Mobile Agents Rendezvous in Spite of a Malicious Agent

Shantanu Das[1], Flaminia L. Luccio[2(✉)], and Euripides Markou[3]

[1] LIF, Aix-Marseille University, Marseille, France
[2] DAIS, Università Ca' Foscari Venezia, Venezia, Italy
luccio@unive.it
[3] DIB, University of Thessaly, Lamia, Greece

Abstract. We examine the problem of rendezvous, i.e., having multiple mobile agents gather in a single node of the network. Unlike previous studies, we need to achieve rendezvous in presence of a very powerful adversary, a malicious agent that moves through the network and tries to block the *honest* agents and prevents them from gathering. The malicious agent can be thought of as a *mobile fault* in the network. The malicious agent is assumed to be arbitrarily fast, has full knowledge of the network and it cannot be exterminated by the honest agents. On the other hand, the honest agents are assumed to be quite weak: They are asynchronous and anonymous, they have only finite memory, they have no prior knowledge of the network and they can communicate with the other agents only when they meet at a node. Can the honest agents achieve rendezvous starting from an arbitrary configuration in spite of the malicious agent? We present some necessary conditions for solving rendezvous in spite of the malicious agent in arbitrary networks. We then focus on the ring and mesh topologies and provide algorithms to solve rendezvous. For ring networks, our algorithms solve rendezvous in all feasible instances of the problem, while we show that rendezvous is impossible for an even number of agents in unoriented rings. For the oriented mesh networks, we prove that the problem can be solved when the honest agents initially form a connected configuration without holes if and only if they can see which are the occupied nodes within a two-hops distance. To the best of our knowledge, this is the first attempt to study such a powerful and mobile fault model, in the context of mobile agents. Our model lies between the more powerful but static fault model of *black holes* (which can even destroy the agents), and the less powerful but mobile fault model of *Byzantine agents* (which can only imitate the honest agents but can neither harm nor stop them).

S. Das—This work has been partially supported by the ANR - MACARON project (anr-13-js02-0002).

F.L. Luccio—This work has been partially supported by the PRIN 2010 Project *Security Horizons*.

E. Markou—Part of this work has been done while this author was visiting Università Ca' Foscari Venezia. This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) — Research Funding Program: THALIS-NTUA (MIS 379414).

P. Bose et al. (Eds.): ALGOSENSORS 2015, LNCS 9536, pp. 211–224, 2015.
DOI: 10.1007/978-3-319-28472-9_16

Keywords: Asynchronous · Mobile agents · Rendezvous problem · Malicious agent

1 Introduction

One of the fundamental problems in distributed computing with mobile robots or agents is the problem of gathering all agents at a single location, known as the *rendezvous* problem. Rendezvous is important for example, for coordination between the agents or for sharing information or for planning a collaborative task. This problem has been well studied for the fault-free environment but there are very few results on solving rendezvous in the presence of faults, in particular, in the presence of a hostile entity that could prevent the agents from achieving their task. As in most previous works, we model the environment as a connected graph with multiple mobile agents moving along the edges of the graph; the objective is to gather them at a single node of the graph. In this context, the hostile entity may be either stationary (e.g. a harmful node in the graph) or mobile (e.g. a virus propagating on a network). Methods for protecting mobile agents from malicious host nodes have been proposed, e.g. based on the identification of the malicious host [14]. However, the issue of protecting a network (hosts and mobile agents) from a malicious and mobile entity is still wide open (see [18] and references therein).

A model for a particularly harmful node which has been extensively studied is the *black hole*, where a node which contains a stationary process destroys all mobile agents upon visiting it, without leaving any trace. In this case, although the hostile entity is very powerful, it is stationary; the mobile agents can simply avoid the black hole once its location is known. Thus, the main issue is locating the black hole [14,17,19]. Locating and avoiding a malicious entity that is also mobile and moves from node to node of the graph, seems to be a more difficult problem. A recent result considers the problem of rendezvous in the presence of *Byzantine agents* [12]. A Byzantine agent is indistinguishable from the legitimate or *honest* agents, except that it may behave in an arbitrary manner and may provide false information to the honest agents in order to induce them to make mistakes, thus preventing the rendezvous of the honest agents. Thus, the issue here is identifying the Byzantine agents and distinguishing them from the honest agents. Note that the Byzantine agent cannot actively harm the agent or physically prevent the agents from gathering. In this paper, we consider a more powerful adversary called a *malicious agent* which can actively block the movement of an honest agent to the node occupied by the malicious agent. For example, when two honest agents are close to each other, the malicious agent can enter the path between the two agents and prevent them from meeting. We investigate the feasibility of rendezvous in the presence of such a powerful adversary. In particular, the malicious agent is more powerful than the honest agents; it can move arbitrarily fast through the graph, has full information about the current configuration (i.e. the graph and location of the agents), and has knowledge of the next action to be taken by each honest agent. On the other hand, the honest

agents are relatively weak; they are anonymous finite automata, they move asynchronously without any prior knowledge of the graph and they can communicate only locally on meeting another agent at the same node. We remark here that the malicious agent is distinguishable from the honest agents, so the question of identifying the malicious agent (as in Dieudonne et al. [12]), does not arise here.

We believe this is an interesting model for studying mobile faults in a graph, that has never been considered before. In some sense this model can be seen as an extension of the model of networks with delay faults. For example, Chalopin et al. [8] consider the problem of rendezvous in the presence of an adversary that can prevent an agent from moving for an arbitrary but finite time. In their case, the agent cannot be blocked forever as in our scenario. Our model can also be contrasted with the model for network decontamination or, cops and robbers search games on graphs, where a team of good agents (called cops) tries to capture a fast fugitive (robber). The fugitive or hostile entity is exterminated as soon as one of the cops reaches it. Thus the behavior of the hostile entity, in this case, is opposite to that of the malicious agent in our model – instead of blocking the honest agents, the hostile entity tries to get away from the good agents.

In terms of practical motivation for this research, we can think of the malicious agent as representing a virus that may spread around the network. While in the classical decontamination problem the aim is to extinguish the virus, in our setting the virus cannot be extinguished and has to be contained in one part of the network, thus dividing the network into *unstrusted* and *trusted* subnetworks. This scenario can be compared to the problem of *botnets*, i.e. a subnet of compromised computers (bots), typically used for denial-of-service attacks on the internet. The untrusted subnetwork in our model can be seen as a botnet, and the *botmaster* who controls the bots represents the malicious agent. An honest agent that resides on a node protects the trusted network from the untrusted one by running some protection mechanism (e.g. a firewall, an intrusion detection mechanism, etc.). Thus the malicious agent cannot enter a node already occupied by an honest agent. On the other hand the botnet is dynamic, and it may reduce its dimension (i.e., when the botmaster leaves the host) or it may increase it only on hosts not occupied, i.e., not protected by an agent. Honest agents may expand towards the untrusted hosts which are not controlled by the botmaster anymore by running botnet detection mechanisms (see, e.g., [25]). We are then interested in solving the rendezvous problem in the trusted subnetwork, and we want to study how this malicious behaviour affects the solvability of the *Rendezvous* problem.

Related Work: The rendezvous problem has been studied for agents moving on graphs [2] or for robots moving on the plane [9], using either deterministic or randomized algorithms. In the fault-free scenario, the rendezvous problem can be solved relatively easily, even in asynchronous networks, when the network has an asymmetry (e.g., a distinguished node), and can be explored by the agents, since the mobile agents can simply be instructed to meet at such a distinguished node. However, this is not the case for symmetric networks, or when

the agents is incapable of visiting all nodes of the network, and the rendezvous problem in such settings is non-trivial and not always solvable even in simple topologies such as the ring network [21]. Symmetry-breaking for the rendezvous problem can be achieved by attaching unique identifiers to the agents (see, e.g., [10,24]), or in the anonymous case using tokens as in e.g., [6,11]. With respect to hostile environments, the *Rendezvous* problem has been studied when there is a black hole or other stationary faults in the network [7,13,23]. Another model for hostile nodes has been presented in [3,20], where the authors have studied how a more severe (than a black hole) behaviour of a malicious host affects the solvability of the *Periodic Data Retrieval* problem in asynchronous networks. A well studied problem in the context of a mobile adversary is the problem of graph searching where a team of mobile agents has to decontaminate the infected sites and prevent any reinfection of cleaned areas. This problem is equivalent to the one of capturing a fast and invisible fugitive moving in the network. For results on this and related problems see, e.g., [4,15,16,22].

Gathering of mobile agents has been also studied in the plane when there are faulty agents which may crash [1,5] and in networks with delay faults [8] or in the presence of Byzantine agents [12], as mentioned before. However, to the best of our knowledge, the rendezvous problem has never been studied under the presence of hostile agents that may block other agents from having access to parts of the network.

Our Results: In this paper we consider a network modelled as a connected undirected graph with multiple honest agents located at distinct nodes of the graph. There is also a hostile entity which is mobile, called the malicious agent. It cannot harm the honest agents but can prevent them from visiting a node: an honest mobile agent cannot visit a node which is occupied by a malicious agent and vice versa. We are interested in solving the rendezvous of all honest agents in this hostile environment. Our objective is to study the feasibility of rendezvous with minimal assumptions. Thus, we consider the weakest possible model for the honest agents. The honest agents are finite state automata with local communication capability and having no prior knowledge of the network. In Sect. 2 we show some configurations in which the problem is unsolvable and we discuss properties that must be respected by any correct algorithm for the problem. For the rest of the paper, we consider ring and mesh networks – two topologies that can be explored even by a finite automaton. In Sect. 3 we present a rendezvous algorithm for ring networks. For oriented rings, we have a universal algorithm that achieves rendezvous starting from any initial configuration, despite the existence of a malicious agent. We prove that the problem is unsolvable for any even number of agents in unoriented rings. Finally, we present an algorithm for rendezvous of any odd number of agents in unoriented rings, thus solving the problem in all solvable instances. In Sect. 4 we consider oriented mesh topologies and we prove that the problem can be solved when the agents initially form a connected configuration without holes if and only if they can detect which are the occupied nodes within a distance of two hops. We show that this latter capability is necessary to achieve rendezvous even for connected configurations

without holes. We conclude in Sect. 5 with a discussion about future research directions for this new model. For space limitation, proofs of some lemmas and theorems have been omitted; these can be found in the full version of the paper.

2 Preliminaries

2.1 Our Model

We represent the network by a graph $G = (V, E)$ composed by $|V| = n$ anonymous nodes or *hosts* and $|E|$ edges or connections between nodes. Each host is connected to other hosts by bidirectional asynchronous FIFO links (i.e., an agent cannot overtake another agent moving in the same edge), and it is capable of serving agents by a mutual exclusive mechanism (i.e., an agent at a node u must finish its computation and move or decide to stay, before any other agent at u starts its computation or another agent visits u). The links incident to a host are distinctly labelled but this port labelling (unless explicitly mentioned), is not globally consistent. In the network there are some *mobile agents* which are independent computational processes with some constant internal memory. The agents are initially scattered in the network (i.e., at most one agent at a node), and can move along its edges. An agent arriving at a node u, learns the label of the incoming port, the degree of u and the labels of the outgoing ports. We assume there are k *honest* anonymous identical agents $A_1, A_2, \ldots A_k$, and one *malicious* agent M which may deviate from the proper operations. The initial locations of the honest and malicious agents are decided by an adversary. We describe below the capabilities and behaviour of honest and malicious agents.

Honest Agents: An honest agent located at a node u can see all other agents at u (if any), and can also read their states. It can also read the degree of u and the labels of the outgoing ports. The agents are anonymous, cannot exchange messages and cannot leave messages at nodes. They are identical finite state automata, hence they have some constant memory. The agents do not know n and k. Two agents travelling on the same edge in different directions do not notice each other, and cannot meet on the edge. Their goal is to rendezvous at a node.

Malicious Agent: We consider a worst case scenario in which the malicious agent M is a very powerful entity compared to honest agents: It can move arbitrarily fast inside the network (since the model is asynchronous and the adversary is combined with the malicious agent) and it can permanently 'see' the positions of all the other agents. It has unlimited memory and knows the transition function of the honest agents. When it resides at a node u it prevents any honest agent A from visiting u (i.e., it "blocks" A): if an agent A attempts to visit u it receives a signal that M is in u (botnet detection) and in that case we say that A *bumps* into M. The malicious agent can neither visit a node which is already occupied by some honest agent, nor cross some honest agent in a link. It also obeys the FIFO property of the links (i.e., it cannot overpass an honest agent which is moving on a link).

We call a node u *occupied* (respectively, *free* or *unoccupied*) when one or more (no) *honest* agents are in u. We notice here that some of our impossibility results hold even for stronger models, e.g., when honest agents have unlimited memory, distinct identities, knowledge about the size of the network, visibility, etc. Our algorithm for the ring topology only requires the capabilities of the honest agents mentioned above while for the mesh topology we assume that the honest agents also have the ability to scan whether a node within a two-hops distance, is occupied or not.

2.2 Basic Properties

In this section we show a special class of configurations for which the problem is unsolvable. Intuitively, those are configurations in which the malicious agent can keep separated at least two agents forever.

Definition 1. *Let $\mathcal{C}$ be a configuration of a number of agents in a graph G with a malicious agent. The configuration $\mathcal{C}$ is called* **separable** *if there is a connected vertex cut-set F composed of free nodes which, when removed, disconnects G so that not all occupied nodes are in the same connected component.*

Lemma 1. *Rendezvous is impossible for any initial configuration in a graph G which is separable, even if the agents have unlimited memory, distinct identities and can always see their current configuration.*

Proof. Let $\mathcal{C}$ be an initial configuration which is separable, and let F be a connected vertex cut-set, whose removal disconnects G so that not all occupied nodes are in the same connected component. Let u, v be two occupied nodes which are in different connected components of G and let A, B be the honest agents located at u, v respectively. Due to asynchronicity an adversary can introduce delays to A's and B's movements while at the same time the malicious agent, which has been initially placed at a node in F, can move everywhere in F (since F has only free nodes and it is connected) preventing agents A, B, from visiting any node in F. Since all paths between u and v include at least one node of F, agents A, B can never meet, no matter how powerful they are. □

Hence for every initial separable configuration the problem is unsolvable. A natural question is whether there are non-separable initial configurations for which the problem is unsolvable. The answer is yes and one can easily find such configurations. We now state sufficient conditions under which the problem is unsolvable for a separable (initial or not) configuration of agents.

Definition 2. *Let $\mathcal{C}_t$ be a configuration at time $t \geq 0$ (i.e., initial or not) of a number of agents in a graph G with a malicious agent. The configuration $\mathcal{C}_t$ is called* **separating** *if $\mathcal{C}_t$ is separable and either $\mathcal{C}_t$ is an initial configuration or the following conditions hold:*

- *there is a node $x_m \in F_t$ (F_t is any vertex cut-set of $\mathcal{C}_t$ as defined in Definition 1) and a path of nodes $(x_0, x_1, \ldots, x_m)$ so that x_0 is free at time 0 and,*

- *the sequence of nodes* $(x_0, x_1, \ldots, x_m)$ *can be partitioned in* $k \leq t+1$ *contiguous subsequences* $(x_0^0, \ldots, x_i^0), (x_{i+1}^1, \ldots, x_j^1), \ldots, (x_{l+1}^k, \ldots, x_m^k)$*, where* $0 \leq i < j < l < m$ *and,*
- *the nodes* $(x_w^s, \ldots, x_r^s)$ *belonging to the same subsequence* s *are free at time* s*, where* $0 \leq s \leq k$ *and nodes* $(x_w^k, \ldots, x_r^k)$ *are free at time* t*.*

Lemma 2. *Rendezvous is impossible for any* separating *configuration in a graph* G*, even if the agents have unlimited memory, distinct identities and can always see their current configuration.*

Intuitively, Lemma 2 states that if $\mathcal{C}_t$ is a separable configuration, and in $\mathcal{C}_t$ there is a free node x so that either: (i) x has been always free or, (ii) there are paths of nodes which eventually become free and they form a connection between a free node at $\mathcal{C}_0$ and x, then there are at least two agents in $\mathcal{C}_t$ which will never meet. Hence, any correct algorithm for the solution of the problem should avoid creating a separating configuration.

3 Rendezvous in a Ring Network

In this section we will study the rendezvous problem in bidirectional rings with a malicious agent M. Notice that in a ring topology there are no separable (and hence no separating either) configurations, since there cannot exist a connected cut-set composed of free nodes whose removal would disconnect the ring. However, since the ring is highly symmetric, rendezvous is impossible even if the agents have unlimited memory and have full knowledge of the configuration, since an adversary can keep synchronized the agents so that they always take the same actions at the same time and therefore they maintain their initial distances (the malicious agent can keep on moving synchronized with the honest agents). Thus, in order to solve the problem we need to add some constraints to the model. A natural step is to assume that there is a special node labeled o^* in the ring which can be recognized by the agents. Note that the malicious agent is so powerful that it could place itself on o^* and never move from there. Our strategies also work under this scenario. We now present algorithms for solving the problem in oriented and unoriented rings.

3.1 Oriented Ring

In an oriented ring, the two incident edges at each node are labelled as clockwise or counter-clockwise in a consistent manner; so, all agents agree on the ring orientation.

The idea of the algorithm is the following. Each agent moves in the clockwise direction until it meets o^* or bumps into M. For the first three times that the agent bumps into M without meeting o^*, it reverses its direction and continues moving in the opposite direction. Due to the FIFO property and the fact that the agent cannot pass over M, we can show that if an agent bump into M after reversing directions at least three times, then the other agents should

Algorithm 1. RV-OR : Rendezvous of $k \geq 2$ agents in oriented rings

```
Let i := 0;
DIR := Clockwise;
while not Stopped do
    Move DIR until you bump into M or meet o* or a stopped agent;
    i=i+1;
    if you met a Stopped agent then
        Become Stopped and Exit loop;
    if i = 1 or i = 2 then
        if Current node is o* then
            Become Stopped and Exit loop;
        else if Bumped into M then
            Reverse direction (DIR := inverse(DIR));
    if i = 3 then
        if Current node is o* or Bumped into M then
            Reverse direction (DIR := inverse(DIR));
    if i = 4 then
        if Current node is o* or Bumped into M then
            Become Stopped and Exit Loop ;
```

have bumped into M at least twice, without meeting the special node o^* (see Lemma 3). After an agent has already bumped into M three times, the next time it bumps into M or meets o^* it stops. On the other hand, if the agent meets o^* before it bumps into M twice, then the agent stops at o^*, and all the other agents will arrive at o^* after bumping into M at most once. The algorithm called RV-OR is presented below.

Lemma 3. *During the execution of the algorithm, if an agent bumps into M in the fourth iteration of the* while *loop, then any other agent must have bumped into M at least two times.*

Lemma 4. *Algorithm RV-OR solves rendezvous of $k \geq 2$ agents in spite of one malicious agent, in any oriented ring containing one special node o^*.*

3.2 Unoriented Rings

In unoriented rings, each agent has its own notion of clockwise and the agents may not agree on the clockwise direction. In this case rendezvous is not always feasible.

Lemma 5. *For any even number $k \geq 2$, the rendezvous problem for k honest agents and one malicious agent cannot be solved in any bidirectional unoriented anonymous ring with a special node o^*, even if the agents know k.*

We now present an algorithm for solving rendezvous of k agents, for any *odd* integer k, in an unoriented asynchronous ring network. Notice that in an unoriented ring, if we follow an algorithm similar to Algorithm RV-OR it is possible that the agents form two distinct groups that gather at two distinct nodes. However, since the total number of agents is odd, exactly one of the two groups would have even number of agents, thus one of the agents of this group could move to collect all the other agents. The algorithm must ensure that there are at most two groups of agents, i.e. there are at most two distinct nodes where the agents stop in the initial phase. In our algorithm, an agent stops at o^* only if it has seen it at least three times, while moving in the same direction. This implies that this agent has traversed the complete ring two times and while M has moved at least once around the ring. So, there could be no agents moving in the opposite direction. On the other hand if some agent stops while bumping into M, then any agent moving in the same direction would reach this node with the stopped agent before reaching M or o^*. In all cases, there will be at most two nodes where the agents stop. When two or more agents have gathered at a node v, one of the agents called the *searcher*[1] reverses direction and moves to search for the other agents. The searcher only stops when it reaches the other node w containing stopped agents. If the number of agents gathered at node w is even then the searcher becomes a *Collector* and it collects all agents and returns to node v. Note that the agent does not need to count the number of other agents as the algorithm depends only on the parity of the size of the group of agents. The complete algorithm, called RV-UR is presented in a following table.

Lemma 6. *Consider an anonymous ring consisting of n nodes, including a special node o^* and one malicious agent. If $k \geq 2$ honest agents execute Algorithm* RV-UR*, then, after a total number of $O(kn)$ edge traversals the honest agents correctly rendezvous, if k is odd.*

The following result summarizes the results of this section:

Theorem 1. *In any anonymous and asynchronous ring with a special node o^* and one malicious agent, k honest agents having constant memory and no knowledge about their number, can solve the rendezvous problem if and only if either the ring is oriented or k is odd.*

We briefly consider the case when there could be multiple malicious agents in the network. In this case, rendezvous is feasible only if all the malicious agents are located in a continuous segment of the ring with no honest agent in between. This scenario is equivalent to the one with a single malicious agent and thus the same algorithm would work in this case.

4 Rendezvous in an Oriented Mesh Network

We now study the problem in an oriented mesh network. In view of Lemma 1, rendezvous is impossible for separable initial configurations. Hence, in this section

[1] We select as searcher the second agent that arrives at node v.

Algorithm 2. RV-UR : Rendezvous in unoriented rings

Case 0. *Initial* state
Move clockwise until:
Case 0.1. You meet node o^* unoccupied for the third time:
Change state to *stopper*;
Case 0.2. You bump into M trying to move from a node that hosts only you:
Change state to *stopper*;
Case 0.3. You meet an agent not at node o^*:
Case 0.3.1. The agent you meet is alone and is a *stopper*:
Change state to *transformer-1*;
Case 0.3.2. Every other agent at the node is at state *final*:
Change state to *stopper*;
Case 0.3.3. You meet a *stopper* and at least one agent at state *final*:
Change state to *transformer-2*;

Case 1. State *transformer-1*
Wait until all other agents change to state *final*;
Change state to *searcher*;

Case 2. State *searcher*
Move counter-clockwise until you bump into M while you try to move from a node u:
Case 2.1. You see one or more agents at u and all of them are at state *final*:
Change state to *stopper*;
Case 2.2. You see no agent at u or an agent not at state *final*:
Change state to *collector*;

Case 3. State *stopper*
Wait until:
Case 3.1. You see a *transformer-1* or *transformer-2*: Change state to *final*;
Case 3.2. You see a *collector*: Follow *collector*;
Case 3.3. You see a *terminator*: Change state to *terminator*;

Case 4. State *collector*
Wait until every other agent at the node changes its state to *stopper*;
Collect everyone;
Move clockwise collecting every agent you meet, until you meet an agent at state *final*;
Change state to *terminator*;

Case 5. State *final*
Wait until:
Case 5.1. You see a *collector*: Change state to *stopper*;
Case 5.2. You see a *terminator*: Change state to *terminator*;

Case 6. State *transformer-2*
Wait until every other agent at the node changes its state to *final*;
Change state to *final*;

Case 7. State *terminator*
Wait until every other agent at the node changes its state to *terminator*;
Exit;

we study the problem for a special class of non separable initial configurations and we give an algorithm that solves the problem for this type of configurations. In particular, we focus on initial configurations where the induced subgraph of

the occupied nodes is connected without holes, i.e., there is no connected set of unoccupied nodes surrounded by occupied nodes. At the end of the section we discuss the solvability of the problem in other classes of initial non separable configurations.

First observe that even in configurations that consist of a simple path of occupied nodes, the problem is unsolvable in the considered model due to network asynchronicity: Initially all agents have the same input and thus (following any potentially correct algorithm), they should all try to move; however, an adversary may slowdown all agents, except for one not located at the endpoints of the path, hence creating a separating configuration. Thus, by Lemma 2 the problem is unsolvable. Therefore, the agents need to be able to gain some knowledge about their current configuration before they move in order to avoid creating a separating configuration. We enhance our model by giving the agents, the capability to discover all occupied nodes within a distance of d-hops.

Definition 3. *We say that an agent A located at a node x* can see (or scan) at a distance *d or it has* d-visibility *if A can decide for any node u within a distance of d hops from x, whether u is occupied or not by an honest agent.*

We emphasize that, if a node u scanned by agent A is occupied, A cannot tell how many agents are in u, or read their states. When the agents have a $d-$visibility capability we assume that moves are instantaneous, i.e., an agent cannot be traveling along an edge while another agent is scanning its neighbourhood. Unfortunately, as we show below, even when the agents have $1-$visibility (i.e., they can only scan their neighbours), the problem remains unsolvable for some connected without holes configurations.

Lemma 7. *The rendezvous problem is unsolvable in an oriented mesh with a malicious agent for initial connected without holes configurations, even when the agents are capable of scanning their adjacent nodes.*

Hence we further equip the agents with the capability of discovering the occupied nodes within a two-hops distance. In that case, as we show below, the problem can be solved for any connected without holes initial configuration.

We present an algorithm which instructs the agents to move only to occupied nodes in a way that they maintain the connectivity and they do not create holes. In order to describe the algorithm we define eleven local configurations as shown in Fig. 1. In these configurations, empty circles represent free nodes, while circles containing black dots represent occupied nodes. The remaining vertices on the figures represent nodes which may be either occupied or free. The agent (let us call it A) which is located below a horizontal arrow in cases (a-g), moves horizontally as depicted by the arrow. The agent (let us call it B) which is located left of a vertical arrow in cases (h-m), moves vertically as depicted by the arrow. Hence the algorithm can be described as follows:

Algorithm RV-Mesh: *If an agent has a view (within two hops) like the one of agent A or B described before, then this agent moves towards the direction shown by the corresponding arrow; otherwise the agent does not move.*

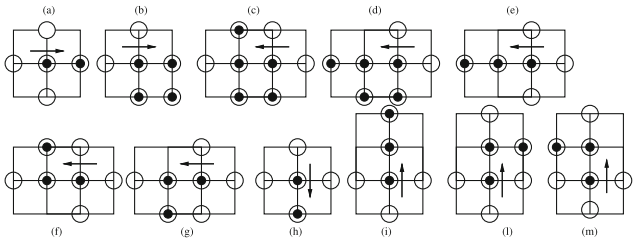

Fig. 1. View of the scanning agent located below (cases a-g) or left (cases h-m) of the depicted arrow. Occupied nodes are depicted as cycles containing black dots, while free nodes are depicted as empty cycles. Nodes which are within two hops from the scanning agent but not shown, can be either occupied or free. The scanning agent will move East in cases (a, b), West in cases (c, d, e, f, g), South in case (h), and North in cases (i, l, m).

Nodes which are within two hops from the scanning agent and are not shown in those configurations can be either occupied or free. If the location of the scanning agent is close to the border of the mesh and some of the nodes in those eleven configurations do not exist, then the agent acts as it would act if those nodes existed in its view and were free. Moreover, while an agent A located at a node u is executing its scan or compute phase then no other operation can take place at u before A moves or decides to stay (i.e., no other agent at u can start scanning and no other agent can arrive at u). That is, operations at a node u are executed in mutual exclusion. Notice that if two adjacent agents want to swap positions they can only do it at the same time.

Lemma 8. *Given an $n \times m$ oriented mesh, for any connected configuration without holes of at least three occupied nodes, there is at least one agent whose view is in one of the configurations depicted in Fig. 1.*

Lemma 9. *Given an $n \times m$ oriented mesh, consider a connected configuration of k agents in two occupied nodes. According to Algorithm* RV-Mesh*, after a total number of at most $k+1$ edge traversals there will be only one occupied node.*

Lemma 10. *Given an $n \times m$ oriented mesh, consider any connected configuration without holes of k agents occupying at least 3 nodes. After any number of moves according to Algorithm* RV-Mesh*, the resulting configuration is also connected without holes. Furthermore, the number of occupied nodes will strictly decrease after at most k edge traversals, reaching the value of only one occupied node after at most $O(k^2)$ edge traversals.*

In view of Lemmas 7, 8, 9 and 10 we have:

Theorem 2. *The rendezvous problem for $k \geq 2$ agents can be solved for any initial connected without holes configuration of agents in an $n \times m$ oriented mesh if and only if the agents are able to discover the occupied nodes within a distance of two-hops.*

If the initial non separable configuration is different from the one considered above, then even the 2-visibility capability is not sufficient anymore to solve rendezvous. In fact the problem remains unsolvable for connected configurations with holes even when the agents are able to discover the occupied nodes within any constant distance. The problem is also unsolvable for some disconnected non separable configurations. Hence it appears that for many initial non separable configurations in an oriented mesh, the combination of the asynchronicity and the limited view (to any constant fraction of the complete view) makes the problem unsolvable.

5 Conclusion

In this paper we studied deterministic protocols for the rendezvous of $k \geq 2$ honest agents in asynchronous networks with a malicious agent which can prevent the agents from reaching any node it occupies. We have presented algorithms for oriented and unoriented ring networks which gathers the honest agents within $O(kn)$ edge traversals for all feasible instances of the problem. We have also presented a deterministic protocol for oriented $n \times m$ meshes which leads the agents to rendezvous within $O(k^2)$ edge traversals for any initial connected without holes configuration when the agents can discover the occupied nodes within a distance of two-hops (which is a necessary condition). Given the novelty of the model there are many interesting open questions. The first is whether the problem can be solved in unoriented meshes for connected configurations without holes when the agents are capable of scanning within a constant distance. It would be also interesting to study randomized protocols for some of the unsolvable cases, and also to study this problem in synchronous networks with unit-speed cooperating agents and unit-speed/infinite-speed malicious agents. Finally, it would be interesting to study the problem in $(m+1)$-connected graphs in the presence of m malicious agents, or in the solved cases presented in this paper in the presence of malicious agents that show a more severe behaviour.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM J. Comput. **36**(1), 56–82 (2006)
2. Alpern, S., Gal, S.: Searching for an agent who may or may not want to be found. Oper. Res. **50**(2), 311–323 (2002)
3. Bampas, E., Leonardos, N., Markou, E., Pagourtzis, A., Petrolia, M.: Improved periodic data retrieval in asynchronous rings with a faulty host. In: Halldórsson, M.M. (ed.) SIROCCO 2014. LNCS, vol. 8576, pp. 355–370. Springer, Heidelberg (2014)
4. Barriere, L., Flocchini, P., Fomin, F.V., Fraigniaud, P., Nisse, N., Santoro, N., Thilikos, D.: Connected graph searching. Inf. Comput. **219**, 1–16 (2012)
5. Bouzid, Z., Das, S., Tixeuil, S.: Gathering of mobile robots tolerating multiple crash faults. In: IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8–11 July 2013, Philadelphia, Pennsylvania, USA, pp. 337–346 (2013)

6. Chalopin, J., Das, S.: Rendezvous of mobile agents without agreement on local orientation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 515–526. Springer, Heidelberg (2010)
7. Chalopin, J., Das, S., Santoro, N.: Rendezvous of mobile agents in unknown graphs with faulty links. In: Pelc, A. (ed.) DISC 2007. LNCS, vol. 4731, pp. 108–122. Springer, Heidelberg (2007)
8. Chalopin, J., Dieudonné, Y., Labourel, A., Pelc, A.: Fault-tolerant rendezvous in networks. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part II. LNCS, vol. 8573, pp. 411–422. Springer, Heidelberg (2014)
9. Cohen, R., Peleg, D.: Convergence of autonomous mobile robots with inaccurate sensors and movements. SIAM J. Comput. **38**, 276–302 (2008)
10. Czyzowicz, J., Labourel, A., Pelc, A.: How to meet asynchronously (almost) everywhere. In: Proceedings of 21st Annual ACM-SIAM Symposium on Discrete Algorithms (2010)
11. Das, S., Mihalák, M., Šrámek, R., Vicari, E., Widmayer, P.: Rendezvous of mobile agents when tokens fail anytime. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) OPODIS 2008. LNCS, vol. 5401, pp. 463–480. Springer, Heidelberg (2008)
12. Dieudonne, Y., Pelc, A., Peleg, D.: Gathering despite mischief. ACM Trans. Algorithms **11**(1), 1 (2014)
13. Dobrev, S., Flocchini, P., Prencipe, G., Santoro, N.: Multiple agents rendezvous in a ring in spite of a black hole. In: Papatriantafilou, M., Hunel, P. (eds.) OPODIS 2003. LNCS, vol. 3144, pp. 34–46. Springer, Heidelberg (2004)
14. Dobrev, S., Flocchini, P., Prencipe, G., Santoro, N.: Mobile search for a black hole in an anonymous ring. Algorithmica **48**(1), 67–90 (2007)
15. Flocchini, P., Huang, M.J., Luccio, F.L.: Decontamination of chordal rings and tori using mobile agents. Int. J. Found. Comput. Sci. **3**(18), 547–564 (2007)
16. Flocchini, P., Huang, M.J., Luccio, F.L.: Decontamination of hypercubes by mobile agents. Networks **3**(52), 167–178 (2008)
17. Flocchini, P., Ilcinkas, D., Santoro, N.: Ping pong in dangerous graphs: optimal black hole search with pebbles. Algorithmica **62**(3–4), 1006–1033 (2012)
18. Flocchini, P., Santoro, N.: Distributed security algorithms for mobile agents. In: Cao, J., Das, S.K. (eds.) Mobile Agents in Networking and Distributed Computing, Chap. 3, pp. 41–70. Wiley, Hoboken (2012)
19. Klasing, R., Markou, E., Radzik, T., Sarracco, F.: Hardness and approximation results for black hole search in arbitrary graphs. TCS **384**(2–3), 201–221 (2007)
20. Královič, R., Miklík, S.: Periodic data retrieval problem in rings containing a malicious host. In: Patt-Shamir, B., Ekim, T. (eds.) SIROCCO 2010. LNCS, vol. 6058, pp. 157–167. Springer, Heidelberg (2010)
21. Kranakis, E., Krizanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. Synthesis Lectures on Distributed Computing Theory. Morgan and Claypool Publishers, San Rafael (2010)
22. Luccio, F.L.: Contiguous search problem in sierpinski graphs. Theory Comput. Syst. **44**, 186–204 (2009)
23. Yamauchi, Y., Izumi, T., Kamei, S.: Mobile agent rendezvous on a probabilistic edge evolving ring. In: ICNC, pp. 103–112 (2012)
24. Yu, X., Yung, M.: Agent rendezvous: a dynamic symmetry-breaking problem. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 610–621. Springer, Heidelberg (1996)
25. Zeng, Y., Hu, X., Shin, K.: Detection of botnets using combined host- and network-level information. In: IEEE/IFIP DSN 2010, pp. 291–300, June 2010