting, interestingly, the link between gather and symmetry breaking is even more clear: gathering is in fact equivalent to the *election* problem [5]. Few papers have examined gathering in the asynchronous setting (see [6–8]) usually assuming two agents with distinct IDs.

Recall that, in our setting, both nodes and agents are *anonymous*, and the system is fully *asynchronous*.

Black Hole Search. Among the severe security threats faced in systems supporting mobile agents, a particularly troublesome one is a *harmful host*; that is, the presence at a network site of harmful stationary processes. The problem posed by the presence of a harmful host has been intensively studied from a programming point of view (e.g., see [9–11]), and also from an algorithmic prospective [12]. Obviously, the first step in any solution to such a problem must be to *identify*, if possible, the harmful host determining its location. Depending on the nature of the danger, the task to identify the harmful host might be difficult, if not impossible, to perform.

A particularly harmful host is a *black hole*: a host that disposes of visiting agents upon their arrival, leaving no observable trace of such a destruction. Note that this type of highly harmful host is not rare; for example, the undetectable crash failure of a site in an asynchronous network transforms that site into a black hole [3].

The *Black Hole Search* problem is to determine the location of the black hole; it is solved if, within a finite amount of time, at least one agent survives and knows the location of the black hole. This problem has been extensively investigated, both in synchronous

and asynchronous networks (e.g., [13–25]; see [26] for a recent survey). The case when the network is an *asynchronous ring* (the setting considered here) has been investigated (e.g., [13,27,28]); however, with the exception of [28], these investigations assume that all agents start from the same node (i.e., they are initially gathered). The best known bounded time complexity for this problem is $O(n \log n)$ [13].

Summarizing, in spite of the literature on both gathering and black hole search in asynchronous settings (see [4,26] for recent surveys on the two topics), no results exist on gathering in the presence of a black hole.

1.3. Contributions

In this paper, we consider the *gathering* problem in the following setting: k asynchronous anonymous agents dispersed in a totally symmetric ring network of n anonymous sites, one of which is a *black hole*. The presence of the black hole makes it deterministically unfeasible for all agents to gather. So, our research goal is to determine how many agents can gather and under what conditions.

More precisely, we study the *gathering* problem $RV(f(k))$, which requires at least $f(k) \leq k$ agents to gather and enter a terminal state in the same node, where f is an arbitrary monotonically non-decreasing function. We also investigate the related *close-gathering* problem $G(f(k), d)$, requiring at least $f(k)$ agents to enter a terminal state located within a distance d from each other. Clearly, $G(f(k), 0) = RV(f(k))$.

We establish a complete characterization of the conditions under which the two problems can be solved (proving the theoretical results previously claimed in [29]). In particular, we determine the maximum number of agents that can be guaranteed to gather in the same location depending on whether k or n is unknown (at least one must be known). All our possibility proofs are constructive: we provide mobile agent algorithms that allow the agents to gather within a predefined distance under the specified conditions. The possibility results are summarized in the table shown in Table 1. These results are *tight*: in each case, gathering with one more agent is impossible. The analysis of the time costs of these algorithms shows that they are *optimal*.

Table 1. Summary of possibility results.

	<i>n</i> Unknown, <i>k</i> Known		<i>n</i> Known, <i>k</i> Unknown	
ORIENTED		$RV(k - 1)$		$RV(k - 2)$
	k odd	$RV(k - 2)$	k odd or n even	$RV(k - 2)$
UNORIENTED	k even	$RV((k - 2)/2)$	k even and n odd	$RV((k - 2)/2)$
	$\forall k$	$G(k - 2, 1)$	$\forall k$	$G(k - 2, 1)$

Some of these results are unexpected. For example, in an oriented ring, all but one agents can indeed gather, even if the ring size n is not known, a condition that makes black hole location deterministically unfeasible [13].

In an unoriented ring, at most $k - 2$ agents can gather; surprisingly, if they cannot, there is no guarantee that more than $(k - 2)/2$ will. It is, however, always possible to bring all $k - 2$ agents within a distance of 1 from each other.

Our gathering algorithm for the case of unknown k is also a solution for the *black hole location* problem. Interestingly, its bounded time complexity is $\Theta(n)$; this is a significant improvement over the $O(n \log n)$ bounded time complexity, shown in [13], of the existing algorithms for the same case.

The paper is organised as follows: in Section 2, definitions and preliminary results are introduced; also, we present a few bounds for the problem. In Sections 3 and 4 we present algorithms that solve the problem, distinguishing two cases: k known and n unknown,

and k unknown and n known; in both cases, results are presented for both oriented and unoriented rings. Finally, we draw some final conclusions in Section 5.

2. Definitions, Basic Properties, and Techniques

2.1. Model

The network environment is a ring \mathcal{R} of n anonymous (i.e., identical) nodes. Each node has two labeled ports and a bounded amount of storage, called a *whiteboard*.

In this network there is a set $\mathcal{A} = \{a_1, \dots, a_k\}$ of k anonymous (i.e., identical) mobile agents. The agents can move from node to neighboring node in \mathcal{R} , have computing capabilities and bounded storage, obey the same set of behavioral rules (the “algorithm”), and all their actions (e.g., computation, movement, etc.) take a finite, but otherwise unpredictable, amount of time (i.e., they are *asynchronous*). Agents communicate by reading from and writing on the whiteboard in the node where they currently are; access to a whiteboard uses fair mutual exclusion. The agents execute an algorithm (the same for all agents) that specifies the computational and navigational steps.

Each agent has a local orientation that allows them to distinguish “left” from “right”. If the orientation is the same for all agents, we say that the ring is *oriented*, otherwise it is *unoriented*. Given an agent a on a node u of the ring, we will denote by $right_a(u)$ and $left_a(u)$ the port labels of u , according to the local orientation of agent a .

Initially, each agent is placed at a distinct node, called its *homebase*, and has a predefined state variable set to *available*. Let us denote by x_i the homebase of agent a_i . Each homebase is initially marked (in the whiteboard) as such by the corresponding agent. Agents can move from node to neighboring node; while crossing an edge, an agent can detect whether there is another agent crossing that edge in the opposite direction at the same time.

The agents are aware of the fact that, in the network there is a *black hole* (BH), a node that *disposes* of visiting agents upon their arrival, leaving *no observable trace* of such a destruction; its location is unknown. The initial location of the agents, the location of the black hole, the duration of each operation of the agents, and, in the case of unoriented rings, the sense of orientation of each agent, are assumed to be under the control of a worst-case adversary. In this environment, we are going to consider the *gathering* problem and the *close-gathering* problem defined below.

The *gathering* problem $RV(f(k))$ requires having at least $f(k) \leq k$ agents gathering in the same site, with f being an arbitrary monotonic and non-decreasing function. There is no a priori restriction on which node will become the gathering point. The problem is solved if at least $f(k)$ agents are at the same node in a terminal state, denoted by *arrived*.

The *close-gathering* problem $G(f(k), d)$ requires having at least $f(k)$ agents within distance d from each other. The problem is solved if at least $f(k)$ agents are within distance d from each other and are in a terminal state. Clearly, $G(f(k), 0) = RV(f(k))$.

The efficiency of a solution algorithm is first and foremost measured in the *size* of the solution, i.e., the number of agents that the algorithm will make gather at the same location in the case of *gathering*, and the number of agents that will be at distance d from each other in the case of *close-gathering*.

A secondary but important cost measure is the amount of *time* elapsed from the beginning to the termination of the algorithm. Since the agents are asynchronous, “real” time cannot be measured. We will use the traditional measure of *bounded time*, where it is assumed that the traversal of a link takes at most one time unit. Usually, the time is measured from the moment the first agent wakes up. However, in our setting, the first agent to wake up might immediately enter the black hole, and an indeterminate amount of time may pass before other agents wake up. In order to avoid this problem, and keep the algorithms and analysis simple, we measure the time complexity from the moment the last *extremal* agent wakes up, where the *extremal* agents are defined as follows:

- In the oriented case, there is a single extremal agent—the leftmost one, measured from the BH.

- In the unoriented case, there are two extremal agents—the leftmost agent within each group of agents with the same understanding of left and right (note that ‘leftmost’ is evaluated using the orientation of the group).

2.2. Cautious Walk

In the following we describe a basic tool, first introduced in [13], that we will use in all our algorithms to minimize the number of agents that disappear in the black hole.

In our algorithms, the ports (corresponding to incident links) of a node are classified as

1. *unexplored*—if no agent has moved across this port,
2. *safe*—if an agent arrived via this port, or
3. *active*—if an agent departed via this port, but no agent has arrived via it.

Clearly, both *unexplored* and *active* links are dangerous in that they might lead to the black hole; the difference is that *active* links are being traversed, so there is in general no need for another agent to go through that link until the link is declared *safe*.

The technique we use, called *cautious walk*, is defined by the following three rules:

Rule 1. When an agent moves from node u to v via an *unexplored* port, this marks the port as *active*; arriving at v , if there is no black hole there, it immediately returns to u , which marks the port as *safe*, and then continues the execution of the algorithm.

Rule 2. An agent never leaves via an *active* port.

Rule 3. An agent can always leave via a *safe* port.

In the following, the agents always move following the above rules.

2.3. Basic Results and Lower Bounds

There are some basic obvious facts:

Theorem 1. *In an anonymous ring with a black hole:*

1. $RV(k)$ is unsolvable;
2. If the ring is unoriented, then $RV(k - 1)$ is unsolvable.

Proof. We distinguish the two cases.

(1) It suffices for the adversary to choose as the location of the black hole the node where the first moving agent goes to: it disappears in the black hole, hence it will never be able to gather.

(2) By contradiction, let \mathcal{A} be a deterministic solution algorithm, let the ring be unoriented, and let a_i be the first moving agent. Let us assume, without loss of generality, that a_i (according to the algorithm) moves to its left. The adversary chooses as the location of the black hole the node to the left of a_i . Then, it places another agent a_j to the immediate left of the black hole, and assigns to a_j a sense of orientation opposite to the one of a_i . Moreover, the adversary does not activate any other agents until both a_i and a_j make their move. This implies that, since \mathcal{A} is deterministic, a_j will choose right as its moving direction. Therefore, both a_i and a_j will eventually disappear into the black hole, a contradiction. \square

Less obvious are the following facts.

Theorem 2. *Let k be unknown. $RV(k - 1)$ is unsolvable even if n is known and the ring is oriented.*

Proof. By contradiction, let \mathcal{A} be an algorithm that solves $RV(k - 1)$ for any $k \geq 2$. Consider an execution of \mathcal{A} where, initially, all agents are activated at the same time. Observe that, in this case, the agents must all move in the same direction; otherwise, if the adversary places the black hole at a distance of one from two agents that move towards each other, then both agents will enter the black hole, making $RV(k - 1)$ impossible. Thus, in this execution of \mathcal{A} all agents initially must move in the same direction, e.g., right.

Let the agents be logically grouped in pairs (h_i, t_i) ($1 \leq i \leq \lfloor k/2 \rfloor$), where h_i and t_i are called the head and the tail of the i -th pair; if k is odd, the remaining agent forms group (h_0) without a tail.

Let the adversary choose the initial distinct locations of the agents as follows: each pair (h_i, t_i) is positioned on two neighboring nodes, x_i and y_i , with the head to the right of the tail; the single group (h_0) (if it exists) is positioned on an unoccupied node x_0 ; the selection of the nodes is made so that, if $k > 3$ at least two pairs have an empty node between them (see Figure 1).

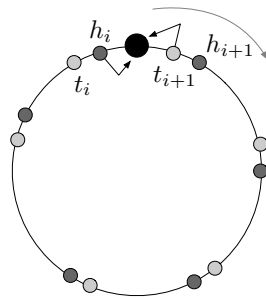


Figure 1. Example used in the proof of Theorem 2.

Once all agents are simultaneously activated in this initial configuration and start moving right, the adversary slows down the movement of the heads, and makes each tail t_i reach its destination x_i simultaneously.

Consider now what \mathcal{A} does at this point if $k = 2$, i.e., there is only one pair (h_1, t_1) . Let us focus on t_1 , which is currently on x_1 . Since, by assumption \mathcal{A} is correct, t_1 must use cautious walk, thus it cannot move to the node z , to the right of x_1 (otherwise the adversary would choose z as the location of the black hole and both agents would disappear); if h_1 has entered the black hole, within a finite amount of time t_1 has to enter a terminal state. How this is achieved depends on the specifics of \mathcal{A} ; there are two possible cases:

Case 1: t_1 becomes *arrived* after a finite amount of time Δ_1 from the start without ever reaching other nodes beside x_1 and y_1 . Note that this can happen even if h_1 has not entered the black hole, e.g., if the adversary has made the delay of its initial movement, and return to x_1 , at least Δ_1 .

In this case, however, the adversary can make the algorithm fail for $k > 3$ by having all tails experiencing the same delays as t_1 , and delaying all heads Δ_1 time units. In this way, all tails enter the terminal state after Δ_1 time, making $RV(k - 1)$ impossible, a contradiction.

Case 2: t_1 reaches other nodes besides x_1 and y_1 before becoming *arrived*. Since it cannot move to the right of x_1 , these nodes are on the left of y_1 . Let w be the node immediately to the left of y_1 , and let t_1 reach it for the first time Δ_2 time units after reaching x_1 the first time. Note that this can happen even if h_1 has not entered the black hole, e.g., if the adversary has made the delay of its initial movement, and return to x_1 , at least Δ_2 .

In this case, however, the adversary can make the algorithm fail for $k > 3$ as follows. The adversary places the black hole between two groups, e.g., (h_i, t_i) and (h_{i+1}, t_{i+1}) (refer to Figure 1), that have one empty node between them (recall that they exist by construction whenever $k > 3$). Again, by having all tails experiencing the same delays as t_1 , and delaying all heads (except h_i) by Δ_2 time units, we have that not only h_i , but also t_{i+1} enter the black hole within a finite amount of time, making $RV(k - 1)$ unachievable, a contradiction. \square

It is known that knowledge of n is necessary to locate the black hole [13]. On the other hand, even without such knowledge, the gathering problem $RV(f(k))$ is *trivially* solvable if $f(k) \leq \lfloor k/2 \rfloor$:

Lemma 1. *Even if neither k nor n are known, $RV(f(k))$ is solvable if $f(k) \leq \lfloor k/2 \rfloor$.*

Proof. The algorithm that lets any robot choose a direction and move for as long as possible in that direction, trivially solves the problem. \square

Now, we are ready to state the following:

Theorem 3. *If neither k nor n are known, $RV(f(k))$ is unsolvable for $f(k) > \lfloor k/2 \rfloor$.*

Proof. By contradiction, let \mathcal{A} be a deterministic algorithm that solves $RV(f(k))$ for $f(k) > \lfloor k/2 \rfloor$ for every $k < n$, without knowledge of k and n .

For any non-constant monotonically non-decreasing function f , there exists a \bar{k} such that $f(\bar{k}) < f(2\bar{k})$: let us consider an execution \mathcal{E} of \mathcal{A} on a ring \mathcal{R} of size $n > \bar{k}$, with \bar{k} agents. By hypothesis, the execution of \mathcal{A} terminates with at least $f(\bar{k})$ agents gathered on the same node of \mathcal{R} , entering the terminal state.

Now, let us build a new ring $\bar{\mathcal{R}}$, having size $2n - 1$ and $2\bar{k}$ agents, composed of two copies (including the initial location of the agents) of \mathcal{R} , connected as depicted in Figure 2.

Consider now the execution \mathcal{E} of \mathcal{A} on $\bar{\mathcal{R}}$. The agents of \mathcal{R}' will behave independently and exactly as the execution by the agents in \mathcal{R} . Thus, within a finite amount of time, two distinct gathering points of size $f(\bar{k})$ will be achieved. Since $f(\bar{k}) < f(2\bar{k})$, we have a contradiction. \square

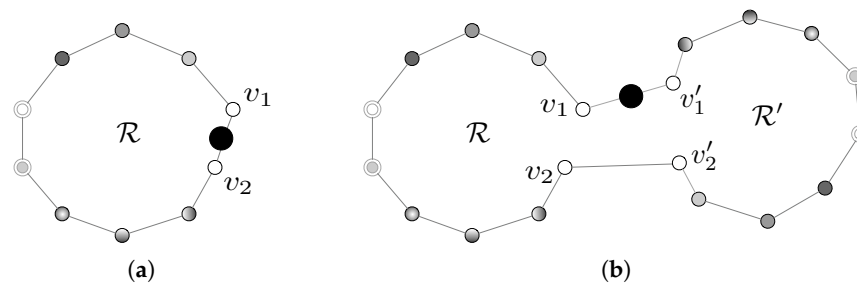


Figure 2. Example used in the Proof of Theorem 3.

Finally, let us state a general lower bound on the number of moves.

Theorem 4. *Let n be known, and the ring be unoriented. The $RV(k)$ problem requires at least $\lfloor \frac{k}{2} \rfloor (n - \frac{k}{2} - 2)$ moves.*

Proof. Consider the situation when the agents are divided in two groups $a_1, \dots, a_{\lfloor \frac{k}{2} \rfloor}$ and $a_{-1}, \dots, a_{\lceil \frac{k}{2} \rceil}$ adjacent to the black hole. Let x be an arbitrary gathering point inside the segment between the two groups at distance d from a_1 . Regardless of the algorithm, $k - 2$ agents have to move to the gathering point for a total of at least $\sum_{i=2}^{\lfloor k/2 \rfloor} (|d - i| + |n - d - 1 - i|)$ moves. Solving, we obtain that the number of moves is at least $\lfloor k/2 \rfloor (n - \lfloor k/2 \rfloor - 2)$. \square

3. Gathering When k Is Known and n Is Unknown

An immediate consequence of the fact that n is unknown is that, by Theorem 3, k must be known for non-trivial gathering to occur. Hence, in the rest of this section we assume that k is known. Another consequence of n being unknown is that, by [13], we cannot locate the black hole!

Let us now examine under what conditions the problem can be solved and how.

3.1. In Oriented Rings

Consider the following algorithm GORIGHT; agents are in two states: *explorer* and *follower*. Initially, every agent is an *explorer*.

Algorithm GORIGHT:

1. An *explorer* at node u moves to the $right(u)$ using cautious walk. (a) if it enters the black hole, it dies; (b) if it enters a node visited by another agent, it becomes a *follower*,

- returns to u , making the link safe; (c) if it enters an unvisited node, it returns to u (maintaining the state *explorer*), it renders the right link safe and goes again to $right(u)$ to continue the cautious walk.
2. A *follower* at node u : (a) if the right link is safe, it moves to the $right(u)$; (b) if the link is active and there are at least $k - 1$ agents at u , it terminates the execution of the algorithm; (c) else, it waits until the condition changes.

Theorem 5. $RV(k - 1)$ can be always solved, and this can be achieved in a time of at most $3(n - 2)$, using at most $nk - k^2/2 + 2n + o(nk)$ moves.

Proof. We first consider the correctness of the algorithm. The rightmost agent before the black hole will remain an *explorer* and eventually it might enter the black hole. All the other agents eventually become *followers*. Since no follower will go past the last safe node explored by the sole remaining explorer, eventually all followers gather at one node.

We now calculate the time complexity. Using cautious walk, the leftmost agent needs at most 3 time steps to traverse a link. Therefore, it will take it (and all other agents) at most $3(n - 2)$ time steps to reach the rightmost safe node and gather there.

Finally, we consider the number of moves employed by the agents. All moves can be divided into *cautious walk* and *right progress*. There are at most 2 cautious walk moves per edge, by all agents combined. An agent i will make at most $s_i - 1$ right progress moves, where s_i is the distance of the i 's starting location from the black hole, measured leftwards (the agent entering the black hole will make one more move). Since in each node, at most one agent starts, all s_i s are different. Summing together, we obtain the claimed bound. \square

Note that there are situations in which the $3(n - 2)$ time bound is indeed achieved: Consider a scenario where there are agents in the two sites neighboring the black hole; the leftmost agent wakes up first and all other agents join the execution only when an agent arrives to their node. Clearly, the leftmost agent must wake up all other agents, and every edge must be traversed using cautious walk.

3.2. Unoriented Rings

Since the ring is not oriented, by Theorem 1, $RV(k - 1)$ cannot be solved, as two agents can immediately disappear in the black hole. Hence, the best we can hope for is $RV(k - 2)$. The result is rather surprising. In fact, either $k - 2$ can gather or no more than $(k - 2)/2$ can, with nothing in between.

We will logically partition the entities in two sets, “clockwise” (or *blue*) and “counterclockwise” (or *red*), where all entities in the same set have a common view of “right”. Notice that each agent, although anonymous, can easily detect whether a message on a whiteboard has been written by an agent in the same set or not: when writing information on the whiteboard, agents also specify an indication of which label of the two local ports the writer considers to be “right”.

3.2.1. k Odd

Consider first the case when k is odd (recall k is known). In this case, only one of the two sets contains at least $(k - 1)/2$ agents, for example, the red one.

The agents of each set first of all execute a modification of algorithm GORIGHT for oriented rings, independently of and ignoring the agents of the other set, terminating this step as soon as $(k - 1)/2$ red follower agents gather in the same node: notice that, since k is odd, there is at most one such node. This node is designated as the *red collection* point: at this time, on the whiteboard of this node it is specified which is the left and right direction of the winning group (in this case, the red one); this direction will be used from now on as the global orientation of the ring. Next, one of the agents at the collection point node is designated to be the *right-collector*. The task of the *right-collector* is to move rightwards, to notify all agents on its way that the collection point has been identified, and to go there for final gathering: notification occurs through the whiteboard, by writing there the safe

direction to follow to reach the collection point. A second agent from the collection point is designated as the *left-collector*: it will behave similarly to the right one, moving leftwards. Upon being notified, an agent goes safely towards the collection point. As we will show, at most two agents (possibly none) enter the black hole; all others achieve the gathering within a finite amount of time. The set of rules describing the algorithm is reported in the following. All agents at the beginning are *explorers* (either red or blue).

Algorithm GR-ODD:

1. A red (resp. blue) *explorer* at node u moves to its right using cautious walk.
 - (a) if it enters the black hole, it dies;
 - (b) if it enters a node visited by another red (resp. blue) agent, it becomes a red (resp. blue) *follower*, returns to u , making the link safe;
 - (c) if it enters a node that is a blue (resp. red) *collection point*, it becomes a blue (resp. red) *follower*;
 - (d) if it enters a node where on the whiteboard there is a message indicating the direction of the collection point, it becomes *notified*;
 - (e) if it enters a node where on the whiteboard there is a message to become a *left-collector*, it cancels the message, and it becomes a *left-collector*;
 - (f) if it enters an unvisited node, it returns to u . If there are less than $(k-1)/2$ red (resp. blue) *followers* at u , it renders the right link safe and goes again to the $right(u)$ to continue the cautious walk. If there are at least $(k-1)/2$ red (resp. blue) *followers* at u , it becomes a *right-collector*; moreover, if u is not a collection point, it registers this node as the red (resp. blue) *collection point* on the whiteboard.
2. A red (resp. blue) *follower* at node u :
 - (a) if u is not a collection point and there are less than $(k-1)/2$ red (resp. blue) *followers*, it moves to its right if its right link is safe;
 - (b) if u is not a collection point and there are at least $(k-1)/2$ red (resp. blue) *followers* at u , it registers u as a collection point; if its right link is safe, it becomes a *right-collector*, otherwise, it becomes a *left-collector*;
 - (c) if u is a collection point and there is no *left-collector*, then it becomes a *left-collector*;
 - (d) if it enters a node where on the whiteboard there is a message indicating the direction of the collection point, it becomes *notified*.
3. The *left-collector* x travels (using cautious walk when necessary) left, writing on the whiteboards of every visiting node the instruction to reach the collection point; it does so until it either dies in the black hole or reaches the last safe node explored by a blue *explorer*. In the latter case, the *left-collector* leaves a message for the blue *explorer* y , informing it of the collection point and instructing it to become a *left-collector*; it then becomes *notified*.
4. The rules for the *right-collector* are exactly those for the *left-collector*, where “left” and “right” are switched and so are “blue” and “red”.
5. A *notified* agent moves in the direction of the collection point.
6. If an agent is at the collection point, and there are both the left-collector and the right-collector, it enters a terminal state.

Notice that, when an agent writes on the whiteboard, it provides all information necessary to the others to complete the required computations (e.g., if the *left-collector* already left, etc.).

Theorem 6. *Within a finite amount of time, a single collection point is formed, and at least $k-2$ agents will gather in the collection point.*

Proof. Since k is odd, one of the two groups is bigger than $\frac{k-1}{2}$. In addition, since all robots in a group move in the same direction, it follows that exactly one collection point is formed.

Moreover, by construction, at any point in time there is at most one right-collector and at most one left-collector, moving in opposite directions. Within a finite amount of time, there will be exactly one right-collector and exactly one left-collector. After this time, and within a finite amount of time, all agents not at the collection point and not at (or moving to) the black hole, are notified by the two collectors. After receiving the notification, an agent starts safely moving to the collection point, reaching it within a finite amount of time. Because of the use of cautious walk, at most two agents can enter the black hole, hence at least $k - 2$ agents gather at the collection point, thus the theorem follows. \square

Let us now examine the time costs of algorithm GR-ODD.

Theorem 7. *Algorithm GR-ODD terminates in a time of at most $5(n - 2)$, using at most $3nk/2 - k^2/4 + O(n) + o(nk)$ moves.*

Proof. Let t_1 be the time when the collection point was formed and let s be the number of edges explored at time t_1 . Since traversing one edge using cautious walk takes at most 3 time steps, we obtain $t_1 \leq 3s$. Let t_2 be the time by which every edge non-incident to the black hole has been traversed by a collector. We obtain $t_2 \leq t_1 + 3(n - 2 - s) + s$, since traversing each of the s already explored edges takes at most one time step. Finally, let t_3 be the time when all agents informed by the collectors arrive to the collection point. Clearly, $t_3 \leq t_2 + n - 2 \leq 3(n - 2) + s + n - 2$. Since $s \leq n - 2$, we obtain $t_3 \leq 5(n - 2)$.

Let us now consider the number of moves. By Theorem 5 the $(k - 1)/2$ followers that triggered the creation of the collection point have spent $nk/2 - k^2/8 + o(nk)$ moves. The collectors spend at most $2n$ moves. The remaining $(k - 1)/2$ agents might have collected at the opposite end (spending $nk/2 - k^2/8 + o(nk)$ moves). Adding the $(n - 2)(k - 1)/2$ moves sufficient to bring them to the collection point yields the theorem. \square

3.2.2. k Even

Consider now the case when k is even (recall k is known). We first observe that $RV((k - 2)/2)$ can always be solved by trivially having each set execute the gathering algorithm GORIGHT for oriented rings, and terminating it when at least $k/2 - 1$ follower agents of the same set gather in the same node. To complete the proof, we need to show that, when k is even, gathering of a greater number of agents cannot be guaranteed.

Lemma 2. *If k is even then $RV(p)$ cannot be solved for $p > (k - 2)/2$.*

Proof. Assume first that n is odd. Consider the symmetry axis passing through the BH and dividing the ring into two sides of equal size (note that when n is odd, the symmetry axis passes through an edge); denote by \bar{v} the symmetric image of node v . Consider now a symmetric but “complemented” distribution of the agents on the two sides of the ring: if there is a blue agent in node v then there is a red agent in node \bar{v} (see Figure 3a); thus, to each agent a with homebase v , corresponds an agent \bar{a} , of opposite color, with homebase \bar{v} . In this distribution, let both neighbors of the BH be occupied by agents. Consider now a synchronous execution (i.e., every move of an agent takes exactly one time step) of any solution algorithm, where all agents start at the same time, and where the first move of the agents neighboring the BH is directed towards the BH. Observe now that, for any move in a given direction by an agent a , there corresponds a move in the opposite direction by agent \bar{a} . In particular, whenever an agent a crosses the symmetry axis, agent \bar{a} will cross the axis in the opposite direction. Since after the first step there are $(k - 2)/2$ agents on each side (two agents went into the BH), the number of agents on each side will always be $(k - 2)/2$, thus rendering $RV(p)$ with $p > (k - 2)/2$ impossible.

If n is even, choose the symmetry axis to be the line dividing the ring into two sides of equal size, passing through the link between the BH and its clockwise neighbor v (see Figure 3b). Force the other link incident to v to be very slow. Since the algorithm does not

know n , it cannot distinguish this situation from the case of n odd, and the same arguments as for n odd would apply. \square

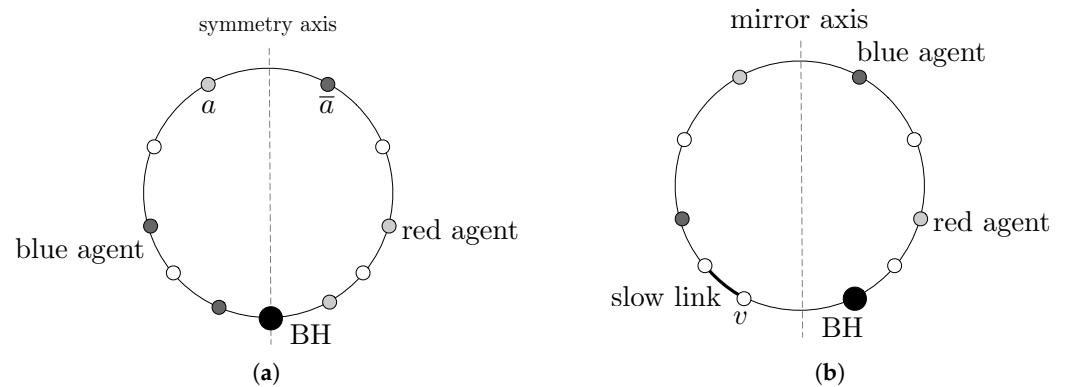


Figure 3. Mirror symmetric distribution for (a) n odd, and (b) n even.

We now show that, although we cannot guarantee that more than half of the surviving agents gather, we can however guarantee that *all* the surviving agents gather within a distance of 1 from each other. To prove this, we use the following strategy, that we will call GR-EVEN.

First, each set of agents executes the gathering algorithm GORIGHT for oriented rings, independently of and ignoring the agents of the other set, and terminates it when (at least) $k/2 - 1$ follower agents of the same set gather in the same node. Notice that it is possible that two (but no more than two) such gathering points will be formed; further, notice that they could be both made of agents of the same color!

Let us concentrate on one of them and assume, without loss of generality, that it is formed of red agents. By definition, associated with it, there is a red explorer that will become a right-collector once it realizes the collection point has been formed; among the followers gathered there, a left-collector has also been selected. Both collectors behave as in GR-ODD except that, now, each of them could encounter a collector from the other group (if it exists). Therefore, we need to add the following additional rules:

1. a collector computes and remembers the distance from its collection point. If passing the role of collector to an explorer, it passes also the distance information;
2. if a collector meets another collector (they must be from different collection points):
 - (a) if they are of the same color, since they both agree on the same orientation of the ring, then they agree on a unique site (e.g., the rightmost of the two) as the final common collection point;
 - (b) if they are of different colors, if the distance between the collection points is odd, they agree on the middle node as the final common collection point; otherwise, each chooses the closest site incident on the middle edge as the final collection point of its group;
 - (c) each collector goes back to its group and notifies all the agents there of their final collection point.

Lemma 3. Strategy GR-EVEN guarantees that $(k - 2)$ agents will either gather in the same node or gather within a distance of 1.

Proof. If only one collection point is formed, the correctness of GR-EVEN follows from the correctness of algorithm GR-ODD.

If two collection points are formed, then two collectors moving in opposite directions will eventually meet, decide on the final collection point (or edge), and inform all the followers. \square

The time and movement costs of strategy GR-EVEN can be easily determined.

Theorem 8. Strategy GR-EVEN terminates in a time of at most $5(n - 2)$.

Proof. If only one collection point is formed, the argument is the same as in the proof of Theorem 7, and the time complexity is at most $5(n - 2)$.

Assume now that two collection points are formed at a distance d from each other (measuring over part of the ring not containing the BH). Let t_1 be the time when the last collection point is formed and let s be the number of edges explored at that moment. Let t_2 be the time when the two collectors meet, and t_3 be the time when all agents gather at the common collection point or edge. Clearly, $t_1 \leq 3s$, and $t_2 \leq t_1 + d/2 + 2(n - 2 - s)$ (since the collectors that meet travel towards each other, and at most $n - 2 - s$ edges must be traversed using cautious walk). It takes less than d time steps for the collectors to come back to their collection points to inform the followers (they could have met quite asymmetrically), and $d/2$ steps for everybody to come to the new common collection point (recall, it is in the middle, between the two old collection points); hence, $t_3 \leq t_2 + 3/2d$. Combining together we obtain $t_3 \leq 3s + d/2 + 2(n - 2 - s) + 3/2d = 2(n - 2) + 2d + s$. Since d and s are at most $n - 2$, then $t_3 \leq 5(n - 2)$. \square

Theorem 9. Strategy GR-EVEN uses at most $3nk/2 - k^2/4 + o(nk)$ moves.

Proof. If there is a single collection point, the proof of Theorem 7 applies directly. If there are two collection points, applying Theorem 5 for the two groups creating collection points yields $2 \times (nk/2 + k^2/8)$ moves. Again, only $O(n)$ moves will be spent by collectors. Incidentally, the cost of moving the agents from the two collection points to the final collection point (or pair of points) is $(n - 2)(k - 2)/2$ regardless of the location of the final collection point, as both collection points contain the same amount of followers. \square

Summarizing:

Theorem 10.

1. If k is odd, $RV(k - 2)$ can always be solved.
2. If k is even, $RV(p)$ cannot be solved for $p > (k - 2)/2$; however, $RV((k - 2)/2)$ can always be solved.
3. $G(k - 2, 1)$ can always be solved.

4. Gathering When k Is Unknown and n Is Known

We now consider the case when k is unknown. Clearly, by Theorem 3, the ring size n must be known for any non-trivial gathering to be possible. Let us examine under what conditions and how the problem can be solved. Since k is unknown, we know by Theorem 2 that there exists no algorithm that solves $RV(k - 1)$, even if n is known. Let us now examine under what conditions the problem $RV(k - 2)$ can be solved.

4.1. Oriented Rings

We now introduce an algorithm, called SHADOW, quite different from the ones used when k is known, that solves $RV(k - 2)$ for $k \geq 4$ (also refer to the example depicted in Figure 4).

The overall idea is to associate with each contiguous block of explored nodes a group of agents, expanding that specific block until either (1) the explored block contains $n - 1$ nodes (in which case a final *collection* phase is initiated, whose goal is to collect the agents into a final designated collection point); or (2) the block merges with a neighboring explored block (in which case the corresponding groups of agents are combined into one group, expanding the new, bigger block).

The group of agents expanding a block consists of at least one and at most four agents. The agents associated with a group are of two kinds: *explorers* and *shadows*—at most one of each type for each direction. The task of the explorers is to expand the explored block in the opposite directions. The shadows scan the explored block between the explorers, counting

the size of the block. Their goal is to detect when the block contains $n - 1$ explored nodes (each node remembers—i.e., on the whiteboard—which types of agents have visited it so far).

At the beginning, each explored block consists of a single node containing an agent starting as an *R-explorer*, whose task is to expand the block by moving *right*(·). As the blocks grow, they eventually touch and their agents are combined, as follows:

- a two-agent block (i.e., created by merging two one-agent blocks) has one *R-explorer* and one *L-explorer*;
- a three-agent block has two explorers (one in each direction) and an *R-shadow*;
- a four-agent block has two explorers and two shadows (one in each direction);
- any additional agent (e.g., after merging two four-agent blocks) becomes *passive*.

The main technical difficulty arises from the fact that the whole process is distributed, asynchronous, and the agents are not immediately aware when their block collides with another block. In addition, both ends of a block might collide with neighboring blocks at about the same time, thus complicating the coordination between the agents of the block.

The adopted technique can be seen as *aging* of agents from initial *R-explorer* through *L-explorer*, *R-shadow*, *L-shadow* and finally *passive*. The goal is to have only one agent of each type per explored block—whenever an agent learns that there is a more extreme (to the right for *R*-agents, to the left for *L*-agents) agent of the same type, the agent is aged. The actual algorithm has a few exceptions to this rule, as sometimes the aging has to be delayed in order to inform waiting shadows to recompute the size of the explored block.

Algorithm SHADOW:

Agent states: *R-explorer*, *L-explorer*, *R*-shadow*, *R-shadow*, *L*-shadow*, *L-shadow*, *passive*, *collector*; *collector* and *shadow* agents maintain counter *size*. At the beginning all agents are *R-explorer*.

Algorithm for agent *a*:

- ***R-explorer*:** Move right using cautious walk as needed until a node *v* already visited by a different *R-explorer* is entered. Become *L-explorer*.
- ***L-explorer*:** Move left using cautious walk as needed until a node *v* already visited by an *L-explorer* is entered. Become *R*-shadow* and set $size_a$ to 0.
- ***R*-shadow*:** Move right until the last safe node *v* is reached, decrementing $size_a$. If *v* has already been visited by an *R-shadow*, set $size_a$ to 0 and become *L*-shadow*. If the *R-shadow* is waiting at *v*, wake it up.

If *v* has not been visited by an *R-shadow*, become *R-shadow*.

- ***R-shadow*:** (Starting at the rightmost safe node in its explored part.) If $size_a = 0$ (i.e., you left this node, traveled across the explored block and returned, and the *R-explorer* meanwhile did not explore a new node), wait until woken up by an *R*-shadow* or the *R-explorer* (which just explored the node to the right and returned to make the link safe).

In the latter case, become *R*-shadow*, otherwise, set $size_a = 0$ and move left until the last safe node is reached, counting in $size_a$ the number of nodes traversed.

If $size_a = n - 1$, then become *collector*, else become *R*-shadow*.

- ***L*-shadow*:** Move left until the last safe node *v* is reached, decrementing $size_a$. If *v* has already been visited by an *L-shadow*, become *passive* (but if the *L-shadow* is waiting at *v*, wake it up before becoming *passive*).

If *v* has not been visited by an *L-shadow*, become *L-shadow*.

- ***L-shadow*:** Analogous to *R-shadow*, but with reversed directions.
- ***collector*:** A *collector* agent traverses the explored part and collects all the agents on the way (if an agent meets a *collector*, it stops its activity and follows the *collector*). Once the whole explored part has been traversed (*collector* counts $n - 1$ links), $k - 2$ agents have been collected and have gathered at the same place. There is a technical detail: it can happen that both shadows can become collectors. In that case, the gathering

point is the node where they meet (or, if they crossed each other on a link, the right endpoint of that link).

- **passive:** Wait to be collected by a collector.

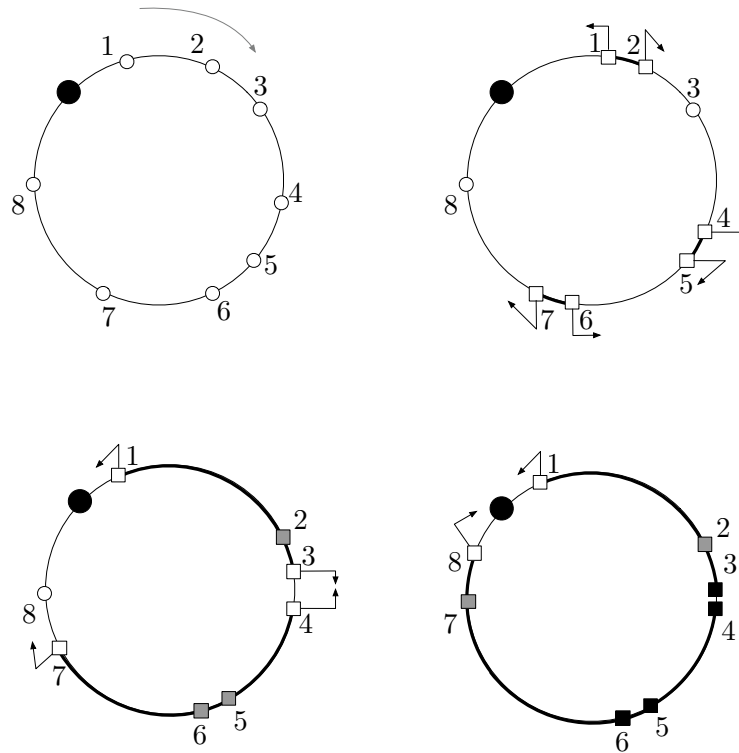


Figure 4. An example for Algorithm SHADOW, where the ring is assumed to be oriented clockwise, with 8 agents. The empty circles represent active agents; the white squares are the explorers, the grey squares the shadows, and the black squares the passive agents. The bold line shows the segments delimited by the explorers. The numbers are placed only to clarify how the agents move and are not used at all during the computation.

Lemma 4. *Within a finite amount of time all nodes will be explored, there will be only one R-explorer and one L-explorer, and they will both enter the black hole.*

Proof. Consider the rightmost agent. From construction, it will remain an *R-explorer* until it reaches the BH. Any other *R-explorer* will eventually become an *L-explorer*. The leftmost of those will remain an *L-explorer* and will eventually enter the BH, all others will become an *R*-shadow*. Of those, the rightmost one will eventually become an *R-shadow*, any remaining one will become an *L*-shadow*. Finally, out of those, the leftmost one will become an *L-shadow*, all others will become *passive*.

Let v be an arbitrary node. If there was an agent to the left of v in the initial configuration, the closest such agent a will reach v as an *R-explorer*. If there were no agents to the left of v , the leftmost *L-explorer* would eventually explore it. □

Lemma 5. *Eventually, at least one agent will become a collector.*

Proof. Consider the rightmost *R-shadow* a (since $k \geq 4$ and from the proof of Lemma 4 we know that there will be an *R-shadow*). From construction, a will remain an *R-shadow* (more precisely, alternating between *R-shadow* and *R*-shadow*) until it becomes a collector or itself is collected by a collector. It remains to be proven that a will not be blocked, forever waiting for the next node to the right to be explored.

Applying Lemma 4, we know that eventually there will be a single explored block. Consider the last step leading to an explored block of $n - 1$ nodes. If the explored block

reaches $n - 1$ nodes by exploring the rightmost node, either a is waiting in the rightmost safe node (in which case it will be awakened by the R -explorer and will traverse the $n - 1$ explored nodes to become a collector), or is currently scanning the explored block and will eventually return to the (new) rightmost explored node, notice that the block has been expanded, and re-scan it. A similar argument applies to the leftmost L -shadow if the explored block reaches $n - 1$ nodes by exploring the leftmost node. Finally, if the explored block reaches size $n - 1$ by the merging of two blocks, either a will detect that directly, or will be awakened by the R^* -shadow created by the collision of the blocks and re-scan to become a collector. \square

We can finally state the following:

Theorem 11. Let $k \geq 4$. Algorithm SHADOW solves $RV(k - 2)$ in $8(n - 2)$ time steps, using at most $4n^2 + nk - k^2/2 + O(n) + o(k^2)$ moves.

Proof. The correctness of the algorithm follows from Lemmas 4 and 5, and from the action of the collector agent.

Consider now the time steps of the algorithm. Let s_0 be the position of the leftmost agent (measured from the BH, going right) and s_1 be the position of the second leftmost agent. Then, at time $3(s_1 - s_0)$ the leftmost agent becomes an L -explorer and enters the BH at time $3s + 4(s_1 - s_0)$. If only the leftmost agent woke up spontaneously, it takes $3(n - s_0 - 1)$ time step for the right exploration to reach the BH. Therefore, all nodes are explored by time $\max(3s + 4(s_1 - s_0), 3(n - s_0 - 1)) < 4(n - 2)$. Note also, that by time $3(n - s_0 - 1)$ there is a single R -explorer, as all other R -explorer agents would have entered a node already visited by an R -explorer. This also means that, by time $3(n - s_0 - 1) + n - 2 < 4(n - 2)$ there will remain only one L -explorer, i.e., all other agents have become shadows. This means by time $5(n - 2)$ there will remain a single right shadow (R^* -shadow or R -shadow) and by time $6(n - 2)$ there will be a single left shadow. If a shadow initiates a scan after time $4(n - 2)$, it will detect that $n - 1$ nodes have been explored and will become a collector. The algorithm would then terminate in at most $n - 2$ additional steps. An unsuccessful scan (without becoming a collector) takes at most $2(n - 3)$ steps, therefore a successful scan would be initiated at time less than $6(n - 2)$ and the algorithm would terminate by time $8(n - 2)$.

Consider now the number of moves employed by the agents. First, note that at most $O(n)$ moves are performed by the explorers. Let us now count the number of moves executed by all agents in states R^* -shadow and R -shadow. Let us call *recharge* the events of an R -shadow being awakened by an R^* -shadow or R -explorer. Note that, there are at most $2n$ recharge events for all right shadows, as in each event either an R^* -shadow becomes an L^* -shadow, or the R -explorer has made progress. Moreover, an R -shadow (or R^* -shadow) can make at most $2(n - 2)$ moves between recharges—if it is not recharged it turns into an L^* -shadow. This means that the total number of moves by all agents in states R -shadow and R^* -shadow can be bound by $2n^2$. Using analogous arguments, it is easy to see that the total number of moves by all agents in states L -shadow and L^* -shadow is at most $2n^2$ as well. The total cost of collection can be bound by $nk - k^2/2 + o(k^2)$, using similar arguments as in the proof of Theorem 7. \square

4.2. Unoriented Rings

Interestingly, we discover for the unoriented case better conditions than those we have found when k was known instead of n . As in the oriented case, let us consider $k \geq 4$.

The main ideas of algorithm SHADOW carry over to algorithm UN-SHADOW, however, several technicalities have to be dealt with differently, due to lack of agreement on orientation. Again, we color the agents red and blue according to their initial orientation. In general, the agents follow algorithm SHADOW, ignoring the agents of the different color, however, these exceptions apply:

1. when an *R-explorer* comes to a node already visited by an *L-explorer* of the opposite color, it becomes an *R*-shadow*;
2. when an *L-explorer* comes to a node already visited by an *R-explorer* of the opposite color, it becomes an *R*-shadow*;
3. the L- and R- for shadows are used only for aging purposes. The tests are made against the shadows working in the same direction. For example, when a red *R*-shadow* a arrives to the last explored node v in its direction, it checks whether v has been visited by a red *R-shadow* or blue *L-shadow*. If yes, a becomes an *L*-shadow* (and possibly awakens the shadow at v , regardless of its color), otherwise a becomes an *R-shadow*;
4. when (if) two collectors cross each other on an edge, they may not agree on which endpoint to meet (if the situation is symmetric), therefore, only $G(k - 2, 1)$ is guaranteed, not $RV(k - 2)$.

Note that if all agents are of the same color, the algorithm UN-SHADOW behaves exactly as algorithm SHADOW and its correctness follows. Therefore, in the rest we assume there is at least one agent of each color. Let us also assume that a red *R-explorer* moves to the right and a blue *R-explorer* moves to the left.

Lemma 6. *Within a finite amount of time, all nodes will be explored.*

Proof. Consider an initial configuration. Let v be an arbitrary node and let a_l and b_l be the closest and second closest agents to v from the left, and a_r and b_r be the closest agents from the right, respectively. (Not all of them might be present, but at least one side has at least two agents—because $k \geq 4$.)

From construction, if a_l or b_l is a red agent, it will travel to the right and either reach v or meet a left-moving agent that already visited v . Similarly, v will be explored if a_r or b_r is blue.

Without loss of generality, assume the left side has at least two agents and both a_l and b_l are blue. b_l starts moving left as an *R-explorer* and when it enters the starting node of a_l , it will become an *L-explorer* and start moving right. Therefore, v will eventually be explored either by b_l or by an agent coming from the right. \square

Lemma 7. *Within a finite amount of time, an agent will become a collector.*

Proof. Since the only time an agent is waiting is either when it is passive, or as an *R-shadow* or *L-shadow*, and because $k \geq 4$, eventually there will be a shadow for the right direction (red *R-shadow* or blue *L-shadow*) and a shadow for the left direction. The rightmost such right shadow and the leftmost left shadow will not age anymore, because other shadows will age due to their rightmost/leftmost mark.

When all nodes become explored (Lemma 6 tells us that this will eventually happen), at least one of those extreme agents will be awakened, traverse the explored nodes, and become a *collector* (using the same arguments as in Lemma 5). \square

Consider first the case when k is odd or n is even.

Lemma 8. *There will be a unique collection point.*

Proof. If there is only one *collector*, there will be a single collection point—when the *collector* traveled $n - 1$ nodes. If there are two collectors, when they meet, they can agree on the collection point: if n is even, they select the point directly opposite to the BH; if k is odd, the two groups of agents have a different size, and the two collectors agree on a single collection point by exploiting this asymmetry. \square

From Lemma 8, we have:

Corollary 1. *At least $k - 2$ agents will gather in the collection point.*

Consider now the case when k is even and n is odd.

Lemma 9. *If k is even and n is odd, then $RV(p)$ cannot be solved with $p > (k - 2)/2$.*

Proof. Consider a ring where $k/2$ red entities are positioned at one side of the black hole, and $k/2$ blue agents are on the other side. Consider now a synchronous scheduler that sends the two agents closest to the black hole to move towards the black hole; under this scheduler there will always be the same number of agents at the two sides of the ring. \square

Lemma 10. *If k is even and n is odd, then $RV(p)$, $RV((k - 2)/2)$, and $G(k - 2, 1)$ can be achieved.*

Proof. From Lemma 7 we know that there will be a collector, however, there might be two of them—one for each direction. A single collector would collect $k - 2$ agents. If there are two collectors, at the moment they meet (or cross each other on a link), both $RV((k - 2)/2)$ and $G(k - 2, 1)$ are achieved. \square

Combining Lemmas 1, 9, and 10 yields the following:

Theorem 12.

1. *If k is odd or n is even, $RV(k - 2)$ can always be solved*
2. *If k is even and n is odd, $RV(f(k))$ cannot be solved for $p > (k - 2)/2$; however, $RV((k - 2)/2)$ can always be solved.*
3. *$G(k - 2, 1)$ can always be solved.*

Since the algorithm UN-SHADOW differs from the algorithm SHADOW only by accelerated aging and final collection, the proof of Theorem 11 carries almost without changes also for UN-SHADOW.

Theorem 13. *The algorithm UN-SHADOW for unoriented rings terminates in at most $8(n - 2)$ time steps, using at most $4n^2 + nk - k^2/2 + O(n) + o(k^2)$ moves.*

5. Conclusions

In this paper we studied the problem of gathering of mobile agents in a ring network in the presence of a black hole, and established a complete characterization of the conditions under which the problem can be solved. In particular, we determined the maximum number of agents that can be guaranteed to gather in the same location, depending on whether k or n is unknown. These results are tight: in each case, gathering with one more agent is deterministically unfeasible. All our possibility proofs are constructive: we provided mobile agent algorithms that allow the agents to gather within a predefined distance under the specified conditions. The analysis of the time costs of these algorithms showed that they are optimal. Among the interesting observations from our results, let us stress the facts that either k or n must be known, that not knowing k makes the problem much harder than not knowing n , and that parity of n and k influences the number of agents that can gather.

Our gathering algorithm for the case of unknown k is also a solution for the black hole location problem. Interestingly, its bounded time complexity is $\Theta(n)$; this is a significant improvement over the $O(n \log n)$ bounded time complexity of the existing algorithms for the same case.

The results for the case k unknown assumed $k \geq 4$. If the number of agents is 3 (but unknown), from Theorem 2 it follows that at most 1 agent can gather. In such a case, the algorithm of [13] can be modified to solve the problem; however, its time complexity is at least $O(n \log n)$. Proving that it is impossible to perform gathering in $O(n)$ time—if it is not known that there are at least 4 agents—is an interesting open problem.

In this paper, we have assumed that, initially, all agents are at distinct nodes. If this is not the case, in each node where initially there is more than one agent, one of them can

become “leader” (e.g., using the fair mutual exclusion of accessing the whiteboard), and the others will “follow the leader” (except on unsafe links). With careful attention to details, all the proposed algorithms can be modified to correctly handle this situation.

These results open several research directions. In particular, it would be interesting to investigate the gathering problem in the presence of a black hole in networks whose topology is more complex (e.g., hypercubes) or unknown (e.g., arbitrary biconnected graphs). Moreover, an entire unexplored research area is the examination of problems other than gathering (e.g., scattering, pattern formation) in the presence of a black hole, both in synchronous and asynchronous networks.

Author Contributions: All authors have equally contributed to conceptualization, methodology, validation, formal analysis, investigation, resources, and writing; project administration, N.S.; funding acquisition, P.F., G.P. and N.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Natural Science and Engineering Research Council (Canada) under the Discovery Grant program, by the Italian National Group for Scientific Computation GNCS-INdAM, and by Progetto PRA_2022_81 (Università di Pisa, Italy).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cao, J.; Das, S. *Mobile Agents in Networking and Distributed Computing*; Wiley: Hoboken, NJ, USA, 2012.
2. Flocchini, P.; Prencipe, G.; Santoro, N. *Distributed Computing by Mobile Entities*; Springer: Berlin/Heidelberg, Germany, 2019.
3. Peng, M.; Shi, W.; Corriveau, J.P.; Pazzi, R.; Wang, Y. Black hole search in computer networks: State-of-the-art, challenges and future directions. *J. Parallel Distrib. Comput.* **2016**, *88*, 1–15. [\[CrossRef\]](#)
4. Pelc, A. Deterministic Rendezvous Algorithms. In *Distributed Computing by Mobile Entities*; Flocchini, P., Prencipe, G., Santoro, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Chapter 17.
5. Barrière, L.; Flocchini, P.; Fraigniaud, P.; Santoro, N. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory Comput. Syst.* **2007**, *40*, 143–162. [\[CrossRef\]](#)
6. Bampas, E.; Blin, L.; Czyzowicz, J.; Ilcinkas, D.; Labourel, A.; Potop-Butucaru, M.; Tixeuil, S. On asynchronous rendezvous in general graphs. *Theor. Comput. Sci.* **2019**, *753*, 80–90. [\[CrossRef\]](#)
7. Czyzowicz, J.; Pelc, A.; Labourel, A. How to meet asynchronously (almost) everywhere. *Acm. Trans. Algorithms* **2012**, *8*, 37:1–37:14. [\[CrossRef\]](#)
8. Dieudonné, Y.; Pelc, A.; Villain, V. How to meet asynchronously at polynomial cost. *Siam J. Comput.* **2015**, *44*, 844–867. [\[CrossRef\]](#)
9. Chattopadhyay, T.; Prasad, G. Mobile agent security against malicious hosts: A survey. *Comput. Sci.* **2022**, *3*, 160. [\[CrossRef\]](#)
10. Sander, T.; Tschudin, C.F. Protecting mobile agents against malicious hosts. In *Mobile Agent Security*; Vigna, G., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 44–60.
11. Vitek, J.; Castagna, G. Mobile computations and hostile hosts. In *Mobile Objects*; Tsichritzis, D., Ed.; University of Geneva: Geneva, Switzerland, 1999; pp. 241–261.
12. Flocchini, P.; Santoro, N. Distributed Security Algorithms for Mobile Agents. In *Mobile Agents in Networking and Distributed Computing*; Cao, J., Das, S., Eds.; Wiley: Hoboken, NJ, USA, 2012; Chapter 3.
13. Dobrev, S.; Flocchini, P.; Prencipe, G.; Santoro, N. Mobile search for a black hole in an anonymous ring. *Algorithmica* **2007**, *48*, 67–90. [\[CrossRef\]](#)
14. Chalopin, J.; Das, S.; Labourel, A.; Markou, E. Black hole search with finite automata scattered in a synchronous torus. In Proceedings of the 25th International Symposium on Distributed Computing, Rome, Italy, 20–22 September 2011; pp. 432–446.
15. Chalopin, J.; Das, S.; Labourel, A.; Markou, E. Tight bounds for Black Hole Search with scattered agents in a synchronous ring. *Theor. Comput. Sci.* **2013**, *509*, 70–85. [\[CrossRef\]](#)
16. Czyzowicz, J.; Kowalski, D.; Markou, E.; Pelc, A. Complexity of searching for a Black Hole. *Fundam. Inform.* **2006**, *71*, 229–242.
17. D’Emidio, M.; Frigioni, D.; Navarra, A. Explore and repair graphs with black holes using mobile entities. *Theor. Comput. Sci.* **2015**, *605*, 129–145. [\[CrossRef\]](#)
18. Dobrev, S.; Flocchini, P.; Kralovic, R.; Prencipe, G.; Ruzicka, P.; Santoro, N. Optimal search for a black hole in common interconnection networks. *Networks* **2006**, *47*, 61–71. [\[CrossRef\]](#)
19. Dobrev, S.; Flocchini, P.; Královič, R.; Santoro, N. Exploring an unknown dangerous graph using tokens. *Theor. Comput. Sci.* **2013**, *472*, 28–45. [\[CrossRef\]](#)
20. Dobrev, S.; Flocchini, P.; Prencipe, G.; Santoro, N. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distrib. Comput.* **2006**, *19*, 1–35. [\[CrossRef\]](#)

21. Flocchini, P.; Kellett, M.; Mason, P.; Santoro, N. Map construction and exploration by mobile agents scattered in a dangerous network. In Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium, Rome, Italy, 23–29 May 2009; pp. 1–10.
22. Flocchini, P.; Kellett, M.; Mason, P.; Santoro, N. Fault-tolerant exploration of an unknown dangerous graph by scattered agents. In Proceedings of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Toronto, ON, Canada, 1–4 October 2012; pp. 299–313.
23. Klasing, R.; Markou, E.; Radzik, T.; Sarracco, F. Approximation bounds for Black Hole Search problems. *Networks* **2008**, *52*, 216–226. [\[CrossRef\]](#)
24. Kosowski, A.; Navarra, A.; Pinotti, M. Synchronous black hole search in directed graphs. *Theor. Comput. Sci.* **2011**, *412*, 5752–5759. [\[CrossRef\]](#)
25. Markou, E.; Paquette, M. Black hole search and exploration in unoriented tori with synchronous scattered finite automata. In Proceedings of the 14th International Conference on Principles of Distributed Systems, Rome, Italy, 18–20 December 2012; pp. 239–253.
26. Markou, E.; Shi, W. Dangerous Graphs. In *Distributed Computing by Mobile Entities*; Flocchini, P., Prencipe, G., Santoro, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Chapter 18.
27. Balamohan, B.; Flocchini, P.; Miri, A.; Santoro, N. Time optimal algorithms for black hole search in rings. *Discret. Math. Algorithms Appl.* **2011**, *3*, 1–15. [\[CrossRef\]](#)
28. Dobrev, S.; Santoro, N.; Shi, W. Using scattered mobile agents to locate a black hole in a unoriented ring with tokens. *Int. J. Found. Comput. Sci.* **2008**, *19*, 1355–1372. [\[CrossRef\]](#)
29. Dobrev, S.; Flocchini, P.; Prencipe, G.; Santoro, N. Multiple agents rendezvous in a ring in spite of a black hole. In Proceedings of the 7th International Conference on Principles of Distributed Systems, La Martinique, France, 10–13 December 2003; pp. 34–46.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.