

Perpetual Exploration of a Ring in Presence of Byzantine Black Hole

Pritam Goswami ✉ 


Sister Nivedita University, Kolkata, India

Adri Bhattacharya ✉ 

Indian Institute of Technology Guwahati, India

Raja Das ✉ 

Jadavpur University, Kolkata, India

Partha Sarathi Mandal ✉ 

Indian Institute of Technology Guwahati, India

Abstract

Perpetual exploration stands as a fundamental problem in the domain of distributed mobile agent algorithms, where the objective is to ensure that each node within a graph is visited by at least one agent infinitely often. While this issue has received significant attention, particularly concerning ring topologies, the presence of malicious nodes, referred to as black holes, adds more complexity. A black hole can destroy any incoming agent without leaving any trace of its existence.

In [2, 19], the authors have considered this problem in the context of *periodic data retrieval*. They introduced a variant of a black hole called gray hole (where the adversary chooses whether to destroy an agent or let it pass) among other variants, and showed that 4 asynchronous and co-located agents are necessary and sufficient to solve the periodic data retrieval problem (hence perpetual exploration) in the presence of such a gray hole if each of the nodes of the ring has a whiteboard.

This paper investigates the exploration of a ring by introducing a realistic variant of a gray hole, called a “Byzantine black hole”. In addition to the usual capabilities of a gray hole, the adversary can also choose whether to erase any previously stored information on that node.

Note that in [2, 19], this problem was considered with only one particular initial scenario (i.e., agents are initially co-located) and one specific communication model (i.e., whiteboard). Now, there can be many other initial scenarios where all agents might not be co-located (i.e., they may be scattered). Also, there are many weaker communications models such as *Face-to-Face* and *Pebble*, where this perpetual exploration problem is yet to be investigated in the presence of a Byzantine black hole.

The main results of our paper focus on minimizing the number of agents while guaranteeing that they perform the perpetual exploration on a ring even in the presence of a Byzantine black hole under different communication models and for different starting scenarios. On the positive side, as a byproduct of our work, we achieved a better upper and lower bound result (i.e., 3 agents) for perpetual exploration in the presence of a Byzantine black hole (which is a more generalized version of a gray hole), by trading-off the scheduler capability, when the agents are initially co-located, and each node contains a whiteboard.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Mobile Agents, Exploration, Ring, Black Hole, Malicious host, Byzantine Fault

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2024.17

Related Version *Full Version*: <https://arxiv.org/pdf/2407.05280> [17]

Funding *Adri Bhattacharya*: Adri Bhattacharya: Supported by CSIR, Govt. of India, Grant Number: 09/731(0178)/2020- EMR-I.



© Pritam Goswami, Adri Bhattacharya, Raja Das, and Partha Sarathi Mandal;
licensed under Creative Commons License CC-BY 4.0

28th International Conference on Principles of Distributed Systems (OPODIS 2024).

Editors: Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni; Article No. 17; pp. 17:1–17:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Exploring a set of nodes in a network is one of the fundamental tasks in the domain of distributed computing by mobile agents, formulated in the year of 1951 by Shannon [21]. Now, the security of these mobile agents while exploring these networks is one of the important issues that needs to be addressed. Among all the possible security threats that are addressed yet in literature, two among them are the most prominent, specifically, the threats from a *malicious agent* [18] and the threats from a *malicious host* [13]. In this paper, we are interested in the latter case, where the threats are from a malicious host. This host is a stationary node in the network, that has the ability to destroy any incoming agents without leaving any trace of their existence. So, the first task of the mobile agents operating in the network must be to locate this malicious node. Note that the most trivial optimization parameter to ensure while locating this malicious host (also termed as a black hole) is that a minimum number of agents gets destroyed by this node. This problem of locating the black hole by mobile agents is termed as *black hole search* (also termed as BHS problem) problem. The BHS problem has been studied since the year 2006, when Dobrev et al. [12] first introduced it. After this, till date, there have been many variations to this problem, some of them are [7, 8, 11, 13, 15]. This problem has various real-life implications, for example the black hole can be a virus in the network or it can be some crash failure, such that this node resembles the characteristic of a black hole, after failure.

Observe that, to detect the black hole, there needs to be some agent that has to visit that particular node. Further, since any agent visiting the node gets destroyed, there must be some communication tool that can convey this information to other alive agents, such that at least one agent remains alive knowing the location of the black hole. Three such communication tools have been predominantly used in literature: the *whiteboard* model [14], in which there is a storage capacity at each node, which an agent can use to leave a message by reading its contents and writing some new information, the *pebble* model [16], where an agent can carry a movable token from one node to another, and the *face-to-face* model [10], where an agent can share and communicate with another agent when they are at the same node at the same time. In addition to the communication tools, the initial locations of the agents (i.e., whether the agents are initially scattered [15] or they are co-located [10]) is also one of the important parameters, generally studied in the literature.

Further, the most studied version of a black hole has a fairly basic nature, i.e., only destroying any incoming agent. Note that, in reality black holes may not be so simple; they may have many ways to disrupt the movement or harm an agent. Considering this phenomenon, we in this paper have tried to consider a black hole that has more capabilities other than just destroying any incoming agent. In our case a black hole may or may not kill any incoming agent; it may do so based on an adversary that decides when to destroy an incoming agent and when not to. Whenever it decides not to destroy an agent, it simply behaves like any other node in the network, disguising (or hiding) it among the rest of the nodes by creating no anomaly for the visiting agent. In addition to this, we have also considered that the black hole has further capabilities; it can also choose whether to destroy the message (i.e., stored data in case of a whiteboard, and a placed token in case of pebble) at that node along with the incoming agent. This choice is also maintained by an adversary as well. We call this kind of black hole a *Byzantine black hole*.

Our aim in this paper is to solve the problem of perpetual exploration in a network, i.e., visiting every node in the network infinitely often except for the Byzantine black hole, by the mobile agents. Previously, in [2, 19] the authors introduced a set of models for a

black hole, which has more capabilities other than just destroying an agent, refer to these malicious nodes as *gray hole*, *gray⁺ hole* and *red hole*, respectively, based on their capabilities. They considered the following characteristics: they can fake agents (i.e., copy their software code), change the whiteboard contents, change the ports different from the requested ones, or can change the FIFO ordering as well. In this context, they solved perpetual exploration in a ring (which they term as *periodic data retrieval* problem) by a team of asynchronous mobile agents under these aforementioned various black hole characteristics, in which the agents are initially co-located and each node in a network has a whiteboard. On the other hand, the results of above-mentioned papers only holds for the case when initially the agents are co-located and each node of the ring has a whiteboard. Note that there can be many other initial positions for the agent to start with, and also, whiteboard is a very powerful communication tool in this domain of study of mobile agents. So, in this paper, we investigate these gaps in the context of perpetual exploration, considering the presence of one Byzantine black hole using synchronous agents. Also note that the position of a Byzantine black hole can be arbitrary (except the starting locations of the agent) but fixed. The Byzantine black hole we considered is a generalized version of the gray hole, as it also can choose whether to erase any previous data stored at that node, the moment it acts as a black hole.

1.1 Related Works

The black hole search (i.e., BHS) problem is a prominent variation of exploration problem studied in the literature, a survey of which can be found in [20]. This problem is investigated under various topologies (such as trees [8], rings [13], tori [5], and in arbitrary and unknown networks [7, 11]). All these discussed networks are static in nature. Recently, there has been a lot of interest in dynamic networks. The papers [3, 4, 10], studied the BHS problem on a dynamic ring, dynamic torus and dynamic cactus graph, where the underlying condition is that, irrespective of how many edges are dynamic in nature, the network must remain connected at any time interval (which is also termed as *1-interval connected*). In rings, the BHS problem has been studied for different variants; the most predominant among are choice of schedulers (i.e., synchronous [6] and asynchronous [1]), communication tools (i.e., face-to-face [7], pebble [16] and whiteboard [1]), and initial position of the agents (i.e., co-located [1] and scattered [6]).

The most relevant papers related to our work are the papers by Královič et al. [19] and by Bampas et al. [2]. The paper by Královič et al. [19] is the first to introduce a variant of this black hole, where the black hole has the ability to either choose to destroy an agent or let it pass (which they term as *gray hole*). Further they extended the notion of a gray hole, where the gray hole has the following additional capabilities: it has the ability to alter the run-time environment (i.e., changing the whiteboard information), or it has the ability to not to maintain communication protocol (i.e., to not maintain the FIFO order). They solved this problem under an asynchronous scheduler on a ring only when the agents are initially co-located and each node in the network has a whiteboard. The following results are obtained by them. They gave an upper bound of 9 agents for performing periodic data retrieval (i.e., which is equivalent to perpetual exploration) in the presence of a gray hole; further, in addition to gray hole, when the whiteboard is unreliable as well, they proposed an upper bound of 27 agents. Next, Bampas et al. [2] significantly improved the earlier results. They showed a non-trivial lower bound of 4 agents and 5 agents for gray-hole case and for case of a gray hole with unreliable whiteboard, respectively. Further, with 4 agents as well, they obtained an optimal result for the gray hole case, whereas with 7 agents they proposed a protocol for the case with a gray hole and unreliable whiteboard. As far as we are aware,

we are the first to investigate the perpetual exploration problem of a ring under different communication tools (i.e., face-to-face, pebble and whiteboard) as well as for different initial positions (i.e., co-located and scattered), for a variant of a gray hole, where it can erase any previously stored information but can not alter it. We term this type of gray hole a Byzantine black hole. In the following part, we discuss the results we have obtained.

Our Contribution. In this paper, we investigate the perpetual exploration problem, by a team of synchronous mobile agents, of a ring R of size n , in the presence of a *Byzantine black hole*. First, we consider the case when the agents are initially co-located. We obtain the following results.

- A:** For *Pebble* model of communication, we obtain that 3 agents are necessary and sufficient to perpetually explore R .
- B:** For *Face-to-Face* model of communication, we obtain that 5 agents are sufficient to perpetually explore R .
- C:** For *Whiteboard* model as well, we achieve the same lower and upper bounds as mentioned in **A**. This result shows that, by considering the scheduler to be synchronous instead of asynchronous (as assumed in [2]), the bound on the number of agents to perpetually explore R , reduces from 4 to 3 rendering it tight.

Next, we consider the case when the agents are initially scattered, and in this context, we obtain the following results:

- D:** For *Pebble* model of communication, we show that 4 agents are necessary and sufficient to explore R perpetually .
- E:** For *Whiteboard* model of communication, we obtain an improved bound of 3 agents (in comparison to **D**), which is necessary and sufficient to explore the ring R perpetually.

In the following Table 1, we have summarized the results.

■ **Table 1** Summary of our results.

		Whiteboard	Pebble	Face-to-Face
Co-located	Upper Bound	3	3	5
	Lower Bound	3	3	3
Scattered	Upper Bound	3	4	–
	Lower Bound	3	4	Non-Constant [9]

Organisation. Rest of the paper is organised as follows. Section 2 presents the model and preliminaries. In Section 3, we discuss some impossibility results. In Section 4 and 5, we propose perpetual exploration algorithms for the agents when the agents are initially co-located and scattered, respectively. Lastly, we conclude in Section 6. Due to page limitations we are omitting the detailed descriptions of the algorithms and the proofs of the theorems and lemmas. These can be found in the full version [17].

2 Model and Preliminaries

In this paper, we consider the underlying topology of the network as an oriented ring $R = \{v_0, v_1, \dots, v_{n-1}\}$. Each node v_i (where $i \in \{0, 1, \dots, n-1\}$) is unlabeled and has two ports connecting $v_{(i-1) \bmod n}$ and $v_{(i+1) \bmod n}$, labeled consistently as *left* and *right*. A set $A = \{a_0, a_1, \dots, a_{k-1}\}$ of k agents operates in R . We consider two types of initial positions for the set A of agents. In the first type, each agent in A is *co-located* at a node, which we

term as their *home*. In the second type, the agents can start from several distinct nodes, which we term as *scattered* initial positions. Each agent has knowledge of the underlying topology R and possesses some computational capabilities, thus requiring $\mathcal{O}(\log n)$ bits of internal memory. The agents have unique IDs of size $\mathcal{O}(\log k)$ bits taken from the set $[0, k^c]$ (c is a constant), which are perceived by other agents when they are co-located. The agents are autonomous and execute the same set of rules (i.e., they execute the same algorithm).

We consider three types of communication that the agents have in order to communicate with other agents. *Face-to-Face* (F2F): In this model, an agent can communicate with another agent when they are co-located. *Pebble*: In this model, the agents are equipped with a movable token (also termed as “pebble”), which signifies a single bit of information. The agents can carry a pebble from one node in a fairly mutually exclusive way and also can drop it on any other node. Agents use this pebble to mark some special nodes, for other agents to distinguish it. *Whiteboard*: In this case, each node of R contains $\mathcal{O}(\log n)$ bits of memory, which can be used to store and maintain information. Any agent can read the existing information or write any new information on the whiteboard of its current node. Note that fair mutual exclusion is maintained, i.e., concurrent access to the whiteboard data is not permitted.

The agents operate in synchronous rounds, and in each round, every agent becomes active and takes a local snapshot of its surroundings. For an agent at a node v in some round r , the snapshot contains two ports incident to v , already stored data on the memory of v (if any, only in the case of the whiteboard model of communication), the number of pebbles located at v (if any, only in the case of the pebble model of communication), contents from its local memory, and IDs of other agents on v . Based on this snapshot, an agent executes some action. This action includes a *communication* step and a *move* step. In a communication step, an agent can communicate implicitly or explicitly with other agents according to the communication models discussed above. In the move step, the agent can move to a neighbouring node by following a port incident to v . Thus, if an agent at node v in round r decides to move during the move step, in round $r + 1$, it resides on a neighbour node of v . All these actions are atomic, so an agent cannot distinguish another agent concurrently passing through the same edge; instead, it can only interact with another agent (based on its communication model) when it reaches another node.

A *black hole* is a stationary malicious node in an underlying graph, which has the ability to destroy any visiting agent without leaving any trace of its existence. Furthermore, the black hole nature of the node is controlled by an adversary. In addition to this, whenever the black hole nature is activated, the adversary can also choose to destroy any information stored at that node. We term this kind of node a *Byzantine Black Hole*.

In this paper, we assume that the underlying graph contains a single Byzantine black hole, while the other nodes are normal nodes, termed as *safe nodes*. It is assumed that the starting positions of each agent must be a safe node. This Byzantine black hole node is unknown to the agent. Here, we assume that if the adversary decides to activate the black hole nature of the Byzantine black hole node, it does so at the beginning of its corresponding round, and the node retains that nature until the end of this current round. Furthermore, we have considered that our Byzantine black hole has the ability always to destroy any incoming agent during its black hole nature and also choose to destroy any information present on that node. This paper aims to perpetually explore the ring R with the minimum number of agents. Next, we formally define our problem:

► **Definition 1** (PEREXPLORATION-BBH). *Given a ring network R with n nodes, where one node (v_b) is a Byzantine black hole, and with a set of agents A positioned on R , the PEREXPLORATION-BBH, asks the agents in A to move in such a way that each node of R , except v_b , is visited by at least one agent infinitely often.*

3 Impossibility Results

Here, at beginning, we state the first impossibility result which gives us a lower bound on minimum number of agents required to solve PERPEXPLORATION-BBH.

► **Theorem 2.** *A set of two synchronous agents in a ring R of size n (where $n \geq 4$) cannot solve PERPEXPLORATION-BBH, even in the presence of a whiteboard if number of possible consecutive black hole positions is at least 3.*

The main idea of the proof of the above Theorem 2 is that we create a scenario using adversarial techniques, where after one agent is destroyed by the Byzantine black hole v_b (say), the other and only alive agent ends up with at least two possible choices for the position of the black hole, which in turn creates confusion for the agent. Hence it is unable to correctly detect the black hole node among these two sets of nodes. In this scenario it is impossible for the only alive agent to successfully explore the whole ring except v_b perpetually, either without correctly detecting the Byzantine black hole position, or without getting destroyed as well. As a direct implication of Theorem 2, we can have the following corollaries. (Please see Section 7 in the full version [17], for the detailed proof)

► **Corollary 3.** *A set of three synchronous agents are required to solve the PERPEXPLORATION-BBH on a ring R with n (where $n \geq 4$) nodes, where each node of R has a whiteboard.*

Note that this lower bound of 3 agents is also true for agents with the pebble model of communication, as the pebble model of communication can be easily simulated in the whiteboard communication model.

► **Corollary 4.** *A set of two agents, each equipped with $O(\log n)$ pebbles, can not solve the PERPEXPLORATION-BBH problem on a ring R with n nodes.*

Our next result further improves the lower bound for number of agents, when agents are scattered and have pebble model of communication.

► **Theorem 5.** *A set of 3 scattered agents, each equipped with a pebble, can not solve the PERPEXPLORATION-BBH problem on a ring R with n nodes.*

We have proved Theorem 5 using an adversarial technique. We created an instance of three copies of the same ring R (i.e., R_1 , R_2 and R_3) in such a way that the position of Byzantine black hole (v_b) are different in each of them but they are consecutive in R . Also, in each case, the agents are initially placed at the nodes of R , which are at a distance of equal length between each other. Now, when an agent is destroyed by v_b , the other two agents can not distinguish between these three copies (due to same information on nodes for each of them). So, the problem reduces to two agents with 3 pebbles solving PERPEXPLORATION-BBH where a number of possible consecutive black hole positions is at least three. Then by Corollary 4, we can have the desired result (For detailed proof, see Section 8 in the full version [17]). Next, corollary is a direct consequence of Theorem 5.

► **Corollary 6.** *A set of four scattered agents, each with a pebble, are required to solve PERPEXPLORATION-BBH on a ring R with n nodes.*

4 Perpetual Exploration with Co-located Agents

In this section we assume that initially agents are co-located at a node which is termed as *home*. On the basis of this assumption here we investigate the sufficiency for the number of agents to solve PERPEXPLORATION-BBH problem under different model of communication.

4.1 Pebble Model of Communication

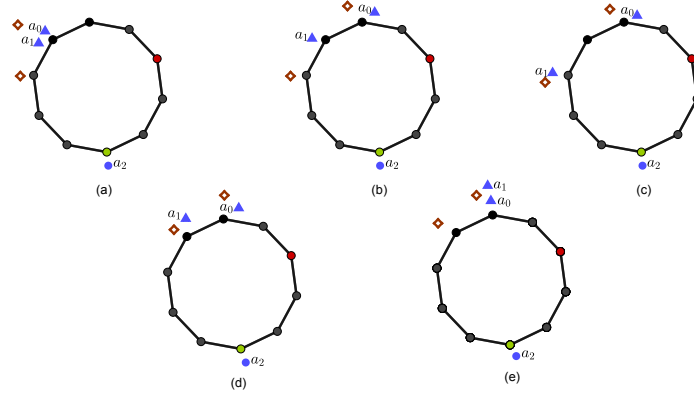
In this section, we consider the communication model, where the starting node has $k - 1$ identical and movable tokens (termed as pebbles), where k is the total number of agents deployed. A pebble can be carried by an agent from one node to another. This pebble acts as a mode of communication for the agents, as the agents can perceive the presence of a pebble at the current node. Moreover, an agent can also perceive the presence of other agents which are co-located at the current round (i.e., gather the IDs of the other co-located agents). Our main aim is to prove the following theorem in this section.

► **Theorem 7.** *A team of 3 co-located, synchronous agents are necessary and sufficient to solve PERPEXPLORATION-BBH on a ring R , with n nodes under the pebble model of communication, with the presence of two pebbles initially co-located with the agents.*

The necessary part follows from Corollary 4. For the sufficiency part we present an algorithm, PERPEXPLORE-COLOC-PBL. We have provided a very brief description of the algorithm. For detailed description and pseudo code see Section 9 in the full version [17].

Brief Description of the Algorithm. Given a ring R with a Byzantine black hole node v_b , a set of agents, $A = \{a_1, a_2, a_3\}$ (where we assume, ID of $a_1 < \text{ID of } a_2 < \text{ID of } a_3$), are initially co-located at *home*. The agents have no knowledge about the position of the node v_b , the only set of knowledge the agents have are: the total number of nodes in the underlying topology R (i.e., n), and also the knowledge that the initial node, i.e., *home* is safe. So, the remaining arc of $n - 1$ nodes in R is a suspicious region (which we term as S , where the cardinality of S i.e., $|S|$ denotes the length or size of the suspicious region) for each of the three agents.

The main idea of this algorithm is as follows: initially two agents a_1 and a_2 explore the ring perpetually, while a_3 waits at the *home*. a_1 and a_2 explore the ring R , executing the rule described as follows. If all the three agents are at *home* along with two pebbles, then in the r -th round ($r \geq 0$), a_1 moves clockwise with one pebble and it waits at round $r + 1$ for a_2 to arrive. In round $(r + 1)$, a_2 , leaving the remaining pebble at *home* moves to the next node along clockwise direction to meet a_1 . In the subsequent round, i.e., $(r + 2)$ -th round, a_1 after meeting a_2 moves again to the next node in the clockwise direction and waits for 3 rounds (i.e., till $(r + 5)$ -th round) for a_2 if and only if a_1 is not at *home*. In the mean time at $(r + 3)$ -th round, a_2 leaves its current node, moves one step in counter-clockwise direction, collects the left behind pebble. Then in $(r + 4)$ -th round a_2 moves again to the earlier node in clockwise direction. Subsequently, in $(r + 5)$ -th round, a_2 again leaves the pebble at its current node, and moves in a clockwise direction, to meet a_1 (which is waiting at the current node for a_2 to arrive), for better reference see Fig. 1. Note that, when a_2 meets a_1 outside *home*, the pebble it was carrying is left by a_2 at the nearest counter-clockwise node of the meeting node (i.e., where a_2 meets with a_1). Now from round $(r + 6)$ onwards to $(r + 9)$ -th round, the same execution repeats, as it has occurred between round $(r + 2)$ to round $(r + 5)$. The only difference is that, if both agents (i.e., a_1 and a_2) at round $r + 2$ were at a node v_1 , at round $r + 6$, they are on the nearest clockwise node of v_1 (i.e., at v_2 , say). This process continues until a_1 reaches *home*. In this case a_1 does not move clockwise, until a_2 meets it with the pebble it was carrying, in that case, it waits further until a_2 brings that pebble back at *home*. For this to happen after reaching *home*, 5 rounds are sufficient for waiting. When a_2 reaches *home* with the pebble, then all 3 agents are again at the *home* with 2 pebbles. This way the 3 agents explore the ring. The exploration can hamper if either a_1 or, a_2 , or a pebble is destroyed by the Byzantine black hole v_b while the above mentioned procedure is executed by them.



■ **Figure 1** An execution of PERPEXPLORE-COLOC-PBL, starting from the configuration where a_0 and a_1 are together on a vertex, to the configuration where a_0 and a_1 are on the same vertex again, which is the clockwise neighbour of the earlier vertex. The red node is the byzantine black hole, the green node is the *home* and red boxes are pebbles.

Firstly, let only a_1 is destroyed at some round t . Then there must exists a round $t' < t$, when a_2 was with a_1 at some node v_0 (say) for the last time. This means at round t' , a_1 moved clockwise to the next node v_1 along with the pebble it was carrying, and it must have been destroyed before or at the round when a_2 reaches v_1 again (in this case v_1 is v_b). Now when a_2 reaches v_1 there can be two cases. Either a_2 remains alive or (as the adversary may choose not to destroy a_2), it can get destroyed as well. For the initial case, a_2 identifies the Byzantine black hole by not seeing a_1 there and then it can leave that node and can explore the ring perpetually avoiding the node. The latter case is equivalent to both a_1 and a_2 are destroyed by the Byzantine black hole $v_b = v_1$. Note that since outside *home* whenever a_2 reaches the same node of a_1 , it leaves behind a pebble at the nearest counter-clockwise node (here v_0). For this case, another agent (i.e., a_3) which was waiting at *home*, finds none of the two exploring agents returns at *home*, even after waiting for a sufficient number of rounds. This incident triggers a_3 to move clockwise until it finds a pebble (one that is left behind by a_2 at v_0). Whenever the pebble is found, a_3 knows that the next clockwise node is the Byzantine black hole and it starts exploring R avoiding that node.

Now, there can be two other cases that can hamper the above mentioned exploring procedure. For the first case, let only a_2 is destroyed at some round $t > 0$. Let $t' < t$ be the last round before t when both a_1 and a_2 were together at a node v_0 . At round t' , a_1 , moves clockwise to the next node v_1 . Now if a_2 is not destroyed in the next 3 rounds, then it meets a_1 at v_1 again, contrary to our assumption. So, if only a_2 is destroyed then it must be at any of the rounds $t' + 1, t' + 2$ or $t' + 3$. In this 3 round a_2 can be either on v_0 or v_{-1} where, v_{-1} is the counter-clockwise nearest node of v_0 . In this case, when a_1 finds a_2 has not arrived at v_1 even after waiting for 3 rounds, in which case, it understands that either v_0 or v_{-1} is the Byzantine black hole. This knowledge triggers a_1 to move clockwise until it meets a_3 at *home*, leaving the pebble along with it behind at v_1 . When a_3 sees that even after certain round of waiting only a_1 is able to arrive at *home*, and that too without the pebble it was supposed to be carrying, a_3 understands that a_1 must have detected an anomaly. In this case both a_1 and a_3 moves counter-clockwise together to find the pebble left by a_1 . When they find it they declare v_1 as *home*. After this they start exploring the ring in different directions in the following way. a_1 moves clockwise until v_{-1} and a_3 starts moving counter-clockwise until v_0 . After that, they both move in opposite direction until they again

reach *home* (which is the new *home*) and waits for sufficient number of rounds to meet each other. This way the exploration keeps going on until one among a_1 or a_3 is destroyed. Let a_3 fails to reach *home*. In that case, a_1 will detect that v_0 must be the Byzantine black hole and it can start exploring the ring R avoiding v_0 . On the other hand if a_1 fails to reach then it must be destroyed by the Byzantine black hole which is nothing but the node v_{-1} . Knowing this a_3 then explores the ring avoiding that node. Now there can be another case when no agents but a pebble is destroyed at v_b . This pebble must be the pebble that is left behind at v_{-1} by the agent a_2 when it reaches v_0 to meet a_1 . In this case, only the pebble can be destroyed before a_2 reaches to collect the pebble back. So, when a_2 reaches v_{-1} , it finds out that there is no pebble. This leads to a_2 , knowing that v_{-1} is the Byzantine black hole and in which case it starts exploring the ring R avoiding that node. If a_2 is also destroyed while it reaches v_{-1} to collect the pebble, this case is similar to the case where we described the algorithm when only a_2 is destroyed.

Sketch of Correctness. To prove the correctness of algorithm PEREXPLORE-COLOC-PBL we prove the following Theorem. Here we just present a sketch of the proof. For details see Section 9.2 in the full version [17].

► **Theorem 8.** *Algorithm PEREXPLORE-COLOC-PBL solves PEREXPLORATION-BBH on a ring R with 3 co-located and synchronous agents under the pebble model of communication with the presence of two pebbles initially co-located with the agents.*

We first prove that if no agents are destroyed then either the agents continue with the exploration of ring R without knowing the exact location of the Byzantine black hole, or there exists at least one agent that knows the exact location of the Byzantine black hole which can then explore the ring indefinitely avoiding the Byzantine black hole (See Lemma 15 in the full version [17]). Next we proved that, if one agent is destroyed while exploring the ring, then either there exists one agent that identifies the Byzantine black hole uniquely and continues to indefinitely explore all nodes of the ring, except the Byzantine black hole or the exploration continues while the length of the suspicious region (i.e., $|S|$) decreases to 2 from $n - 1$ (Lemma 17 in the full version [17]). For the later case we proved that the exploration continues until another agent is destroyed by the Byzantine black hole (Remark 18 in the full version [17]). In this case when two agents are destroyed by the Byzantine black hole, the only alive agent can uniquely identify the Byzantine black hole without moving on to it (Lemma 19 in the full version [17]). Thus it can then perform the exploration of R avoiding the Byzantine black hole. This proves Theorem 8.

4.2 Face-to-Face Model of Communication

In this section, we consider the Face-to-Face (also termed as F2F) model and prove the following theorem.

► **Theorem 9.** *A team of 5 co-located and synchronous agents are sufficient to solve PEREXPLORATION-BBH on a ring R with n nodes under the F2F model of communication.*

In order to prove Theorem 9, we discuss an algorithm with 5 co-located agents, where each agent can communicate among themselves at the same node at the same round (i.e., each agent have F2F model of communication). Initially 3 lowest ID agents, say, a_1 , a_2 and a_3 are chosen, where fourth lowest ID agent say, a_4 , is associated with a_1 and the largest ID agent, say a_5 , is associated with a_2 . The idea of the algorithm resembles to that of the algorithm PEREXPLORE-COLOC-PBL. Note that as in this case there is no existence of

17:10 Perpetual Exploration of a Ring in Presence of Byzantine Black Hole

pebbles, hence, we configure the behaviour of the agents a_4 and a_5 in such a way that they act as pebbles, associated with a_1 and a_2 in PEREXPLORE-COLOC-PBL, respectively. In PEREXPLORE-COLOC-PBL when we say that an agent a_i carries a pebble p , in this case we mean that the agent a_i communicates a message `carry` with its associated agent $a_{i'}$ such that both these agents simultaneously move together until further new instruction is communicated. On the contrary as per PEREXPLORE-COLOC-PBL when we say that an agent a_i drops a pebble p at a node v , then in this case we mean that a_i communicates a message `drop` with $a_{i'}$, which in turn instructs $a_{i'}$ not to move further and remain stationary at the node v until further instruction is communicated. Hence, with this terminology, a team of 5 agents can execute the algorithm PEREXPLORE-COLOC-PBL, and the correctness also follows similarly. This shows that a team of 5 co-located and synchronous agents are sufficient to solve PEREXPLORATION-BBH under F2F model of communication. This proves Theorem 9.

4.3 Whiteboard Model of Communication

The algorithm PEREXPLORE-COLOC-PBL can be simulated in whiteboard model of communication. A pebble on a node can be simulated by a bit of information, which is marked on the whiteboard of that node. Note that, collecting a pebble from the node is simulated by erasing the same bit of information, on that node and dropping the pebble can be simulated by marking the node with a bit of information as well on the whiteboard of that node. From this we get the following result for whiteboard model of communication.

► **Theorem 10.** *A team of 3 synchronous co-located agents are necessary and sufficient to solve PEREXPLORATION-BBH on a ring R if each node are equipped with a whiteboard having constant memory.*

Previously in [2], for asynchronous scheduler the tight bound for number of agents to solve this problem was 4. Now from Theorem 10, we get a trade-off between reducing the optimal number of agents required vs the scheduler, in presence of a more generalized version of gray hole as well (i.e., Byzantine black hole).

5 Perpetual Exploration with Scattered Agents

Here in this section, we discuss the problem of PEREXPLORATION-BBH, when the agents are initially scattered on more than one nodes, where each of these starting nodes are assumed to be safe. We investigate this problem, under different communication models. First, we discuss the pebble model of communication and subsequently we discuss the whiteboard model of communication.

5.1 Pebble Model of Communication

Our goal in this section is to prove the following theorem.

► **Theorem 11.** *A team of 4 synchronous scattered agents with one pebble each is necessary and sufficient to solve the PEREXPLORATION-BBH problem on a ring R with n nodes, when the agents are initially scattered on R .*

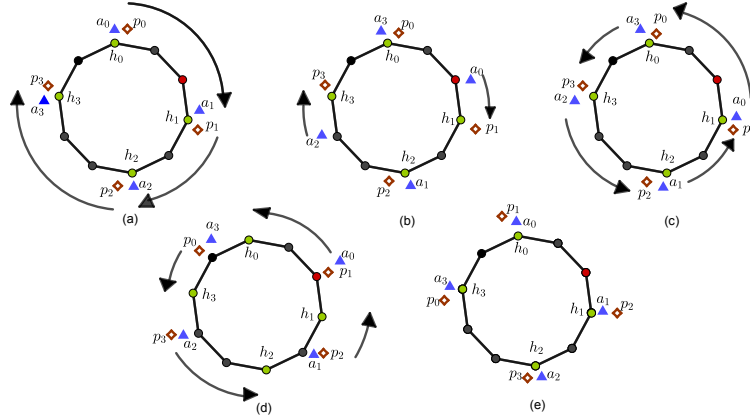
The necessary part follows from Corollary 6. For the sufficient part, we present an algorithm PEREXPLORE-SCAT-PBL, that can solve the PEREXPLORATION-BBH problem in the above context. In this algorithm we assumed that 4 agents are initially scattered at 4

different nodes of R . To include the remaining cases where the agents are initially scattered at less than or equal to 3 nodes, a slight modification of PEREXPLORE-SCAT-PBL is enough which we describe in Remark 26 in the full version [17]. Here we describe only the main idea of the algorithm. The detailed description and pseudocodes of the algorithm can also be found in Section 10.1 in the full version [17].

Brief Idea of the Algorithm. Let us consider, the starting position of a_0 to be the first, after which the starting position of a_1, a_2 and a_3 , respectively follows in clockwise order. Let h_i be the starting node of an agent a_i (i.e., *home* of a_i), where $i \in \{0, 1, 2, 3\}$. By $Seg(a_i)$ we define the clockwise arc starting from the node h_i and ending at $h_{(i+1) \pmod{4}}$ (called *segment* of a_i). If no agents are destroyed by the Byzantine black hole v_b then the exploration of R goes as follows. Note that in this explanation, if we say a_k or h_k , we mean $a_{k \pmod{4}}$ or $h_{k \pmod{4}}$, for some $k \geq 0$.

Agent a_i moves clockwise along $Seg(a_i)$ leaving its pebble at h_i until it reaches $h_{(i+1)}$ (i.e., the end of $Seg(a_i)$). An agent can distinguish the node $h_{(i+1)}$ by seeing the pebble left there by the agent $a_{(i+1)}$. When a_i reaches $h_{(i+1)}$ traversing $Seg(a_i)$ for the first time, it knows the length of the segment $Seg(a_i)$ and stores the length in its own memory. For further traversals it does not depend on seeing pebbles at the end nodes of the segment. It can simply use the length of the segment. After a_i reaches $h_{(i+1)}$, it waits there for a certain number of rounds so that the other agents can in the meantime reach the endpoints of there corresponding segments, while moving along a clockwise direction. After this waiting, a_i collects the pebble (if exists) from $h_{(i+1)}$ (the pebble left by $a_{(i+1)}$) and starts moving counter-clockwise along with the collected pebble until it reaches its own *home*, i.e., h_i along with the pebble it was carrying. The waiting time at $h_{(i+1)}$ also ensures that all agents starts moving counter-clockwise at the same round. Also note that, if a_i does not find any pebble to collect at $h_{(i+1)}$ it starts moving counter-clockwise without any pebble (this case can only happen if $a_{(i+1)}$ is destroyed at the Byzantine black hole node v_b , while it was returning back after collecting the pebble from $h_{(i+2)}$ in some previous round). After a_i reaches h_i moving counter-clockwise, it again waits for another set of rounds before it repeats the whole process again. Again, the waiting time at h_i is configured in such a way that all agents start repeating the process at the same round (for details on the precise value of these waiting times, refer to the detailed description of the algorithm in Section 10.1 in the full version [17]). Note that, since $\cup_{i=0}^3 Seg(a_i) = R$, if no agents are destroyed, the perpetual exploration of R happens. The exploration can be hampered only if an agent is destroyed at the Byzantine black hole node v_b . Note that, since $Seg(a_i) \cap Seg(a_j)$ is either empty or consists of a safe node and in addition to that, as a segment is explored by only one agent. Hence, at most one agent can be destroyed while exploring their respective segments, as described above. Without loss of generality, let $v_b \in Seg(a_j)$, for some $j \in \{0, 1, 2, 3\}$. Now there are two cases.

Case-I: Let a_j is destroyed while moving clockwise. For this case, a_j fails to collect the pebble at $h_{(j+1)}$ left by $a_{(j+1)}$ and return h_j . So, when $a_{(j+1)}$, returns $h_{(j+1)}$ after moving counter-clockwise along with the pebble it has collected from $h_{(j+2)}$, it finds two pebble at $h_{(j+1)}$. This is considered by $a_{(j+1)}$ as an anomaly and it learns that v_b must be in the arc, which is in the counter-clockwise direction starting from $h_{(j+1)}$. In this scenario, $a_{(j+1)}$ waits a certain number of rounds (if needed) to ensure that every other alive agents have reached their corresponding *home*. Then $a_{(j+1)}$ starts moving clockwise with both the pebbles. The aim of this move is to meet and gather with the other alive agents. Note that the other alive agents wait at their corresponding *home*, after moving along the counter-clockwise



■ **Figure 2** (a) h_i 's are home marked as green. Agent a_i is on h_i initially with a pebble. We name the pebble initially at h_i as p_i , but in reality they are anonymous. (a-b) Each agent moves clockwise until the next home (i.e., $h_{(i+1)}$) without carrying any pebble. The agents already reached (here a_1 and a_3) waits for others. (c-e) All agents are on their clockwise nearest home. They start moving counter clockwise together with the pebble present at their current location towards their initial home. The agents which already reaches their home, wait for others to reach their home.

direction. It is because, at their respective home they do not detect any anomaly. The waiting time is provided in such a way that it is enough for $a_{(j+1)}$ to detect anomaly and meet both the remaining alive agents after moving clockwise, while the other agents are waiting at their home. The agent $a_{(j+1)}$ first meets $a_{(j+2)}$ at $h_{(j+2)}$ while it moves clockwise after detecting anomaly. Then both of them moves together, while $a_{(j+2)}$ carries the pebble, which it was earlier carrying back to home. They move until they meet with $a_{(j+3)}$ at $h_{(j+3)}$. Note that at this moment 3 agents are at a node (which is $h_{(j+3)}$), with at least 3 pebbles (one carried by $a_{(j+3)}$ and two carried by $a_{(j+2)}$). In this case they execute the algorithm PEREXPLORE-COLOC-PBL and achieve PEREXPLORATION-BBH of the ring R .

Case-II: Let a_j be destroyed at v_b while it was moving counter-clockwise along with the pebble it collected from $h_{(j+1)}$. Then in this case, when all alive agents return to their corresponding home, none of them finds any anomaly as all agent sees exactly one pebble at their home. So, they wait and start the exploration again at the same round. Note that in this scenario, all agents except a_j , move clockwise to the end points of their corresponding segments, waits there and collects pebble (if any) and moves back to their corresponding home again. Now, since a_j was destroyed earlier, the pebble left by $a_{(j+1)}$ was picked by no agents and thus, while $a_{(j+1)}$ returns back to $h_{(j+1)}$ it finds two pebbles and does the same execution as explained in *Case-I*.

The basic idea of this algorithm is to gather three agents with at least 3 pebbles (refer Remark 20 in the full version [17]) at the expense of one agent and then execute PEREXPLORE-COLOC-PBL to solve the PEREXPLORATION-BBH problem.

Sketch of Correctness. To prove the correctness of the algorithm PEREXPLORE-SCAT-PBL, we have to prove the following theorem. Here we only provide the proof idea. For details see Section 10.2 in the full version [17]). Fig. 2 illustrates the working of the algorithm PEREXPLORE-SCAT-PBL.

► **Theorem 12.** *Algorithm PEREXPLORE-SCAT-PBL solves PEREXPLORATION-BBH problem of a ring R with 4 synchronous and scattered agents under the pebble model of communication where each agents are equipped with a pebble.*

First, we proved that algorithm PEREXPLORE-SCAT-PBL guarantees exploration if no agents are destroyed (Corollary 24 in the full version [17]). This corollary is a direct consequence of the Lemma 23 also present in the full version [17]. Further we showed that if an agent is destroyed then exactly one agent detects anomaly and that agent gathers with the remaining two alive agents within finite rounds This can be found in Lemma 25 in the full version [17]. The rest of the proof follows from Theorem 8.

5.2 Whiteboard Model of Communication

The main aim of this section is to prove the following theorem.

► **Theorem 13.** *A team of 3 synchronous agents are necessary and sufficient to solve the problem PEREXPLORATION-BBH on a ring R with n nodes, when each node of R has a whiteboard of $O(\log n)$ bits of memory, irrespective of their starting location.*

The necessary part follows from Corollary 3. The sufficiency part for co-located initial situation follows from Theorem 10. For the other cases where the agents are initially scattered at more than one starting node, we propose an algorithm PEREXPLORE-SCAT-WHITBRD. This algorithm is designed for the case when all the agents are starting at different nodes. Note that, this algorithm can be easily modified a bit to include the case where initially three agents are scattered at two distinct nodes (refer the modification in Section 11.3 in the full version [17]). For detailed description and pseudo code, see Section 11.1 in [17].

Brief Description of the Algorithm. Let a_0, a_1 and a_2 be three agents starting from the nodes h_0, h_1 and h_2 , respectively, where these nodes are in a clockwise order. Let $Seg(a_i)$ (also called “Segment of a_i ”) be the clockwise arc starting from h_i and ending at $h_{(i+1) \pmod 3}$. The algorithm first ensures that the ring is explored perpetually if no agents are destroyed. In order to do this, the algorithm goes as follows. Note that in this explanation, if we say a_k or h_k , we mean $a_{k \pmod 3}$ or $h_{k \pmod 3}$, for some $k \geq 0$.

An agent, say a_i , first erases all previously stored data (if any) at h_i . Then it writes the message (**home**, $ID(a_i)$) at h_i and starts moving clockwise. This type of message is called a **home** type message that indicates it is *home* of a_i . a_i moves clockwise until it reaches $h_{(i+1)}$. It distinguishes $h_{(i+1)}$ by seeing the **home** type message left by $a_{(i+1)}$. When a_i moves clockwise, it also marks each node of $Seg(a_i)$, except h_i and $h_{(i+1)}$, by writing **right**, after erasing any previous such markings (if at all exists) at each such nodes. After a_i reaches $h_{(i+1)}$ it waits for a certain number of rounds (if needed), so that the other agents (say a_j) gets enough time to reach endpoints of their corresponding segment (i.e., $h_{(j+1)}$). After this waiting, each alive agent a_i write the message (**visited**, $ID(a_i)$) at $h_{(i+1)}$ at the same round. This type of messages is termed as a **visited** type message, which indicates that an agent with its corresponding ID, also mentioned in the **visited** type message, has visited the respective node. Then each a_i waits for n rounds at $h_{(i+1)}$ and then starts moving counter-clockwise at the same round. a_i moves counter-clockwise until it reaches h_i , i.e., its own *home*. While moving counter-clockwise, a_i erases previously written **right** marking (which it marked while moving along clockwise direction) from each nodes of $Seg(a_i)$ and writes **left** there upon arriving (except at h_i and at $h_{(i+1)}$). When a_i reaches its own *home* (i.e., h_i), it waits again for a certain number of rounds, upon seeing the **visited** type message, left there by $a_{(i-1)}$. This waiting period is enough for each of the other alive agents to reach their corresponding *home*. After this waiting period is over, all of the agents, starts repeating the same procedure again together, from the same round. This procedure

17:14 Perpetual Exploration of a Ring in Presence of Byzantine Black Hole

ensures that, if no agent is destroyed at the Byzantine black hole node v_b , then the perpetual exploration of R continues. This can only be hampered, only if an agent gets destroyed at the node v_b . Without loss of generality let $v_b \in \text{Seg}(a_j)$ for some, $j \in \{0, 1, 2\}$. So only a_j can be destroyed at v_b , while performing this exploration. It is because a_j is the only agent to visit each node u of $\text{Seg}(a_j)$, where $u \in \text{Seg}(a_j) \setminus \{h_j, h_{(j+1)}\}$. There can be two cases.

Case-I: Let a_j be destroyed while it is moving in clockwise direction along $\text{Seg}(a_j)$. This implies it fails to reach $h_{(j+1)}$ and also fails to write a `visited` type message there. So, when $a_{(j+1)}$ returns to its *home* (i.e., $h_{(j+1)}$), it finds no `visited` type message (as it should've, if a_j was not destroyed). $a_{(j+1)}$ interprets from this that, the segment which is adjacent to its own segment in counter-clockwise direction has the Byzantine black hole and an agent must have been destroyed there while moving clockwise. This information triggers it to move clockwise with the aim of gathering with the other agent (i.e., $a_{(j+2)}$). Note that, when $a_{(j+1)}$ starts moving, the other agent is waiting and the waiting time is sufficient for the moving agent to meet it. After they meet, $a_{(j+1)}$ shares the information about its direction of movement in the whiteboard of $h_{(j+2)}$. With this, they again start moving clockwise together until they reach h_j . The agents can distinguish h_j by the `home` type message written there by a_j before it was destroyed. Note that in the clockwise direction each node after h_j up to the previous node of v_b were marked `right` by a_j before it was destroyed. So from h_j , $a_{(j+1)}$ and $a_{(j+2)}$ starts moving cautiously clockwise. That is, while both agents (a_L and a_H , where a_L and a_H are the agent with lowest ID and highest ID among $a_{(j+1)}$ and $a_{(j+2)}$ respectively) are at a node v_0 , a_L moves clockwise to the next node, say v_1 while the other agent waits at v_0 . If at v_1 , a_L sees the `right` marking, it interprets v_1 is safe. So, it comes back to v_0 . At v_0 , seeing that a_L has returned, a_H also interprets v_1 to be safe. So, in the next round both a_L and a_H moves to v_1 together and repeats the process from v_1 again. When a_L reaches v_b , it sees no `right` marking there. a_L if alive, interprets this by determining the current node to be Byzantine black hole. So, it starts perpetual exploration of R , except v_b . Otherwise, if a_L gets destroyed at v_b , it does not return to the previous node where a_H is waiting. Even after waiting, when a_H sees a_L has not returned, it interprets this incident as the next clockwise node is the Byzantine black hole and starts exploring the ring R avoiding that node. Thus for this case the perpetual exploration continues.

Case-II: Let a_j be destroyed at v_b while it is moving counter-clockwise. In this case both the alive agents $a_{(j+1)}$ and $a_{(j+2)}$ find `visited` type message after returning to their corresponding *home*. This is because, a_j is destroyed at v_b after writing `visited` type message at $h_{(j+1)}$. So, all alive agents after waiting, starts repeating the exploration procedure again. The agents first erase the whiteboard content at their corresponding *home* and then writes the corresponding `home` type message before it starts moving clockwise until the end of their corresponding segment. Note that since a_j was destroyed earlier (even before the repetition starts), $a_{(j-1)}$ finds the `visited` type message which was written there earlier by it, is still present (which was supposed to be erased if a_j was alive). From this information, $a_{(j-1)}$ deduces that, v_b must be in the segment adjacent to its own segment in the clockwise direction. It also deduces that, an agent must have been destroyed at v_b while it was moving counter-clockwise. This is because, if the agent was destroyed while it was moving clockwise then it would have been detected by the other agents when they are at their *home* (as described in *Case-I*), which is before the start of the next clockwise move (note that the agents have already started the next clockwise move). So, this information triggers $a_{(j-1)}$ to move counter-clockwise with the aim to gather with $a_{(j-2)}$. Since $a_{(j-1)}$ starts its move while $a_{(j-2)}$ waits at *home* and also, since the waiting time is sufficient, $a_{(j-1)}$ meets $a_{(j-2)}$ during the waiting period at $h_{(j-1)}$. So, after they meet, $a_{(j-1)}$, communicates the direction

of its move to $a_{(j-2)}$ on the whiteboard of $h_{(j-1)}$ and they move together until they reach $h_{(j-2)} (= h_{(j+1)})$ together. They distinguish the node by the `home` type message left there by $a_{(j-2)}$ before the start of moving clockwise. Note that, in the counter-clockwise direction, the nodes in $Seg(a_j)$, starting from the next node of $h_{(j+1)}$ up to the previous node of v_b are the only nodes that were marked `left` by a_j before it was destroyed. So, from $h_{(j+1)}$, the alive agents $a_{(j+1)}$ and $a_{(j+2)}$ start moving cautiously in the counter-clockwise direction. Among $a_{(j+1)}$ and $a_{(j+2)}$, an agent with lowest ID is denoted by a_L and the agent with highest ID is denoted by a_H . The cautious walk is same as described in Case-I, except for the fact that here after moving one node in the counter-clockwise direction, the agent a_L searches for the `left` marking. If it finds such marking, it returns to a_H and then both moves counter-clockwise and repeats the process. On the other hand if a_L does not find any `left` marking (it must be v_b) and it stays alive, a_L moves out of that node and starts exploring R avoiding that node. If, on the contrary a_L gets destroyed, then a_H sees that a_L has not returned while it should have. From this incident it interprets next counter-clockwise node is v_b and it starts exploring R avoiding that node.

Sketch of Correctness: Here we give a brief glimpse on how we proved the following theorem. For details see Section 11.2 in the full version [17].

► **Theorem 14.** *Algorithm PEREXPLORE-SCAT-WHITBRD solves PEREXPLORATION-BBH problem of a ring R with n nodes and with 3 synchronous agents initially scattered under the whiteboard model of communication.*

In order to prove this theorem, we first define a *cautious start node*. Let a_i be the agent destroyed at the Byzantine black hole first. We define the *cautious start node* to be h_i if a_i was moving clockwise when destroyed, otherwise it is $h_{(i+1)}$. We first prove that after one agent is destroyed, the remaining two agents gather at the cautious start node. This result can be found in details in Lemma 36 in the full version [17]. Note that a_i marks each node on the arc between the cautious start node and the Byzantine black hole with either `left` or `right` markings depending on the direction it was moving before it was destroyed. We proved that the alive agent can know the direction of a_i before it was destroyed. This can be found in details in Lemma 30 and Lemma 31 in the full version [17]. After the alive agents reach the cautious start node, they start moving cautiously looking for the marking based on the direction of a_i before it was destroyed. We proved that at least an agent among the two moving cautiously will stay alive knowing the exact location of the Byzantine black hole (This result can be found in details in Lemma 37 in the full version [17]). Hence this agent can explore R perpetually avoiding the Byzantine black hole. This proves Theorem 14.

6 Conclusion

The paper addresses perpetual exploration of a ring network in the presence of a malicious node that we call a *Byzantine black hole*. This problem, termed as PEREXPLORATION-BBH, is explored under three communication models (*Face-to-Face*, *Pebble*, and *Whiteboard*) considering various initial scenarios (*co-located* or *scattered* agents) with the aim of minimizing number of agents. We proposed optimal bound (in terms of number of agents) for both the *Pebble* and *Whiteboard* communication models under both initial scenarios. Further, an upper bound of 5 agents and a lower bound of 3 agents is provided for *Face-to-Face* model in the case of co-located agents.

Future research could focus on proposing an optimal bound for *Face-to-Face* co-located scenario, while also proposing constructive lower and upper bounds for *Face-to-Face* scattered scenario. Additionally, investigating this problem in different scheduler models can be another future direction.

References

- 1 Balasingham Balamohan, Paola Flocchini, Ali Miri, and Nicola Santoro. Time optimal algorithms for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 3(04):457–471, 2011. doi:10.1142/S1793830911001346.
- 2 Evangelos Bampas, Nikos Leonardos, Euripides Markou, Aris Pagourtzis, and Matoula Petrolia. Improved periodic data retrieval in asynchronous rings with a faulty host. *Theoretical Computer Science*, 608:231–254, 2015. doi:10.1016/J.TCS.2015.09.019.
- 3 Adri Bhattacharya, Giuseppe F Italiano, and Partha Sarathi Mandal. Black hole search in dynamic cactus graph. In *International Conference and Workshops on Algorithms and Computation*, pages 288–303. Springer, 2024. doi:10.1007/978-981-97-0566-5_21.
- 4 Adri Bhattacharya, Giuseppe F Italiano, and Partha Sarathi Mandal. Black hole search in dynamic tori. *arXiv preprint*, 2024. doi:10.48550/arXiv.2402.04746.
- 5 Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Black hole search with finite automata scattered in a synchronous torus. In *Distributed Computing: 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings 25*, pages 432–446. Springer, 2011. doi:10.1007/978-3-642-24100-0_41.
- 6 Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, 509:70–85, 2013. doi:10.1016/J.TCS.2013.02.010.
- 7 Jurek Czyzowicz, Dariusz Kowalski, Euripides Markou, and Andrzej Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2-3):229–242, 2006. URL: <http://content.iospress.com/articles/fundamenta-informaticae/fi71-2-3-05>.
- 8 Jurek Czyzowicz, Dariusz Kowalski, Euripides Markou, and Andrzej Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability and Computing*, 16(4):595–619, 2007. doi:10.1017/S0963548306008133.
- 9 Giuseppe A Di Luna, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Black hole search in dynamic rings: The scattered case. In *27th International Conference on Principles of Distributed Systems (OPODIS 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.
- 10 Giuseppe Antonio Di Luna, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Black hole search in dynamic rings. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 987–997. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00098.
- 11 Stefan Dobrev, Paola Flocchini, Rastislav Kráľovič, and Nicola Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, 2013. doi:10.1016/J.TCS.2012.11.022.
- 12 Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, 19:1–99999, 2006.
- 13 Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007. doi:10.1007/S00453-006-1232-Z.
- 14 Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007. doi:10.1007/S00453-006-1232-Z.
- 15 Stefan Dobrev, Nicola Santoro, and Wei Shi. Locating a black hole in an un-oriented ring using tokens: The case of scattered agents. In *European Conference on Parallel Processing*, pages 608–617. Springer, 2007. doi:10.1007/978-3-540-74466-5_64.
- 16 Paola Flocchini, David Ilcinkas, and Nicola Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62:1006–1033, 2012. doi:10.1007/S00453-011-9496-3.
- 17 Pritam Goswami, Adri Bhattacharya, Raja Das, and Partha Sarathi Mandal. Perpetual exploration of a ring in presence of byzantine black hole. *arXiv preprint*, 2024. doi:10.48550/arXiv.2407.05280.

- 18 Wayne A Jansen. Intrusion detection with mobile agents. *Computer Communications*, 25(15):1392–1401, 2002. doi:10.1016/S0140-3664(02)00040-3.
- 19 Rastislav Kráľovič and Stanislav Miklík. Periodic data retrieval problem in rings containing a malicious host. In *Structural Information and Communication Complexity: 17th International Colloquium, SIROCCO 2010, Şirince, Turkey, June 7-11, 2010. Proceedings 17*, pages 157–167. Springer, 2010. doi:10.1007/978-3-642-13284-1_13.
- 20 Mengfei Peng, Wei Shi, Jean-Pierre Corriveau, Richard Pazzi, and Yang Wang. Black hole search in computer networks: State-of-the-art, challenges and future directions. *Journal of Parallel and Distributed Computing*, 88:1–15, 2016. doi:10.1016/J.JPDC.2015.10.006.
- 21 Claude E Shannon. Presentation of a maze-solving machine. *Claude Elwood Shannon Collected Papers*, pages 681–687, 1993.