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

Stefan Dobrev [*] Paola Flocchini[†] Giuseppe Prencipe[‡] Nicola Santoro[§]

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 *wake-up*, *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

[*]University of Ottawa, email:sdobrev@site.uottawa.ca
[†]University of Ottawa, email:flocchin@site.uottawa.ca
[‡]Università di Pisa, email:prencipe@di.unipi.it
[§]Carleton University, email:sdobrev@site.uottawa.ca

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, 5, 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*, provides 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, 2, 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.

	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	$RV(\frac{k-2}{2})$
	$\forall k$	$G(k-2,1)$	$\forall k$	$G(k-2,1)$

Figure 1: Summary of possibility results.

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, 9, 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.

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].

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.

[STEFAN:] change here, might need to be refined. We can use also the 'latest agent to wake up', but the result is weaker and needs more work in adjusting the proofs and statements of lemmas and theorems. 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 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 latest *extremal* agent wakes-up, where the *extremal* agents are defined as follows:

- In the oriented case, there is single extremal agent – the left-most one, measured from the Bh.
- In 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 [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 and Lower Bounds

There are some basic obvious facts:

Theorem 2.1. *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.
1. Since the location of the back hole is unknown to the agents, the first agent that move could immediately disappear in the black hole and will never be able to gather.
2. Since there is no directional agreement, two agents could disappear in the black hole from both sides at their first move and will not be able to gather. □

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

Theorem 2.2. *If k is unknown, non-trivial rendez-vous requires locating the black hole.*

Proof. If the black hole is not located, then there can be unexplored nodes (i.e., nodes whose ports are all unexplored) that can contain agents not participating to the gathering. □

Furthermore,

Theorem 2.3. *If k is unknown, then $RV(k-1)$ cannot be solved.*

Proof. Partition the agents into two equal size groups A and B. Denote by R_A and R_B the regions of the ring explored by the agents A and B respectively. Since k is unknown, the region R_A must increase; w.l.o.g., let it first increase to the right (let x be the first node visited in this exploration). Because of symmetry, also B must first increase to the right (let $r(B)$ be the first node visited in this exploration).

If the next expansion of A is again to the right (towards node $r(x)$), the adversary slows down this expansion to a halt. Within finite time, some agent in A must be expanding to the left (towards node $l(A)$); the reason is that in a scenario where the agents in B did not exist and the black hole were indeed in $r(x)$, the agents in A would not be able to detect the black hole without proceeding in the opposite direction, contradicting Theorem 2.2. In fact, since the regions R_A and R_B are disjoint, for the agents in A such a scenario is indistinguishable from the one where the two groups exist.

Consider now the setting of A and B as shown in Figure **??**.

This means that at least one agent in A, moving towards $l(A)$ will enter the black hole. Consider now the execution of B; by symmetry the agents of B will behave in the same way as the agents in A, hence, an agent in B going right to $r(B)$ will also enter the black hole. □

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

Theorem 2.4. [8] *If n is not known, black hole location is unsolvable.*

it follows that

Theorem 2.5. *Either k or n must be known for non-trivial rendez-vous.*

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

Theorem 2.6. *Let n be known, and the ring be unoriented. The RV 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 RV 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 RV point for a total of at least $\sum_2^{\lfloor \frac{k}{2} \rfloor} (|d-i| + |n-d-1-i|)$ moves. Solving, we obtain that the number of moves is at least $\lfloor \frac{k}{2} \rfloor (n - \lfloor \frac{k}{2} \rfloor - 2)$. □

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 2.5, *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 2.4, we can *not* locate the black hole!

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

3.1.1 In Oriented Rings

Theorem 3.1. *$RV(k-1)$ 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 3.2. *Protocol* GoRight *solves* $RV(k-1)$.

Proof. The rightmost agent before the black hole will remain *explorer* and eventually enter the black hole. All 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. □

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

Lemma 3.3. *Protocol* GoRight *terminates in time at most* $3(n-2)$ *since the start of the leftmost agent.*

Proof. 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 rendezvous there. □

Lemma 3.4. *Protocol* GoRight *uses at most* $nk - k^2/2 + 2n + o(nk)$ *moves.*

Proof. 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 yields the lemma. □

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 3.1 now immediately follows from Lemmas 3.2 and 3.3.

3.1.2 Unoriented Rings

Since the ring is not oriented, by Theorem 2.1, $RV(k-1)$ 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 3.5.
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 3.6. *There is only one collection point.*

By construction of algorithm Gr-Odd we have

Lemma 3.7. *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 3.7, yields the following:

Lemma 3.8. *$k-2$ agents will gather in the collection point.*

Hence, by Lemmas 3.6 and 3.8, part (1) of Theorem 3.5 holds. Before proceeding with the proof of the other parts of Theorem 3.5, let us examine the time costs of Protocol Gr-Odd.

Theorem 3.9. *Protocol* Gr-Odd *terminates in time at most* $5(n-2)$.

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 get $t_1 \leq 3s$. Let t_2 be the time by which every edge non-incident to the black hole has been traversed by a collector. We get $t_2 \leq t_1 + 3(n-2-s) + s$, 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 get $t_3 \leq 5(n-2)$. □

Theorem 3.10. *Protocol* Gr-Odd *uses at most* $3nk/2 - k^2/4 + O(n) + o(nk)$ *moves.*

Proof. By Lemma 3.4 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. □

Consider now the case when k is even (recall k is known). To prove part (2) of Theorem 3.5 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 3.11. *If k is even then $RV(p)$ can not 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; denote by $\overline{v}$ 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 $\overline{v}$ (see Figure 2.a)); thus, to each agent a with homebase v corresponds an agent $\overline{a}$, of opposite color, with homebase $\overline{v}$. 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), where all agents start at the same time, and where the first move of the agents neighboring the Bh be directed towards the Bh. Observe now that, to any move in a given direction by an agent a it corresponds a move in the opposite direction by agent $\overline{a}$. In particular, whenever an agent a crosses the symmetry axis, agent $\overline{a}$ 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 $(k-2)/2$.

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 2.b)). Force the 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. □

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.

First, each set of agents executes the rendezvous algorithm GoRight for oriented rings, independently of and ignoring the agents of the other set, 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 generalization, 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:

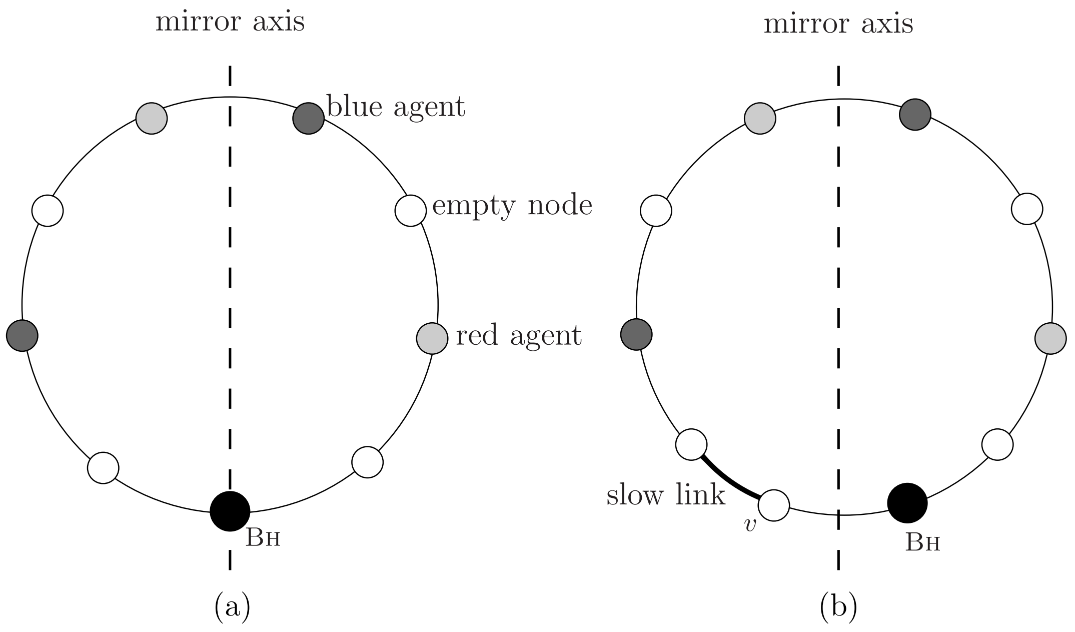

Figure 2: Mirror symmetric distribution for (a) n odd, and (b) n even.

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

Proof. If only one collection point was formed, the protocol Gr-Even is correct for the same reasons as protocol Gr-Odd.

If two collection points were formed then two collectors moving in opposite directions will eventually meet, decide on the final collection point (or edge) and inform about that all other followers. □

This completes the proof of Theorem 3.5. The time and movement costs of Protocol Gr-Even can be easily determined:

Theorem 3.13. *Protocol* Gr-Even *terminates in time at most* $5(n-2)$.

Proof. If only one collection point was formed, the reasoning is the same as in the proof of Theorem 3.9, and the time complexity is at most $5(n-2)$.

Assume now that two collection points were formed at a distance d from each other (traversing the part of the ring not containing BH). Let t_1 be the time when the last collection point was formed and let s be the number of edges explored at that moment. Let t_2 be the time when the two collectors met, and t_3 be the time when all agents gathered 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 $t_3 \leq 5(n-2)$. □

Theorem 3.14. *Protocol* GR-EVEN *uses at most* $3nk/2 - k^2/4 + o(nk)$ *moves.*

Proof. If there is single collection point, the proof of Theorem 3.10 applies directly. If there are two collection points, applying Lemma 3.4 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. □

3.2 RendezVous when k is unknown

An immediate consequence of the fact that k is unknown is that, by Theorem 2.5, the ring size n must be known for any non-trivial rendez-vous to be possible.

Another consequence is that, by Theorem 2.2, 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.

3.2.1 Oriented Rings

Theorem 3.15. *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.

We associate with each contiguous block of explored nodes a group of agents expanding that block until either (1) the explored block contains $n-1$ nodes (in which case a final *collection* phase is initiated, collecting the agents into a designated collection point) or (2) the block merges with a neighbouring 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 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 a *R-explorer*. As the blocks grow, they eventually touch and their agents are combined, maintaining the following rules:

- 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 a *R-shadow*.
- A four-agent block has two explorers and two shadows (one in each direction).
- Any additional agents (i.e. if merging two four-agent blocks) become *passive*.

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

The technique used can be seen as *aging* of agents from initial *R-explorer* through *L-explorer*, *R-shadow*, *L-shadow* and finally *passive*. The aim 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 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.

Protocol SHADOW:
Agent states: *R-explorer*, *L-explorer*, *R^*-shadow*, *R-shadow*, *L^*-shadow*, *L-shadow*, *passive*, *collector*. *collector* and *shadow* agents maintain counter *size*.

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 a *R-shadow*, set $size_a$ to 0 and become *L^*-shadow*. If the *R-shadow* is waiting at v, wake him up.

  If v has not been visited by a *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, travelled across the explored block and returned, and the *R-explorer* meanwhile did not explore a new node), wait until waken up by a *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*.

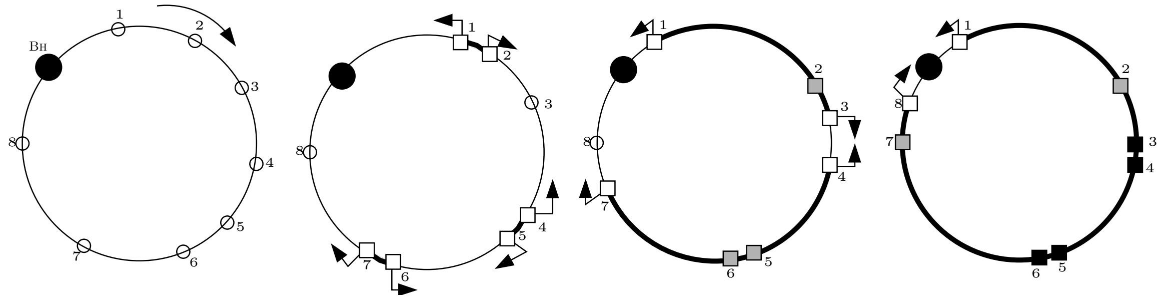

Figure 3: Protocol Shadow, where the ring is assumed to be oriented clockwise. The empty circles represent *active* agents; the white squares are the *explorers*, the grey squares the *shadows*, and the black squares the *passive* agents. The fat line evidences the segments delimited by the explorers. The numbers are placed only to clarify how the agents move, and are not used at all during the computation.

- ***L^*-shadow*:** Move left until the last safe node v is reached, decrementing $size_a$. If v has already been visited by a *L-shadow*, become *passive*. If the *L-shadow* is waiting at v, wake him up.

  If v has not been visited by a *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*.

Lemma 3.16. *Within finite 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 a *R-explorer* until it reaches the Bh. Any other *R-explorer* will eventually become *L-explorer*. The leftmost of those will remain *L-explorer* and will eventually enter the Bh, all others will become R^**-shadow*. Of those, the rightmost one will eventually become *R-shadow*, any remaining one will become L^**-shadow*. Finally, out of those, the leftmost one will become *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 *R-explorer*. If there were no agents to the left of v, the leftmost *L-explorer* would eventually explore it. □

Lemma 3.17. *Eventually, at least one agent will become* collector*.*

Proof. Consider the rightmost *R-shadow* a (since $k \geq 4$ and from the proof of Lemma 3.16 we know that there will be a *R-shadow*). From construction, a will remain *R-shadow* [1] 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 3.16 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 awaken by the *R-explorer* and will traverse the $n-1$ explored nodes to become 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 merging of two blocks, either a will detect that directly, or will be awaken by the *R^*-shadow* created by the collision of the blocks and re-scan to become *collector*. □

The proof of Theorem 3.15 now follows quite straightforwardly from the construction of *collector* agents.

Let us now examine the time costs of this protocol.

Theorem 3.18. *The protocol* Shadow *terminates in at most* $8(n-2)$ *time steps.*

Proof. 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 *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 the time step $\max(3s + 4(s_1 - s_0), 3(n - s_0 - 1) < 4(n-2)$. Note also that by time step $3(n - s_0 - 1)$ there is a single *R-explorer*, as all other *R-explorer* agents would have entered a node already visited by a *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 became 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 scan after time $4(n-2)$, it will detect that $n-1$ nodes have been explored and will become *collector*. The algorithm would then terminate in additional at most $n-2$ steps.

An unsuccessful scan (without becoming *collector*) takes at most $2(n-3)$ steps, therefore a successful scan would be initiated at time less then $6(n-2)$ and the algorithm would terminate by time $8(n-2)$. □

Let us stress that protocol Shadow solves the *black hole location* problem (by Lemma 3.17). 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.

Theorem 3.19. *The protocol* Shadow *locates the* Bh *using at most* $4n^2 + nk - k^2/2 + O(n) + o(k^2)$ *moves.*

Proof. 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 a *R-shadow* being awaken by a *R^*-shadow* or *R-explorer*. Note that there are at most $2n$ recharge events for all right shadows, as in each event

[1] more precisely, alternating between *R-shadow* and *R^*-shadow*

either a R^*-*shadow* becomes a L^*-*shadow*, or the *R-explorer* has made progress. Moreover, a *R-shadow* (or R^*-*shadow*) can make at most $2(n-2)$ moves between recharges – if it is not recharged, it turns into 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 3.10.

Summing together yields the theorem. □

3.2.2 Unoriented Rings

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

Theorem 3.20.
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 prove Theorem 3.20 constructively by presenting a protocol Un-Shadow. The main ideas of protocol Shadow carry over to protocol Un-Shadow, however, several technicalities have to be dealt with differently due to lack of agreement on orientation. Again, we colour the agents red a blue, according to their initial orientation. In general, the agents follow protocol Shadow ignoring the agents of the different colour, however these exceptions apply:

1. when a *R-explorer* comes to a node already visited by a *L-explorer* of the opposite colour, it becomes R^*-*shadow*.
2. when a *L-explorer* comes to a node already visited by a *R-explorer* of the opposite colour, it becomes 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 a L^*-*shadow* (and possibly awakes the shadow at v, regardless of its colour), otherwise a becomes *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 colour, the protocol Un-Shadow behaves exactly as protocol Shadow and its correctness follows. Therefore, in the rest we assume there is at least one agent of each colour. Let us also assume that red *R-explorer* moves to the right and blue *R-explorer* moves to the left.

Lemma 3.21. *Within finite 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.

W.l.o.g. assume the left side has at least two agents and both a_l and b_l are blue. b_l starts moving left as a *R-explorer* and when it enters the starting node of a_l, it will become *L-explorer* and start moving right. Therefore, v will be eventually explored either by b_l or by an agent coming from the right. □

Lemma 3.22. *Within finite time, an agent will become* 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 (ref *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 any more, because other shadows will age due to their rightmost/leftmost mark.

When all nodes become explored (Lemma 3.21 tells us that this will eventually happen), at least one of those extreme agents will be awaken, traverse the explored nodes and become *collector* (using the same arguments as in Lemma 3.17). □

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

Lemma 3.23. *There will be a unique collection point.*

Proof. If there is only one *collector*, there will be a single collection point – when the *collector* travelled $n-1$ nodes. If there are two collectors, when they meet, they can agree on the collection point (i.e. directly opposite the Bh). □

Corollary 3.24. $k-2$ *agents will gather in the collection point.*

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

Lemma 3.25. *If k is even and n is odd then $RV(p)$ can not be solved for $p > (k-2)/2$; however, $RV((k-2)/2)$ and $G(k-2,1)$ can be achieved.*

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.

From Lemma 3.22 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 $G(k-2,1)$ is achieved. □

Combining Lemmas 3.24 and 3.25 yields Theorem 3.20

Using the same approach as in the proof of Theorem 3.18 we get

Theorem 3.26. *The protocol* Un-Shadow *for unoriented rings terminates in at most* $8(n-2)$ *time steps.*

Since the Un-Shadow differs from the protocol Shadow only by accelerated aging and final collection, the proof of Theorem 3.19 carries almost without changes also for Un-Shadow.

Theorem 3.27. *The protocol* Un-Shadow *locates the* Bh *using at most* $4n^2 + nk - k^2/2 + O(n) + o(k^2)$ *moves.*

4 Conclusions

We need some real conclusions, so far we have just this short discussion how to extend the results. We have assumed that the agents are initially dispersed, with each agent in a different homebase.

If at the beginning there are two or more agents at the same node, we immediately group them and elect a "leader" among them (e.g. using the mutual exclusion in writing to the whiteboard); the others in the group will "follow the leader" (except on unsafe links).

With careful attention to details, all the proposed protocols can be modified so to handle this situation.

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.