Multiple Agents RendezVous In a Ring in Spite of a Black Hole

Stefan Dobrev[1], Paola Flocchini[1], Giuseppe Prencipe[2], and Nicola Santoro[3]

[1] University of Ottawa, {sdobrev,flocchin}@site.uottawa.ca
[2] Università di Pisa, prencipe@di.unipi.it
[3] Carleton University, santoro@scs.carleton.ca

Abstract. The *Rendezvous* of anonymous mobile agents in a anonymous network is an intensively studied problem; it calls for k anonymous, mobile agents to gather in the same site. We study this problem when in the network there is a *black hole*: a stationary process located at a node that destroys any incoming agent without leaving any trace. The presence of the black hole makes it clearly impossible for all agents to rendezvous. So, the research concern is to determine how many agents can gather and under what conditions.

In this paper we consider k anonymous, *asynchronous* mobile agents in an anonymous ring of size n with a black hole; the agents are aware of the existence, but not of the location of such a danger. We study the rendezvous problem in this setting and 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 for any non-trivial rendezvous). These results are *tight*: in each case, rendezvous with one more agent is impossible.

All our possibility proofs are constructive: we provide mobile agents protocols that allow the agents to rendezvous or near-gather under the specified conditions. The analysis of the time costs of these protocols show that they are *optimal*.

Our rendezvous protocol for the case when k is unknown 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 protocols for the same case.

Keywords: Mobile Agents, RendezVous, Gathering, Black Hole, Harmful Host, Ring Network, Asynchronous, Anonymous, Distributed Computing.

1 Introduction

In networked systems that support autonomous *mobile agents*, a main concern is how to develop efficient agent-based *system protocols*; that is, to design protocols 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 *wakeup*, *traversal*, *gathering*, *election*. The coordination of the agents necessary to perform these operations is not necessarily simple or easy to achieve. 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 conditions for the solvability of these problems and to the discovery of efficient algorithmic solutions; e.g., see [4–6, 15].

At an abstract level, these environments, which we shall call *distributed mobile systems*, can be described as a collection $\mathcal{E}$ of autonomous mobile entities located in a graph G. Depending on the context, the entities are sometimes called *robots* or *agents*; in the following, we use the latter. The agents have computing capabilities and bounded storage, execute the same protocol, and can move from node to neighboring node. They are *asynchronous*, in the sense that every action they perform takes a finite but otherwise unpredictable amount of time. Each node of the network, also called *host*, provide a storage area called *whiteboard* for incoming agents to communicate and compute, and its access is held in fair mutual exclusion. The research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost.

In this paper, we focus on a fundamental task in distributed mobile computing, *rendezvous* in the simplest symmetric topology: the *ring* network. We will consider its solution in presence of a severe security threat: a *black hole*, a network site where a harmful process destroys all incoming agents without leaving a trace.

Rendezvous. The *rendezvous* problem consists in having all the agents gather at the same node; upon arriving there, each agent terminally sets its variable to *arrived*; there is no a priori restriction on which node will become the rendezvous point.

This problem (sometimes called *gathering*, *point-formation*, or *homing*) is a fundamental one in distributed mobile computing both with agents in graphs and with robots in the plane.

In the case of agents in the graph, the rendezvous problem has been extensively investigated focusing on more limited settings (e.g., without whiteboards) with *two* agents; e.g., see [1–3, 6, 7, 13, 19]. Almost from the start it became obvious that the possibility (and difficulty) of a solution is related to the possibility (and difficulty) to find or create an *asymmetry* in anonymous and symmetric settings, like the one considered here, to break symmetry in the problem and thus ensure a rendezvous solutions researchers have used randomization (e.g., [2]), or different deterministic protocols for the two agents (e.g., [19]), or indistinguishable tokens [13]. The case of more than two agents has been investigated in [11, 14, 17], with only [11] providing a fully deterministic solutions for anonymous ring networks.

Let us stress that *all* these investigations assume *synchronous agents* and this assumption is crucial for the correctness of their solutions.

In contrast, in our setting, both nodes and agents, besides being *anonymous*, are also fully *asynchronous*. The only known results for this setting are about the relationship between *sense of direction* and possibility of *rendezvous* [6]; interestingly, the link between rendezvous and symmetry-breaking is even more clear: rendezvous is in fact equivalent to the *election* problem [6].

Black Hole Location. 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 [12, 16, 18]), and recently also from an algorithmic prospective [8, 9]. Obviously, the first step in any solution to such a problem must be to *identify*, if possible, the harmful host; i.e., to determine and report 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. The task is to develop a mobile agents protocol to determine and report the location of the black hole; the task is completed if, within finite time, at least one agent survives and knows the location of the black hole. The research concern is to determine under what conditions and at what cost mobile agents can successfully accomplish this task, called the *black hole location* problem. Note that this type of highly harmful host is not rare; for example, the undetectable crash failure of a site in a asynchronous network transforms that site into a black hole.

The black hole location problem has been investigated focusing on identifying conditions for its solvability and determining the smallest number of agents needed for its solution [8–10]. In particular, a complete characterization has been provided for ring networks [8].

Our Contributions. In this paper we consider the *rendezvous* problem in a more difficult setting: k asynchronous anonymous agents dispersed in a totally symmetric ring network of n anonymous sites, one of which is a *black hole*.

Clearly it is impossible for all agents to gather since an adversary (i.e., a bad scheduler) can immediately direct some agents towards the black hole. So, the research concern is to determine how many agents can gather. We study this problem and establish a complete characterization of the conditions under which the problem can be solved. The possibility results are summarized in the table shown in Figure 1; these results are *tight*: in each case, rendezvous with one more agent is impossible. It is interesting to observe that at least one of k and n must be known to the agents; however, knowledge of both is not necessary.

	n unknown, k known		n known, k unknown	
ORIENTED	$\forall k$	$RV(k-1)$	$\forall k$	$RV(k-2)$
UNORIENTED	k odd	$RV(k-2)$	k odd or n even	$RV(k-2)$
	k even	$RV(\frac{k-2}{2})$	k even and n odd	$RV(\frac{k-2}{2})$
	$\forall k$	$G(k-2,1)$	$\forall k$	$G(k-2,1)$

Fig. 1. Summary of possibility results.

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

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

All our possibility proofs are constructive: we provide mobile agents protocols that allow the agents to rendezvous or near-gather under the specified conditions.

Our rendezvous protocol, for the case when k is unknown, is also a solution for the black hole location problem. Interestingly, its bounded time complexity is $O(n)$; this is a significant improvement over the $O(n \log n)$ bounded time complexity of the existing protocols for the same case [8].

Due to space limitation all the proofs are omitted.

2 Definitions, Basic Properties and Techniques

2.1 The Framework

The network environment is a ring $\mathcal{R}$ of n *anonymous* (i.e., identical) nodes. Each node has two ports, labelled *left* and *right*; if this labelling is globally consistent, the ring will be said to be *oriented*, *unoriented* otherwise. Each node has a bounded amount of storage, called *whiteboard*.

In this network there is a set $a_1, \ldots, a_k$ 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 "protocol"), 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 whiteboards; access to a whiteboard is done in mutual exclusion. The agents execute a protocol (the same for all agents) that specifies the computational and navigational steps. 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 by the corresponding agent.

The agents are aware of the fact that in the network there is a *black hole* (BH); its location is however unknown. In this environment, we are going to consider the *Rendezvous* problem and the *Near-Gathering* problem defined below.

The *Rendezvous* problem $RV(p)$ consists in having at least $p \leq k$ agents gathering in the same site. There is no a priori restriction on which node will become the rendezvous point. Upon recognizing the gathering point, an agent terminally sets its variable to *arrived*. We consider a solution algorithm terminated when at least p agents become *arrived* (explicit termination).

The *Near-Gathering* problem $G(p,d)$ consists in having at least p agents within distance d from each other. As for the Rendezvous problem we consider the algorithm terminated when at least p agents know that they are within distance d from each other and change their state to a terminal state. Clearly, $G(p,0) = RV(p)$.

The efficiency of a solution protocol is obviously first and foremost measured in the *size* of the solution, i.e. the number of agents that the algorithm will make rendezvous at the same location. 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. During the computation some agents will disappear in the black hole, some will survive and eventually gather; for the purposes of bounded time complexity we will consider that the overall computation starts (i.e., we will start to count time) when the first surviving agent starts the algorithm.

2.2 Cautious Walk

In the following we describe a basic tool, first introduced in [8], that we will use in all our protocols to minimize the number of agents that disappear in the back hole.

In our algorithms, the ports (corresponding to the incident links) of a node can be classified as (a) *unexplored* – if no agent has moved across this port, (b) *safe* – if an agent arrived via this port or (c) *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 two rules:
Rule 1. when an agent moves from node u to v via an *unexplored* port (turning it into *active*), it immediately returns to u (making the port *safe*), and then goes back;
Rule 2. no agent leaves via an *active* port.

In the following, agents will either move only on safe links or move using cautious walk.

2.3 Basic Results

There are some basic obvious facts:

Theorem 1. *In an anonymous ring with a* Bh
1. $RV(k)$ *is unsolvable;*
2. If the ring is unoriented, then $RV(k-1)$ *is unsolvable.*

Less obvious are the following facts. Rendez-vous problem $RV(p)$ is said to be *non-trivial* if p is a non-constant function of k.

Theorem 2. *If* k *is unknown, non-trivial rendez-vous requires locating the black hole.*

In view of the fact that knowledge of n is necessary for locating a black hole

Theorem 3. [8] *If* n *is not known, BH location is unsolvable.*

it follows that

Theorem 4. *Either* k *or* n *must be known for non-trivial rendez-vous.*

3 Characterization and Tight Bounds

3.1 RendezVous when n is unknown

An immediate consequence of the fact that n is unknown is that, by Theorem 4, k *must be known* for non-trivial rendezvous to occur. Hence, in the rest of this section we assume that k is known. Another consequence of n being unknown is that, by Theorem 3, we can *not* locate the black hole!

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

In Oriented Rings

Theorem 5. *$RV(k-1)$ can be always solved, and this can be achieved in time at most $3(n-2)$.*

To prove this theorem, consider the following protocol **GoRight!**; agents are in two states: *explorer* and *follower*.

PROTOCOL **GoRight!**

1. Initially, everybody is an *explorer*.
2. An *explorer* moves right using cautious walk. If it enters a node visited by another agent, it becomes a *follower*.
3. A *follower* moves right, traversing only safe links.
4. If there are $k-1$ *followers* in one node, the agents there terminate the execution of the protocol.

Lemma 1. *Protocol* **GoRight!** *solves $RV(k-1)$.*

Note that the rendezvous site is not necessarily next to the black hole.

Lemma 2. *Protocol* **GoRight!** *terminates in time at most $3(n-2)$ since the start of the leftmost agent.*

Note: 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 left most agent must wake-up all other agents, and every edge must be traversed using cautious walk.

Theorem 5 now immediately follows from Lemmas 1 and 2.

Unoriented Rings Since the ring is not oriented, by Theorem 1, $RV(k-1)$ can *not* 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 that $(k-2)/2$ can, with nothing in between.

Theorem 6.
1. If k is odd, $RV(k-2)$ can always be solved
2. If k is even, $RV(p)$ can not 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.

To prove this theorem, 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 (e.g, each message contains also an indication of which of the two local ports the writer considers to be "right").

Consider first the case when k is odd (recall k is known).

We have the following protocol **GR-Odd**.

1. The agents of each set first of all execute the rendezvous algorithm **GoRight!** for oriented rings, independently of and ignoring the agents of the other set, terminating as soon as $(k-1)/2$ *follower* agents of the same set gather in the same node.
   (Notice: this will eventually happen, and only to one set, as there is only one set with at least $(k+1)/2$ agents, and eventually only one of those agents will remain *explorer*).
   Without loss of generality, let this happens to the *red agents.*
2. The node where the $(k-1)/2$ red *followers* have gathered becomes the *collection point*, and one of the *followers* is selected as *left-collector.*
3. Every *follower* or blue *explorer* arriving at the collection point joins the group.

4. The *left-collector* x travels (using cautious walk when necessary) left and tells every *follower* and red *explorer* it encounters to go to the collection point; it does so until it reaches the black hole or 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 meeting point, and instructing it to become *left-collector*; it then returns to the collection point. If/when the *explorer* y returns to that node, it finds the message, becomes *left-collector* and acts accordingly.
5. A red *explorer* returning to the collection point during its cautious walk (notice: there is only one) becomes now a *right-collector*.
6. The rules for the *right-collector* are exactly those for the *left-collector*, where "left" is replaced by "right", and viceversa.

Since k is odd, we get

Lemma 3. *There is only one collection point.*

By construction of algorithm **GR-Odd** we have

Lemma 4. *Every edge non-incident to the black hole will be traversed by a collector.*

Because of cautious walk, at most 2 agents will enter the black hole; this fact, combined with Lemma 4, yields the following:

Lemma 5. $k-2$ *agents will gather in the collection point.*

Hence, by Lemmas 3 and 5, part (1) of Theorem 6 holds. Before proceeding with the proof of the other parts of Theorem 6, let us examine the time costs of Protocol**GR-Odd**.

Theorem 7. *Protocol* **GR-Odd** *terminates in time at most* $5(n-2)$.

Consider now the case when k is even (recall k is known). To prove part (2) of Theorem 6 we first observe that $RV((k-2)/2)$ can always be solved by trivially having each set execute the rendezvous 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, rendezvous of a greater number of agents can not be guaranteed.

Lemma 6. *If* k *is even then* $RV(p)$ *can not be solved for* $p > (k-2)/2$.

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

We will first of all have each set execute the rendezvous algorithm **GoRight!** for oriented rings, independently of and ignoring the agents of the other sets, and terminate 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 generalitazion, 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 rules:

1. a *collector* keeps the distance from its collection point. When passing the role of collector to an *explorer*, it passes also the distance information.
2. when a *collector* meets another *collector* (notice: they must be from different groups; further notice, they might actually "jump" over each other):
   (a) if they are of the same color, then they agree on a unique site (e.g., the rightmost of the two ones) 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 goes back to its group and notifies all the agents there of their final collection point.

Lemma 7. *Protocol* **GR-Even** *guarantees that* $(k-2)$ *agents will either rendezvous in the same node or gather within distance* 1.

This completes the proof of Theorem 6. The time efficiency of Protocol **GR-Even** can be easily determined:

Theorem 8. *Protocol* **GR-Even** *terminates in time at most* $5(n-2)$.

3.2 RendezVous when k is unknown

An immediate consequence of the fact that k is unknown is that, by Theorem 4, the ring size n must be known for any non-trivial rendez-vous to be possible.

Another consequence is that, by Theorem 2, if we want to rendez-vous we *must* locate the black hole! Let us examine under what conditions and how the problem can be solved.

Oriented Rings

Theorem 9. *Let* $k \geq 4$. *Then* $RV(k-2)$ *can always be solved.*

To prove this theorem we design a protocol, called *Shadow*, quite different from the ones used when k is known. The "virtual" structure of the protocol is rather simple:

1. There will be a pair of agents, a *right explorer* and a *left-explorer*, that will transform every link not leading to the BH into a safe one by moving (using cautious walk) in opposite directions along the ring. Clearly, both these *explorers* will disappear in the BH.
2. Associated to the pair of *explorers* there is a pair of agents, a *right shadow* and a *left shadow*, whose function is to detect *when* both *explorers* have moved to the BH.
3. The *shadows* will then become *collectors*, traverse the safe area, collecting the other entities, and meet in a site that becomes the collection point.

To make this "virtual" structure real we must solve several problems, summarized by the following questions: Q1. how do we create a unique pair of *explorers* ? Q2. how do we create a unique pair of *shadows* ? Q3. how do the *shadows* detect termination ? Q4. how do the *collectors* make all other entities gather at a unique collection point ?
The first two questions are particularly important since initially the agents are all in the same identical state (recall, they are anonymous) and execute the same protocol. An informal description of the algorithm and thus of the answers to these questions follows.

Initially, each agent is *active* and moves "right" using cautious walk. It is possible that an *active* agent x enters the BH (because of cautious walk, this can happen to at most one agent); in this case, x will be considered the final *right-explorer* (one of the two we are looking for). If instead an *active* agent x reaches the last safe point of another *active* entity y, it transforms itself into a *left-explorer* and moves "left" using cautious walk: a pair is born.

So, after some time, pairs of *left-* and *right- explorers* will start to be formed (see Figure 2), the two members of each pair moving in opposite directions. Notes that each pair delimits a contiguous segment of safe nodes in the ring. Sooner or later, a *right-explorer* x will meet the *left-explorer* z belonging to a different pair; i.e., two segments meet. When this happens, the two agents cease to be *explorers* and each becomes a *shadow looking for a master*; as a consequence , out of their two pairs they create a single pair composed of the *right-explorer* to the right of z and the a *left-explorer* to the left of x; i.e., the two segments merge.

Eventually, only one pair of *explorers* will be left (note that when this happens, they are possibly both in the BH already).

Initially, all *explorers* are *without a shadow*; as soon as an *explorer* has a *shadow* (and is aware of this fact) it is said to be *complete*. What will x and y do when they become *shadows looking for a master* is to check if the *explorers* of their segment have a *shadow*: if they are not *complete*, they will become so: x becomes the *shadow* of one of them, and y of the other; if they are both *complete* already, x and y will become *passive* and wait to be collected; if only one of them is *complete*, only one of x and y will become a *shadow*, the other *passive* (see Figure 2). Note that, in any case, the segment where x and y are will have two *complete explorers*. This means that, eventually, there will be only one couple of *right-shadow* and *left-shadow*.

The task of a *shadow* x is to check the size of the segment it belongs to. In particular, a *right-shadow* x waits until its *right-explorer* z has increased the size of the segment by one; at this point, x goes to the *left-explorer* of the segment, and measures the size of the segment: if the size is $n-1$ (i.e., all nodes except the BH have been explored), x becomes *collector* and starts moving in the opposite direction; otherwise, it goes back to its *right-explorer*. The *left-shadow* follow similar rules. Only the two final *shadows* can become *collectors*.

The task of a *collector*, is to collect all the passive entities it encounters traversing the safe area, transforming them into *collected*; those entities will then follow the *collector*. In this way, when the two *collectors* meet, a unique gathering point has been formed, and all *collected* agents will gather there.

Following is the detailed description of the protocol. Initially, all entities are *active*.

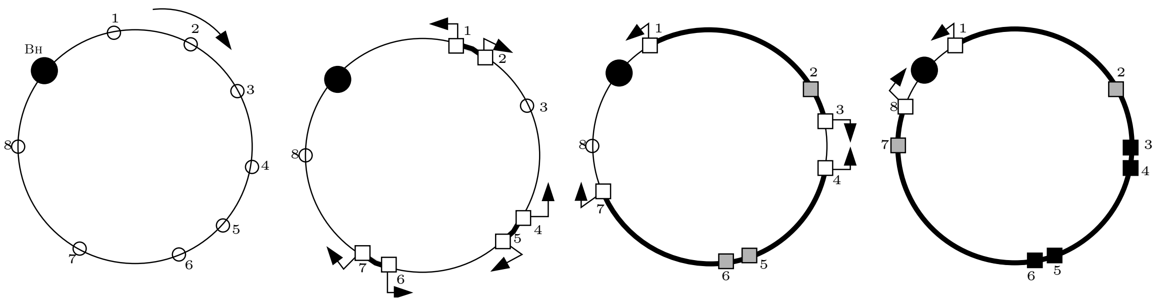

Fig. 2. The Shadow Protocol, where the ring is assumed to be oriented clockwise. The empty circles represent *active* agents; the white squares are the *explorers*, the grey squares the *shadows*, and the black squares the *passive* agents. The fat line evidences the segments delimited by the explorers. The numbers are placed only to clarify how the agents move, and are not used at all during the computation.

PROTOCOL **Shadow**

1. An *active* agent x moves to the right until one of the following will happen:
   (a) It reaches the “last safe site” of another *active* entity y; in this case it leaves a message “Become Right Explorer”for y, and becomes a *left-explorer without shadow*.
   (b) It finds (returning to its “last safe site” during cautious walk) the message “Become Right Explorer”. If there is no *shadow*, it becomes a *right-explorer without shadow*, else it becomes a *complete right-explorer*; in either case, it moves to the right.
   (c) It meets a *left-explorer* z (or reaches its “last safe site”); in this case, x becomes a *shadow looking for master*.
   (d) It enters the BH; in this case, x will be considered to have become a *right-shadow*.
2. An (left- or right-) *explorer* x continues to move in the assigned direction using cautious walk until one of the following will happen:
   (a) It meets (or reaches the “last safe site” of) an *explorer* coming in the opposite direction. In this case x becomes a *shadow-looking-for-explorer*;

(b) It finds (returning to its "last safe site" during cautious walk) the message "You Have a Shadow". It becomes a *complete explorer* and continues to move in the assigned direction.
(c) [Only for *left-explorer*.] It reaches the last safe site of an *active* entity containing the message "Become Right Explorer"; in this case it becomes *shadow looking for a master*.
(d) [Only for *right-explorer*.] It reaches the "last safe site" of an *active* entity y; in this case, x leaves a message "Become Right Explorer", and becomes a *shadow looking for a master*.
(e) It enters the Bh.

3. A *shadow x looking for a master* does the following:
   (a) It goes to check if the *right-explorer* is without a *shadow*. If this is the case, x leaves a message "You Have a Shadow", becomes *right-shadow*, and is assigned to that *explorer*;
   (b) If the *right-explorer* is already complete, x goes to check if the *left-explorer* is without a *shadow*. If this is the case, x leaves a message "You Have a Shadow", becomes *left-shadow*, and is assigned to that *explorer*.
   (c) If both *explorers* already have shadows, x becomes *passive* and waits to be collected by a *collector*.
4. A (left- or right-) *shadow* x assigned to an *explorer* z does the following:
   (a) It waits in the "last safe site" of the *explorer* of the assigned direction, until the "safe" area is increased by (at least) one site (since last time x checked);
   (b) It then goes towards the "last safe site" of the opposite *explorer* y counting the size of the safe area; when it arrives there:
      i. If the size of the safe area is $n-1$ then both *explorers* are in the Bh, and the final part of the protocol starts: x becomes a *collector* and moves in the assigned direction.
      ii. Otherwise, x goes back to the "last safe site" of the *explorer* in its assigned direction, to check whether the "safe" area has increased since x left. If in this travel it encounters another *shadow* of the same type (i.e., right-shadow encounters right-shadow), x checks if there is a *left-shadow*; if so, x becomes *passive* and waits to be collected by a *collector*; otherwise x becomes *left-shadow*, is assigned to the "left" *explorer* and leaves a message "You Have a Shadow" for it.
      iii. If reaches a site traversed by a *collector*, it becomes *collected* and follows the *collector*.
5. A *collector* x moves in the assigned direction until it encounters a *collector* traveling in the opposite direction or travels $n-1$ steps. During this travel, if it finds a *passive* agent, it transform it into *collected*. A *collected* agent follows its *collector*.
   (a) If a *collector* finds the other *collector* z: If they both meet at the same node, that node becomes the unique collection point: all other agents will stop there. If the collectors jumped over each other over an edge, the right endpoint of the edge is chosen as the collection point.
   (b) If it travels distance $n-1$ without encountering the other *collector*, that site becomes the unique gathering point.

Let us now examine the correctness of protocol *Shadow*.

Lemma 8. *Within finite time there will be only one* right-explorer *and one* left-explorer*, and they will both enter the black hole.*

Lemma 9. *Within finite time there will be only one* right-shadow *and one* left-shadow.

Lemma 10. *At least one* shadow *will become* collector*, and a* collector *knows the location of the black hole.*

Lemma 11. *Every edge non-incident to the* Bh *will be traversed by a collector.*

Lemma 12. *There will be a unique collection point.*

Lemma 13. $k-2$ *agents will gather in the collection point.*

This completes the proof of Theorem 9. Let us now examine the time costs of this protocol.

Theorem 10. *The protocol* Shadow *terminates in at most* $8(n-2)$ *time steps since the wake-up of the leftmost agent.*

Let us stress that protocol *Shadows* solves the *black hole location* problem (by Lemma 10). This means that we have obtained a significant improvement over the $O(n \log n)$ time complexity of the existing protocols for the black hole search.

Unoriented Rings Interestingly, we discover for the unoriented case better conditions than those we have found when k was known instead of n.

Theorem 11.
1. If k odd or n even, $RV(k-2)$ can always be solved
2. If k even and n is odd, $RV(p)$ can not be solved for $p > \lfloor (k-2)/2 \rfloor$; however, $RV(\lfloor (k-2)/2 \rfloor)$ can always be solved.
3. $G(k-2, 1)$ can always be solved.

We will again logically partition the entities in two sets, "clockwise" or *blue* and "counterclockwise" *red*, where all entities in the same sat have a common view of "right".

To prove part 1. of Theorem 11 we consider PROTOCOL *Blue-Red Shadows*. This is protocol *Shadows* expanded to take care of few additional cases that can occur.

NEW CASES:
- In Rule 2 of *Shadows*: A red *active* can "encounter" a blue *active* entity; in this case they both become *explorers* each continuing to move in its current direction.
- In Rule 5.b.ii of *Shadows*: A red *right-shadow* (resp. *left-shadow*) x returning to its assigned *explorer* can encounter a blue *left-shadow* (resp. *right-shadow*) y ; in this case, x acts as it has encountered a red *left-shadow* (resp. *right-shadow*).

Lemma 14. *Lemmas 8 to 11 hold also for protocol* Blue-Red Shadows *in unoriented rings.*

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

Lemma 15. *There will be a unique collection point.*

Lemma 16. $k-2$ *agents will gather in the collection point.*

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

Lemma 17. *If k is even and n is odd then $RV(p)$ can not be solved for $p > (k-2)/2$; however, $RV((k-2)/2)$ can be achieved.*

We now show that, although we cannot guarantee that more than half of the surviving agents rendez-vous, we can however guarantee that *all* the surviving agents gather within distance 1 from each other.

Lemma 18. *Protocol* **Blue-Red Shadows** *guarantees that* $(k-2)$ *agents will either rendez-vous in the same node or gather within distance* 1.

Using the same approach as in the proof of Theorem 10 we get

Theorem 12. *The modified protocol* Shadow *for unoriented rings terminates in at most* $8(n-2)$ *time steps.*

References

1. S. Alpern. The Rendezvous Search Problem. SIAM Journal of Control and Optimization, Volume 33, 673 - 683, 1995.
2. S. Alpern and V. Baston and S. Essegaier. Rendezvous Search on a Graph. Journal of Applied Probability, 36, No.1, 223-231, 1999.
3. E.J. Anderson and R.R.Weber. The Rendezvous Problem on Discrete Locations. Journal of Applied Probability, 28, 839-851, 1990.
4. E. Arkin, M. Bender, S. Fekete, and J. Mitchell. The freeze-tag problem: how to wake up a swarm of robots. In 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02), pages 568–577, 2002.
5. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA '02), 200-209, 2002.

6. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Election and rendezvous in fully anonymous systems with sense of direction. In 10th Colloquium on Structural Information and Communication complexity (SIROCCO '03), 17-32, 2003.
7. A. Dessmark, P. Fraigniaud and A. Pelc. Deterministic rendezvous in graphs. In 11th Annual European Symposium on Algorithms (ESA'03) 2003.
8. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile agents searching for a black hole in an anonymous ring. In 15th Int. Symposium on Distributed Computing (DISC 2001), 166–179, 2001.
9. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks. In 21st ACM Symposium on Principles of Distributed Computing (PODC '02), 153-162, 2002.
10. S. Dobrev, P. Flocchini, R. Král'ovič, G. Prencipe, P. Ružička, and N. Santoro. Searching for a black hole in hypercubes and related networks. In 6th International Conference on Principles of Distributed Systems (OPODIS '02),171-182, 2002.
11. P.Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Multiple Mobile Agent Rendezvous in a Ring. In LATIN '04 (accepted for publication).
12. F. Hohl. A Framework to Protect Mobile Agents by Using Reference States. In International Conference on Distributed Computing Systems (ICDCS '00), 2000.
13. E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Mobile Agent Rendezvous in a Ring. In 23rd International Conference on Distributed Computing Systems (ICDCS'03), 2003.
14. W.S. Lim and A. Beck and S. Alpern. Rendezvous search on the Line with More Than Two Players. Operations Research, 45, pp. 357-364, 1997.
15. P. Panaite and A. Pelc. Exploring unknown undirected graphs. Journal of Algorithms, 33:281–295, 1999.
16. T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In: Mobile Agents and Security, LNCS 1419, pp. 44-60, 1998.
17. L.C. Thomas amd P.B. Hulme. Searching for Targets Who Want to be Found. Journal of the Operations Research Society, 48, Issue 1, pp. 44-50, 1997.
18. Jan Vitek and Giuseppe Castagna. Mobile Computations and Hostile Hosts. In: D. Tsichritzis (Ed.), Mobile Objects, University of Geneva, pp. 241-261, 1999.
19. X. Yu and M. Yung. Agent Rendezvous: A Dynamic Symmetry-Breaking Problem. In ICALP '96, LNCS 1099, 610-621, 1996.